Dipartimento di Elettronica, Informazione e Bioingegneria

Software Engineering II Project DD

DREAM

Author:
Farimah Anvari: 10772192
Sajedeh Firouzizadeh: 10771435

Supervisors:
Prof. Elisabetta Di Nitto
Prof. Matteo Rossi

January 2022

Contents

1 Introduction

1.1 Purpose

This is the Design Document (DD) of DREAM application. The purpose of this document is to discuss more technical aspects regarding architectural and design choices that must be made, so as to follow well-oriented implementation and testing processes. This document is to provide more technical and detailed information about the software discussed in the RASD document.
In this DD we present hardware and software architecture of the system in terms of components and interactions among those components. Furthermore, this document describes a set of design characteristics required for the implementation by introducing constraints and quality attributes. It also gives a detailed presentation of the implementation plan, integration plan and the testing plan.
In this document we mostly talk about:

- The High-level architecture of the system.
- Main components of the system.
- Interfaces provided by the components.
- Design patterns.
- final representation

1.2 Scope

The software wants give them the possibility to see the details of their lands and the predictions, start discussions and ask their problems.

Basic service Farmers can visualize data relevant to them and also allow farmers to follow the progress of their lands and the details about the irrigation and products based on the information that the application gave them. Also, they can insert the data about their lands and also the updates about their products.

Advance Function1 Second functionality point out about the farmers that can create discussions with other farmers. Also, they can insert any problem that they are faced.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Policy Maker	A person who observe farmers progress
Farmers	A person who has a farm land on Telengana
Water-Irrigation System	An organization who observe the amount of used water
Sensor	A device that sense a physical phenomena

1.3.2 Acronyms

DD	Design Document
UI	User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Over Secure Socket Layer
REST	Representation State Transfer
JSON	JavaScript Object Notation
RASD	Requirement Analysis and Specification Document
GPS	Global Positioning System
App	Application
API	Application Programming Interface
DBMS	Database Management System
ODBC	Open Database Connectivity
MVC	Model, View, Controller
SDK	Software Development Kit

1.3.3 Abbreviations

BS	Basic Service of DREAM
AF1	Advance Function 1 of DREAM
Rn	Requirement number n

1.4 Revision history

Date	Modifications
09/01/2021	First version

1.5 Reference Documents

- Specification Document: "R&DD Assignment A.Y. 2021-2022.pdf"
- Slides of the lectures.

1.6 Document Structure

This document is divided in seven sections.

- **Chapter 1** describes the scope and purpose of the DD, including the structure of the document and the set of definitions, acronyms and abbreviations used.
- **Chapter 2** contains the architectural design choice, it includes all the components, the interfaces, the technologies (both hardware and software) used for the development of the application. It also includes the main functions of the interfaces and the processes in which they are utilised (Runtime view and component interfaces). Finally, there is the explanation of the architectural patterns chosen with the other design decisions.
- **Chapter 3** shows how the user interface should be on the mobile and web application.
- **Chapter 4** describes the connection between the RASD and the DD, showing the matching between requirements described previously with the elements which compose the architecture of the application.
- **Chapter 5** traces a plan for the development of components to maximize the efficiency of the developer team and the quality controls team. It is divided in two sections: implementation and integration. It also includes the testing strategy.
- **Chapter 6** shows the effort spent for each member of the group.
- **Chapter 7** include the reference documents.

2 Architectural Design

2.1 overview

The three-tier architecture model, which is the fundamental framework for the logical design model, segments an application's components into three tiers of services. These tiers do not necessarily correspond to physical locations on various computers on a network, but rather to logical layers of the application. How the pieces of an application are distributed in a physical topology can change, depending on the system requirements.

Other benefits (compared to single- or two-tier architecture) include:
Faster development: Because each tier can be developed simultaneously by different teams, an organization can bring the application to market faster, and programmers can use the latest and best languages and tools for each tier.

Improved scalability : Any tier can be scaled independently of the others as needed.

Improved reliability: An outage in one tier is less likely to impact the availability or performance of the other tiers.

Improved security: Because the presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall, preventing SQL injections and other malicious exploits.

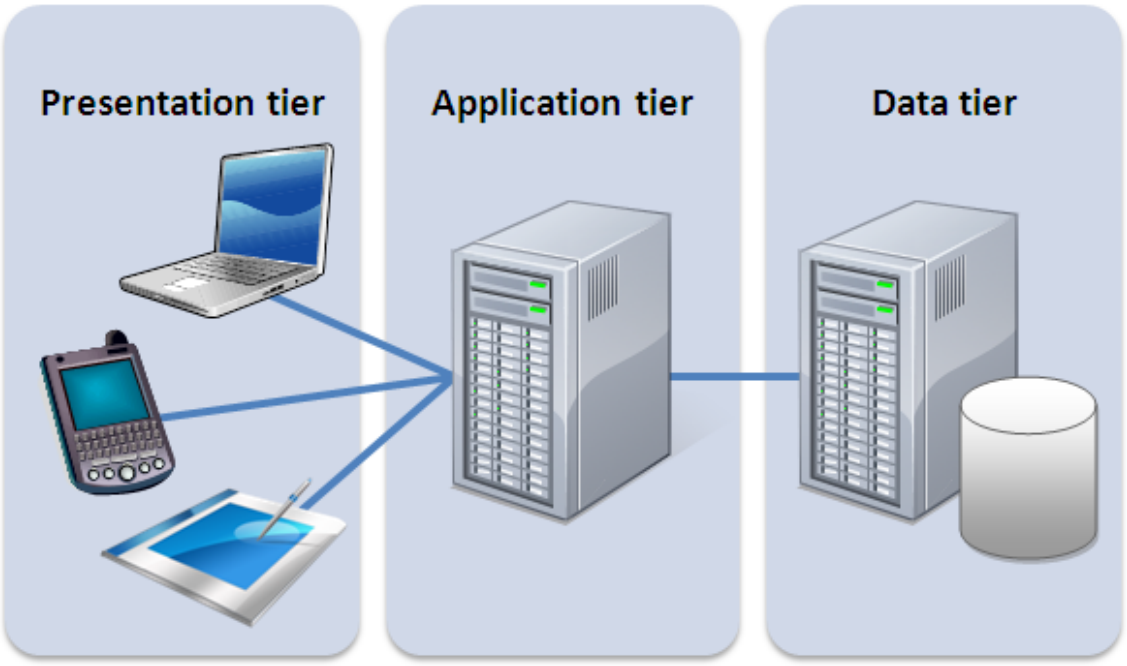


Figure 1: Three tier Architecture

2.2 High Level Components

The architecture of the application is structured according to three logic layers:

presentation tier: or user services layer, gives a user access to the application. This layer presents data to the user and optionally permits data manipulation and data entry. The two main types of user interface for this layer are the traditional application and the Web-based application. Web-based applications now often contain most of the data manipulation features that traditional applications use. This is accomplished through use of Dynamic HTML and client-side data sources and data cursors.

The business tier: or middle services layer, consists of business and data rules. Also referred to as the business logic tier, the middle tier is where COM+ developers can solve mission-critical business problems and achieve major productivity advantages. The components that make up this layer can exist on a server machine, to assist in resource sharing. These components can be used to enforce business rules, such as business algorithms and legal or governmental regulations, and data rules, which are designed to keep the data structures consistent within either specific or multiple databases. Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require. For example, simple edits can be placed on the client side to minimize network round-trips, or data rules can be placed in stored procedures.

The data tier: or data services layer, interacts with persistent data usually stored in a database or in permanent storage. This is the actual DBMS access layer. It can be accessed through the business services layer and on occasion by the user services layer. This layer consists of data access components (rather than raw DBMS connections) to aid in resource sharing and to allow clients to be configured without installing the DBMS libraries and ODBC drivers on each client.

