POLITECNICO
MILANO 1863

DREAM - Data-dRiven PrEdictive FArMing in Telangana

Design document

Version 1.0

Venturini Luca
Vernola Dario
Vittorini Francesco

Release date: January 9th, 2022

Table of contents

1. Introduction

1.1. Purpose

This document provides a description of DREAM software architecture, starting from an abstract view of the system and then going into the details of each component. It is shown a set of graphical user interfaces to better represent the features of the application. Finally, a discussion about implementation, integration and test of the components is presented.

DREAM is an application that aims to help farmers of Telangana, a region of India, dense with farmers who every day face different types of problems, starting from an unstable economic situation to changing weather conditions, that put at risk the harvest, and more in general their well-being.

The application will be used in a large area, since Telangana counts 35 million people, part of which are farmers or people involved in agriculture.
Moreover this sector is expanding every year, and so the applications will grow in number of users, and so it is expected to scale and serve more and more people.

The DREAM app has been proposed, in agreement with the government, in order to handle in a smart way the production of food by farmers. In fact in the next few years a huge increment of demand regarding food is expected, and the government wants to face it at the best of its possibilities, safeguarding the economy and, at the same time, the wellness of the farmers.

The objective of this application is to help farmers, involving other figures, like agronomists and policy makers.
Both of these actors will propose steering initiatives in order to improve the production and well being of farmers.
Agronomists will act at the local level, keeping in touch with farmers of the same zone, and helping them on the basis of their requests and their performances.
Policy makers act instead at a higher level, checking that such initiatives are effective and evaluating the farmers’ performances.

1.2. Scope

The application will provide a set of services to the different actors involved.

The farmers will be able to interact with their peers, they can access data that can help them in production, like weather forecasting or data retrieved by sensors and they will be encouraged to contact agronomists if needed.
Agronomists will be required to monitor the farmers living in their same zone, visiting the farms multiple times during the year, to check how they are working and answering their requests or helping them if needed.

Finally Policy Makers monitor the data produced by farmers and agronomists, to understand if the steering initiatives are working properly, and intervene when necessary introducing new strategies.

1.3. Definitions, Acronyms, Abbreviations

API: Application Programming Interface. It is a type of software interface, offering a service to other pieces of software;
DBMS: DataBase Management System. It is software designed to store, retrieve, define, and manage data in a database.
Sink/Base station: it is a term used to describe a computer or any other medium capable of receiving data.
Daily plan: Plan made by agronomists that contains a list of farms to visit on a specific day.
Sensor: We refer with this term to all the embedded devices used to get information from parameters of the surrounding environment
DREAM: It is the name of the application in object
Dataset: collection of data accessed by the application
Client-server architecture: a type of architecture which relies on the communication between a usually high computational power device called server and one or more clients (devices with low computational power in our case).
Three tier architecture: a client-server architecture which separates the logic behind the application in three distinct and separated tiers.
Thin client: a solution which implies that all of the processing and elaboration of data done by the application is executed through the servers, with the client responsible only to show the results of such elaborations through an interface.
Layered structure: an abstraction of a system which shows its components divided into layers stacked on top of each other based on their functionalities

1.4. Revision history

Version	Description
1.0	Initial version

1.5. Reference Documents

All the references are presented in **section 7**.

1.6. Document Structure

In **Section 1** a brief and general description of the purpose and scope of the DREAM project is presented, identifying the main objectives of the application and describing why it is useful and necessary.

Section 2 represents the core of this document. In this section the design decisions and the resulting DREAM software architecture are shown, described and motivated. First, a general overview of the architecture is presented. Then, the architecture is better described by component view, runtime views, deployment view, component interfaces and other useful details.

In **Section 3** the graphical user interfaces of the main functionalities are shown.

In **Section 4** the traceability matrices of requirements are shown. They map each component to the requirements which are satisfied using that particular component (and possibly others).

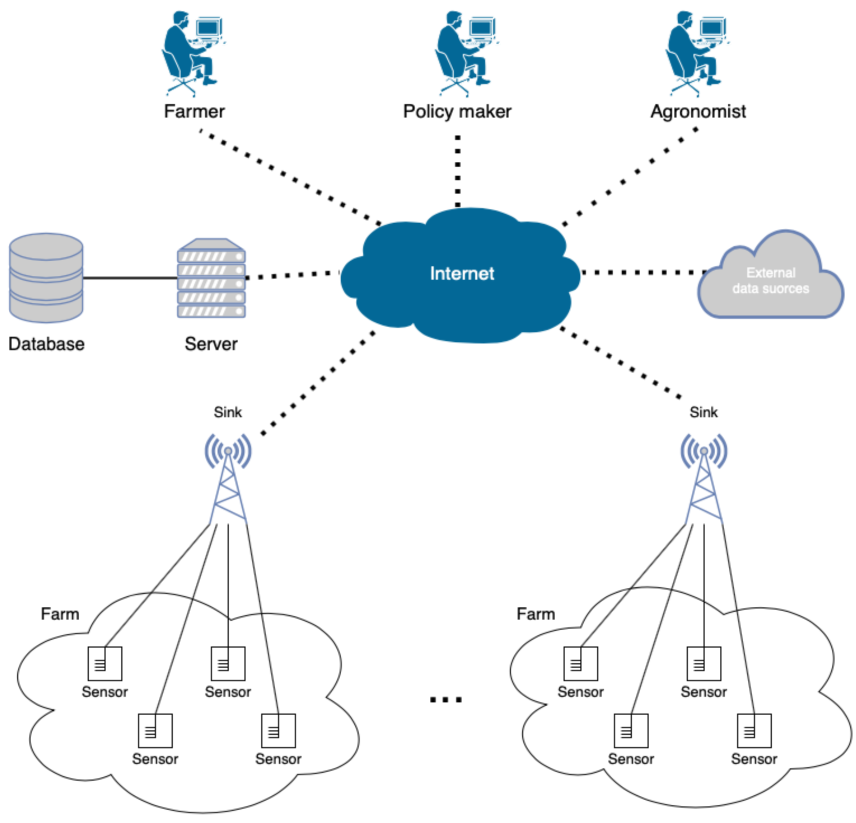
In **Section 5** a discussion about the implementation, integration and test processes is presented.

In **Section 6** the time spent by each group component to realize this document can be found.

In **Section 7** there is a list of the reference documents used.

2. Architectural Design

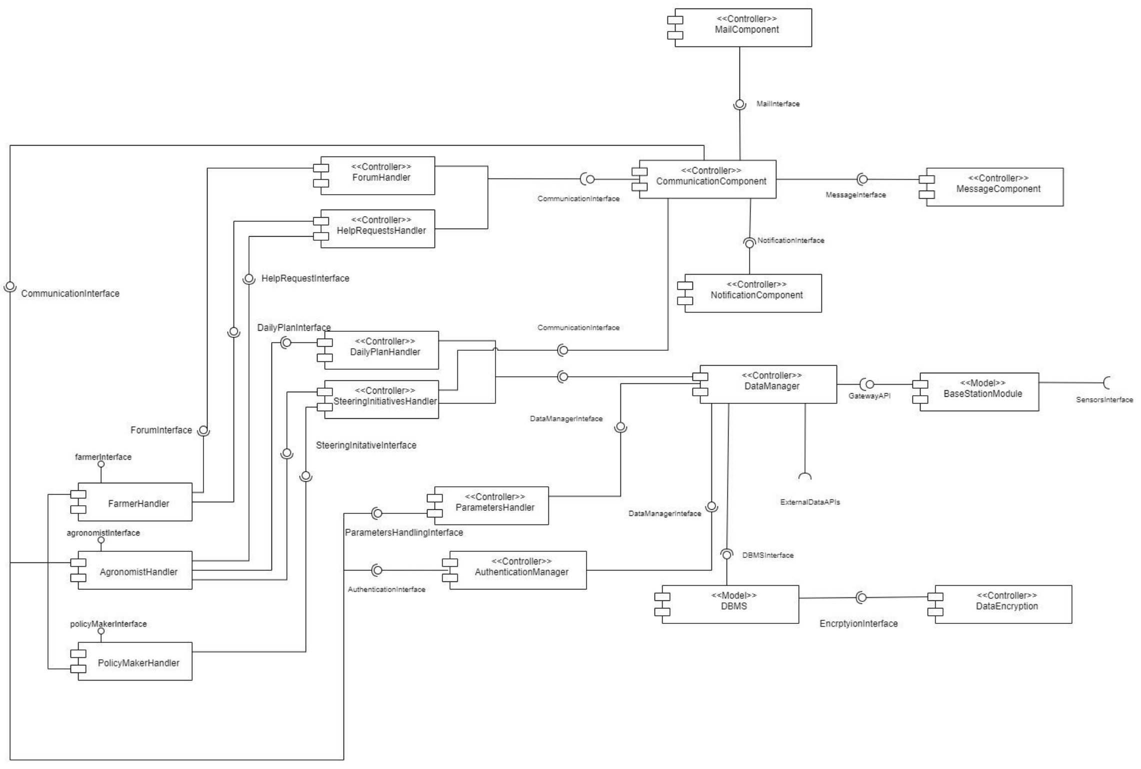
2.1. Overview



This figure represents the general idea behind DREAM architecture. The general schema is a three-tier client-server architecture. Each farm is equipped with various kinds of sensors to collect all relevant data. A particular device, the sink, is responsible for the coordination among all the sensors in a farm and for sending the data to the DREAM server through the Internet. Other relevant data, such as weather forecasts, come from external data sources and datasets which communicate through the Internet with the server. All the users can have access to the application by PCs or mobile devices equipped with an Internet connection. Due to the wide variety of devices which can be used by users, the business logic is implemented entirely in the server, with thin clients which are only responsible for managing the front-end part of the application.

