POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

DREAM - Data-dRiven PrEdictive FArMing in Telangana

DD
SOFTWARE ENGINEERING 2

TEAM MEMBERS:

Simone Brunello - 10831750
Nicholas Nicolis - 10867841

Academic Year: 2021-22

Contents

1 | INTRODUCTION

1.1. Purpose

This document is called Design Document, its purpose is to describe an overall view to the architecture of the software product already discussed in the RASD.
It contains a general description of the structure chosen to design the DREAM system. In addiction to this there are a representation of the UI and a list of the architectural components present in this document. It also contains a set of design characteristics required for the implementation by introducing constraints, quality attributes and it gives a presentation of the implementation, integration and testing plan.
The DD will be given to the development team indeed it has the purpose to guide them through the entire production process.

1.2. Scope

In order to manage more efficiently the communication between farmers and policy makers our DREAM application will provide an easy way to access the system which will make available to all different users a dedicate set of tools and information.

Farmers will be able to monitor weather conditions, crops and fertilized suggestions. They will have the possibility to send direct requests to expert or other farmers in order to receive advice. The ease of communicating their production data and problems will be a key point.

Telangana's policy makers will be able to monitor farmers performances and decide if current policies are providing good results. They will also be supported in the visualization of critical situations in order to intervene in advance.

1.3. Definitions

Definition	Description
Farmer	A farmer registered in the system.
Policy Maker	An authorized user who works for the government.
Farmer Report	Document containing information about production, expenses and incomes of farmers.
Policy Maker Report	Document containing an economical/production analysis towards a specific farmer. It includes also management suggestions about what they could do to improve or to maintain the situation.
Archive	A collection of technical information about plants, fertilizers, practical tools and techniques.
Forum	An application's section where users can hold conversations in the form of posted messages.
Ticket	A special message which farmers can use in order to directly contact policy makers, usually to ask help.
Mockup	A realization for illustrative or merely display purposes of the application's UI.
Land registry	A set of data and documents which attest the owner of a certain property or other goods , in this case they attest the owner of the farms.
Gantt diagram	This is a particular diagram who take his name from Henry Lawrence Gantt and is often used in the project management activities.

Table 1.1: Table of Definitions

Acronyms	Description
RASD	Requirements Analysis and Specification Document
DD	Design Document
UI	User Interface
GUI	Graphic User Interface
API	Application Programming Interface
JEE	Java Enterprise Edition
EIS	Enterprise Information System
EJB	Enterprise Java Bean
JSP	Java Scripting Preprocessor
DBMS	Data Base Management System
REST	REpresentational State Transfer

Table 1.2: Table of Acronyms

Abbreviations	Description
alt	alternative

Table 1.3: Table of Abbreviations

1.4. Document Structure

This document is composed of six chapters:

1. **Chapter 1: Introduction.** This part contains the scope the purpose of the DD document. This chapter also includes the structure of the document and a set of abbreviations, acronyms and definitions used.

2. **Chapter 2: Architectural Design.** This part contains the architectural design choices, there are an overview of the designed architecture, all the components, the interfaces, and the technologies used for the design of the application. This chapter also includes the functions of the interfaces and the process in which they are utilized. At the end there is an explanation of the design pattern chosen to develop the application.

3. **Chapter 3: User Interface Design.** This part contains a representation of how the UI should look like. It completes the User Interfaces subsection present inside the RASD.

4. **Chapter 4: Requirements Traceability.** This part contains a description of how the requirements defined in the RASD map to the design elements described in this document.

5. **Chapter 5: Implementation, Integration and Test Plan.** This part contains the plan which describes how to implement the subcomponents of the system and in which order. In addiction to this there is also the test planning for the system.

6. **Chapter 6: Effort Spent.** In this part there are information about how many hours each member of the group spent for every part of this document.

7. **Chapter 1: References.** Here there is just a list of the references used to complete this document.

2 | ARCHITECTURAL DESIGN

2.1. Overview

The application which is going to be implemented is a distributed one and it will has three logic software levels: presentation, application and data.

1. **Presentation:** It includes the components which have the aim to provide the final user interfaces. If the user logs in through the mobile application the interfaces are provided internally by the application. If they logs in through the web browser the interfaces are requested and then provided by the DREAM web server.

2. **Application:** It includes all the components used in order to define the business logic of the DREAM application. All the computations are located here.

3. **Data:** It includes all the parts which implement the storage and the management of data. So it is composed by DBMS and the database. In order to avoid losses of data there is a backup database. Part of the data are not stored internally, but retrieved from external APIs.

In order to to guarantee the security of sensible data, firewalls and an external authentification server have been designed.
Since this system will be used by a huge audience, cache servers will help to reduce the traffic through the main server. It's possible to add also a load balancer in case of necessity.
The application level also interacts with external APIs, which are represented as external servers like wheather server, map server, land registry server, news server and archive server.
This architecture is called three-tier which allows to implement specific characteristics as maintainability, scalability and reliability. Different hardware machines will be used for each tier.

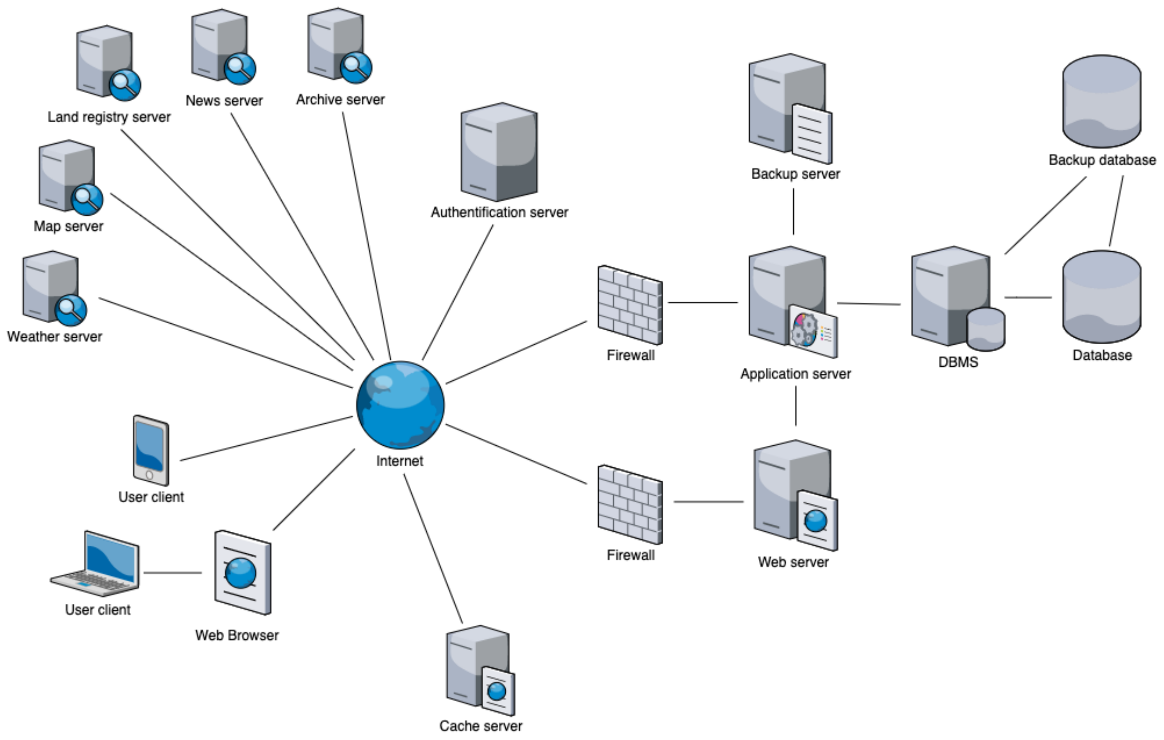


Figure 2.1: high-level architectural overview

2.2. Component view

2.2.1. High level component diagram

The following diagram explain how the main components of the system interact with the interfaces.

In our case we have 2 sides:

1. **Client Side:** It's composed by 2 parts and the user can use both the **Mobile Application** and the **Web Application**.
   The Web Application provides exactly the same behaviour of the Mobile Application (allServices) with an adapted graphic user interface for the browser (allServices + web_interface).
2. **Server Side:** It's composed by 3 parts. The **Main Logic** is the main component where all the computation takes place. It has 2 subsystems, **Farmer Services** and **Policy Maker Services**, which provide specific services.

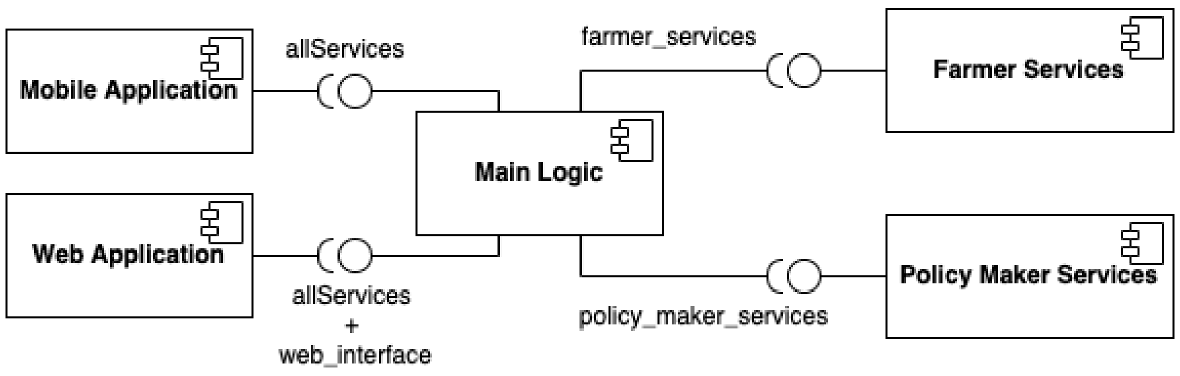


Figure 2.2: high level - component diagram

2.2.2. Main Logic

The following diagram shows how the subsystem Main Logic is built and how its modules communicate with the external ones.
The subsystem has 2 modules and each one provides to the application some interfaces.