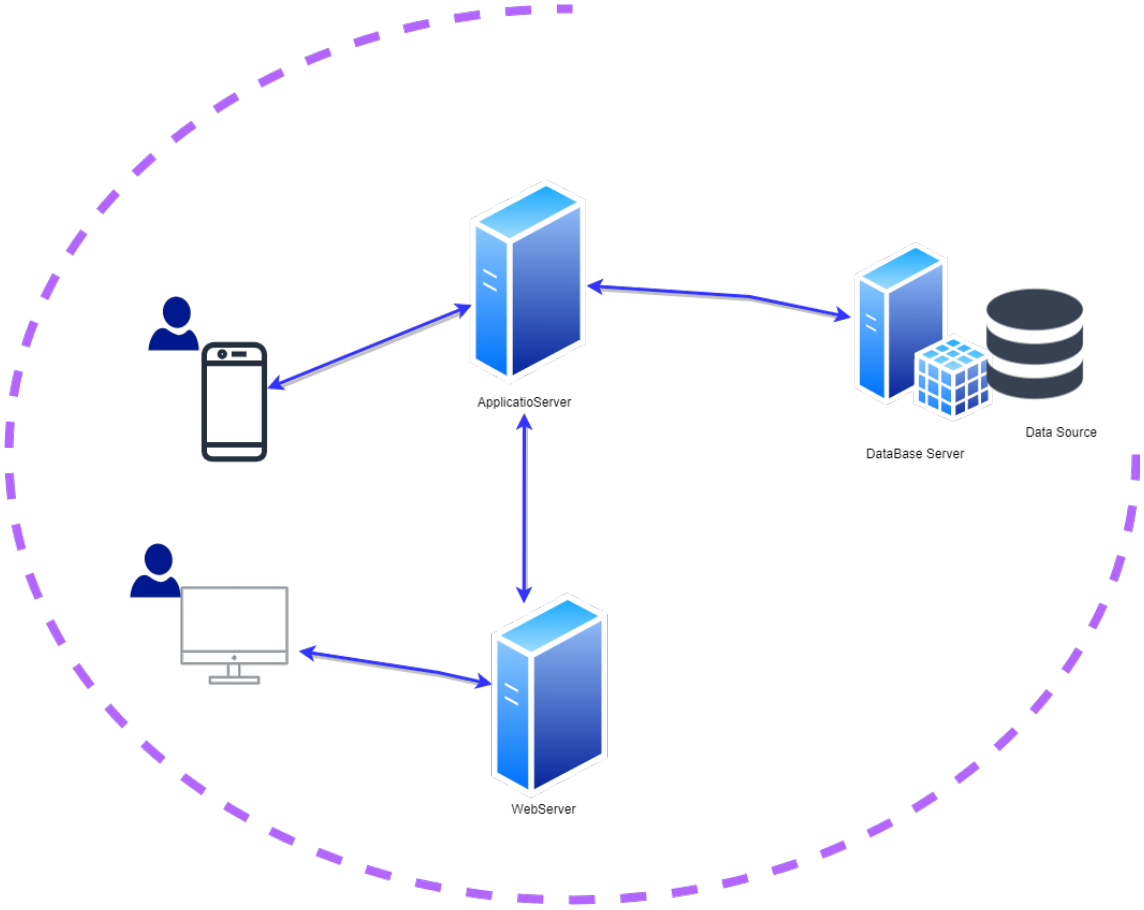


Figure 2: the system Architecture

During an application's life cycle, the three-tier approach provides benefits such as re-usability, flexibility, manageability, maintainability, and solubility. You can share and reuse the components and services you create, and you can distribute them across a network of computers as needed. You can divide large and complex projects into simpler projects and assign them to different programmers or programming teams. You can also deploy components and services on a server to help keep up with changes, and you can redeploy them as growth of the application's user base, data, and transaction volume increases. High Level Components

2.3 Component View

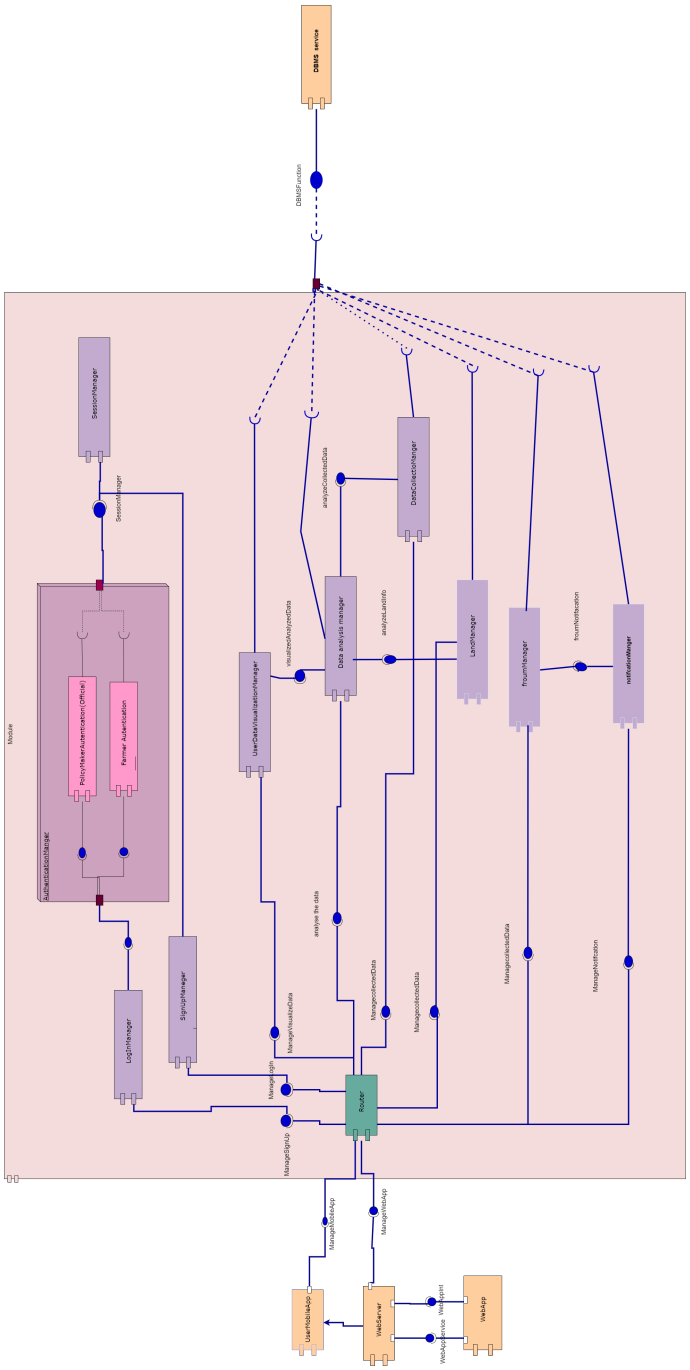


Figure 3: Component Diagram

components' functions contained in the 'ApplicationServer'focusing on the representation of the internal structure of the application server, showing how its components interact. are described in the following:

NotificationManager: it manages the notification which is used to send to users when its going to be close to be their turn and also their turn to enter the super-

market so it is connected to Estimator component.

DBMSManager: this is the component that allows every other component in the system to interact with the database. The Interface provided by this component contains all useful methods to store, retrieve, update data into the database from different actors. Every internal component of the application server uses some methods of its interface.

Router: this component simply dispatches the requests and calls to methods from the users and the managers to the core of the application server. Every method is redirected to the proper component that can handle it. Also, responses and data sent back pass through this component to reach the applicant.

SignUpManager: this component contains all the procedures to allow the farmer to register to **DREAM** expressing also to which service they want to register for.It has to interact with DBMS to store data about the registration and performing controls about the chosen username and password.

LoginManager: it manages all the logic inherent to the authentication of the customers. It interacts with the DBMS to check that the authentication parameters match the stored ones.

Authentication component: include two main sub component as follow:

- **policymaker authentication (official):** this component check the credential and access level with official organization and authorize them.
- **farmer authentication:** farmer authentication with saved data in database and check the access level.
- **session manager:** this component for authenticated user create the session and log the necessary information in database.also this delete the session after the user log out.

session manager: this component for authenticated user create the session and log the necessary information in database.also this delete the session after the user log out.

forum manager: this component responsible for manage the forum such as create, delete topics and save the result on the database or manage the putting comment or manage the question and answer and send the notification to related user.

land manager: responsible to related land function such as add new land,delete land or modify the production or fertilizer of land by given the location ID and save the changes in database.

data collection manager: This component responsible to measure the environmental variable of farm land such as used water ,today weather and soil humidity and save in database.Also preparation data for analysing unit to have better performance.

data analysis manager: this section analyze the function of the all farmer by environmental input such as used water,weather,soil humidity and out put is assess the farmer performance to show to the policymaker to encourage the best one and help to weak performance farmer

data visualization manager: this section visualize the analyzed data about farmer progress(such as product and fertilizer recommendation) to show farmers to monitor the condition and improve the performance

2.4 Deployment View

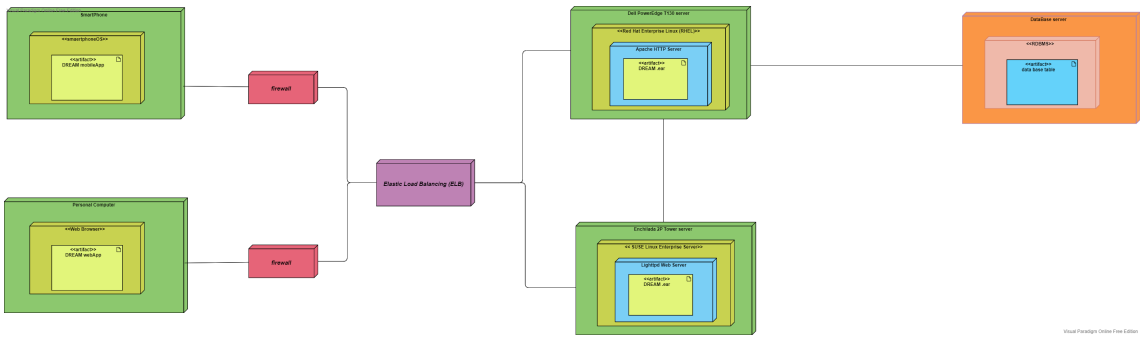


Figure 4: deployment view

In the Deployment diagram in the figure are shown the most important components. A better explanation of these ones follows here:

Smartphone: The farmers can use the smartphone to login and check the status of the their land or their progress on the other hand policy makers can monitor farmer performance by their smartphone .the information send to the Application Server using HTTPS.

Computer: another device to that farmer and policymakers can use to enter in**DREAM** is computer to have better experience to monitor of environmental situation and the progress.

Firewall: Provides safety access to the internal network of the system as part of the safety of the system against external attacks.

Load balancer: Distributes the workloads across multiple servers to increase capacity (concurrent users) and reliability of applications trying to avoid the overload of any single server.