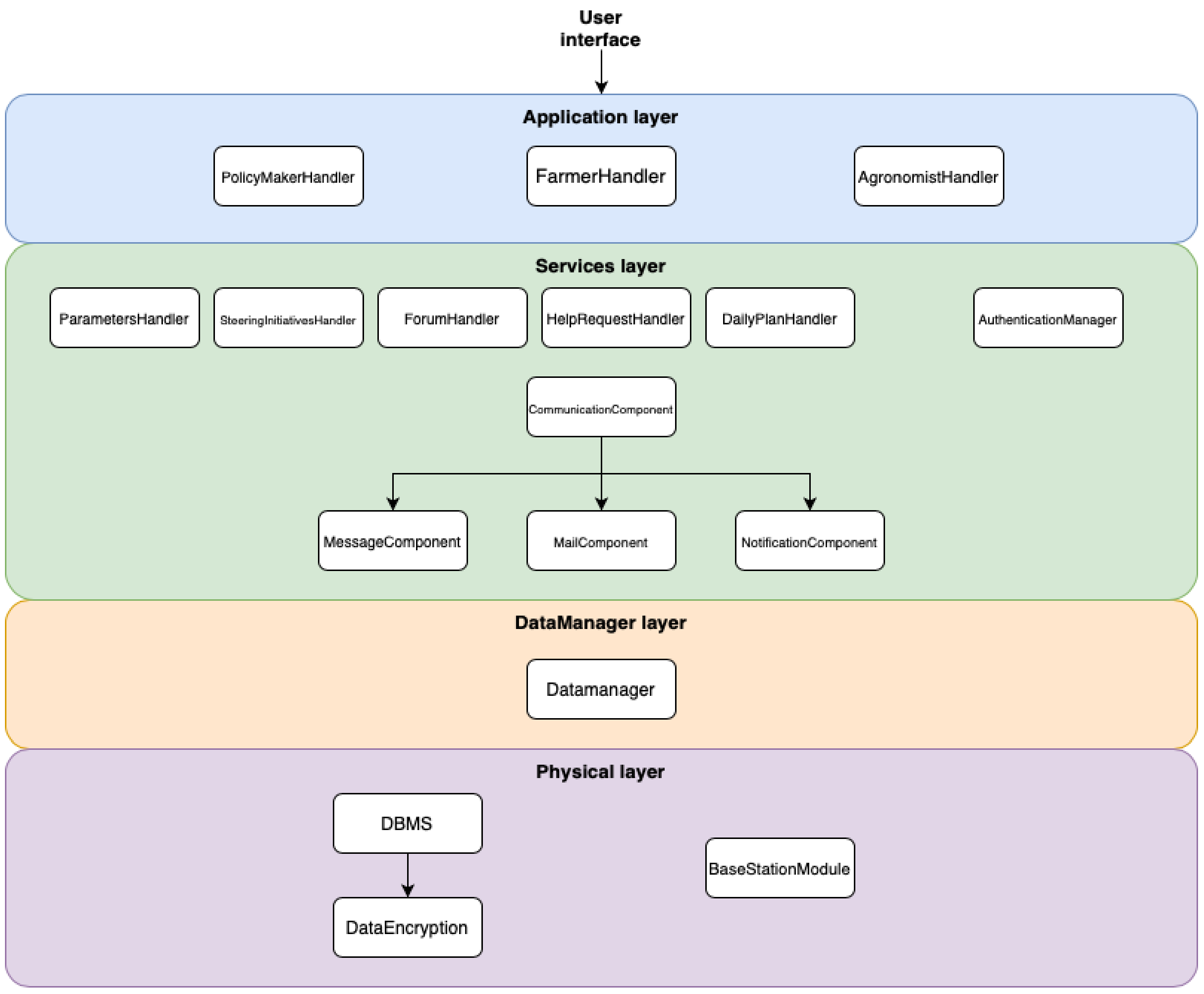
2.2. Component view

2.2.1. Component diagram



<<Controller>> MailComponent
MailInterface
CommunicationInterface
<<Controller>> ForumHandler
<<Controller>> HelpRequestsHandler
CommunicationInterface
<<Controller>> CommunicationComponent
MessageInterface
<<Controller>> MessageComponent
NotificationInterface
<<Controller>> NotificationComponent
HelpRequestInterface
DailyPlanInterface
<<Controller>> DailyPlanHandler
CommunicationInterface
<<Controller>> SteeringInitiativesHandler
<<Controller>> DataManager
GatewayAPI
<<Model>> BaseStationModule
SensorsInterface
DataManagerInteface
ForumInterface
SteeringInitativeInterface
farmerInterface
FarmerHandler
<<Controller>> ParametersHandler
DataManagerInteface
ExternalDataAPIs
ParametersHandlingInterface
agronomistInterface
AgronomistHandler
<<Controller>> AuthenticationManager
DBMSInterface
AuthenticationInterface
<<Model>> DBMS
<<Controller>> DataEncryption
EncrptyionInterface
policyMakerInterface
PolicyMakerHandler

Starting from the component view, a layered structure can be derived. In the diagram below the layered structure of the application is presented.
For more readability, the relationships among components are not shown in the layered structure. They are only shown in the component diagram.



Application layer components

FarmerHandler: this high level component allows farmers to access all the features provided by the application that are destined to them.

AgronomistHandler: this high level component allows agronomists to access all the features provided by the application, it is a sort of high level controller that can use the various components to satisfy the agronomists' requests.

PolicyMakerHandler: this high level component allows policy makers to access all the features provided by the application. It can be seen as the module that collects all the services that the application can offer to each policy maker.

Services layer components

ParametersHandler: the parameters handler component grants access to the DataManager component to all the users of the application; moreover, it manages access to different functions of the DataManager based on the user type. Specifically:

> -**farmers** are provided ways to access parameters such as water consumption, weather forecast, soil humidity and other ones available. At the same time they're able to upload production values via this handler, allowing the storage of such values in the DBMS in the future.
>
> -**agronomists** and **policy makers** are provided ways to check some of the parameters available to the farmers and to access their performances. This module implements a method to determine if a specific farmer is well-performing or under-performing, based on the associated parameters. Furthermore, it allows the policy maker to mark a farmer as a model or as "to-be-helped".

HelpRequestHandler: this controller allows **agronomists** and **farmers** to help each other through help requests. Specifically, it allows farmers to issue help requests on the platform and for other farmers and agronomists to visualize help requests and reply to them.

ForumHandler: the forum handler is the component of the system that manages the forum in its organization, providing the possibility to create topics and reply to existing ones. It also allows moderation of users and the assignment of the "moderator" user status.

DailyPlanHandler: this component is used to handle the daily plan feature used by agronomists. It allows an agronomist to create a new daily plan and it stores it inside the DB calling the needed interfaces.
Moreover it allows the user to check already created plans and handle them, changing the schedule for the following days or confirming the execution of a plan.

SteeringInitiativeHandler: this component is related to the creation and maintenance of steering initiatives. It is used by Policy makers and agronomists. It allows the creation of steering initiatives, selecting the users to which it is directed.
It also allows the maintenance of the created initiatives with the opportunity to check the trend of the involved users.

AuthenticationManager: the authentication manager allows farmers and agronomists to sign up. Policy makers have their own institutional credentials. This module allows all the users to log in and log out of the application, granting different permissions based on the type of users.

CommunicationComponent: this component allows communication between different actors and its behaviour varies depending on the component that requires its usage. More of its specifical functionality will be discussed in the "component interfaces" section. It is also responsible for storing such messages in a database.

MessageComponent: the message component grants the possibility to send and receive messages to all users, based on the kind of feature currently in use.

NotificationComponent: the notification component generates and shows notification to certain users based on the feature currently in use.

MailComponent: this component is responsible for all the emails sent in the processes illustrated in the *RASD* document, which take place after the user is registered to the system.

DataManager layer component

DataManager: This component communicates with the DBMS and other data sources and handles the storing, retrieving and updating data.

Physical layer components

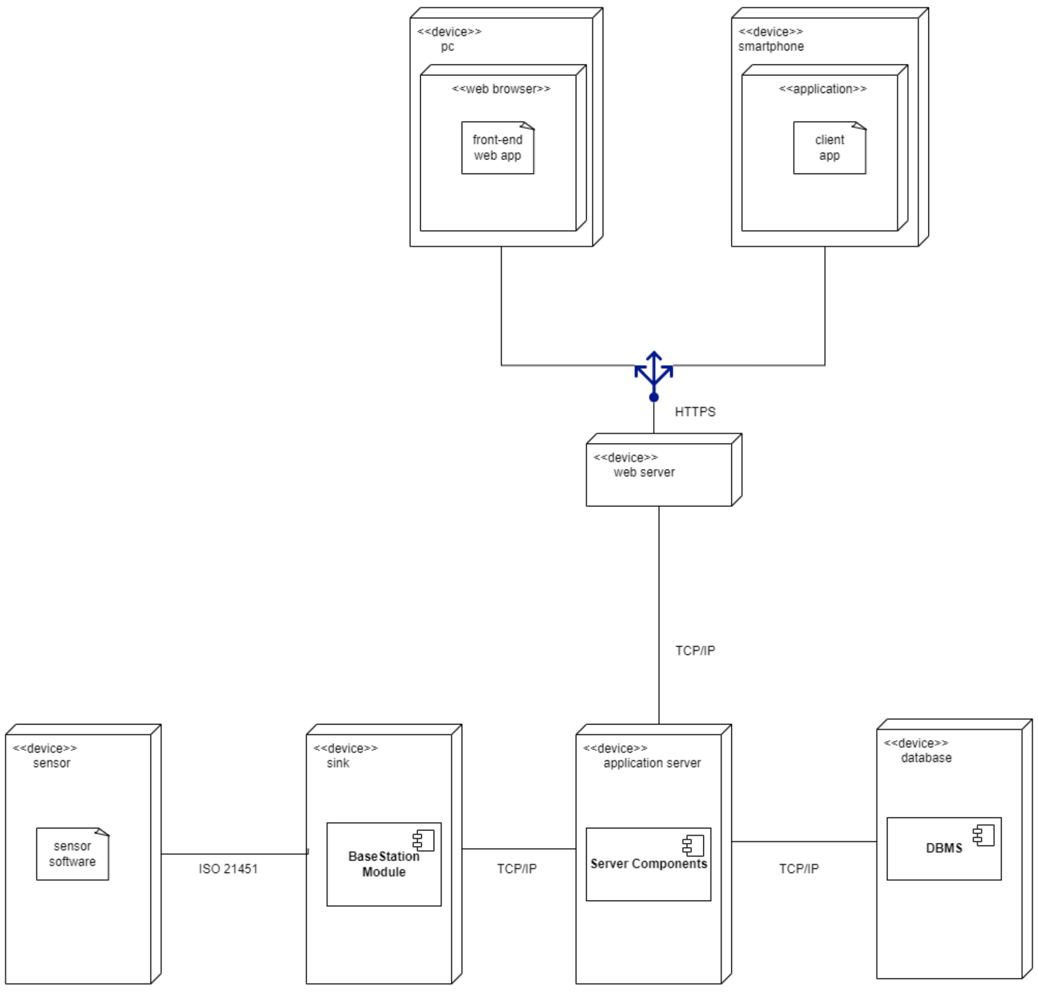
BaseStationModule: the base station module component is the part of the system that collects data from sensors dispatched in the farm it is monitoring. This module can allow changes in modality and in data collection frequency to fit best the needs of the governance.

DBMS: It is the module responsible for the physical storing of the data.

DataEncryption: this component is used by the DBMS in order to efficiently encrypt data and guarantee access to sensible information only to those with the correct permissions.

2.3. Deployment view

The following diagram shows how the system will be implemented through hardware devices.



This diagram shows the three tier architecture implementation.
The client is identified through any user device, such as a smartphone or a PC. For the latter implementation, a web browser is required in order to access the application content, while for the former a native application will be made available.

The **Web Server** is required to handle HTTPS requests and act as a medium between the client and the **application server**.

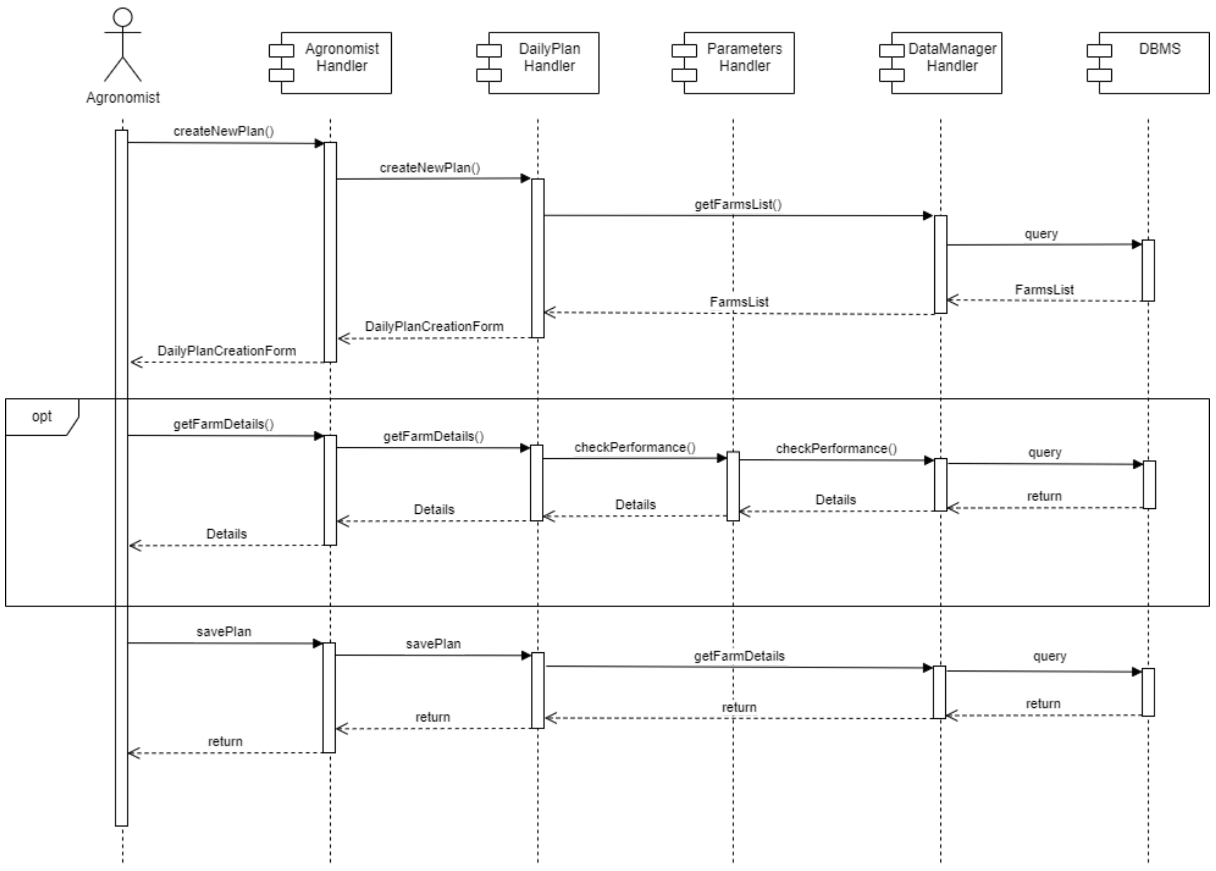
The system is intended as a thin client architecture. For this reason, the **application server** contains all the server logic components described in the component diagram in section 2.2. The application server and the farms' **sinks** communicate through the internet, using the standard TCP/IP protocols.