1. **WeatherMap Module:** show_wheather, show_forecast, show_map
2. **News Module:** show_news
3. **Account Module:** access_manager
4. **Notification Module:** send_notification

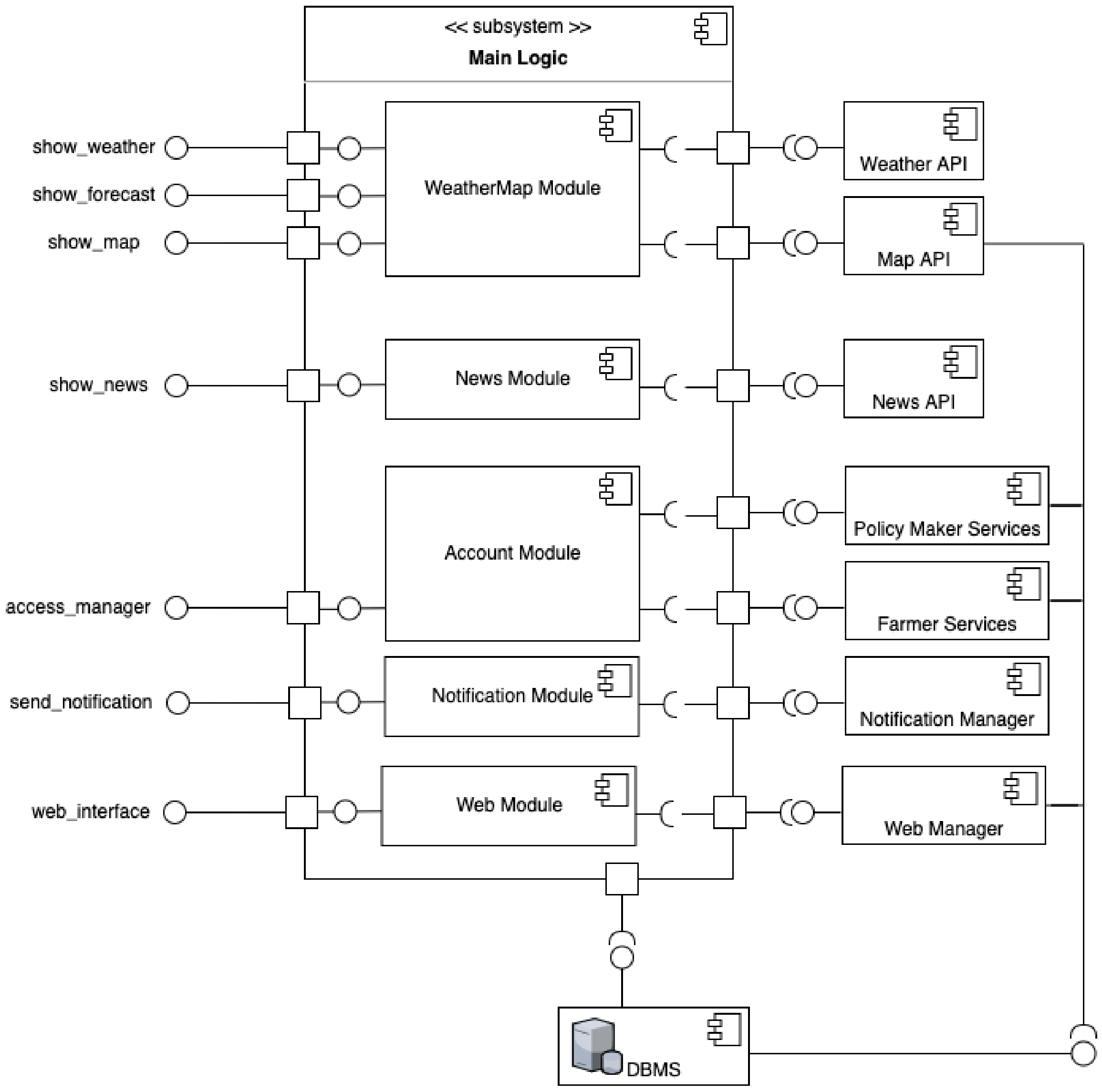


Figure 2.3: main logic - component diagram

2.2.3. Farmers services

The following diagram shows how the subsystem Farmer Mobile Services is built and how its modules communicate with the external ones.
The subsystem has 4 modules, and each one provides to the application some interfaces.

1. **Ticket Module:** send_ticket, read_ticket_answer
2. **Report Module:** send_report, read_answer
3. **Forum Module:** post_on_thread, read_thread, close_thread
4. **Archive Module:** find_in_archive

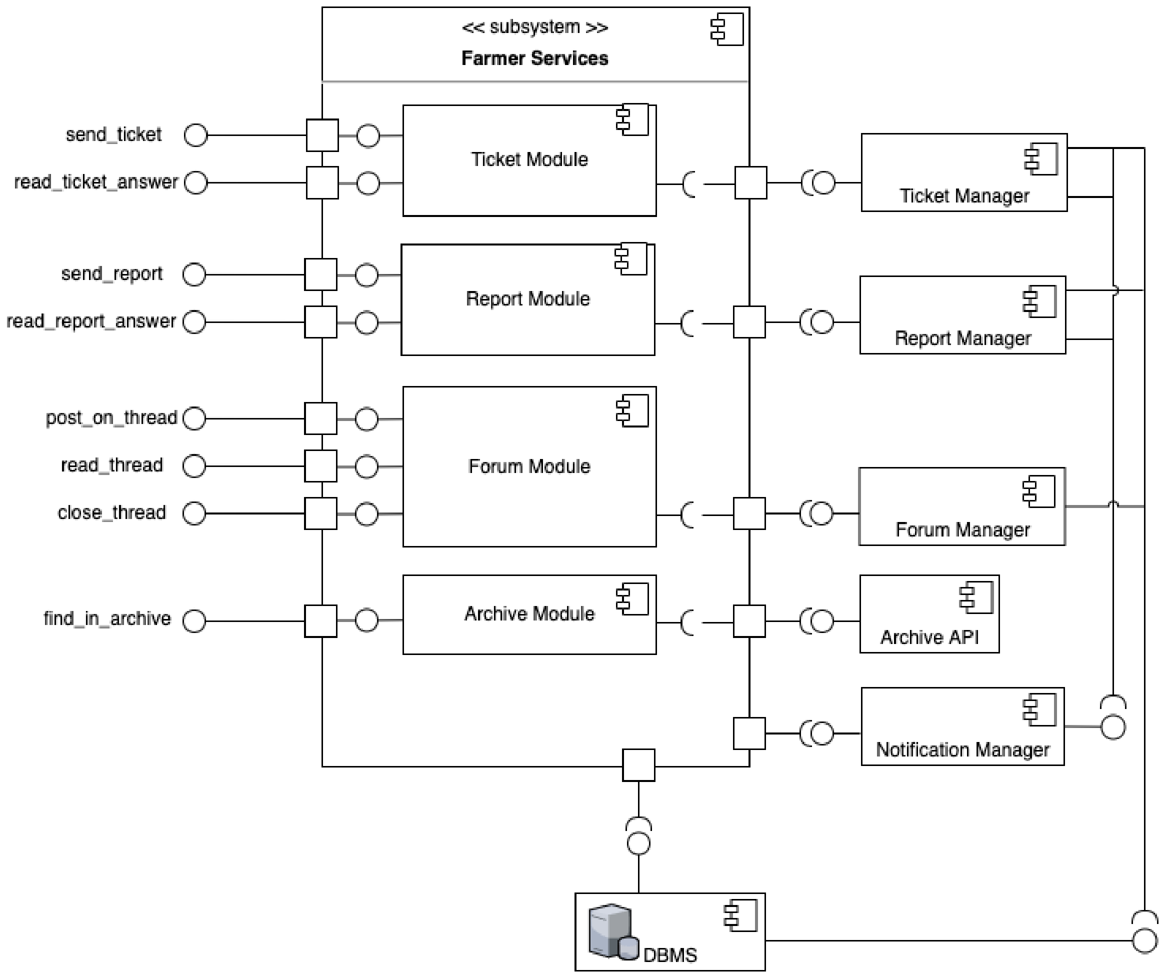


Figure 2.4: farmer services - component diagram

2.2.4. Policy maker services

The following diagram shows how the subsystem Policy Maker Mobile Services is built and how its modules communicate with the external ones.
The subsystem has 5 modules, and each one provides to the application some interfaces.

1. **Ticket Module:** answer_ticket, read_ticket
2. **Report Module:** answer_report, read_report
3. **Contacts Module:** read_contacts
4. **Forum Module:** read_thread
5. **Statistics Module:** aggregate_data, show_data