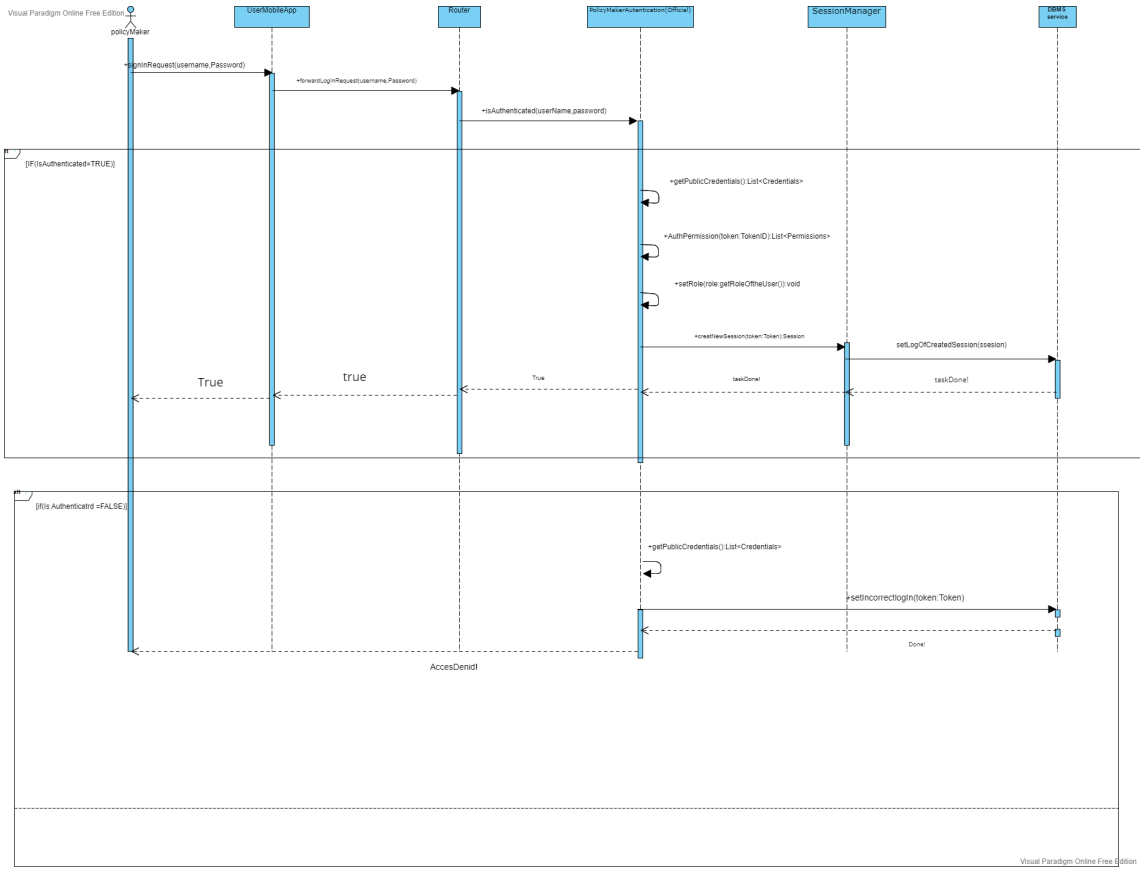
Web Server: Receives contents and requests from clients through web app and sends data received, using Java RMI, to the Application Server for processing. Moreover, it is replicated to avoid a single point of failure and to guarantee a better performance.

Application/web Server: Here we have the application logic, but not all because also the client has a part of this. The Application Server handles all the requests and provides the appropriate answers for all the offered services. It is directly addressed by the client app and , it is replicated just for having more accessibility an reliability.

Database Management System (DBMS): is interface between the end user and the database, simultaneously managing the data, the database engine, and the database schema in order to facilitate the organization and manipulation of data.and we use visualized and analyzed information by using provided data from saved raw data in database.

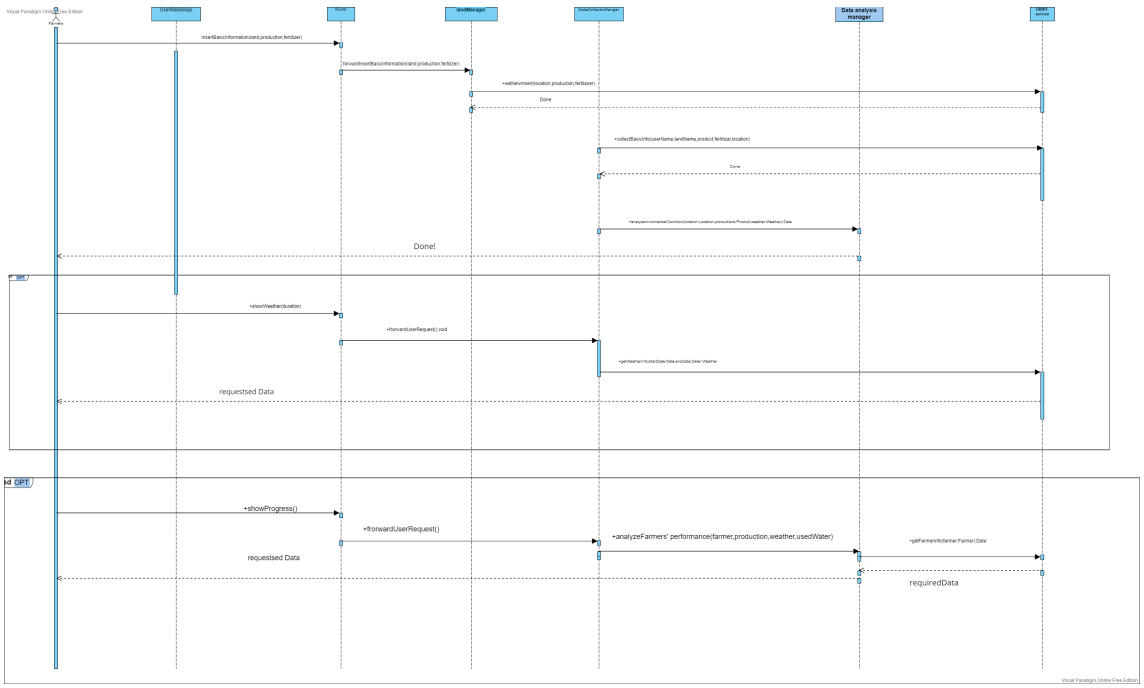
2.5 Run time view

2.5.1 Authenticate user



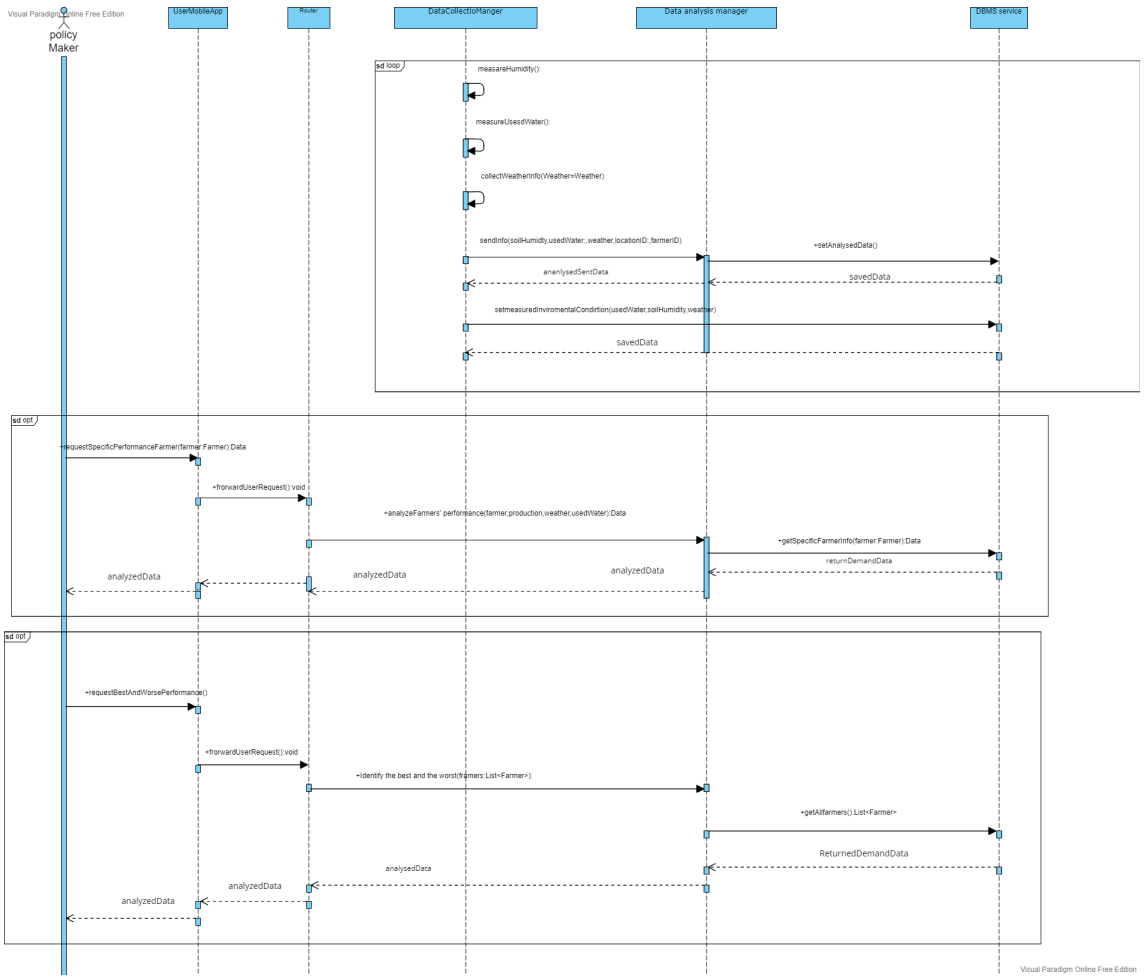
In the sequence diagram shows how user is authenticated in our system. first user request to log In in system by mobile application and with router this request is delivered to the login manager to validate the request and it to authentication component to export credential of the users and authorized them and dedicate the access by considering the their role in our system after the user data with role pass to the session manager to create new session and log necessary info in database .