The communication between sinks and **sensors** is also wireless and is regulated by a wireless sensor communication protocol.

2.4. Runtime view

In this section sequence diagrams of some of the use cases presented in the *Requirements analysis and specification document* (RASD) are presented, showing the interactions between different components involved in such presented use cases.

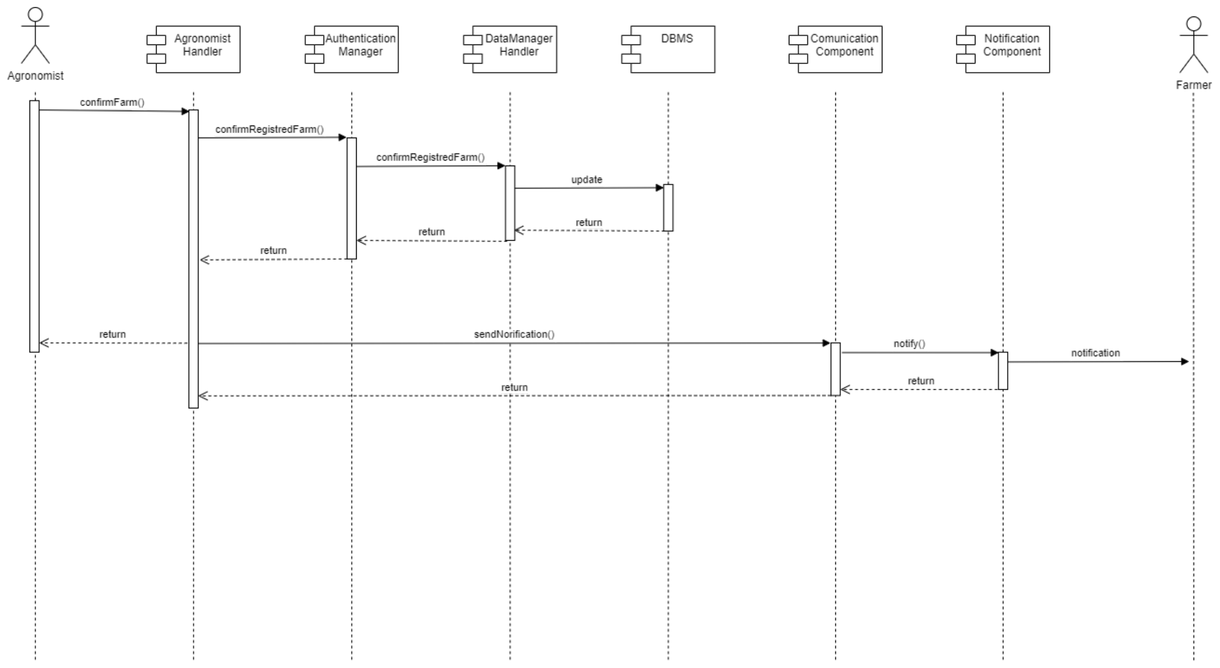
Add a new daily plan



In this diagram it’s described how an Agronomist can create a new daily plan for the following days.
The first step is performed by agronomists when they decide to create a new plan clicking on the related button.
The application will communicate with the server and the component AgronomistHandler will call a method of the specific component DailyPlanHandler.
This component, which is in charge of handling the operations related to the daily plan, requires a list of all the farms to display their names, through a specific method.
The DataManagerHandler, responsible for the data handling, queries the DBMS and retrieves the data.
Optionally the agronomist can ask for details related to one or more farms, the followed path through the components is the same, but another component is involved, the parameterComponent, which gets the data and produces as response a page with the details.
Finally, once inserted all the data, the agronomist can save the daily plan.

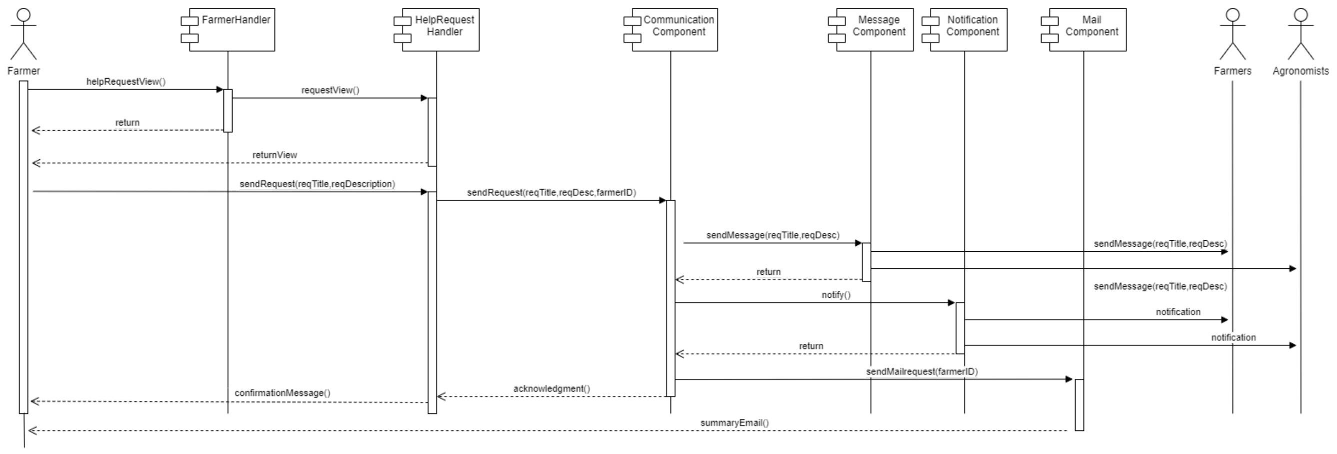
In this case the DailyPlanHandler requires the DataManagerHandler to save the data inside the database.

Confirm Farmer Registration



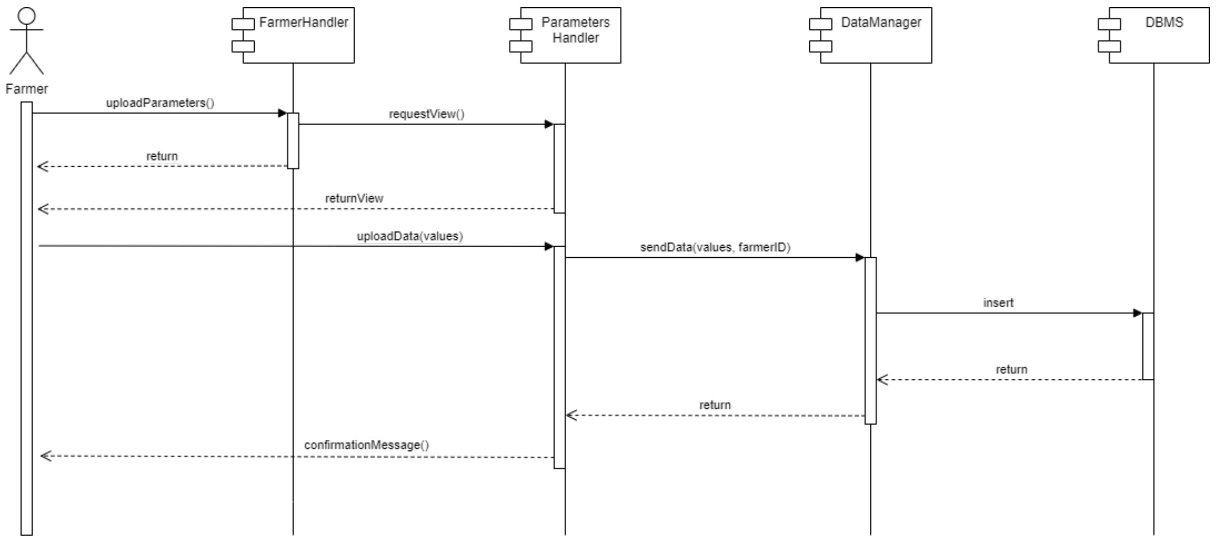
In this diagram it is described how the agronomist can confirm the registration of a farmer. After the farmer signs up, an inspection on site by the agronomist is required to check that everything is correct and to install all the sensors.
After the inspection, the agronomist can confirm the registration of the farmer.
The authentication component, with a specific method, allows to confirm the farmer, and the update on the DBMS is performed by the DataManager, which is called, indeed, by the AuthenticationComponent.
After these operations a notification to the farmer is sent. To do this the agronomistHandler component calls a method of the CommunicationComponent, which is in charge, in turn, of calling the NotificationComponent, that is able to send a notification to the related farmer. Finally, there are exceptional situations that are not represented in the diagram, for example the case in which the farm is not confirmed. In this case a different method of the AuthenticationComponent will be called and consequently the database will delete the record associated with the farm which has been refused.

Asking for help



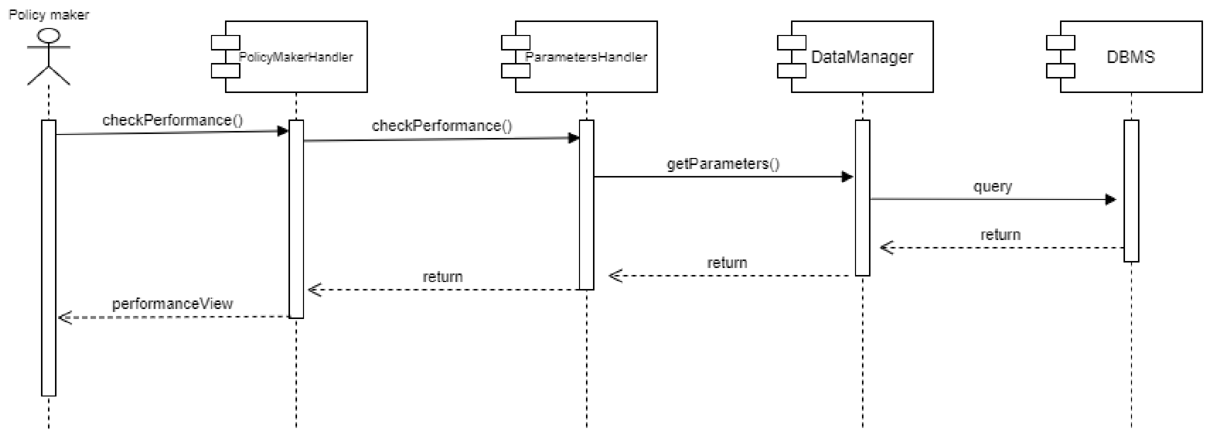
A farmer willing to issue a help request can do so through the high level *FarmerHandler* component which lets the farmer, through a dashboard interface, obtain the help request view of the application provided by the *HelpRequestHandler* component.
Through such view a farmer can type the information he/she wants and confirm the execution. After confirmation the *HelpRequestHandler* component interacts with the *CommunicationComponent* which stores the request for future readings and sends the correspondent message through the *MessageComponent.* A notification to all receiving users is issued through the *NotificationComponent* in order to send communication of the newly issued request to other farmers and to the agronomist appointed to the area of the issuer. The *CommunicationComponent* is also responsible to instruct the *MailComponent* to send a summary email to the help request issuer after a successful operation.