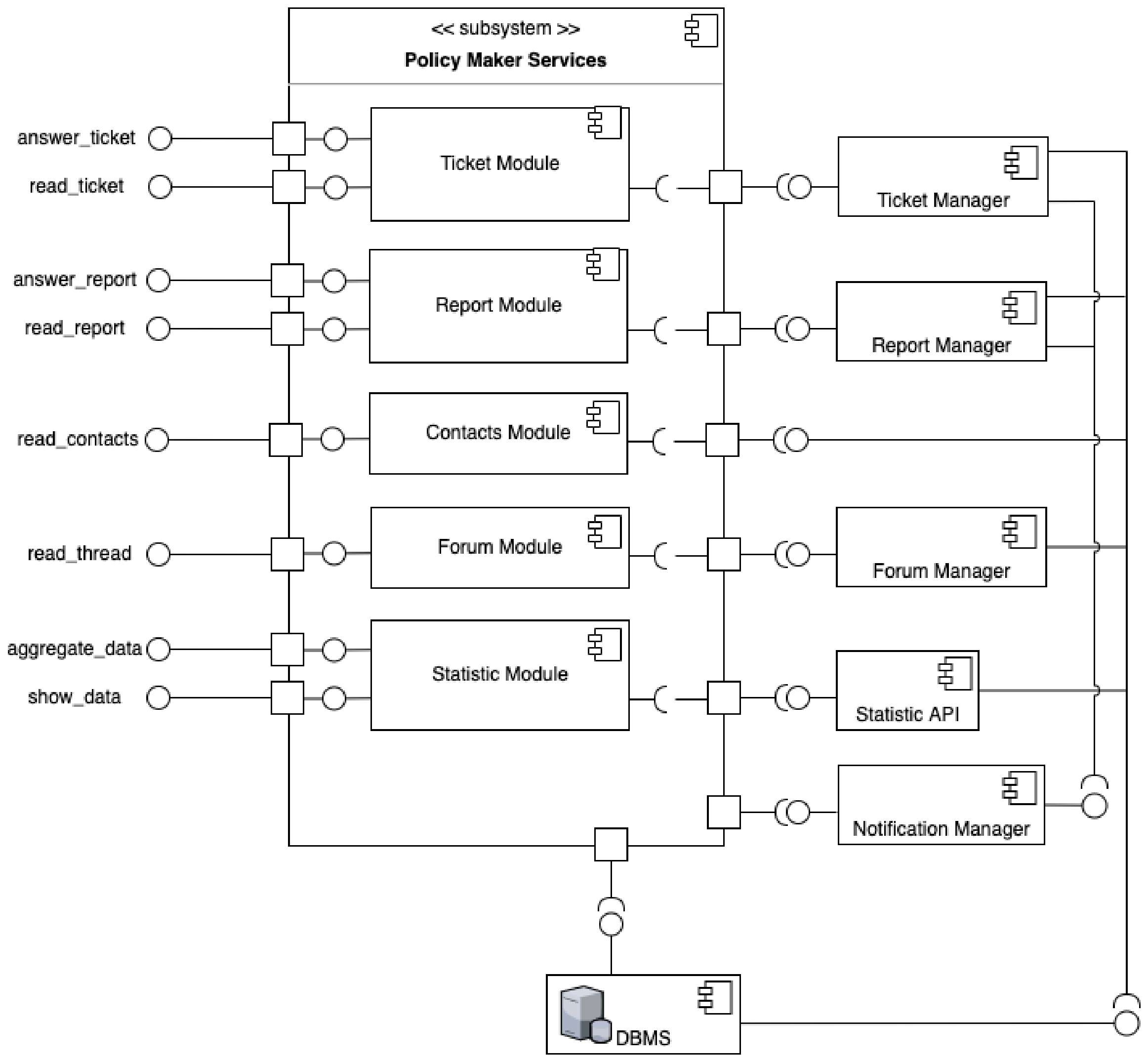


Figure 2.5: policy maker services - component diagram

2.3. Deployment view

The system architecture is divided into 4 tiers using the standard JEE:

1. **EIS Tier:** In this label are collected all the functions which are connected with the database. All the cryptography components are organised here.
   This part is connected with the database and the business tier.
2. **Business Tier:** In this part are collected all the EJB which manage the server-side logic of the dream application.
   This part is connected with the business tier and the web tier.
3. **Web Tier:** In this label are collected all the servlet and the JSP which provide all the interfaces that clients can use.
   This tier is connected with the business tier and with Internet to serve users requests.
4. **Client Tier:** In this tier is possible to find the DREAM application for mobile user and a browser connection point for web users.

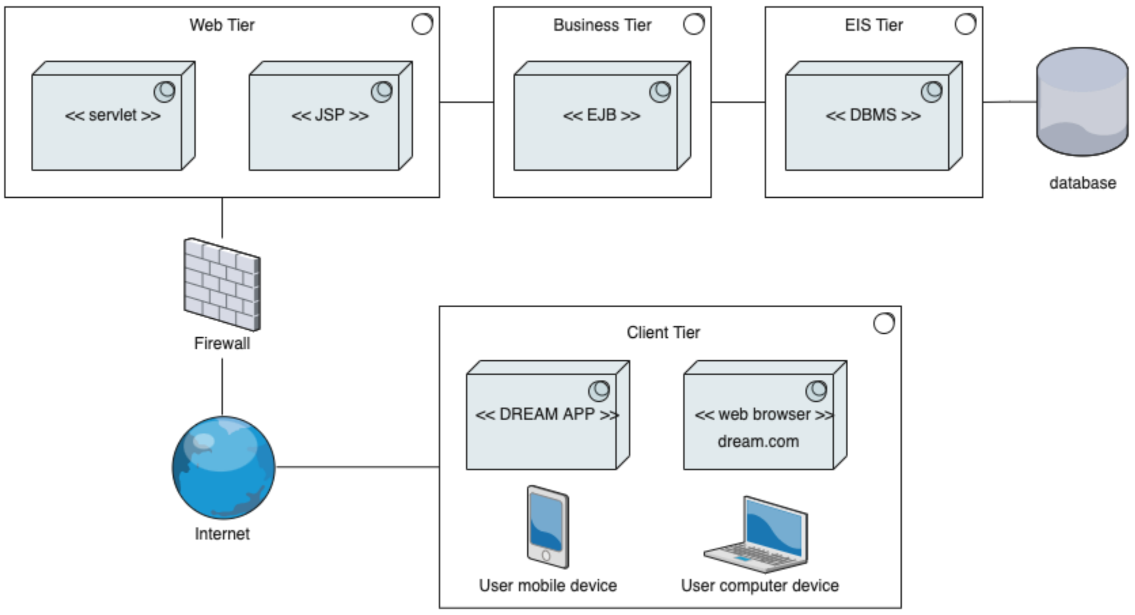


Figure 2.6: high-level deployment overview

2.3.1. Recommended Deployment

All the tiers must be developed as separate elements. This organisation will produce various advantages as easy maintainability because it is possible to work on a single

module without interfering with the others. In this way the deployment can be done faster in separate teams. It will be also possible to upgrade the system in future by adding single elements without modify the general structure, furthermore this approach promote the scalability.
Given that all the tiers are implemented with the JEE indeed all the components are easy and already structured to be integrated with each others.
We recommend to follow the enumeration at the beginning of this section as the order of priority to implement the components of the system. More information about the implementation will be given in the next chapters.

2.4. Runtime view

In the following sequence diagrams we are going to represent the behaviour of the most important components of the DREAM application. The diagrams show just an high level representation of the interaction between the various components. All the other details will be described and implemented during the development process.

2.4.1. Sequence diagrams

In this section some sequence diagrams are presented as an extension of the sequence diagrams already showed in the RASD document. Their purpose is to highlight the internal behaviour of the system, showing the interactions through components and DBMS.