2.5.2 Analyse the Land Info



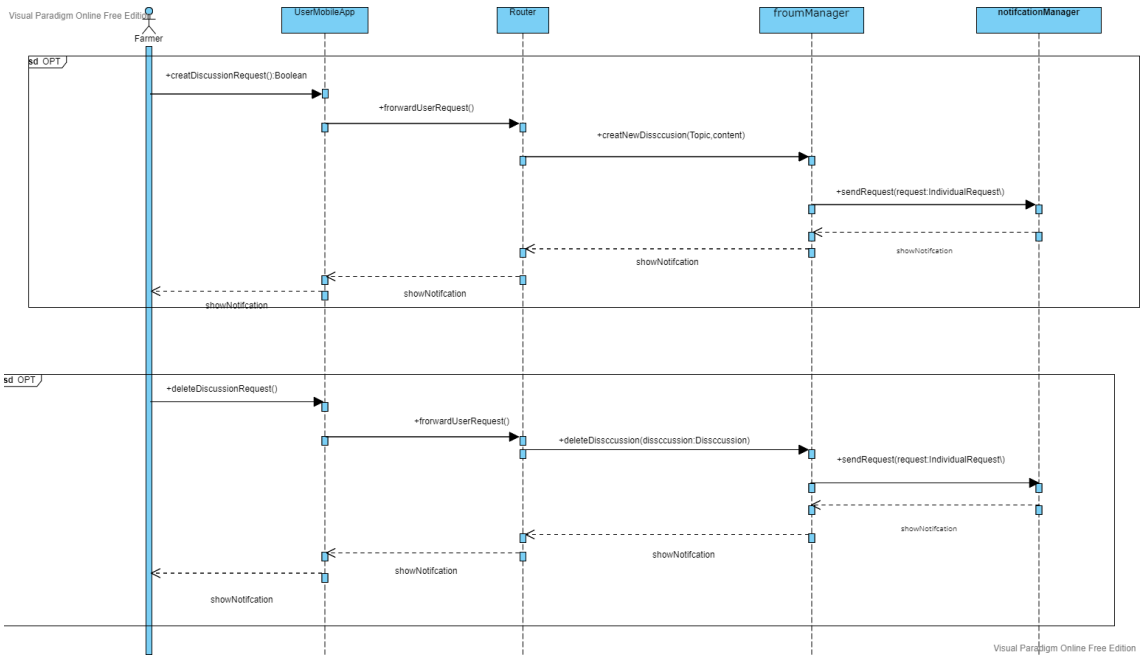
In the sequence diagram shows how system make recommendation essential suggestion to farmer. First farmer insert the her land information such as location and production by the mobile application and router forward this request to land manger that response for manage the affair of land and insert this information in database the collection section gather this info and measure other information such as weather condition and deliver this raw information to analyzer section to process this info and suggest the best production and fertilizer can be used.
Also farmer can check the monitoring information such as his progress and or land weather by mobile application and router forward the request to collection unit to gather appropriate info from database and show to the users also analysing unit by provided info by collection unit can measure the farmer progress .

2.5.3 Assess farmer progress



In the sequence diagram shows how the policy maker assess the farmer progress system continuously measure the the environmental variable by collection unit and get another information from database and deliver this Information to analysis unit and after the saved this analysed data to database and then show to policy maker if request to identify the best and worst farmer or and can request to see the specific farmer performance as the same way.

2.5.4 forum activity



In the sequence diagram shows how the farmer can have a activity in forum . user can request to create the new topic router forward the request to forum manager and dedicate a topic and content and save these formation in the database also notify the user to these process done correctly.Also user can delete the past topic by his mobile application and router forward this request to forum manager and after this component save these change to database ,notify the user this process done correctly.

2.6 Component Interfaces

In the next diagram are described the main methods which can be invoked on the interfaces and their interactions, referring to the most important processes reported in the run time view section.
One aspect is fundamental to be pointed out: in general, methods written in the Component Interfaces diagram are not to be intended exactly as the methods that the developers will write, but they are a logical representation of what component interfaces have to offer. They will be adapted facing the various aspects that will come out during the implementation of the code

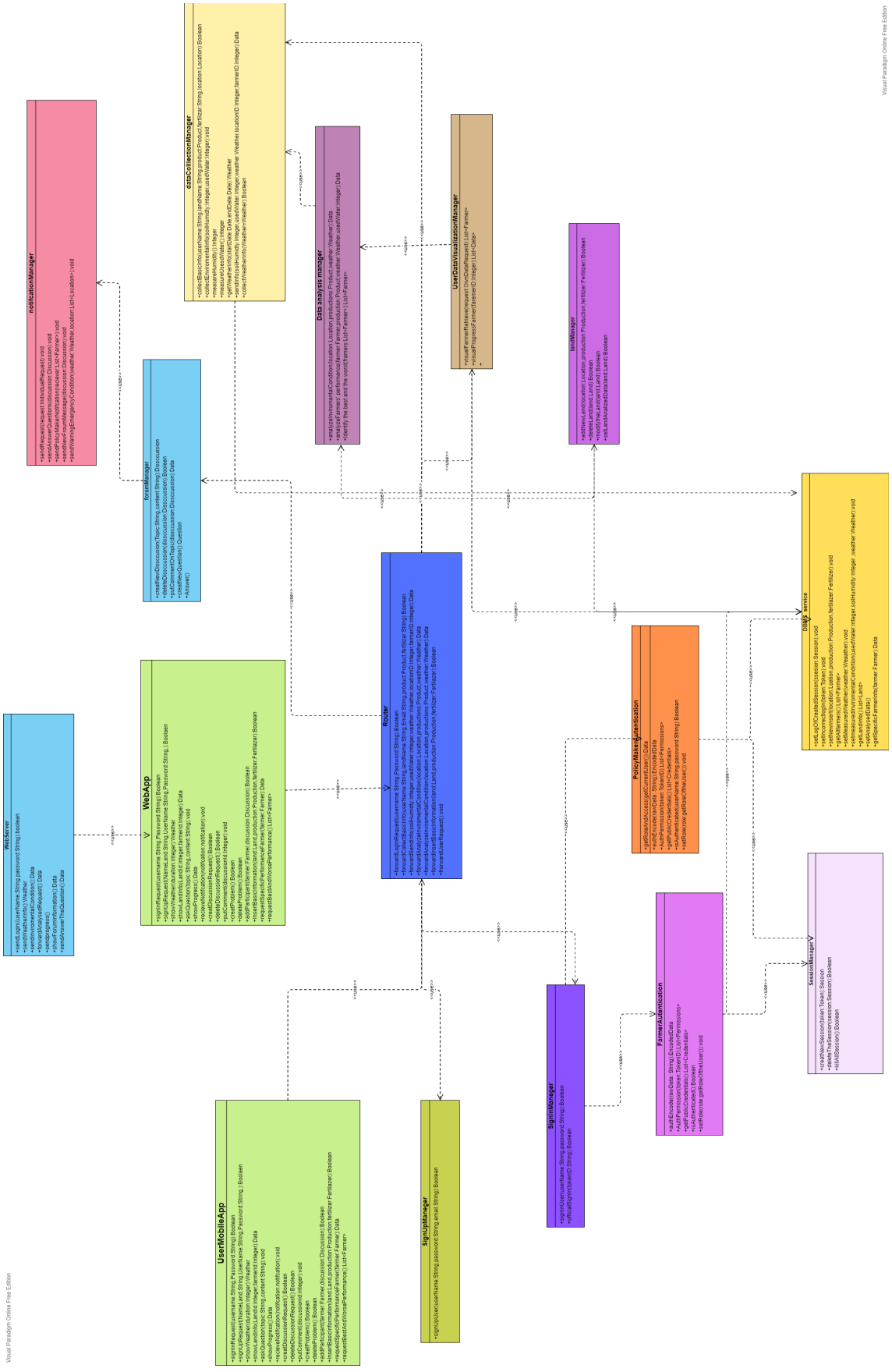


Figure 5: component interface diagram

2.7 Selected Architectural Style and Patterns

Our system is based on a client/server architecture with three layers. Presentation, application and data access are physically separated. We choose this kind of architecture in order to have a modular system, also we use fat client to reduce the server load and smartphones nowadays are very powerful so, the costs of server will be less. Moreover we separated physically the database from the application server, to increase scalability and security, and it's also possible to use a distributed database. This choices promote a major decoupling of the system, increasing the reusability, scalability and exibility.Furthermore, components in the application server have been thought to be cohesive and with low coupling among modules to make the system more comprehensible and modifiable. The communication between the Mobile application and the Server will be done via HTTP requests following REST principles. The communication between modules inside the application server and DB manager is RPC. When we use RPC, the programmer can use procedure call semantics and writing distributed applications is simplified. because RPC hides all of the network code into stub functions.
In addition, we use some caches in the client side so we avoid some interactions between the client and server so we could reduce server loads. For example, in a short period of time users location do not change so, we could cache shops near the user for short amount of time and do not send requests to server. We use JSON format for the communication data because, JSON uses less data overall, so you reduce the cost and increase the parsing speed. Readable: The JSON structure is straightforward and readable. You have an easier time mapping to domain objects, no matter what programming language you're working with. The model we choose for this project is Model View Controller (MVC). This model im- proves the reusability and maintainability of code. One of the most important feature of this design pattern is separation of concerns.