Uploading parameters



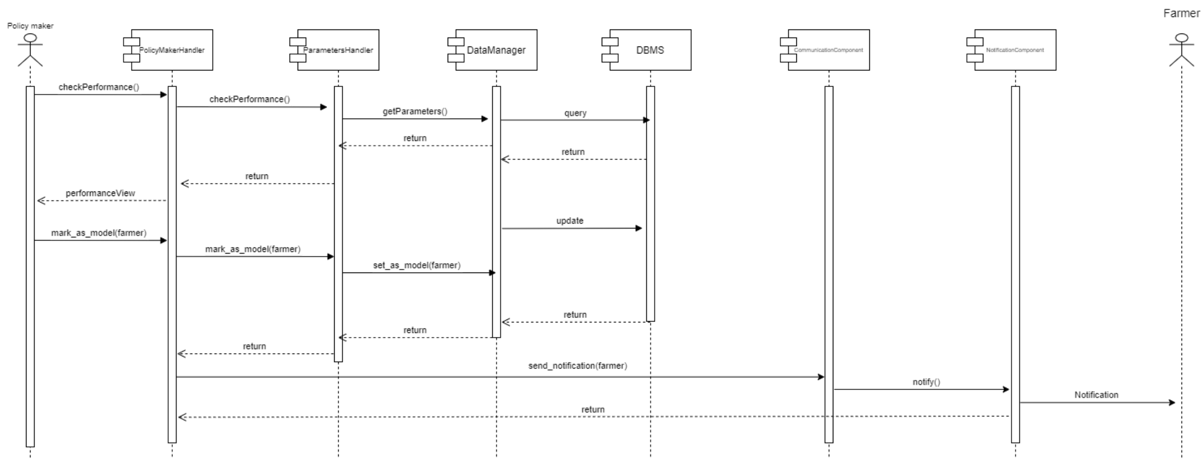
After a day's work, a farmer willing to upload his production values to the platform can do so by accessing the proper view through the dashboard provided by the *FarmerHandler* component accessing the *ParameterHandler*. The farmer can then type the daily measured values of interest alongside some notes and send it to the system. Once the “Submit” button is clicked, the *ParametersHandler* will send the data to the *DataManager* which will recognize the request type and query the *DBMS* inserting the newly generated tuple.

Check farmers’ performances



A policy maker, through his/her device, chooses the option to check farmers’ performances. The PolicyMakerHandler component is devoted to serving this request. It uses the services offered by the ParametersHandler component. It, in turn, asks the DataManager component to get the requested data. The DataManager component can communicate directly with the DBMS by a query to get the data. It returns the obtained data to the ParametersHandler and then the PolicyMakerHandler is ready to present the results to the policy maker. The view is the list of all farmers, highlighting the well-performing farmers and the under-performing ones.

Mark a farmer as a model



To mark a farmer as a model, the policy maker checks farmers’ performances, as described before. Then he/she selects the option to mark as a model a particular farmer. This service is offered by the PolicyMakerHandler component, which, in turn, uses the ParametersHandler to update the state of the farmer. It is the DataManager that can communicate directly with the DBMS to update the farmer’s state. When the update process is completed, the

PolicyMakerHandler uses the CommunicationComponent to send a notification to the farmer. The CommunicationComponent, in turn, uses the services offered by the NotificationComponent which is devoted to sending the notification to the farmer marked as a model.

2.5. Component interfaces

In this section the interfaces of main components are illustrated. What is presented here is a high level overview that explains the main functionalities of principal components, most of which are already presented in the 2.4 section (*Runtime view*). They can be further expanded as needed during the development of the application.

DailyPlanHandlerInterface:
- createNewPlan()
- savePlan(List<Farms> farms)
- updateDailyPlan(List<objects> values)
- getDailyPlan(String date)

SteeringInitiativeHandlerInterface:
- createNewInitiative()
- saveInitiative(List<Farms> farms)
- updateInitiative(List<objects> values)
- getInitiative(String name)

ParametersHandlingInterface:
- getParameters(String dataName, DataType dataType)
- storeDataInDB(int value, DataType dataType)
- getSensorsData(Sensor sensor)
- checkPerformance(String farmerID)
- mark_as_model(String farmerID)

AuthenticationInterface:
- userLogin(string username, string password)
- userLogout()
- userSignUp(string name, string surname, string password, string email)
- confirmFarmerRegistration(Farmer farmer)
- confirmRegisteredFarm()

CommunicationComponentInterface:
- sendNotification(User user, string text)
- sendMessage(User user, string text)
- popUpNotification(User user, string text)
- replyMessage(Message message)

NotificationComponentInterface:
- notify()

MessageComponentInterface:
- sendMessage(String reqTitle, String reqDesc)

MailComponentInterface:
- sendMailRequest(User user)

ForumInterface:
- createNewTopic(Topic topic)
- reply(String text)
- getTopicList()
- suspendUser(Farmer farmer, int time)
- remove(Topic topic) / remove(Reply reply)

HelpRequestInterface:
- sendRequest(String reqTitle, String reqDesc)

DataManagerInterface:
- confirmRegisteredFarm()
- getFarmsList()
- checkPerformance(Farmer user)
- getFarmDetails()
- getParameters()
- sendData(float value, String description, String farmerID)
- set_as_model(String farmerID)

GatewayAPI:
- retrieveData(Sensor sensor)
- refreshForNewData()

DBMSInterface:
- query()
- insert()
- update()

EncryptionInterface:
- encrypt()

The following interfaces are external
ExternalAPIs:
- getWildfireData()
- getSoilTextureData()
- getWeatherForecast()

This interface allows access to datasets made available by the Telangana government regarding some of the parameters accessed by DREAM users. As mentioned in the RASD the datasets available are wildfire, soil texture and weather forecast data but they can be further expanded in the future.
SensorInterface:
- acquisition()

The **FarmerInterface**, **AgronomistInterface** and **PolicyMakerInterface** interfaces allow the users to access the features of the application and provide methods to interact with the lower level components.

2.6. Selected architectural styles and patterns

- **Three tier architecture:** This type of architecture, from a general point of view, divides the application in three levels, each one with a significantly different role.
  The presentation tier is dedicated to the presentation of the data to the users; the business tier implements the logic to make the application works, and it’s a bridge between the user inputs and the operations on the database.
  Finally the data tier handles the data on which the application works.

- **Layered structure:** In this particular architecture each level can be seen independent from the others and it relies on the services offered by lower levels. The lowest levels interact directly with hardware and DBMS. The top most levels, on the other hand, are more abstract and contribute to offer services to the end-user. This architecture allows the software to be highly maintainable due to independence of each level which communicates with the others through external interfaces. A change in the internal structure of a component does not influence the behaviour of other components.

- **Wireless Sensor Network Architecture:** Since the application requires the use of sensors it is important to handle all the data retrieved from them in the most efficient way.
  The proposed architecture, already used in many agricultural implants, allows the positioning of different sensors in the farms, to observe all the parameters needed. Beyond the sensors an important role is played by the Sink, or base station. This component can be considered as the root of the network of the sensors. Its main function is to communicate with all the sensors placed in the farm and get data from them.
  Once the data is in the Sink, which is connected to the net, they can be sent over the internet and so they become accessible from the authorized users of the DREAM application.

  The architecture is composed of different levels, each one of them with specific characteristics regarding for example the data transport, error detection, and so on.

  Finally a particular attention is given to the energy consumption, in fact some of those sensors are alimented by battery and for this reason it would be important to reduce as much as possible the waste of energy, which is required in particular during the communication phase with the sink.

- **MVC:** The mobile phone application and the web based application are realized using the Model View Controller pattern.
  The View element is responsible for the graphic interface and the update of it consequently to user actions.
  The controller element acts as intermediary between the actions of the user and the consequent changes on the model.
  The model handles the data that will be then proposed to the user.

2.7. Other design decisions

2.7.1. Farmer performance analysis

The DREAM application is a data intensive application, one of the main objectives of the application is the analysis of datas coming from different sources to produce a score, this score will be used to highlight well performing and bad performing users.
Then the Policy maker, manually, will have to mark models or farmers to be helped, but this work will be facilitated, as stated before, by the preprocessing of data conducted by the application.

For this reason an approach based on Machine Learning could be a good solution.
The algorithm that determines the score of farmers will base the evaluation on all the data related to their, applying on them data mining techniques in order to understand not only the single value but also the differences with respect to the past, taking into account for example the periodicity of data, the trend and the usual statistical fluctuation that makes data always different.

Moreover the algorithm should also integrate models to predict the trend in the future, this is important to give an evaluation which is not only based on the old data but also on the future perspective, that may change across the days.

All the outputs produced by the Machine Learning models will be compared with some target values set to separate well or bad performers. After this comparison a score will be produced and will be assigned automatically to each farmer.

If the score is particularly good or worse this will result in a highlight when the Policy Makers the Agronomists will visit the list of farmers.

2.7.2. Sensor data collection and communication methods

Sensor data is collected, as stated earlier in this document, through systems called *sinks* which act as “interfaces” between the DREAM system and the sensors. The communication between the sink and the sensors is wireless and thus the sensors need to be able to communicate in such a way.
In order to create a connection that is easily scalable (for additional measurements in further application developments) and maintainable (since sensors are subject to deterioration caused by water and air exposition), the sensors need to be compliant to the same communication protocol. In this case a protocol such as the ISO 21451, already mentioned in the *RASD* would suffice.
Data collection is issued by the sinks periodically, with an adjustable measurement frequency based on the type of intended data manipulation. The data is stored in the sink for a 24 hours period of time and then they are sent, through the internet, to the web server of the web application and shortly after they are deleted, in order to save memory in case numerous acquisitions take place during the same day. The acquisitions, though, need to

take place with a high frequency; since farmers will be able to check acquired measurements at any given time, the data provided to the issuer cannot be obsolete.

2.7.3. Design decisions related to nonfunctional requirements

In order to guarantee a sufficient performance and availability, as mentioned in the *RASD,* the workload will be shared among multiple server machines. Such machines need to be located in different geographical positions in order to prevent system faults caused by environmental disasters, which are frequent in the Telangana region.

The same concept holds with respect to the data persistence, which can be guaranteed through a RAID 6 solution, with database copies located far from each other.

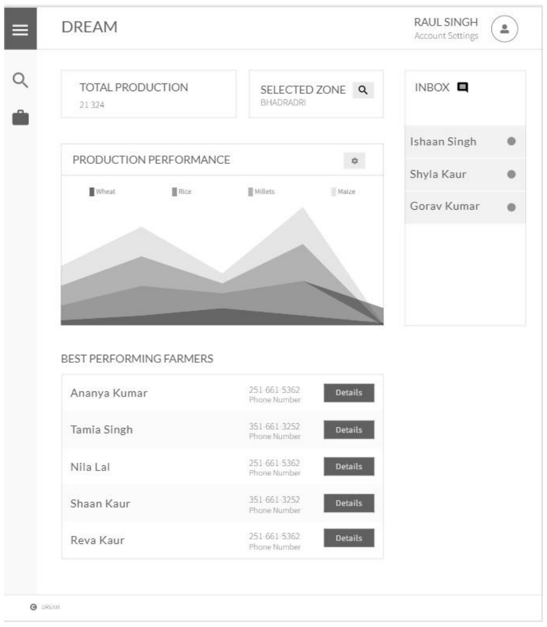
Sensible data shall also be protected through SHA-3 data encryption algorithm and transmissions shall be protected through the SSL protocol.

3. User interface design

Here some graphical interfaces are presented.
Since the application should be usable by a wide range of people with different capabilities and technological skills, the application is accessible both from PCs and smartphones.

Here we present the graphic interfaces for PCs regarding the Policy Makers and Agronomists, in fact they more likely will use a desktop PC.
For farmers we present instead the graphic interface of the mobile app, that should be used by the majority of farmers, being simpler to use and more immediate to access.

First we present the interfaces accessible by PCs and related to Agronomists and Policy makers, then the interfaces relative to the part fo Framer.