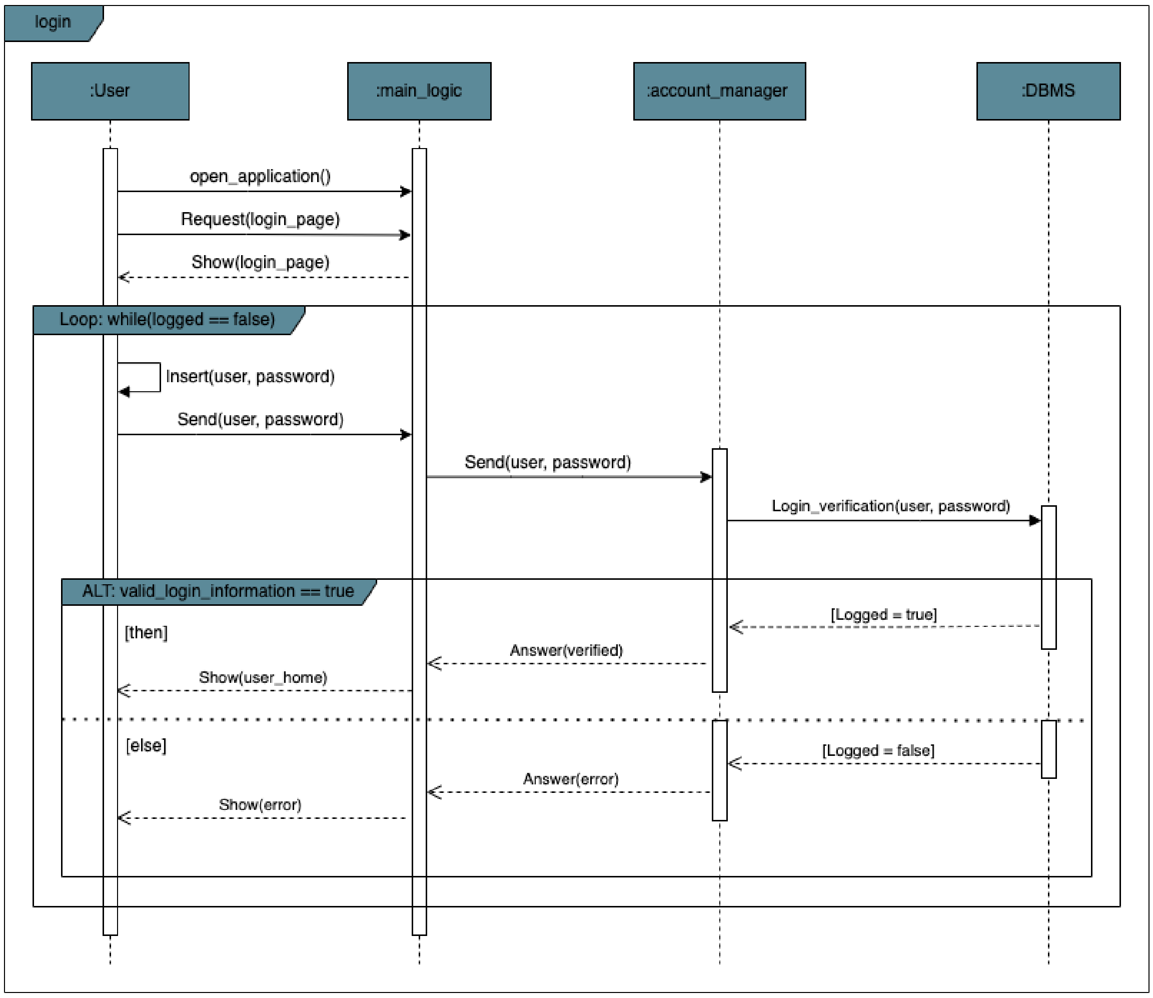


Figure 2.7: login - sequence diagram

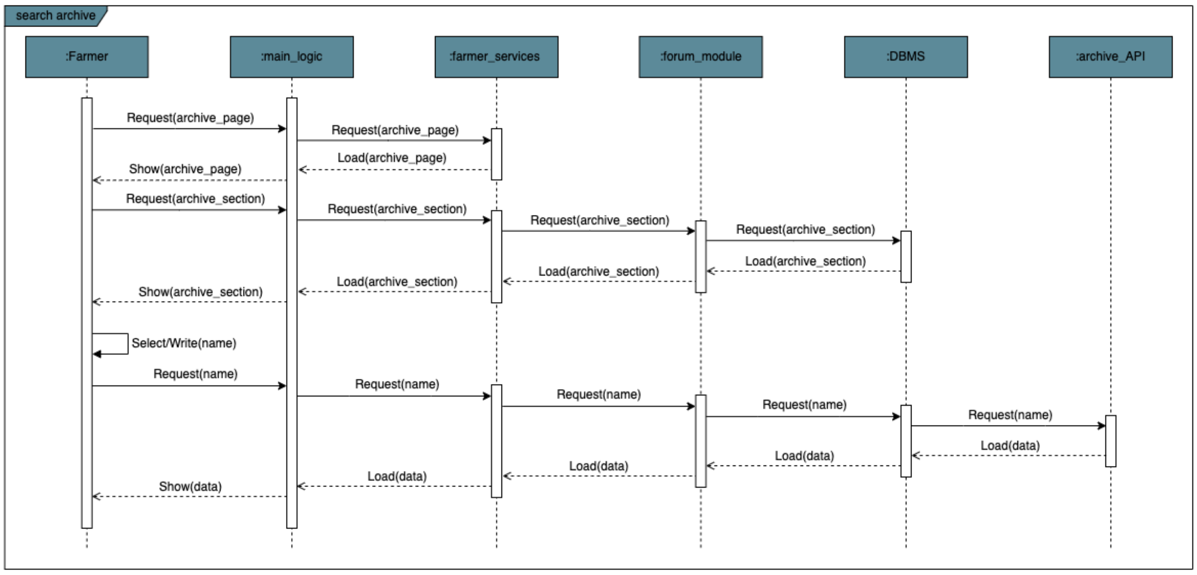


Figure 2.8: search archive - sequence diagram

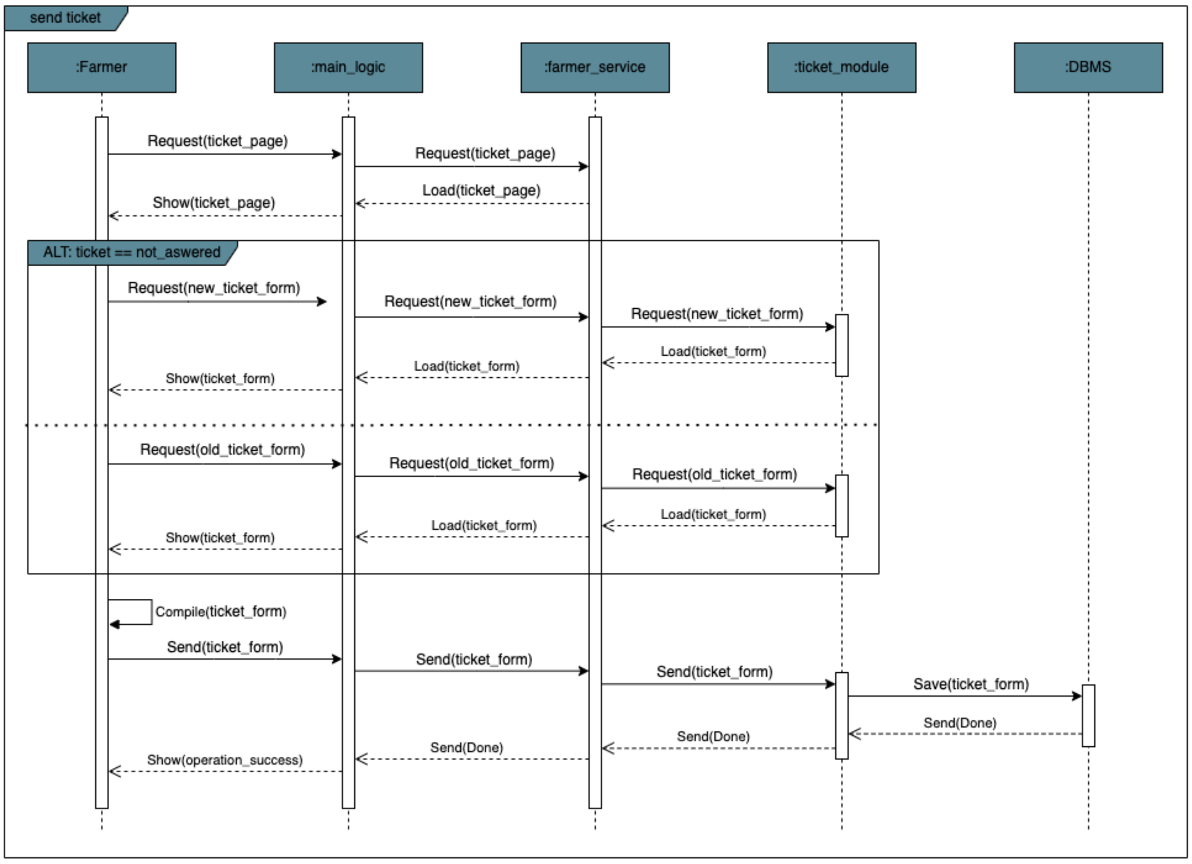


Figure 2.9: send ticket - sequence diagram

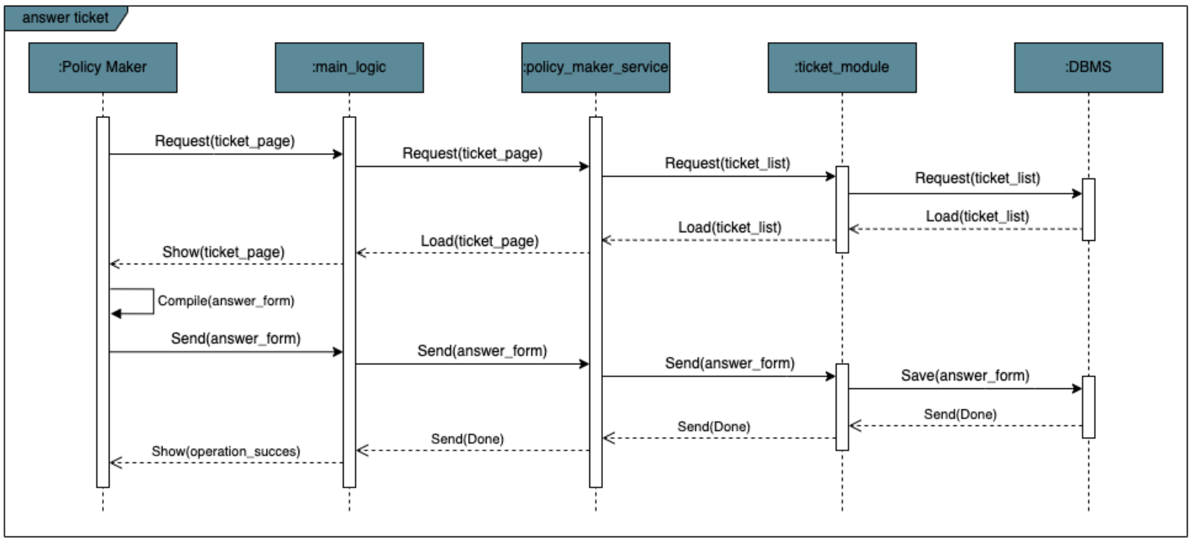


Figure 2.10: answer ticket - sequence diagram

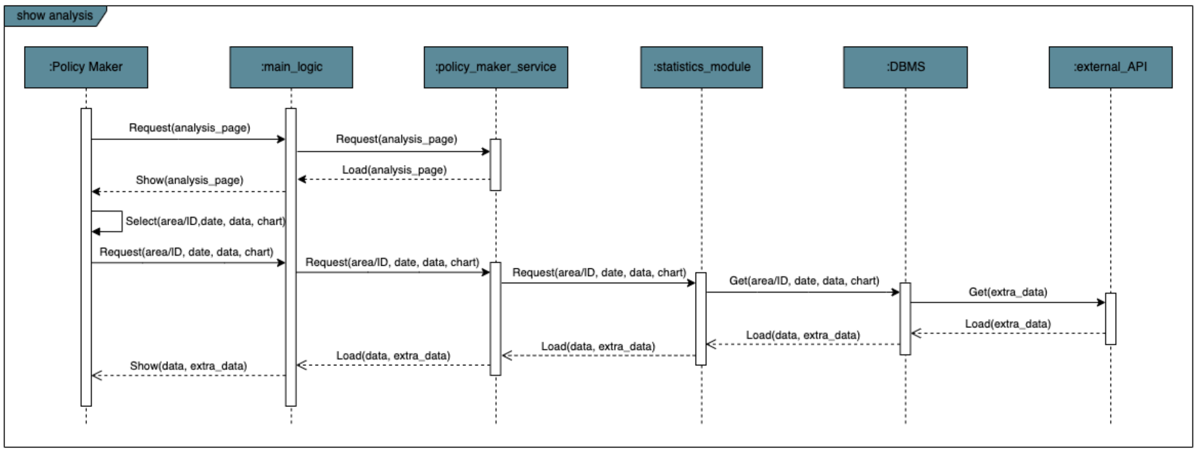


Figure 2.11: show analysis - sequence diagram

2.5. Component interfaces

In this section the interfaces of the various components will be analyzed. There are diagrams for the **Main Logic** and for its subsystems: **Farmer Services** and **Policy Maker Services**.
Each interface contains possible fields and functions which are necessary to provide the desired behaviour of the application.