•**View:** is that part of the application that represents the presentation of data.Views are created by the data collected from the model data. A view requests the model to give information so that it resents the output presentation to the user.

• **Controller** is that part of the application that handles the user interaction.

•**Model** component stores data and its related logic. It represents data that is being transferred between controller components or any other related business logic.

2.7.1 Other Design Decision

In this section, we elaborate some decision we take.

Flutter: Flutter is a free, open-source mobile SDK that can be used to create native-looking Android and ioS apps from the same code base. Flutter helps app developers build cross platform apps faster by using a single programming language, Dart, an object-oriented, class defined programming language. We chose to use this Framework in order to have a mobile app that works both with ioS and Android, with performances similar to the native development for each platform, but only having to write the code once. In Flutter, every piece of the user interface is a widget: like text, buttons, check boxes, images. There are also, "container widgets" that contain other widgets. Widgets can be stateless, which are immutable, or stateful, which have a mutable state and are used when we are describing a part of the user interface that can change dynamically.
Lastly we want to remember that Flutter is not the only framework that can be used to build cross-platform mobile apps, another option can be React Native, based on JavaScript. We decided to use Flutter because of it's advantages in comparison with React Native and other UI software development kit, which are: better performances and increasing popularity due to it's simplicity of develop.

Golang: Go is a statically typed, compiled programming language. Go has the same performance as C, but it is much easier to maintain than Java. Without the need for a virtual machine, Go boasts easier maintenance and no warming up period. These and many other characteristics are what make Golang stand out from its competitors.

Docker: Containers work a little like VMs, but in a far more specific and granular way. They isolate a single application and its dependencies|all of the external software libraries the app requires to run|both from the underlying operating system and from other containers. All of the containerized apps share a single, common operating system (either Linux or Windows), but they are compartmentalized from one another and from the system at large. The benefits of docker:

- Docker enables more efficient use of system resources.
- Docker enables faster software delivery cycles.
- Docker enables application portability.
- Docker shines for micro services architecture

3 User Interface Design

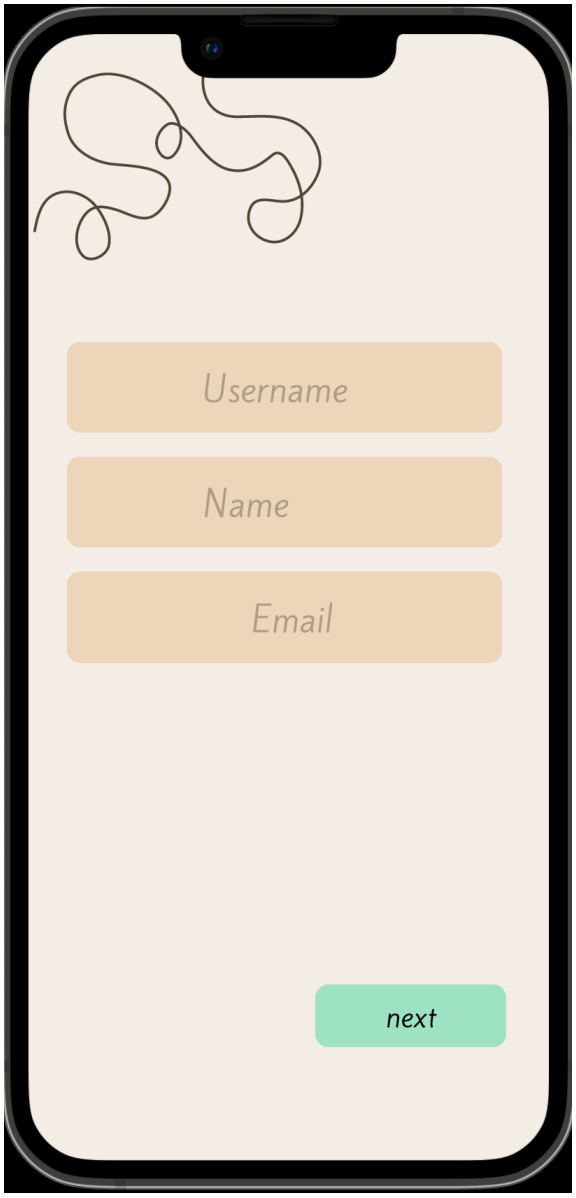
3.1 Mockups

In the following section you can see the look of the mobile and web application that can be used by the farmers and policymakers.