DREAM

RAUL SINGH
Account Settings

TOTAL PRODUCTION
21 324

SELECTED ZONE
BHADRADRI

INBOX
Ishaan Singh
Shyla Kaur
Gorav Kumar

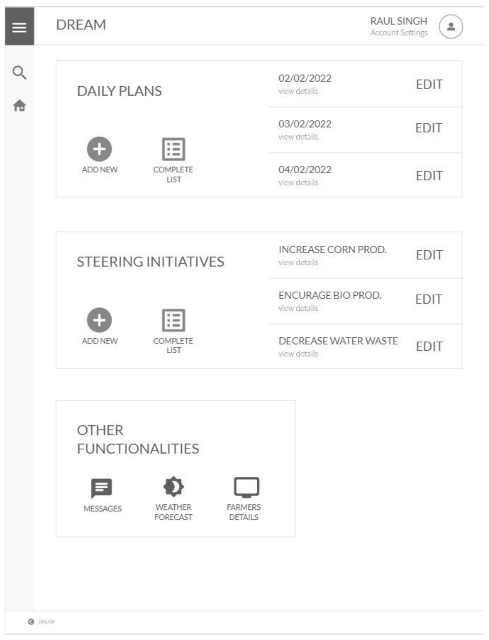
PRODUCTION PERFORMANCE

BEST PERFORMING FARMERS

Ananya Kumar	251-661-5362 Phone Number	Details
Tamia Singh	351-661-3252 Phone Number	Details
Nila Lal	251-661-5362 Phone Number	Details
Shaan Kaur	351-661-3252 Phone Number	Details
Reva Kaur	251-661-5362 Phone Number	Details

DREAM

This is the main page, after the login, seen by Agronomists and Policy makers, it shows an overview of all the features and data.


DREAM

RAUL SINGH
Account Settings

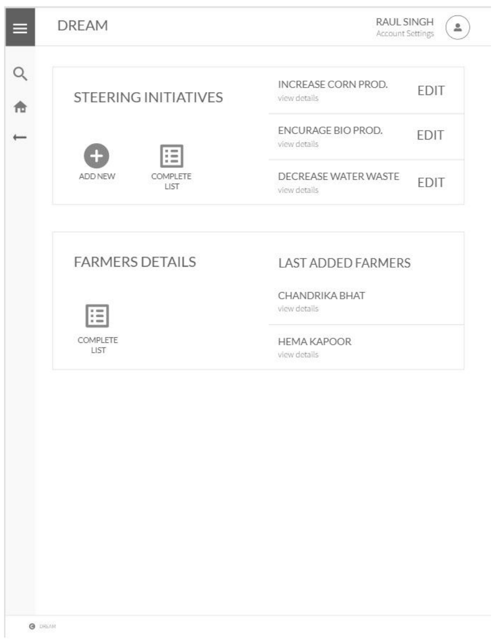
DAILY PLANS

02/02/2022 view details	EDIT
03/02/2022 view details	EDIT
04/02/2022 view details	EDIT

STEERING INITIATIVES

INCREASE CORN PROD. view details	EDIT
ENCURAGE BIO PROD. view details	EDIT
DECREASE WATER WASTE view details	EDIT

OTHER FUNCTIONALITIES


DREAM

RAUL SINGH
Account Settings

STEERING INITIATIVES

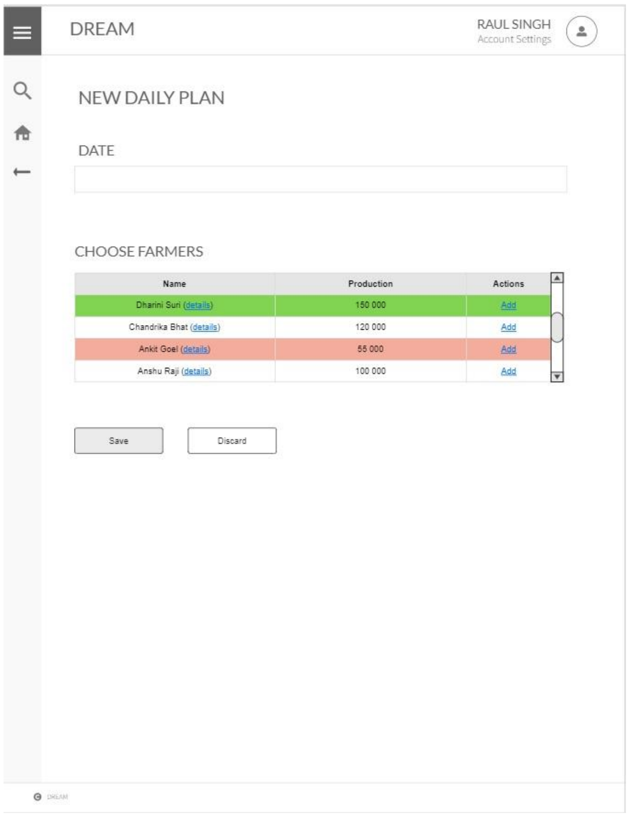
INCREASE CORN PROD. view details	EDIT
ENCURAGE BIO PROD. view details	EDIT
DECREASE WATER WASTE view details	EDIT

FARMERS DETAILS

LAST ADDED FARMERS
CHANDRIKA BHAT
view details
HEMA KAPOOR
view details

The two interfaces above are accessible by the main page, and allows agronomists (left) and policy makers (right) to access the main functionalities of their competence.

Daily plan



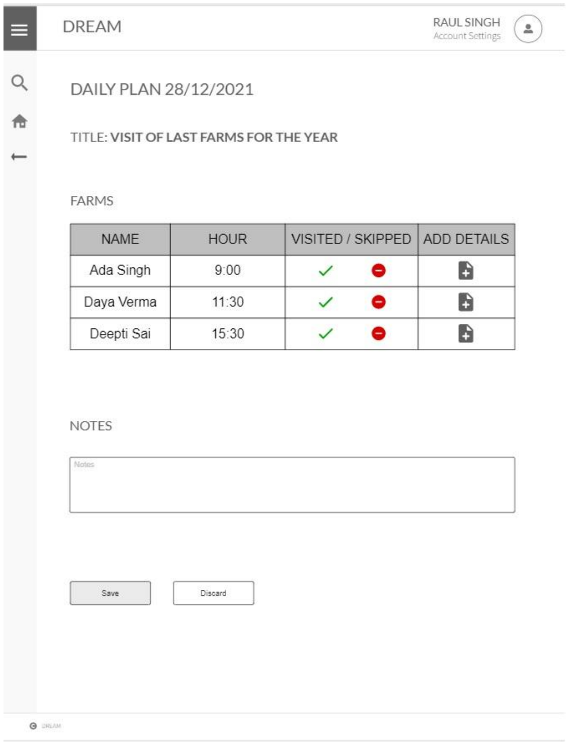
NEW DAILY PLAN

DATE

CHOOSE FARMERS

Name	Production	Actions
Dharini Suri (details)	150 000	Add
Chandrika Bhat (details)	120 000	Add
Ankit Goel (details)	55 000	Add
Anshu Raji (details)	100 000	Add

With the interface proposed on the left the agronomist can create a new daily plan, selecting the farmers to visit, also on the basis of the data proposed by the application.



DAILY PLAN 28/12/2021

TITLE: VISIT OF LAST FARMS FOR THE YEAR

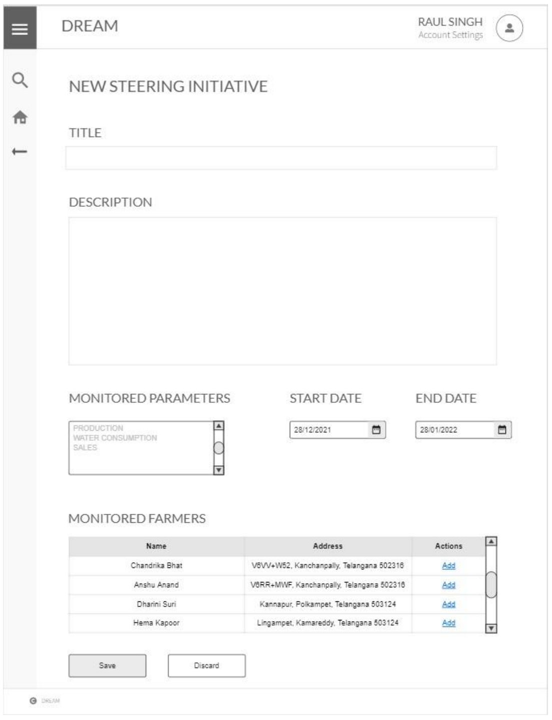
FARMS

NAME	HOUR	VISITED / SKIPPED	ADD DETAILS
Ada Singh	9:00	✓ ⊖	⊞
Daya Verma	11:30	✓ ⊖	⊞
Deepti Sai	15:30	✓ ⊖	⊞

NOTES

With this interface the agronomist can confirm the execution of a daily plan, marking the visited farms and adding notes if needed.

Steering initiative



NEW STEERING INITIATIVE

TITLE

DESCRIPTION

MONITORED PARAMETERS: PRODUCTION, WATER CONSUMPTION, SALES

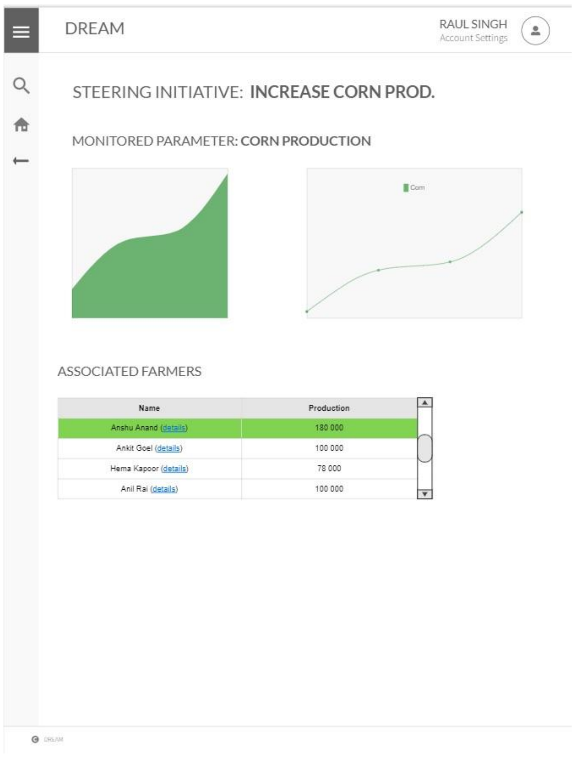
START DATE: 28/12/2021

END DATE: 28/01/2022

MONITORED FARMERS

Name	Address	Actions
Chandrika Bhat	V6VV+W52, Kanchanpally, Telangana 502316	Add
Anshu Anand	V6RR+MWF, Kanchanpally, Telangana 502316	Add
Dharini Suri	Kannapur, Polkampet, Telangana 503124	Add
Hema Kapoor	Lingampet, Kamareddy, Telangana 503124	Add

With this interface a new steering initiative can be created, this interface is accessible both from policy makers and agronomists.



STEERING INITIATIVE: **INCREASE CORN PROD.**

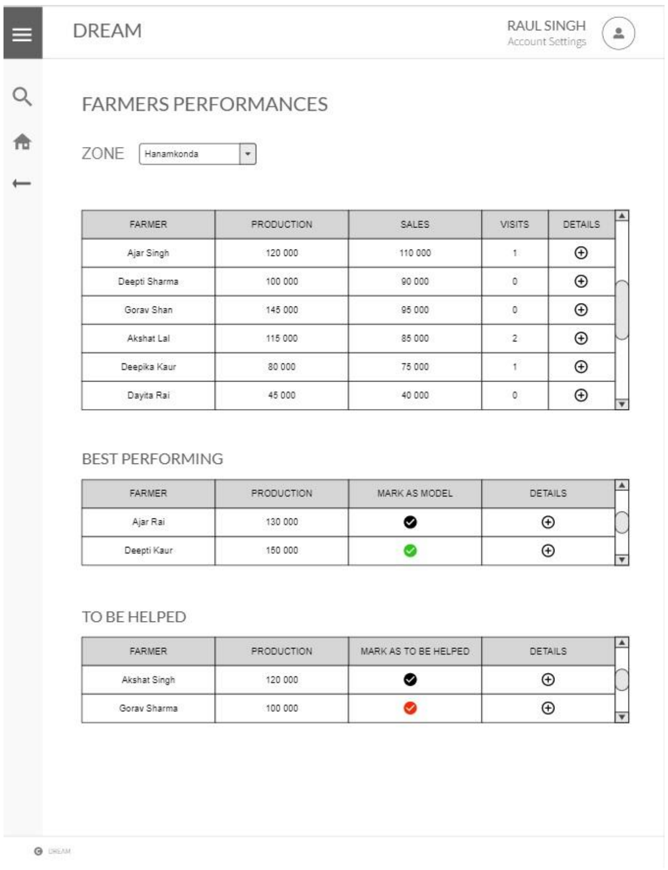
MONITORED PARAMETER: **CORN PRODUCTION**

ASSOCIATED FARMERS

Name	Production
Anshu Anand (details)	180 000
Ankit Goel (details)	100 000
Hema Kapoor (details)	78 000
Anil Rai (details)	100 000

This interface shows the details of an inserted steering initiative, it illustrates the trend of the monitored parameter and the farmers involved.