2.5.1. Main Logic

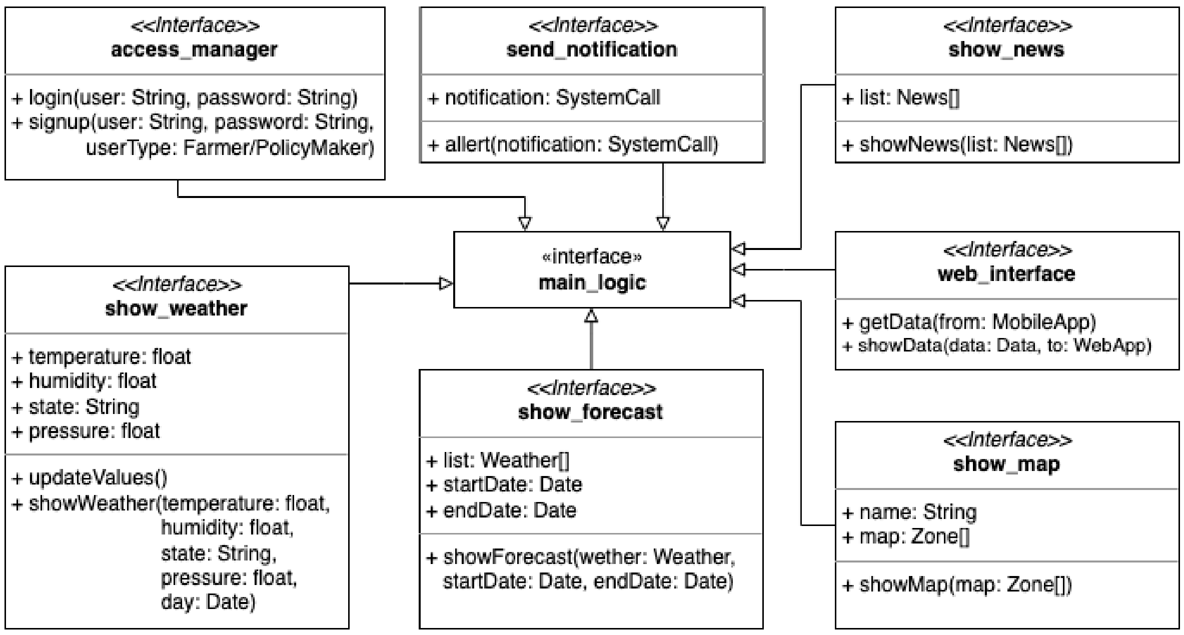


Figure 2.12: main logic - interface diagram

2.5.2. Farmer Services

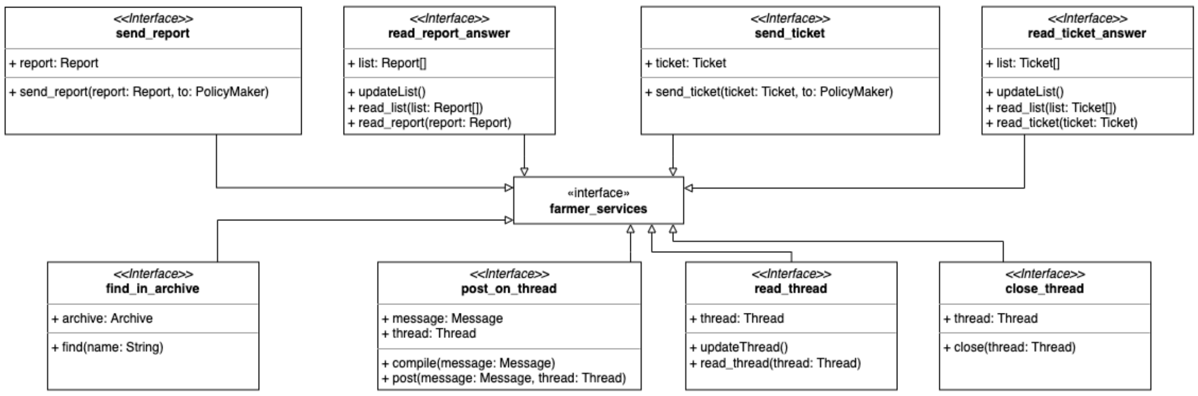

<<Interface>> send_report
+ report: Report
+ send_report(report: Report, to: PolicyMaker)
<<Interface>> read_report_answer
+ list: Report[]
+ updateList()
+ read_list(list: Report[])
+ read_report(report: Report)
<<Interface>> send_ticket
+ ticket: Ticket
+ send_ticket(ticket: Ticket, to: PolicyMaker)
<<Interface>> read_ticket_answer
+ list: Ticket[]
+ updateList()
+ read_list(list: Ticket[])
+ read_ticket(ticket: Ticket)
«interface» farmer_services
<<Interface>> find_in_archive
+ archive: Archive
+ find(name: String)
<<Interface>> post_on_thread
+ message: Message
+ thread: Thread
+ compile(message: Message)
+ post(message: Message, thread: Thread)
<<Interface>> read_thread
+ thread: Thread
+ updateThread()
+ read_thread(thread: Thread)
<<Interface>> close_thread
+ thread: Thread
+ close(thread: Thread)

Figure 2.13: farmer services - interface diagram

2.5.3. Policy Maker Services

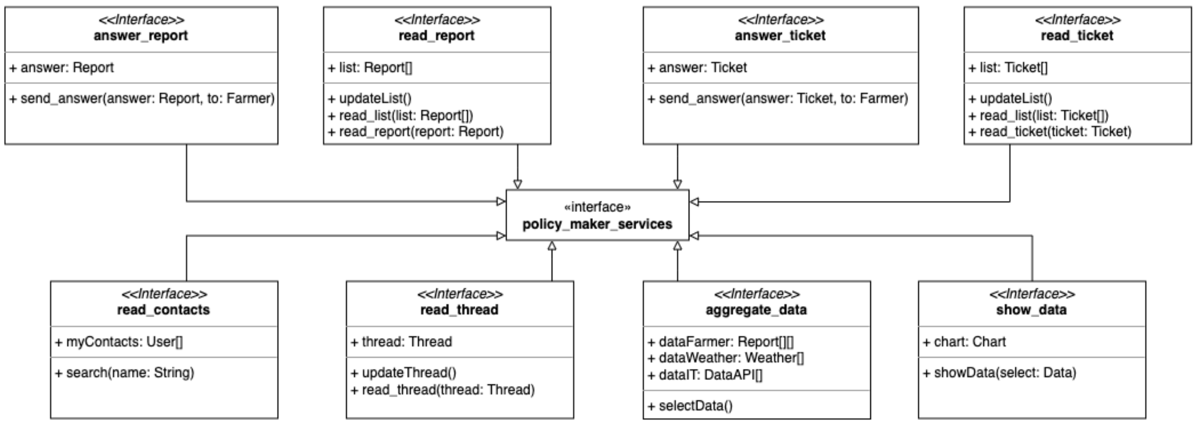

<<Interface>> answer_report
+ answer: Report
+ send_answer(answer: Report, to: Farmer)
<<Interface>> read_report
+ list: Report[]
+ updateList()
+ read_list(list: Report[])
+ read_report(report: Report)
<<Interface>> answer_ticket
+ answer: Ticket
+ send_answer(answer: Ticket, to: Farmer)
<<Interface>> read_ticket
+ list: Ticket[]
+ updateList()
+ read_list(list: Ticket[])
+ read_ticket(ticket: Ticket)
«interface» policy_maker_services
<<Interface>> read_contacts
+ myContacts: User[]
+ search(name: String)
<<Interface>> read_thread
+ thread: Thread
+ updateThread()
+ read_thread(thread: Thread)
<<Interface>> aggregate_data
+ dataFarmer: Report[][]
+ dataWeather: Weather[]
+ dataIT: DataAPI[]
+ selectData()
<<Interface>> show_data
+ chart: Chart
+ showData(select: Data)

Figure 2.14: policy maker services - interface diagram

2.6. Selected architectural styles and patterns

2.6.1. Architectural Style

In the overview section there is a description of how a three-tier client-server architecture has been used to design the system. This choice guarantees to the system an increasing reusability, scalability, security and flexibility.
For the communication between clients (mobile application or web application) and server (main logic) the choice taken is a REST architecture. This architecture gives the possibility to update the server side with all the needed modifications without touching the client side of the system. As a consequence the user experience will be easier because less update on the application will be required.
In order to keep this architecture clean and easy to maintain the system will be structured in modules.

2.6.2. Patterns

Recommended pattern to implement the application:

- **Model-View-Controller Pattern:** It is an architectural pattern which divides the application into three interconnected parts. It is used to separate the how the information are represented inside the system from what the user actually see. It is a really good and easy pattern to use when a system is based on a three tier architecture presented in the previous sections.
- **Facade Pattern:** It simplifies a complicated a complex system by introducing a single interface to a set of interfaces within a subsystem.
- **Factory Pattern:** It simplifies a lot the logic of system because it allows to create Farmers and Policy Makers object by using the User class and to avoid useless redundant parts.
- **Observer Pattern:** It simplifies the communications between objects, notifying changes in the state of other specific objects.

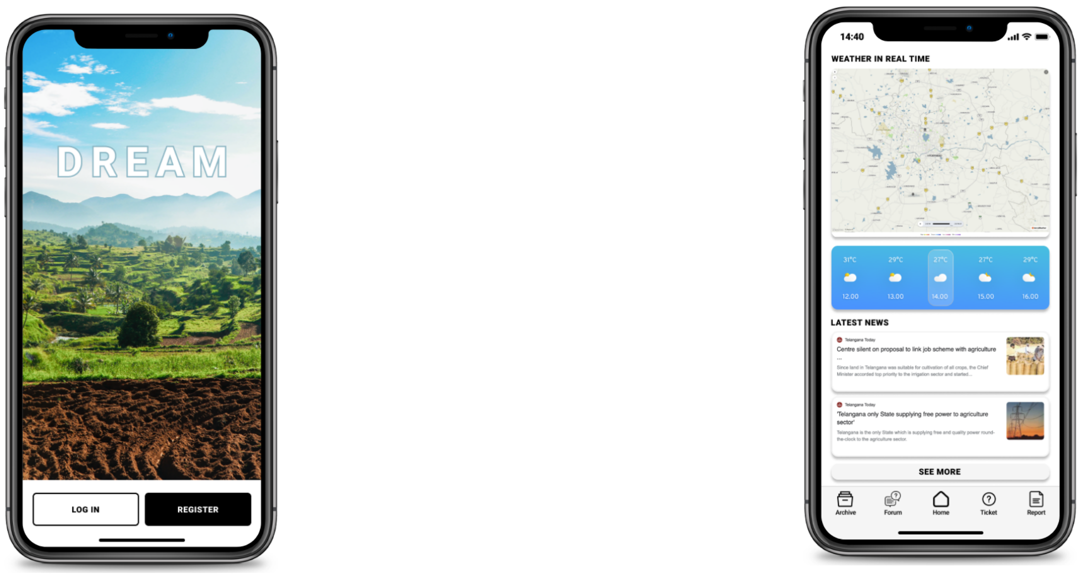
2.7. Other design decisions

- **Geo-location API:** In order to keep track of the position of farms and zones the system needs a geo-location API, in addiction to this it helps to localize the position where it is necessary to make a forecast.

- **Land Registry API:** It is necessary in order to obtain sensible data of every farmer and their land properties.
- **Weather API:** Thanks to this API it is possible to obtain weather information, in addiction to the geo-location API it is possible to understand with enough precision and forewarning when and where a weather event will happen.
- **Chart API:** In order to better organise the data gathered by the policy maker the system use a chart API which let to create different graphs and table to analyze the data.
- **Archive API:** This API let the archive and its inside elements to stay up to date, with the best descriptions possible.
- **News API:** This API let the system to stay up to date with the last information about the weather, the politics and agriculture policies change inside the country.