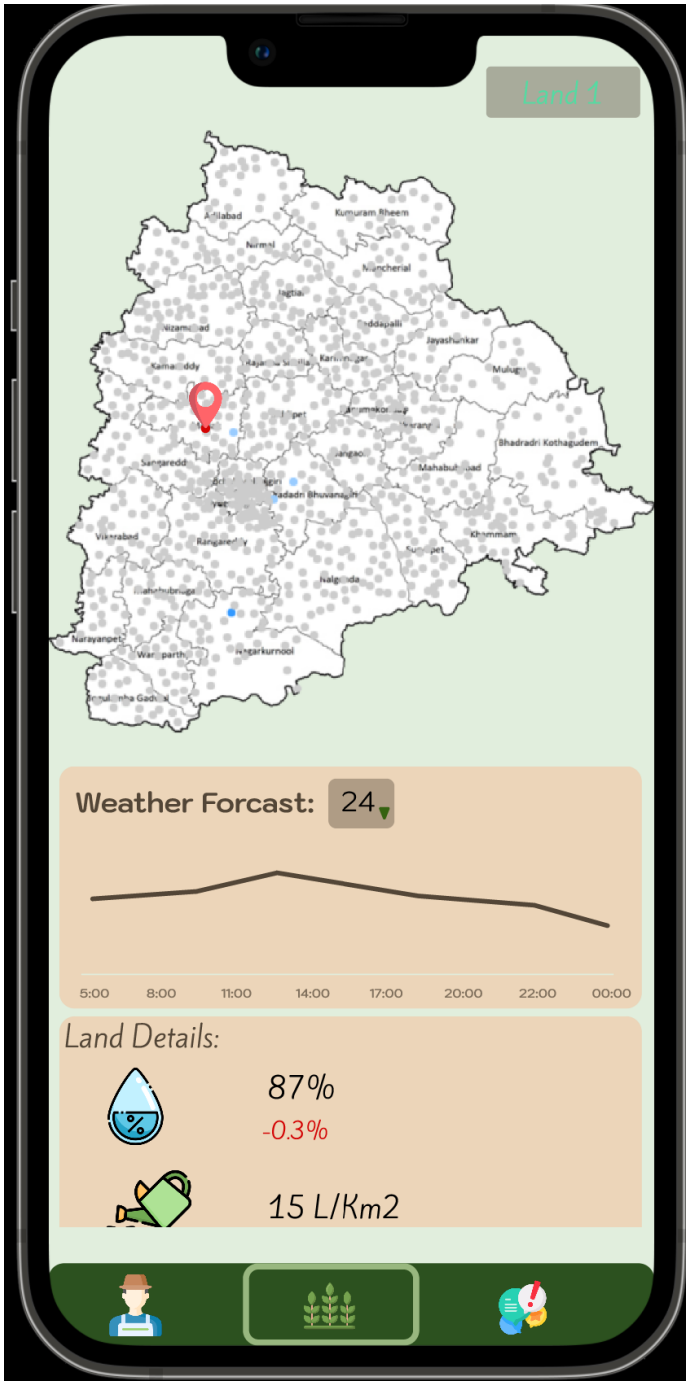
3.1.1 Logo

3.1.2 Farmers Login

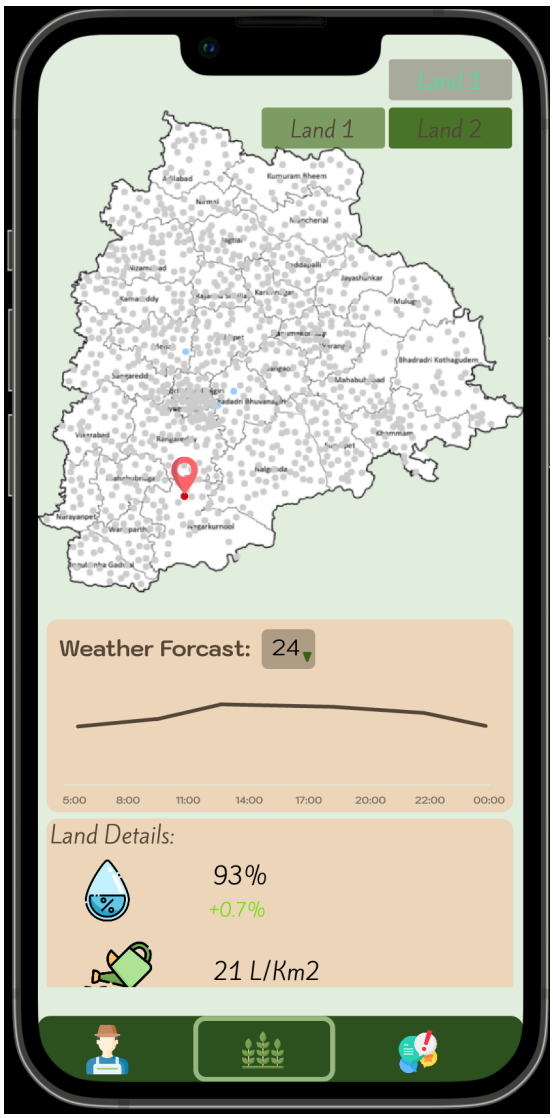
3.1.3 Farmers Signup



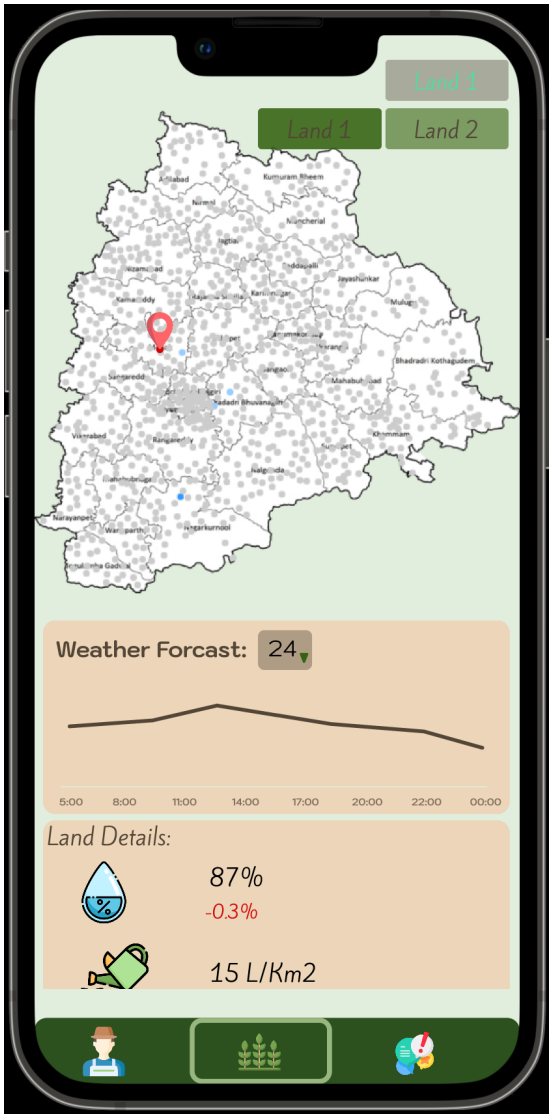
3.1.4 Farmers Home



3.1.5 Change Land

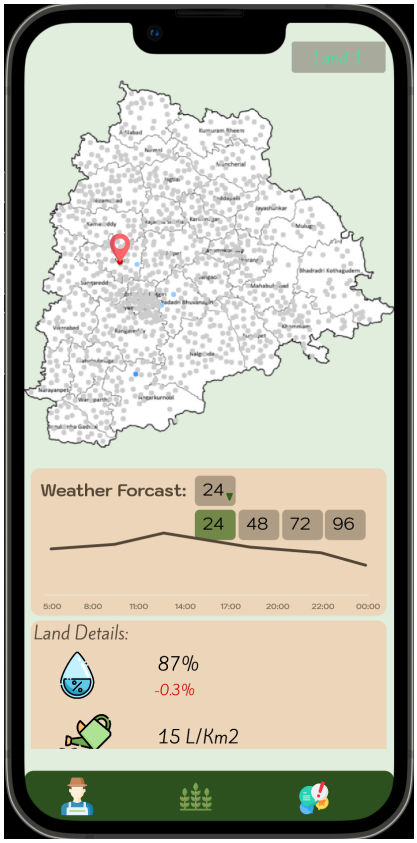


(a) Land 1

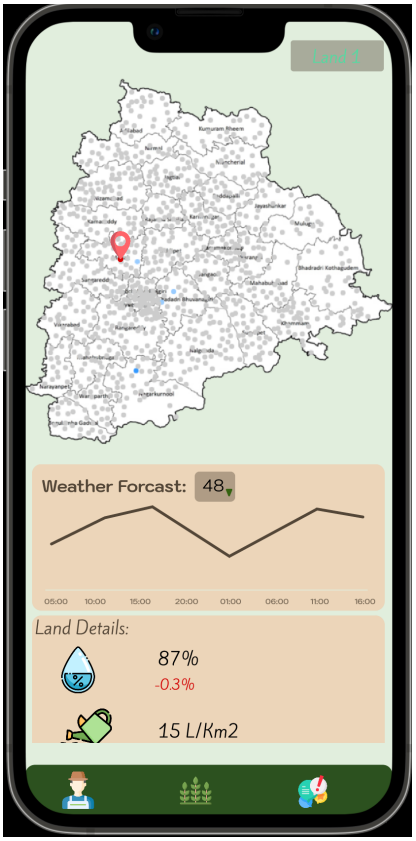


(b) Land 2

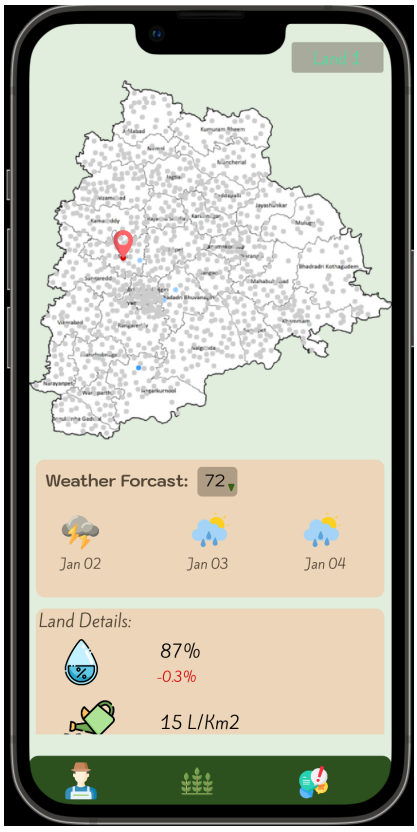
3.1.6 Change Forcast



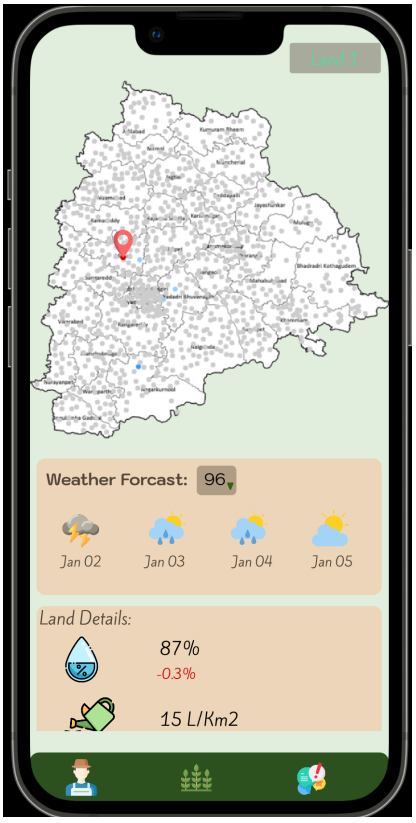
(a) 24 hours



(b) 48 hours



(c) 72 hours

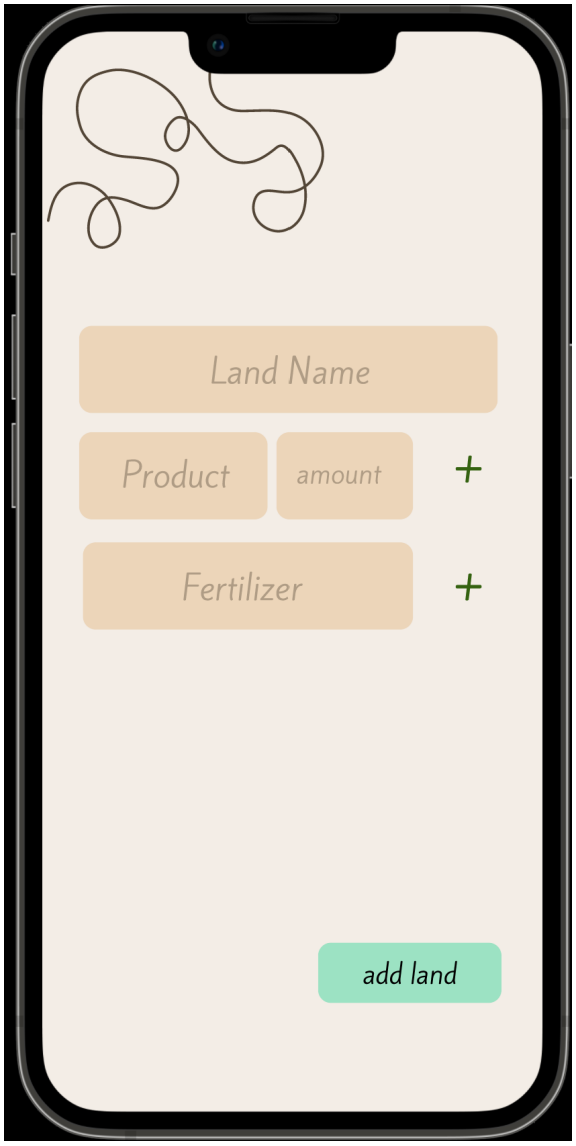


(d) 96 hours

3.1.7 Farmers Profile

3.1.8 Add Land Location

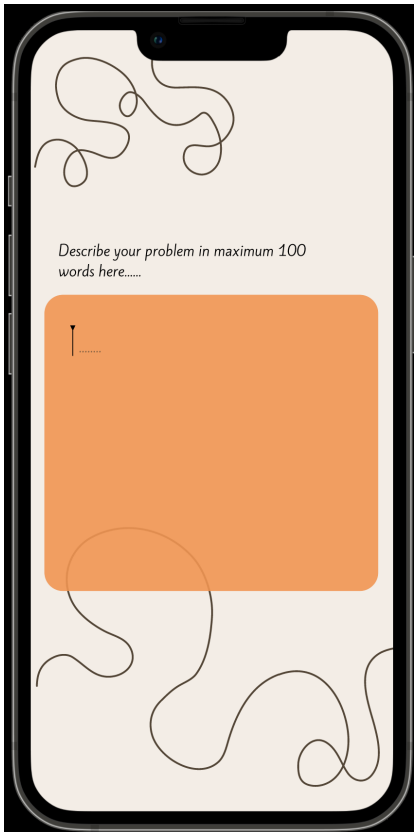
3.1.9 Add Land Details



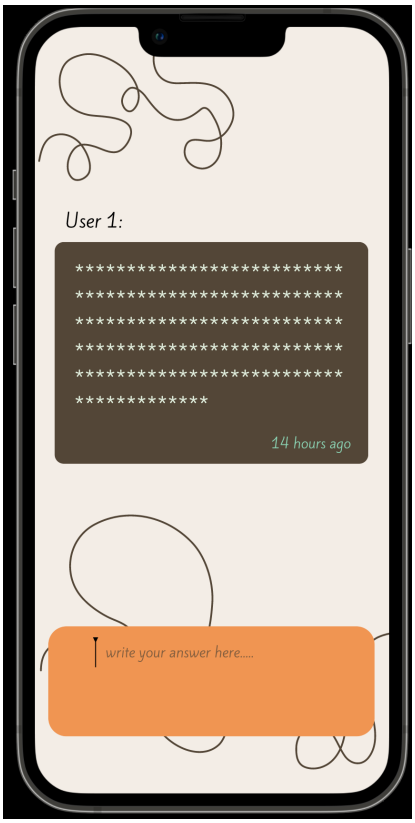
3.1.10 Problems

3.1.11 Discussions

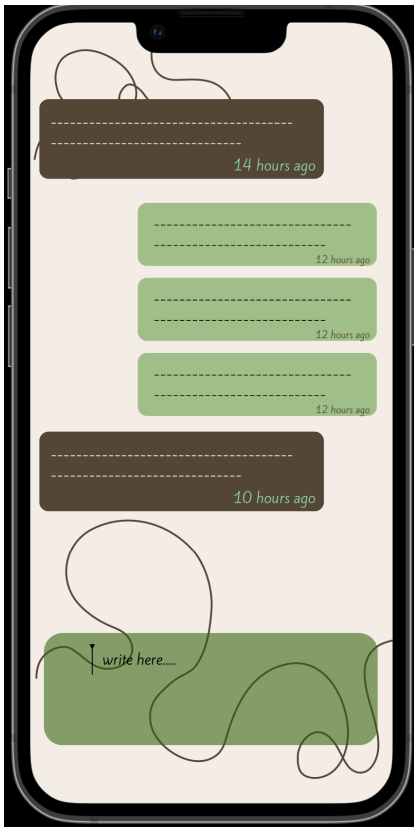
3.1.12 Problems and Discussions details



(a) Ask Problem



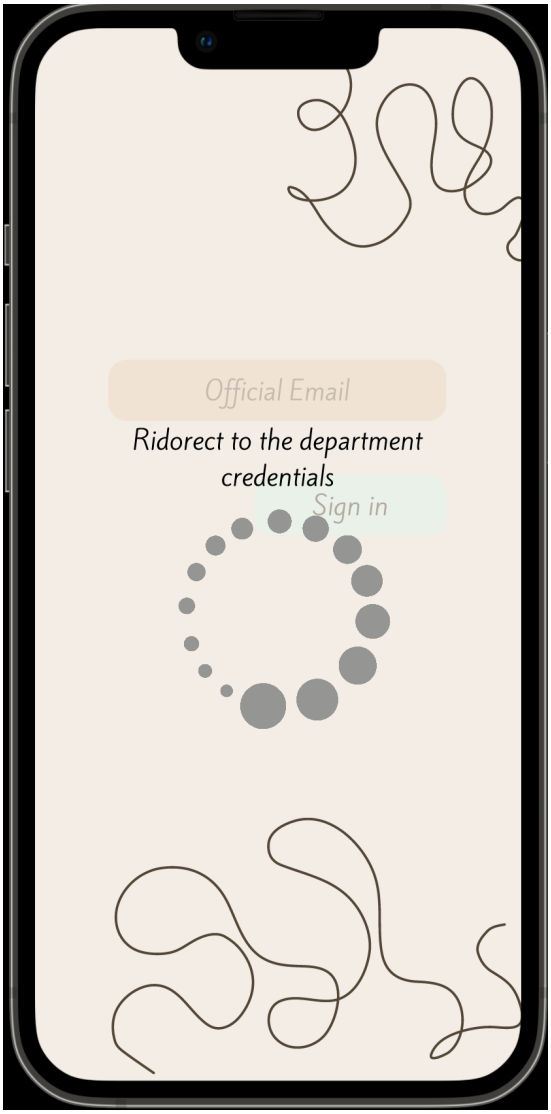
(b) Answer to Problem



(c) Answer to Discussions

3.1.13 Policymaker Login

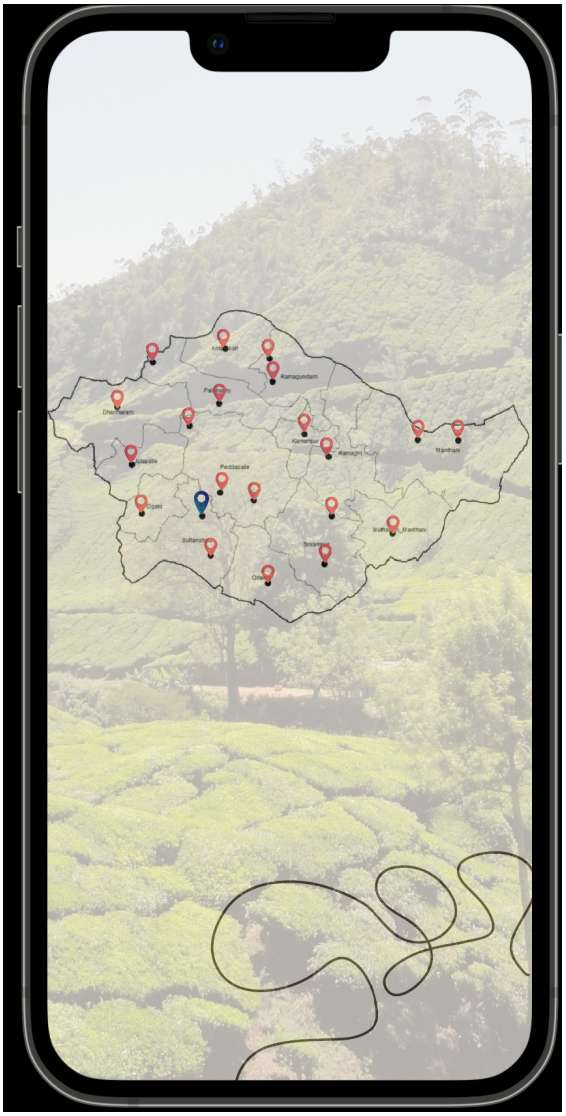
(a) Login



(b) Redirect to organization

3.1.14 Policymaker Home

(a) home page



(b) lands

3.1.15 Check Farmers

3.1.16 Promote or Help Farmers

3.1.17 Farmers Home page Web App

3.1.18 Farmers Home page Web App

Other pages for policymakers and farmers are like mobile with different scales.

4 Requirements Traceability

R1	components: • FarmerAuthentication • SignInManager • DBMSManager
R2	components: • PolicyMakerAuthentication • SignInManager • DBMSManager
R3	components: • PolicyMakerAuthentication • SignInManager • DBMSManager
R4	components: • PolicyMakerAuthentication • SignInManager • UserMobileApp / WebApp • UserDataVisualizationManager • DBMSManager