Farmer performances



FARMERS PERFORMANCES

ZONE: Hanamkonda

FARMER	PRODUCTION	SALES	VISITS	DETAILS
Ajar Singh	120 000	110 000	1	⊕
Deepti Sharma	100 000	90 000	0	⊕
Gorav Shan	145 000	95 000	0	⊕
Akshat Lal	115 000	85 000	2	⊕
Deepika Kaur	80 000	75 000	1	⊕
Dayita Rai	45 000	40 000	0	⊕

BEST PERFORMING

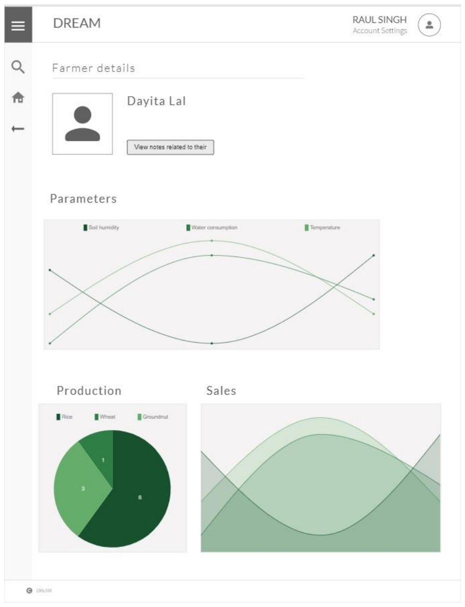
FARMER	PRODUCTION	MARK AS MODEL	DETAILS
Ajar Rai	130 000	✓	⊕
Deepti Kaur	150 000	✅	⊕

TO BE HELPED

FARMER	PRODUCTION	MARK AS TO BE HELPED	DETAILS
Akshat Singh	120 000	✓	⊕
Gorav Sharma	100 000	✓	⊕

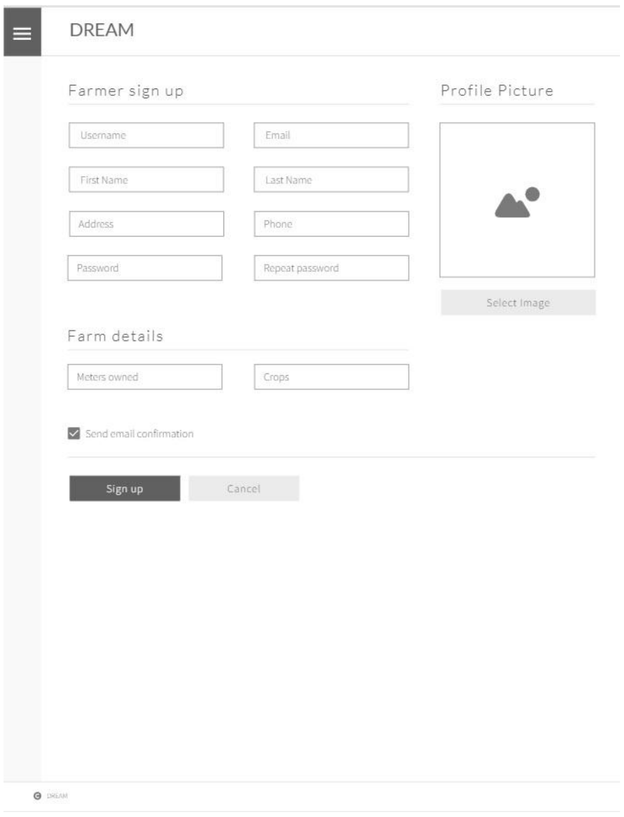
This interface, accessible by Policy makers, allows them to see a list of farmers, and allows them to select farmers as “models” or as “to be helped”, on the basis of their performances.

Farmer details



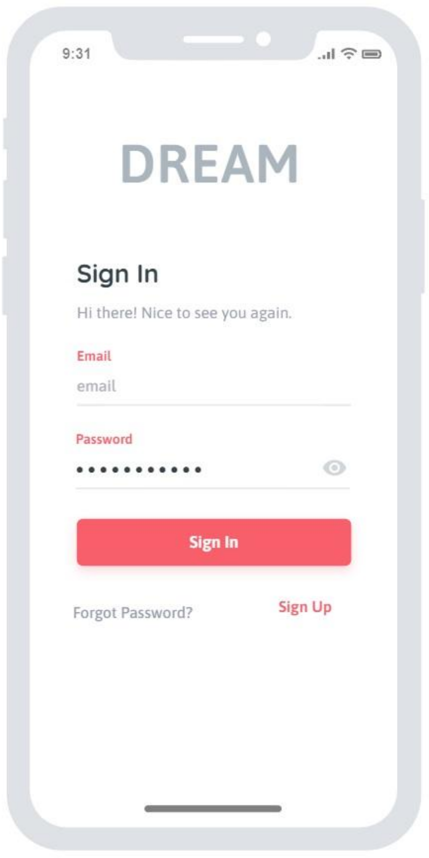
This page shows a detailed presentation of a farmer, this page can be reached when clicking the button “details” near the name of a farmer.

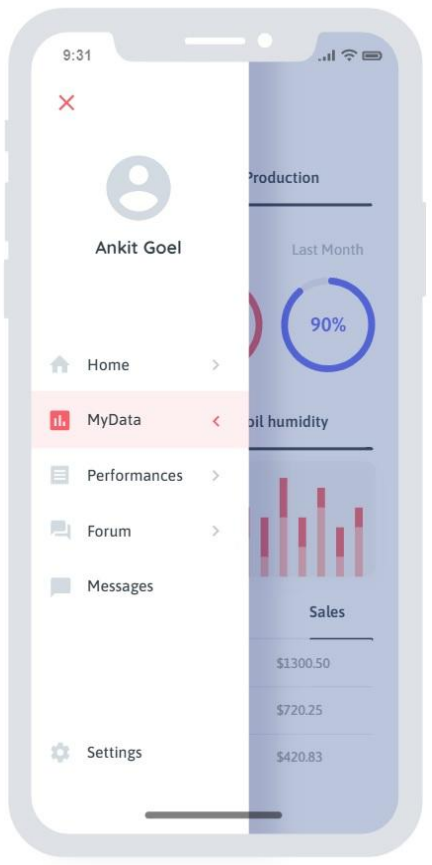
Farmer sign up



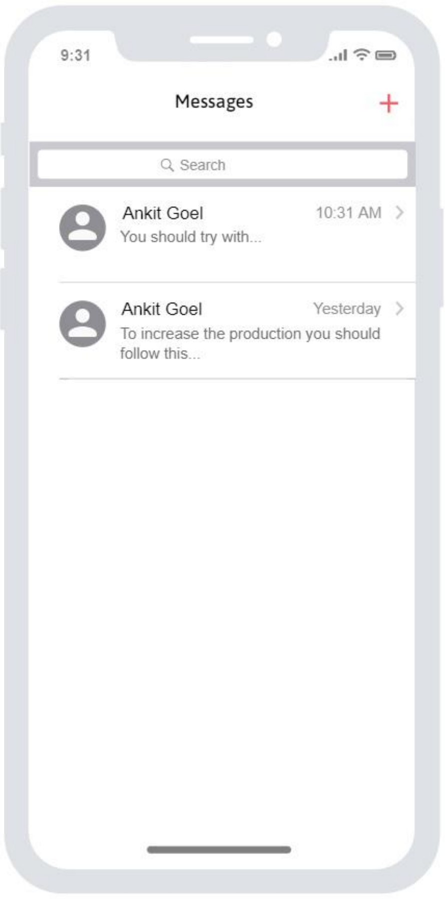
With this page the farmer can register to the application, it has to insert all the data regarding the person and the farm.

Now some interfaces proposed for the Farmers are shown. As said before these interfaces refer to the mobile application version.





The login page and the menu from which the farmer can access all the functionalities of the application.

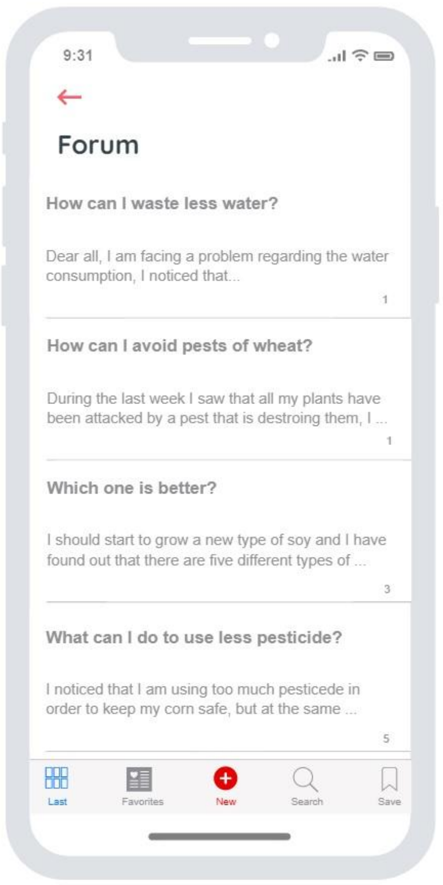

9:31

Messages

Search

Ankit Goel 10:31 AM
You should try with...

Ankit Goel Yesterday
To increase the production you should follow this...


9:31

Forum

How can I waste less water?

Dear all, I am facing a problem regarding the water consumption, I noticed that...

1

How can I avoid pests of wheat?

During the last week I saw that all my plants have been attacked by a pest that is destroing them, I ...

1

Which one is better?

I should start to grow a new type of soy and I have found out that there are five different types of ...

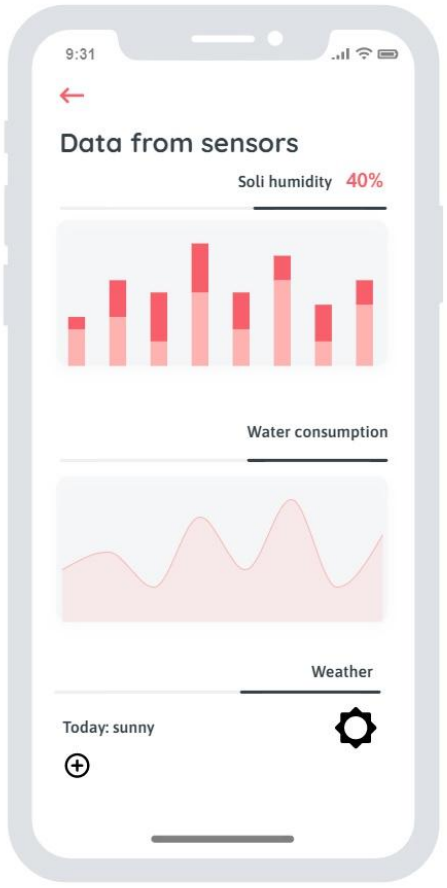
3

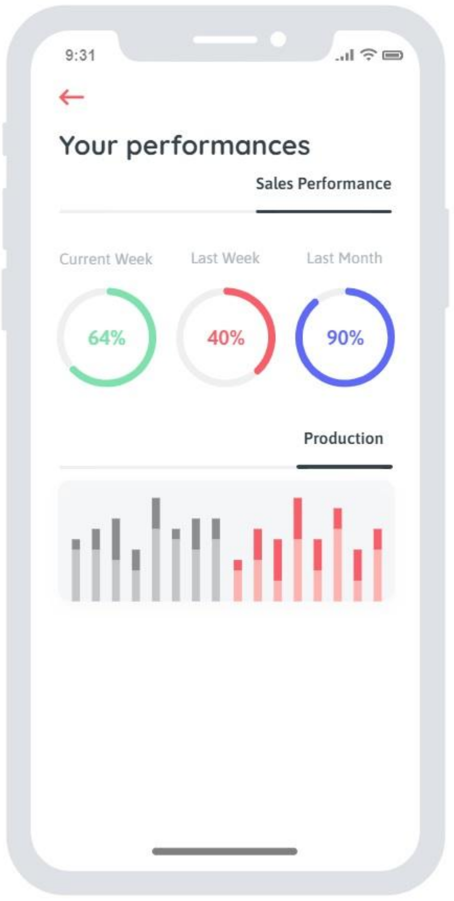
What can I do to use less pesticide?