3 | USER INTERFACE DESIGN

In addiction to the view presented in the RASD, in this part there are other representations of additional mockups which will give a complete overview of how the application must look like.



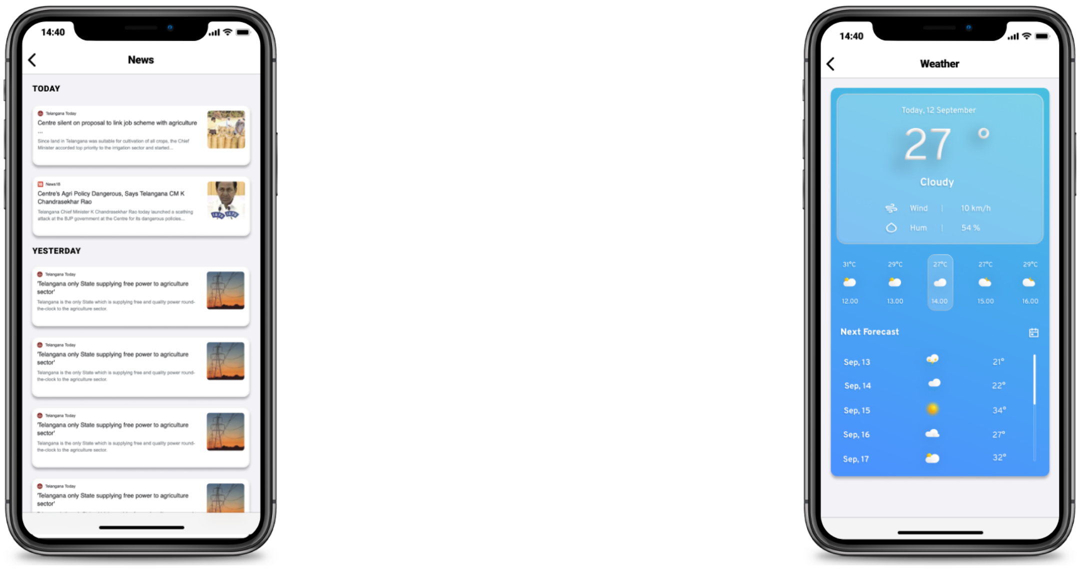
Centre silent on proposal to link job scheme with agriculture ...

Since land in Telangana was suitable for cultivation of all crops, the Chief Minister accorded top priority to the irrigation sector and started...

'Telangana only State supplying free power to agriculture sector'

Telangana is the only State which is supplying free and quality power round-the-clock to the agriculture sector.

Figure 3.1: Mockup: on the left user's log out, on the right home page


14:40

News

TODAY

Telangana Today
Centre silent on proposal to link job scheme with agriculture
...
Since land in Telangana was suitable for cultivation of all crops, the Chief Minister accorded top priority to the irrigation sector and started...

News18
Centre's Agri Policy Dangerous, Says Telangana CM K Chandrasekhar Rao
Telangana Chief Minister K Chandrasekhar Rao today launched a scathing attack at the BJP government at the Centre for its dangerous policies...

YESTERDAY

Telangana Today
'Telangana only State supplying free power to agriculture sector'
Telangana is the only State which is supplying free and quality power round-the-clock to the agriculture sector.

Telangana Today
'Telangana only State supplying free power to agriculture sector'
Telangana is the only State which is supplying free and quality power round-the-clock to the agriculture sector.

Telangana Today
'Telangana only State supplying free power to agriculture sector'
Telangana is the only State which is supplying free and quality power round-the-clock to the agriculture sector.

Telangana Today
'Telangana only State supplying free power to agriculture sector'

14:40

Weather

Today, 12 September
27 °
Cloudy
Wind | 10 km/h
Hum | 54 %

31°C	29°C	27°C	27°C	29°C
12.00	13.00	14.00	15.00	16.00

Next Forecast

Sep, 13	21°
Sep, 14	22°
Sep, 15	34°
Sep, 16	27°
Sep, 17	32°

Figure 3.2: Mockup: on the left News page, on the right Weather page

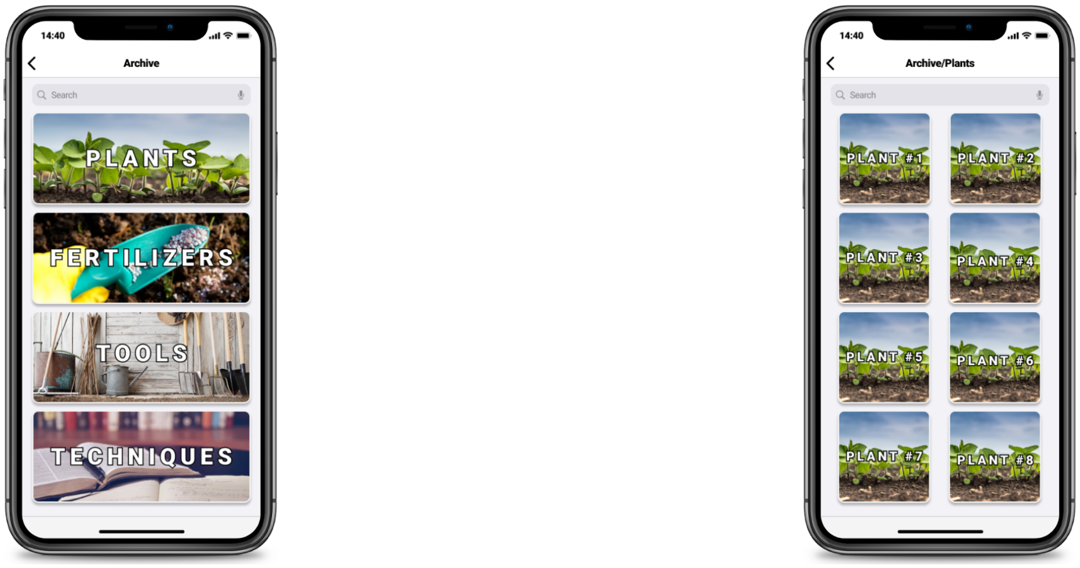
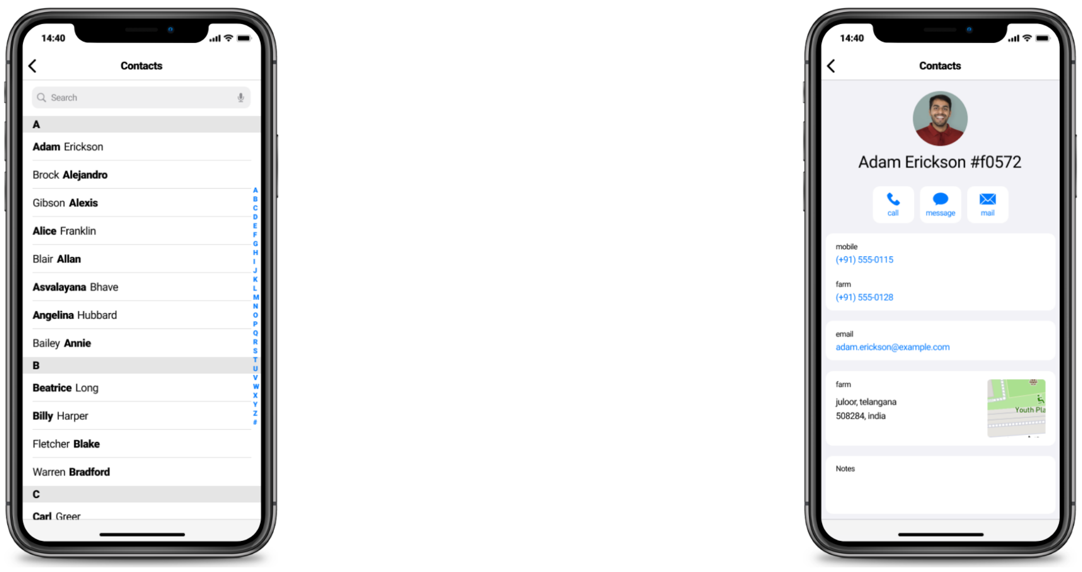


Figure 3.3: Mockup: on the left Archive page, on the right Archive/Plants page


14:40

Contacts

Search

A

Adam Erickson

Brock **Alejandro**

Gibson **Alexis**

Alice Franklin

Blair **Allan**

Asvalayana Bhave

Angelina Hubbard

Bailey **Annie**

B

Beatrice Long

Billy Harper

Fletcher **Blake**

Warren **Bradford**

C

Carl Greer

14:40

Contacts

Adam Erickson #f0572

mobile
(+91) 555-0115

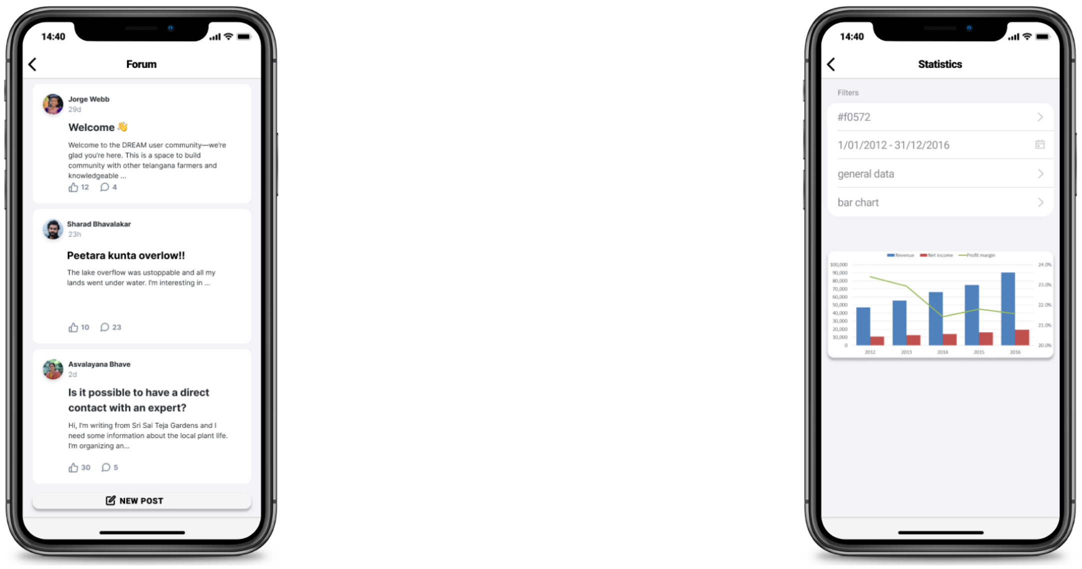
farm
(+91) 555-0128

email
adam.erickson@example.com

farm
juloor, telangana
508284, india

Notes

Figure 3.4: Mockup: on the left Contacts page, on the right page of a single contact


14:40

Forum

Jorge Webb
29d

Welcome 👋

Welcome to the DREAM user community—we're glad you're here. This is a space to build community with other telangana farmers and knowledgeable ...