R5	components: • PolicyMakerAuthentication • UserMobileApp / WebApp • UserDataVisualizationManager • DBMSManager

R6	components: • PolicyMakerAuthentication • UserMobileApp / WebApp • UserDataVisualizationManager • DBMSManager
R7	components: • PolicyMakerAuthentication • UserMobileApp / WebApp • UserDataVisualizationManager • DBMSManager
R8	components: • UserMobileApp / WebApp • DBMSManager
R9	components: • UserMobileApp / WebApp • DBMSManager
R10	components: • UserMobileApp / WebApp • LandManager • DBMSManager
R11	components: • UserMobileApp / WebApp • DataColllectionManager • DBMSManager

R12	components: • UserMobileApp / WebApp • LandManager • DBManager
R13	components: • UserMobileApp / WebApp • DataColllectionManager • DBManager
R14	components: • UserMobileApp / WebApp • ForumManager • DBMSManager
R15	components: • UserMobileApp / WebApp • ForumManager • DBMSManager
R16	components: • UserMobileApp / WebApp • NotifcationManager • ForumManager • DBMSManager

R17	components: • UserMobileApp / WebApp • ForumManager • DBMSManager
R18	components: • UserMobileApp / WebApp • ForumManager • DBMSManager
R19	components: • Routher • DataColllectionManager • DBMSManager
R20	components: • DataAnalysisManager • DBMSManager
R21	components: • DataAnalysisManager • DBMSManager
R22	components: • DataAnalysisManager • DBMSManager
R23	components: • NotifcationManager • DBMSManager

R24	components: • NotifcationManager • DBMSManager
R25	components: • NotifcationManager • DBMSManager
R26	components: • NotifcationManager • DBMSManager