I noticed that I am using too much pesticede in order to keep my corn safe, but at the same ...

5

The left image shows the interface from which the farmer can read the messages received and answer them.
The right image illustrates the graphic of the forum, from which the user can read the topics and create new ones.





The interface on the left allows the user to check data retrieved from sensors, while the one on the right shows the performances of the farmer during the last days, so that he can figure out his trend.

4. Requirements traceability

The following table shows the requirements and their respective ID taken from the RASD of the DREAM application and divided by actor.

Farmers

ID	**FUNCTIONAL REQUIREMENT**
FR1	The system shall allow farmers to register themselves
FR2	The system shall allow registered farmers to register their farms
FR3	The system shall send a confirmation email after the registration process
FR4	The farmer shall be able to check the weather forecast
FR5	The farmer shall be able to check soil humidity values
FR6	The farmer shall be able to check water consumption values
FR7	The farmer shall be able to send help requests to other farmers
FR8	The farmer shall be able to send help requests to agronomists
FR9	The system shall send a summary email to the issuer after the help request
FR10	The farmers shall be able to reply to issued help requests
FR11	The system shall send a summary email to the farmer who responds to a help request
FR12	The farmers shall be able to check the replies given to every given issued request
FR13	The farmers shall be able to close an help request
FR14	The system shall notify those who replied if the issue is still open after the issuer checks the replies
FR15	The farmer shall be able to upload the daily production information to the system
FR16	The farmer shall be able to automatically upload daily water consumption values if available
FR17	The system shall prevent multiple daily uploads
FR18	The system shall provide a discussion forum to all registered farmers with non-customizable sections
FR19	The farmers shall be able to create new topics in any given section
FR20	The farmers shall be able to reply to any given forum topic

FR21	The system shall automatically appoint senior farmers as forum moderators
FR22	The moderators shall be able to delete topic replies
FR23	The moderators shall be able to delete replies
FR24	The moderators shall be able to suspend users
FR25	The system shall send a notification every time a new reply has been added to a recently topic to which an user has replied
FR26	The farmers shall be able to log into the system
FR27	The farmers shall be able to log out of the system

Policy makers

ID	**FUNCTIONAL REQUIREMENT**
PR1	The system shall allow a policy maker to log into the system with his/her institutional credentials
PR2	The system shall allow a policy maker to logout from the system
PR3	The system shall allow a policy maker to upload a new steering initiative
PR4	The system shall show to a policy maker the list of well-performing farmers
PR5	The system shall show to a policy maker the list of under-performing farmers
PR6	The system shall allow a policy maker to mark a well-performing farmer as a model
PR7	The system shall allow a policy maker to mark an under-performing farmer as “to-be-helped”
PR8	The system shall show to a policy maker the results of the farmers involved in a steering initiative
PR9	When a farmer is marked as a model, the system shall send a notification to that farmer inviting him/her to publish him/her best practices on the forum
PR10	When a farmer is marked as “to-be-helped”, the system shall send a notification to that farmer inviting him/her to ask for help from an agronomist or other farmers
PR11	Each time a farmer uploads new data about his/her production, the system performs the performance analysis and marks the farmer as well-performing (or model if he/she is already marked so) if the results are up to a certain threshold, under-performing otherwise (or to-be-helped if he/she is already marked so)
PR12	If a farmer has not uploaded new data about production for a month, the system shall mark his/her state as not defined regardless of the current state

Agronomists

ID	FUNCTIONAL REQUIREMENT
AR1	The system shall allow an agronomist to log in
AR2	The system shall allow an agronomist to log out
AR3	The system shall send a confirmation email to an agronomist once they select the area of responsibility
AR4	The system shall check the consistency of data inserted by users
AR5	The system shall allow the agronomist to send messages to farmers
AR6	The system shall let the Agronomist to confirm the registration of a farmer, after the Agronomist has visited the related farm
AR7	The system shall show the agronomist all and only the farms he can visit during the daily plan creation process
AR8	The system shall correctly store each update of the daily plan
AR9	The system shall allow Agronomist compare farmers performances
AR10	The system shall let Agronomists to delete farms from their future daily plan
AR11	The system shall let Agronomists to add farms to their future daily plan
AR12	The system shall let Agronomist to indicate deviations from their “pre-compiled” daily plan
AR13	The system shall let Agronomists to confirm the execution of a daily plan
AR14	The system shall keep track of how many times a farm has been visited
AR15	The system shall show always the more updated values retrieved from sensor
AR16	The system shall allow Agronomist to monitor created steering initiatives

The following table shows the correspondence between the components presented in this document and the requirements that it shall satisfy.

Farmers

COMPONENT NAME	REQUIREMENT IDs
BaseStationModule	FR5, FR6
ForumHandler	FR18, FR19, FR20, FR21, FR22, FR23, FR24
FarmerHandler	FR4 - FR27
HelpRequestHandler	FR7, FR8, FR12

ParametersHandler	FR4, FR5, FR6, FR15, FR16, FR17
AuthenticationManager	FR1, FR2, FR3, FR26, FR27
DataManager	FR1, FR2, FR4, FR5, FR6, FR15, FR16
DBMS	FR1, FR2, FR4, FR5, FR6, FR15, FR16
CommunicationComponent	FR7, FR8, FR10, FR12, FR19, FR20
MessageComponent	FR7, FR8, FR9, FR10, FR11, FR19, FR20
NotificationComponent	FR14, FR25
MailComponent	FR9

Policy makers

Component	Requirements
PolicyMakerHandler	PR1, PR2, PR3, PR4, PR5, PR6, PR7, PR8
AuthenticationManager	PR1, PR2
SteeringInitiativesHandler	PR3, PR8
ParametersHandler	PR4, PR5, PR6, PR7, PR8, PR11, PR12
CommunicationComponent	PR9, PR10
NotificationComponent	PR9, PR10
DataManager	PR1, PR3, PR4, PR5, PR6, PR7, PR8, PR11, PR12
DBMS	PR1, PR3, PR4, PR5, PR6, PR7, PR8, PR11, PR12

Agronomists

COMPONENT NAME	REQUIREMENT IDs
BaseStationModule	AR15
AgronomistHandler	AR1, AR2, AR4, AR9, AR15, AR16
HelpRequestHandler	AR5
ParametersHandler	AR9, AR15
AuthenticationManager	AR1, AR2, AR6
DataManager	AR4, AR8
DBMS	AR8

CommunicationComponent	AR3, AR5
MessageComponent	AR5
NotificationComponent	AR5, AR6
MailComponent	AR3
DailyPlanHandler	AR4, AR7, AR10, AR11, AR12, AR13, AR14
SteeringInitiativeHandler	AR4, AR16

5. Implementation, Integration and test plan

5.1. Implementation Plan Introduction

A bottom-up approach is suggested in order to efficiently implement and create the application, starting from the components presenting the main core and logic of the application and proceeding with higher level components.

The first component to be tested is the DBMS, alongside with the DataEncryption component, since it is the core of the application and the element on which all the other components rely on.

After successfully choosing and testing the DBMS the DataManager component can be implemented and validated, since it is the other component responsible for data manipulation.

Alongside the testing of the DataManager component the communication components can also be implemented and tested since it is used by most of the higher level components presented.

Afterwards we can proceed testing all the higher level components that are built to satisfy some specific features of the application (DailyPlanComponent, SteeringInitiativesComponent, ParametersHandler, …).

Finally the handlers that manage all the functionalities accessed by the users are tested and verified to work properly.

The application used by users can be tested in a separate way, since the server exposes just APIs, which can be initially simulated and then implemented properly afterwards.

5.2. System testing

Different types of tests must be performed, in order to check all the possible aspects of the components, and all the possible errors or problems that may damage the users.

Static analysis: This type of analysis is performed manually by the software engineer working on the project, it is a review between peers. Since this technique has been proven to be very efficient and time saving, we introduce it so that the written code can be quickly inspected by other programmers, and possible problems are eliminated at the source.

Unit tests: Unit tests are important to perform a dynamic analysis of the component. Each component will be tested separately. If the components with which it interacts haven't been tested or produced yet, they will be replaced with a simulated counterpart.
The unit test will be a large batch of different tests that must be executed automatically before the submit of the produced code.

Integration tests: The components, after the above mentioned tests, will be integrated.
In order to guarantee that the communication between each component behaves correctly and satisfies all the requirements, it is important to perform integration tests, whose aim is to check that the whole system works properly.

Alpha and beta testing: Since the application is directed to Policy makers, Agronomists and Farmers an important role is played by their opinions.
For this reason, before the final product will be released, it's important to gather the feedback from some users for each category.
Some volunteers will be selected among each category and they will be introduced to a prototype of the application they will have to use in the future.
The users will communicate their feedback writing directly to the team, and proposing improvements during organized meetings opened to these testers.

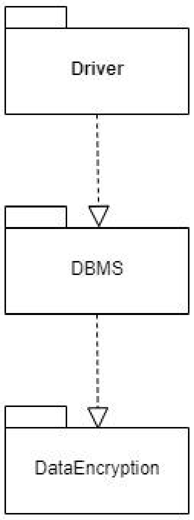
Stress test: The application will be used by a large number of people; the load will probably be considerably high during some specific hours of the day, while it will be possibly neglectable during the night.
For this reason it is important to plan tests in order to cover different situations, with different workloads.

Performance test: The application will be made available to many different users, who live in various economic conditions. For this reason we expect a high heterogeneity regarding the performances of the used devices and the speed of the internet connection.
Some tests are required to verify that, even under bad connectivity/device performance conditions, the application can offer at least the minimal services in an acceptable time, as specified in the *RASD*.

5.3. Integration and testing

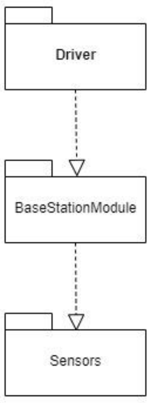
DBMS Integration

The first element to test is the DBMS. The component will interact with the already developed dataEncryption component.
All the other components will be substituted by a Driver that simulates their behaviour.



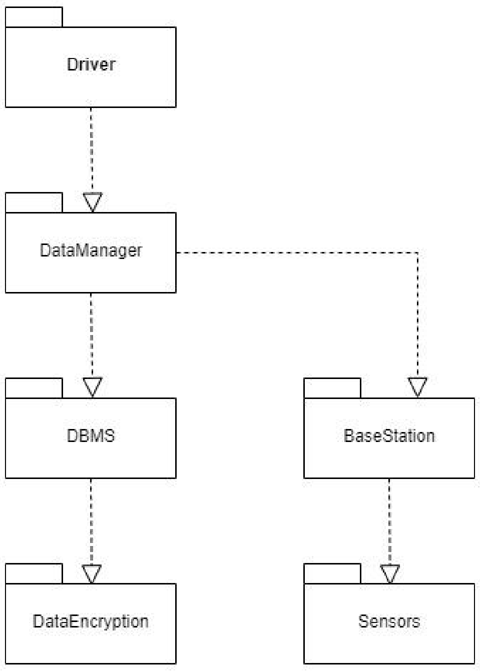
Base station module

A main part of the application is the data handling. Part of this data is retrieved by sensors, located in the various farms.
One important step is to verify that all the sensors used are well integrated with the system.
In particular the communication between the sensors and the sink must work properly and, after this, the sink must be able to send the data to the server.
The integration test for this part requires working sensors and a sink, while the rest will be simulated by the Driver, which simulates an acquisition of data from the sink.