12 4

Sharad Bhavalakar
23h

Peetara kunta overlow!!

The lake overflow was ustoppable and all my lands went under water. I'm interesting in ...

10 23

Asvalayana Bhave
2d

Is it possible to have a direct contact with an expert?

Hi, I'm writing from Sri Sai Teja Gardens and I need some information about the local plant life. I'm organizing an...

30 5

NEW POST

14:40

Statistics

Filters

#f0572

1/01/2012 - 31/12/2016

general data

bar chart

Figure 3.5: Mockup: on the left Forum page, on the right Statistics page

These last two mockups show how the same page will be presented in the mobile application and in the web application. As before described the functionalities remain the same, but the GUI is adapted for a better web user experience.
There are not other web pages mockups because these pages will follow the same structure.

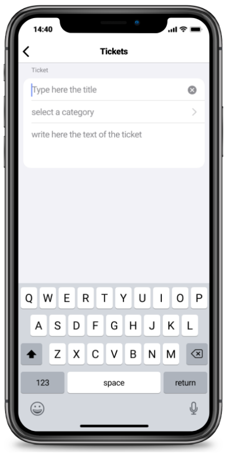


Figure 3.6: Mockup: Ticket page

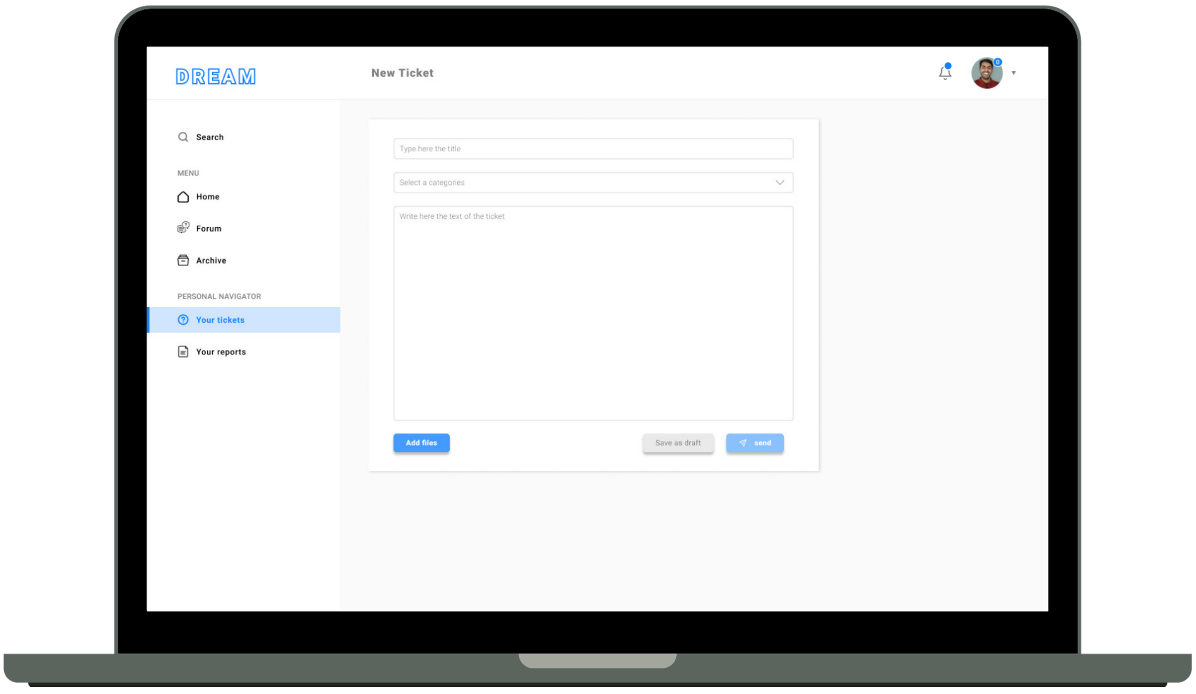


Figure 3.7: Mockup: Web application Ticket page

4 | REQUIREMENTS TRACEABILITY

In this chapter there is a mapping between the requirements specified in the RASD document and the design elements specified in the second chapter of this document.

G1	**Allow farmers to easily check weather condition**
R1	The system shall allow an unregistered farmers to register.
R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.
R7	The system shall allow the farmer to check the weather conditions of the entire week
R8	The system shall allow the farmer to visualize the weather characteristics like the humidity , the temperature and the probability of precipitations in the current day.
R9	The system shall allow the farmer to visualize the general weather map of Telengana.
R10	The system shall allow the farmer to visualize the weather map of his/her farm.
	Main Logic: • WeatherMap Module • Account Module • Web Module • Notification Module

	Weather API
	Map API
	Notification Manager

G2	**Allow farmers to have technical and personalized advices from other farmers or experts of the field.**
R1	The system shall allow an unregistered farmers to register.
R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.
R7	The system shall allow the farmer to check the weather conditions of the entire week.
R8	The system shall allow the farmer to visualize the weather characteristics like the humidity , the temperature and the probability of precipitations in the current day.
R10	The system shall allow the farmer to visualize the weather map of his/her farm.
R11	The system shall allow the farmer to visualize the news.
R12	The system shall allow the farmer to visualize the archive.
R13	The system shall allow the farmer to search inside the archive.
R14	The system shall allow the farmer to visualize the results of the research in the archive.
R20	The system shall allow the farmer to visualize the forum.
R21	The system shall allow the farmer to open a thread in the forum.
R22	The system shall allow the farmer to read a thread in the forum.
R23	The system shall allow the farmer to create a new thread in the forum.

R24	The system shall allow the farmer to answer to a thread in the forum.
R25	The system shall allow the farmer to delete a thread in the forum.
R26	The system shall allow the farmer to read a thread in the forum.
	Main Logic: • WeatherMap Module • Account Module • Web Module • News Module • Notification Module
	Farmer Services: • Archive Module • Forum Module
	Weather API
	Map API
	News API
	Archive API
	Forum Manager
	Notification Manager

G3	**Allow farmer to ask for help to the government.**
R1	The system shall allow an unregistered farmers to register.
R2	The system shall allow an unregistered policy makers to register.
R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.

R6	The system shall send a confirmation email to a policy maker who finishes the registration process in order to confirm the en-tire. procedure.
R15	The system shall allow the farmer to send a ticket request.
R16	The system shall allow the farmer to get notified by an answer of his/her ticket.
R17	The system shall allow the farmer to compile a report.
R18	The system shall allow the farmer to send a report to a policymaker.
R19	The system shall allow the farmer to get notified for a report answer.
R32	The system shall allow the policy maker to get notified when he/she gets a ticket.
R33	The system shall allow the policy maker to answer to a ticket request.
R34	The system shall allow the policy maker to get notified when he/she gets a report from a farmer.
R35	The system shall allow the policy maker to write an answer to a report.
R39	The system shall allow the policy maker to visualize data.
R40	The system shall allow the policy maker to aggregate and disaggregate data.
R41	The system shall allow the policy maker to change the visualization of data.
R42	The system shall allow the policy maker to visualize contacts.
	Main Logic: • Account Module • Web Module • Notification Module

	Farmer Services: • Ticket Module • Report Module
	Policy Maker Services: • Ticket Module • Report Module • Statistic Module • Contacts Module
	Web Manager
	Notification Manager
	Ticket Manager
	Report Manager
	Statistic API

G4	**Improve the communication between farmers through a forum.**
R1	The system shall allow an unregistered farmers to register.
R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.
R20	The system shall allow the farmer to visualize the forum.
R21	The system shall allow the farmer to open a thread in the forum.
R22	The system shall allow the farmer to read a thread in the forum.
R23	The system shall allow the farmer to create a new thread in the forum.
R24	The system shall allow the farmer to answer to a thread in the forum.

R25	The system shall allow the farmer to delete a thread in the forum.
R26	The system shall allow the farmer to read a thread in the forum.
	Main Logic: • Account Module • Web Module • Notification Module
	Farmer Services: • Forum Module
	Web Manager
	Forum Manager
	Notification Manager

G5	**Improve data communication between farmers and government.**
R1	The system shall allow an unregistered farmers to register.
R2	The system shall allow an unregistered policy makers to register.
R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.
R6	The system shall send a confirmation email to a policy maker who finishes the registration process in order to confirm the en-tire. procedure.
R11	The system shall allow the farmer to visualize the news.
R15	The system shall allow the farmer to send a ticket request.
R16	The system shall allow the farmer to get notified by an answer of his/her ticket.

R17	The system shall allow the farmer to compile a report.
R18	The system shall allow the farmer to send a report to a policymaker.
R19	The system shall allow the farmer to get notified for a report answer.
R31	The system shall allow the policy maker to visualize the news.
R32	The system shall allow the policy maker to get notified when he/she gets a ticket.
R33	The system shall allow the policy maker to answer to a ticket request.
R34	The system shall allow the policy maker to get notified when he/she gets a report from a farmer.
R35	The system shall allow the policy maker to write an answer to a report.
R36	The system shall allow the policy maker to visualize the forum.
R37	The system shall allow the policy maker to read a thread in the forum.
R38	The system shall allow the policy maker to read a thread in the forum.
R39	The system shall allow the policy maker to visualize data.
R40	The system shall allow the policy maker to aggregate and disaggregate data.
R41	The system shall allow the policy maker to change the visualization of data.
R42	The system shall allow the policy maker to visualize contacts.