4.1 Functional Requirements

List of Requirements

R1	Farmers certified with an authentication
R2	Policy-Makers certified with an authentication
R3	Policy-Makers should register to the application with extra mandatory fields
R4	Only Policy-Makers can observe all the farm land
R5	Only Policy-Makers can choose best and worst farmers
R6	Policy-Makers can choose farmers who received special incentives
R7	Policy-Makers must identify those farmers who need to be helped
R8	Farmers can be asked to provide useful best practices to the others
R9	Farmers must accept the GPS location for the application
R10	Farmers must insert their product to the system
R11	Farmers can visualize data relevant to them
R12	Farmers must insert the fertilizers that they are using to the system
R13	Farmers could see the list of forecast related to their location
R14	Farmers can insert any problem they face
R15	Farmers can start a discussion
R16	Farmers can request for help and suggestions by other farmers
R17	Farmers can join to an existing discussion
R18	Farmers can terminate their own discussion
R19	The system should receive the data related to meteorological short-term and long-term forecasts from Telengana's governments
R20	The system should suggest fertilizer based on the farmers locations and products
R21	The system should receive the data from sensors periodically
R22	The system should receive data from water irrigation system periodically
R23	The system should send notifications about the farm land to the farmers
R24	The system sends a notification to farmers if they receive new message
R25	The system sends a notification to farmers if the humidity if water decrease
R26	The system sends a notification to farmers if policymakers choose them as worst or best farmers

5 Implementation, Integration and Test Plan

5.1 Overview

This section will discuss the implementation, integration, and testing of the DREAM application. The section aims to prevent any setbacks entailing changes on the project once started its development and do multiple tasks twice. During the time that a team starts to develop a project, they have lots of ups and downs, maybe, they have had lots of bugs in their projects during the development phase and remove them easily. But as you may know, the cost of removing these bugs after publishing the application would be much more, so verification and validation play an important role in the development phase.

5.2 Implementation Plan

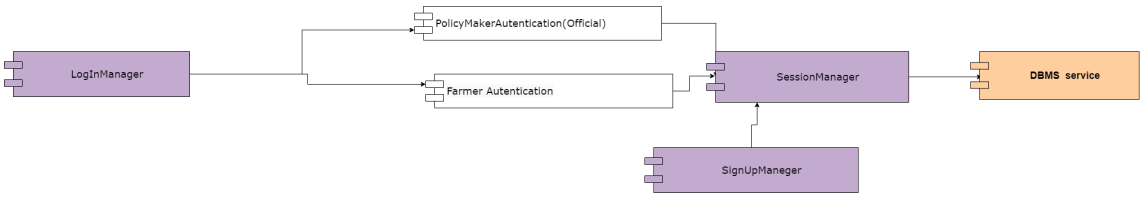
First of all, it is important to divide the project into smaller components in order to have more concrete goals that help keep the developers' motivation high and make a straightforward development of the project. Therefore, we will divide the project into different components, starting from the ones that are responsible for the basic features and going on until the end, where we will develop the most specific ones. It is not a new method, because it has been studied during this course and it is referred to as 'bottom-up' strategy. This method will help to provide better integration of the project tier-by-tier and make different tests of the behavior of the application before it ends.
In the bottom-up approach, the implementation must start from the lower components up to the top because in this approach the implementation is gradual. In the design, there are some components that rely on other components. Thus, it's better first start implementing with the component that other components use it.

Two important components that we have in our application are DataVisualization and DataAnalysis because we predict based on the data that we received from DataCollection. If we don't have enough accuracy in the data that we receive from DataColletion these components wouldn't work properly. For example, if our devices were not good enough to measure accurately, or we predict or conclude based on sparse data, or if these data couldn't fulfill the requirement of our algorithms correctly.

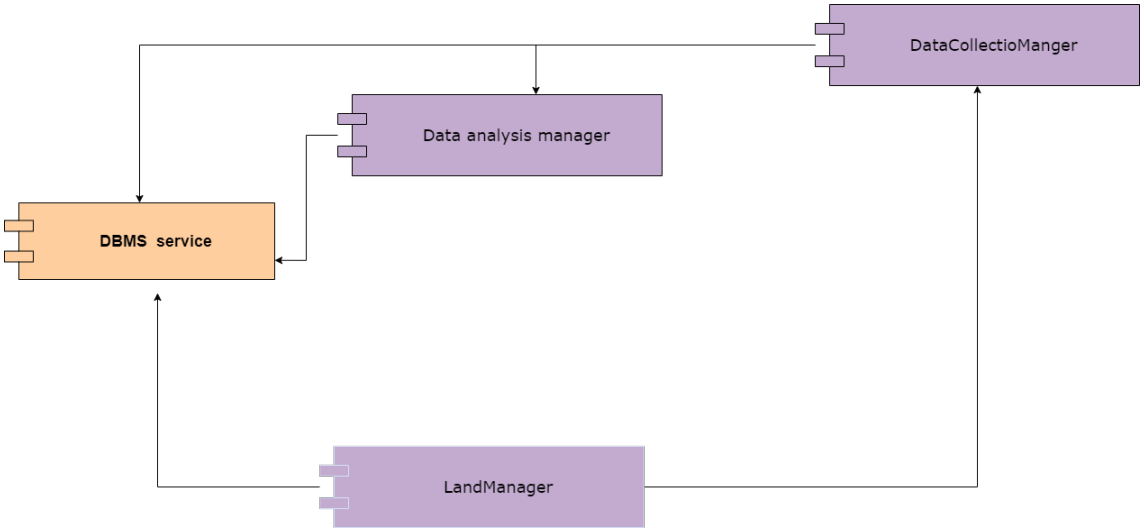
5.3 Integration Strategy

To implement and test the different functionalities of the system a bottom-up approach has been used. The following diagrams describe how the process of implementation and integration testing takes place, according to a bottom-up approach.

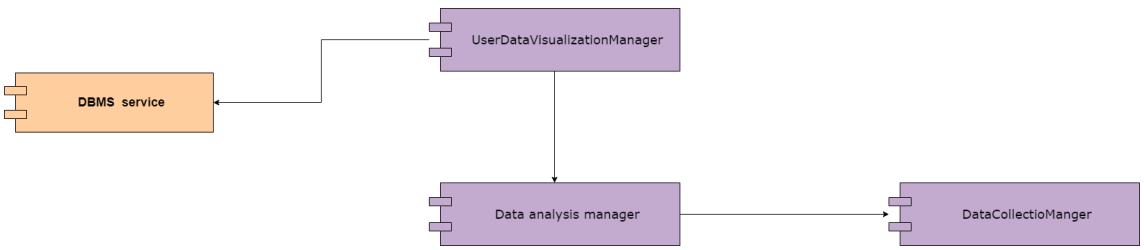
1. At first the authentication is implemented and unit tested using a driver for the components that are still under implementation. It is not a complex component, but other components rely on it to provide some services: that's why it is implemented and tested before the others.



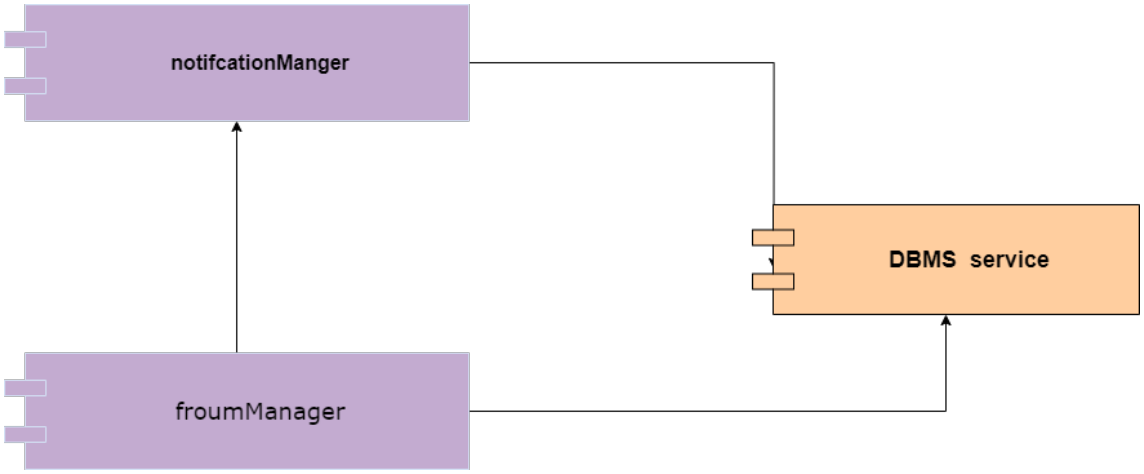
2. Then we implement the data analyzing, because one of the most important entity of our application is analyzing the farmer progress.



3. another integrated components are responsible for visualize the data that analyse by analyzing component and reparation them by data collection component.



4. to implement the functionality of forum for asking the question and send proper answer beside the sending notification to related user to notice them from last changes



6 Effort spent

Farimah Anvari:

Topic	Hours (hh:mm)
Design meeting	3:30
Introduction	2:00
Component View	1:00
Runtime View	1:00
Reading Deployment technology	1:00
User Interface	7:00
Requirement Traceability	4:00
Implementation Plan	3:00
Document Review	3:00

Sajedeh Firouzizadeh:

Topic	Hours (hh:mm)
Design meeting	3:30
Introduction	1:30
Component View	10:00
Runtime View	5:00
Reading Deployment technology	2:00
Deployment View	3:00
Study of patterns	3:00
Requirement Traceability	2:00
Integration Strategy	1:00
Revise Sequence Diagram	1:00

7 References

https://docs.microsoft.com/
https://app.diagrams.net/
https://online.visual-paradigm.com/
https://www.ibm.com/cloud/learn/three-tier-architecture