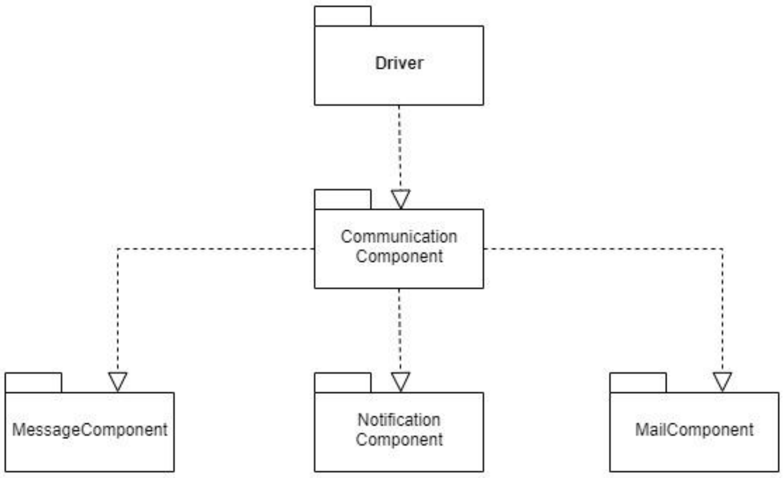
Data manager module

The data manager module is tested and integrated right after the integration of the DBMS and the data encryption module, the Driver will interact with the dataManager simulating higher level components and requesting data modification or acquisition.



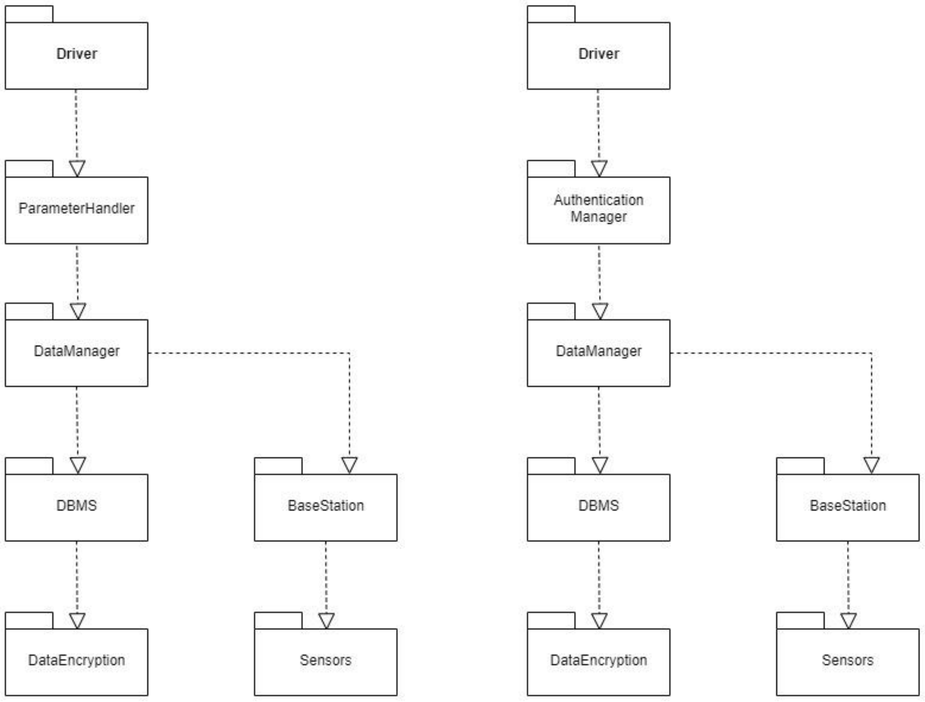
CommunicationComponent

The communication component can be integrated simultaneously to the data handling components, since they do not present any interactions in the system.
The component will use sub components specific to some tasks (email, messages and notifications). Those components must be tested in advance and during this phase we will proceed to their integration with the communication component, that then will be integrated with the rest of the system, simulated by a Driver. Specifically, the driver needs to simulate message exchanges between users, prompting notification messages and performing actions that generate a mail message as a response.



Authentication and Parameters handler

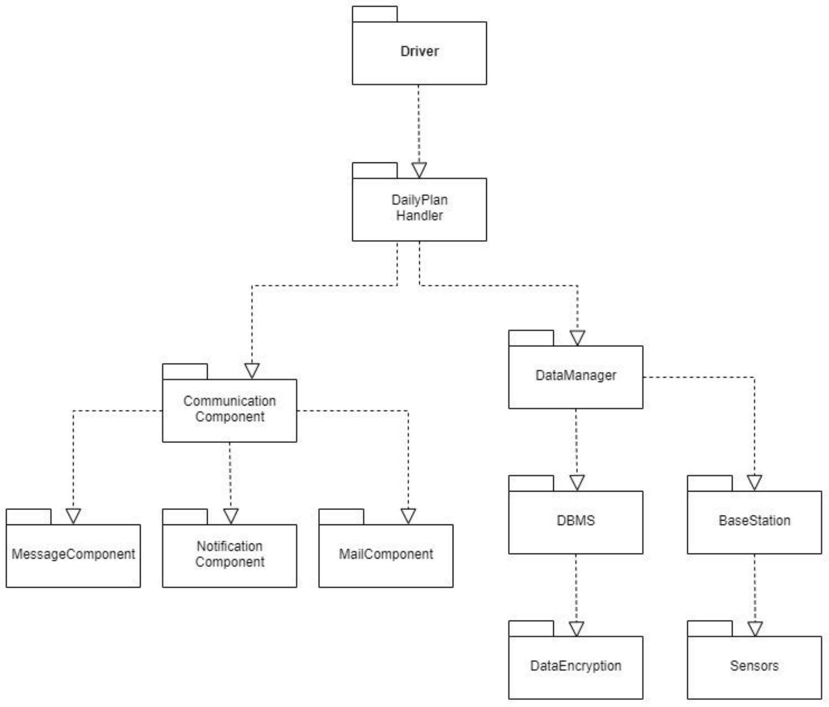
AuthenticationManager and ParametersHandler components have different functionalities, they can be tested and integrated in parallel, since they require access only to the DataManager and they are independent from each other.



Particular attention needs to be given to the logic behind the ParametersHandler component, since it is the part of the system that implements the farmers performance analysing algorithm. In particular, the algorithm needs to be validated beforehand using different test sets.

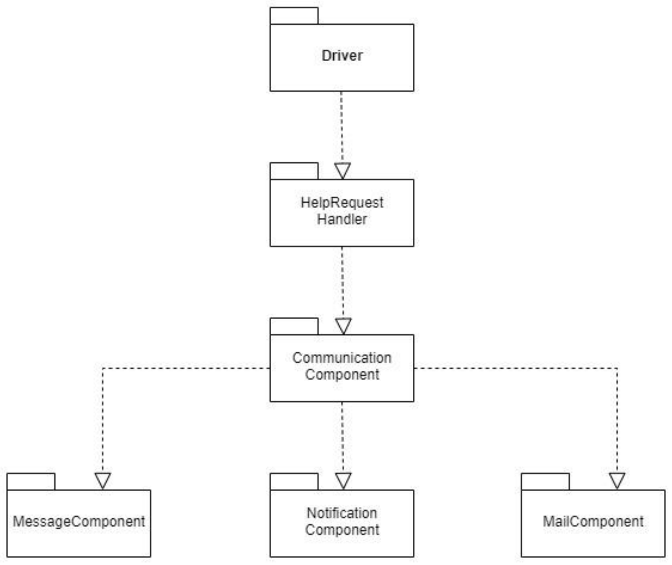
DailyPlan and SteeringInitiative handlers

The DailyPlanHandler component and the SteeringInitiativeHandler component need the access to all the subcomponents mentioned earlier, for this reason they are integrated afterwards to the system.
The image below illustrates, in the case of the DailyPlanHandler component, the sub components needed, while the rest of the application is simulated by a driver.
The same picture holds also for the SteeringInitiativeHandler, just substituting the DailyPlanHandler with the SteeringInitiativeHandler.



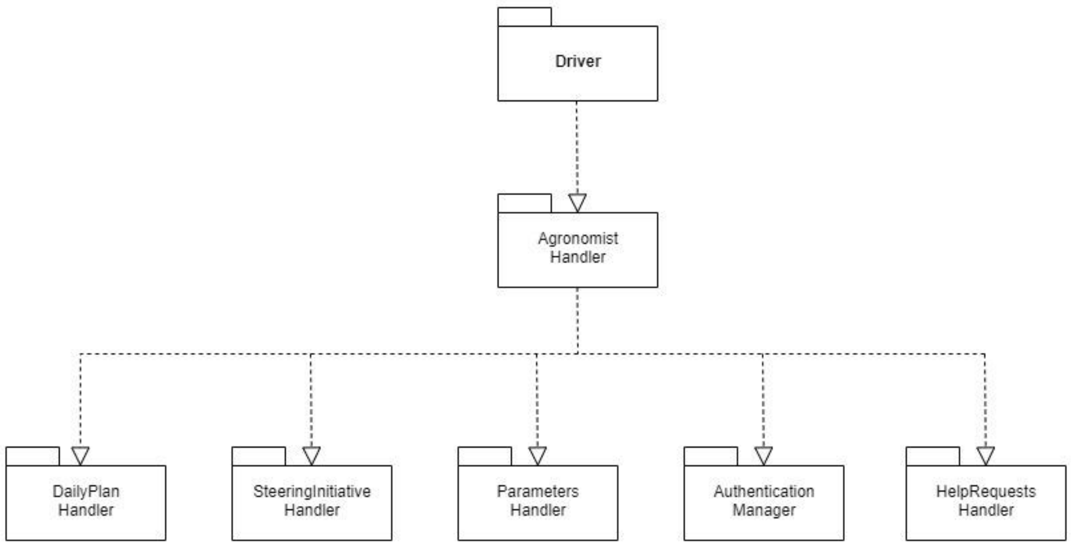
Forum and help requests handler

The forumHandler component and the requestsHandler component need to access the communication component, for this reason they will be integrated after the in the system.
The image below shows, in the case of the requestsHandler component, the component that must be already implemented to test this part of the application, while the upper part will be simulated by a Driver which exploits the component.
The same applies also in the case of the forumHandler component.



Handlers

One of the last components to be tested and integrated are the PolicyMakerHandler, the AgronomistHandler and the FarmerHandler, these high level components are at the top of each feature and for this reason must be deeply tested, once all the sub-components have been integrated and are working properly.
The image shows the case for the AgronomistHandler, but the same holds also for Policy Makers and Farmers handlers, with the respective associated modules.



Applications

While testing all the server components, the client can be tested separately but, once all the integration server side is ended, the two macro components need to be integrated and tested, to check if the whole application works.

Finally, the whole application is integrated and tested, to check if it is ready to be published as a fully working beta version.

6. Effort spent

All

First meeting	4h
Meeting	3h
Wrap up meeting	3h

Luca Venturini

Graphic Interfaces	4h
Component Diagrams	2h 30min
Tests	2h
Interfaces	1h
General text	1h

Dario Vernola

Component diagram description, traceability matrix, fixes	2h 30min
Sequence diagrams	2h
Modifications to the document	1h
Descriptions	1h
Sequence diagram fixes, Interface fixes, section 1 modification and references	2h 30min

Vittorini Francesco

Overview, Component diagram description	2h
Sequence diagrams	2h
General descriptions	1h 30min
Layered structure and traceability matrix	2h

7. References

Agriculture in Telangana
https://www.niti.gov.in/writereaddata/files/Telangana_Presentation_0.pdf

Jackson, M., & Zave, P. (1995). Deriving Specifications from Requirements: an Example.
https://www.fceia.unr.edu.ar/asist/jackson01.pdf

Data meteorological forecasts
https://www.tsdps.telangana.gov.in/aws.jsp

Why India has introduced the new Personal Data Protection Bill
https://www.datacenterdynamics.com/en/opinions/why-india-has-introduced-the-new-personal-data-protection-bill/

Dall'idea al codice con UML 2 - Guida all'utilizzo di UML attraverso esempi
https://www.researchgate.net/publication/344569853_Dall'idea_al_codice_con_UML_2_Guida_all'utilizzo_di_UML_attraverso_esempi

Information technology — Smart transducer interface for sensors and actuators — Part 1: Network Capable Application Processor (NCAP) information model
https://www.iso.org/standard/54357.html

ESA CCI Soil Moisture website
https://www.esa-soilmoisture-cci.org/

PalmNET: An open-source wireless sensor network for oil palm plantations
https://www.researchgate.net/publication/317619871_PalmNET_An_open-source_wireless_sensor_network_for_oil_palm_plantations

Slides course and materials uploaded on WeBeep