	Main Logic: • Account Module • Web Module • News Module • Notification Module
	Farmer Services: • Ticket Module • Report Module
	Policy Maker Services: • Ticket Module • Report Module • Forum Module • Statistic Module • Contacts Module
	Web Manager
	News API
	Ticket Manager
	Notification Manager
	Report Manager
	Forum Manager
	Statistic API

G6	**Allow the government to improve the analysis and the sharing of important data concerning agriculture.**
R1	The system shall allow an unregistered farmers to register.
R2	The system shall allow an unregistered policy makers to register.

R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.
R6	The system shall send a confirmation email to a policy maker who finishes the registration process in order to confirm the en-tire. procedure.
R15	The system shall allow the farmer to send a ticket request.
R17	The system shall allow the farmer to compile a report.
R18	The system shall allow the farmer to send a report to a policymaker.
R23	The system shall allow the farmer to create a new thread in the forum.
R24	The system shall allow the farmer to answer to a thread in the forum.
R27	The system shall allow the policy maker to check the weather conditions of the entire week.
R28	The system shall allow the policy maker to visualize the weather characteristics like the humidity , the temperature and the probability of precipitations in the current day.
R29	The system shall allow the policy maker to visualize the general weather map of Telengana.
R30	The system shall allow the policy maker to visualize the weather of the area he/she checking.
R31	The system shall allow the policy maker to visualize the news.
R32	The system shall allow the policy maker to get notified when he/she gets a ticket.
R33	The system shall allow the policy maker to answer to a ticket request.
R34	The system shall allow the policy maker to get notified when he/she gets a report from a farmer.

R35	The system shall allow the policy maker to write an answer to a report.
R36	The system shall allow the policy maker to visualize the forum.
R37	The system shall allow the policy maker to read a thread in the forum.
R38	The system shall allow the policy maker to read a thread in the forum.
R39	The system shall allow the policy maker to visualize data.
R40	The system shall allow the policy maker to aggregate and disaggregate data.
R41	The system shall allow the policy maker to change the visualization of data.
	Main Logic: • Account Module • Web Module • WeatherMap Module • News Module • Notification Module
	Farmer Services: • Ticket Module • Report Module • Forum Module
	Policy Maker Services: • Ticket Module • Report Module • Forum Module • Statistic Module

	Web Manager
	Ticket Manager
	Report Manager
	Weather API
	Map API
	News API
	Notification Manager
	Forum Manager
	Statistic API

G7	**Allow the government to have specific data about farmers.**
R1	The system shall allow an unregistered farmers to register.
R2	The system shall allow an unregistered policy makers to register.
R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.
R6	The system shall send a confirmation email to a policy maker who finishes the registration process in order to confirm the en-tire. procedure.
R15	The system shall allow the farmer to send a ticket request.
R17	The system shall allow the farmer to compile a report.
R18	The system shall allow the farmer to send a report to a policymaker.
R32	The system shall allow the policy maker to get notified when he/she gets a ticket.
R33	The system shall allow the policy maker to answer to a ticket request.

R34	The system shall allow the policy maker to get notified when he/she gets a report from a farmer.
R35	The system shall allow the policy maker to write an answer to a report.
R39	The system shall allow the policy maker to visualize data.
R40	The system shall allow the policy maker to aggregate and disaggregate data.
R41	The system shall allow the policy maker to change the visualization of data.
R42	The system shall allow the policy maker to visualize contacts.
	Main Logic: • Account Module • Web Module • Notification Module
	Farmer Services: • Ticket Module • Report Module
	Policy Maker Services: • Ticket Module • Report Module • Statistic Module • Contact Module
	Web Manager
	Ticket Manager
	Report Manager
	Notification Manager

	Statistic API

G8	**Allow the policy makers to easily recognise critical and virtuous situations.**
R1	The system shall allow an unregistered farmers to register.
R2	The system shall allow an unregistered policy makers to register.
R5	The system shall send a confirmation email to a farmer who finishes the registration process in order to confirm the entire procedure.
R6	The system shall send a confirmation email to a policy maker who finishes the registration process in order to confirm the en-tire. pro-cedure.
R15	The system shall allow the farmer to send a ticket request.
R17	The system shall allow the farmer to compile a report.
R18	The system shall allow the farmer to send a report to a policymaker.
R32	The system shall allow the policy maker to get notified when he/she gets a ticket.
R33	The system shall allow the policy maker to answer to a ticket request.
R34	The system shall allow the policy maker to get notified when he/she gets a report from a farmer.
R35	The system shall allow the policy maker to write an answer to a report.
R39	The system shall allow the policy maker to visualize data.
R40	The system shall allow the policy maker to aggregate and disaggregate data.
R41	The system shall allow the policy maker to change the visualization of data.

	Main Logic: • Account Module • Web Module • Notification Module
	Farmer Services: • Ticket Module • Report Module
	Policy Maker Services: • Ticket Module • Report Module • Statistic Module
	Web Manager
	Ticket Manager
	Report Manager
	Notification Manager
	Statistic API

5 | IMPLEMENTATION, INTEGRATION AND TEST PLAN

In this section we specify the order of implementation of each component with a Gantt diagram. It also allows to understand how each component communicates with the others. Furthermore it highlightes the general hierarchy of components and sub-components in our system.
These sequences have to be respected also as testing phases, in order to avoid a "big bang" scenario.

5.1. Implementation and Component Integration

The chronological sequence of components' implementation are illustrated in this Gantt diagram with a relative time scale and not an absolute one. That highlights the sequence of component implementation and the critical paths but not the real effort which need to be spent on each part. If the forecast of time or resources spent are important, it's suggested to prepare a risk plan and a 3 world case scenario (optimal - normal - worst).

Details of the diagram:

- The DBMS component is the first to be implemented because the majority of the other components has have to communicate with the database.
- The Ticket Manager, Report Manager and Forum Manager are showed twice because the second group has to be intended as an alias of the first one. This has been made just for readability purposes. Instead Ticket Module, Report Module and Forum Module are represented twice in Policy Maker Services and in Farmer Services, but they implements different interfaces, as showed in previous chapters.
- The Web Module, which provides the web interface of the application, has to be the last component implemented in the main logic.

- Account, WeatherMap, News and Notification Module can be implemented and integrated in the system in parallel because they interact with each other but their behaviour is independent from other components. This planning gives the possibility to work in parallel with different development teams.
  The same reasoning it is used to Policy Maker services and Farmer Services, with the difference that they can be seen as sub components of Account Module.

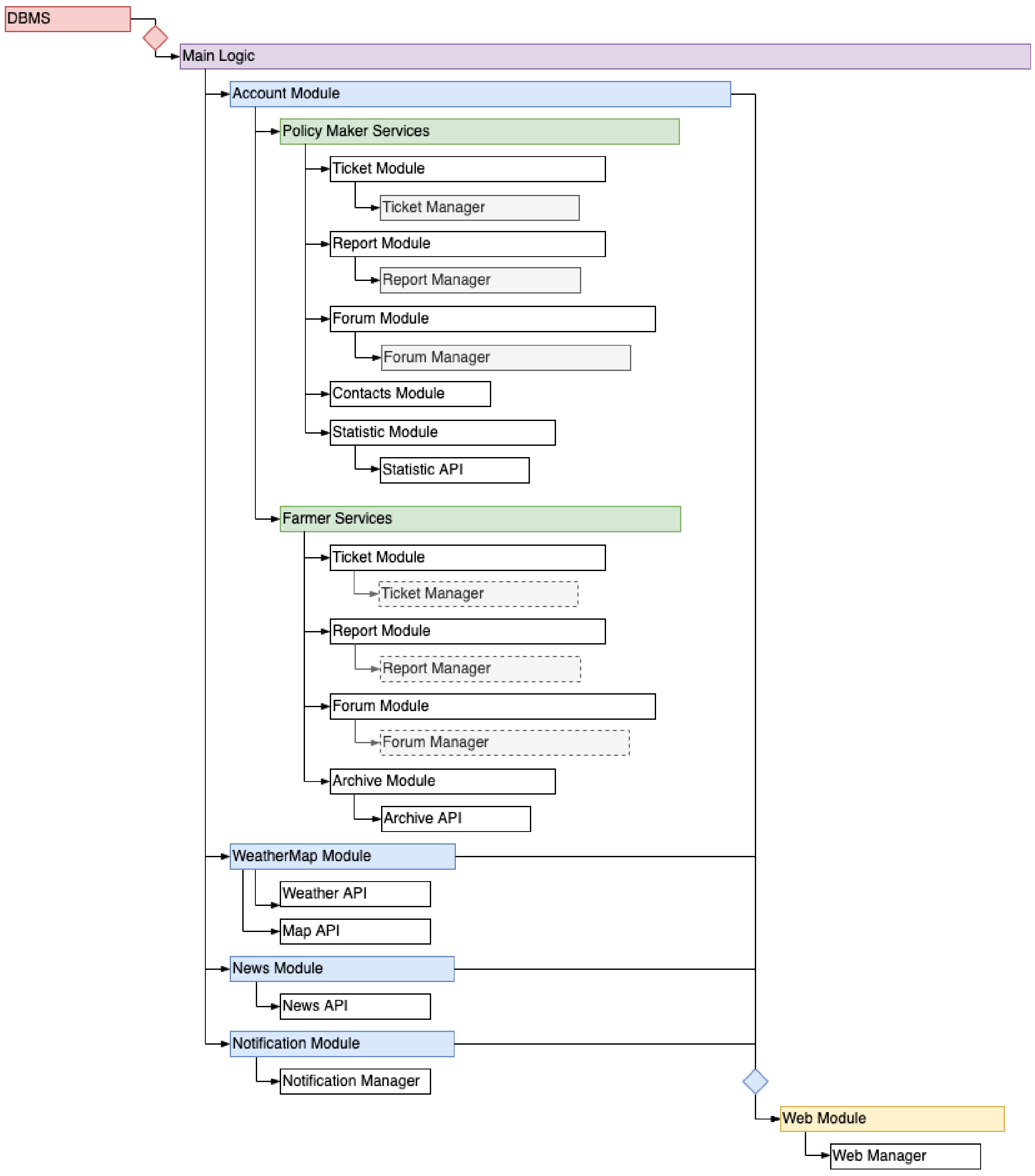


Figure 5.1: Gantt diagram

5.2. System Testing

Once the system is implemented and all of its components are implemented and tested, it must be tested as a unique entity. it should respects all the requirements and specification which are present inside the RASD document. If this phase has come means that the project is almost at the end. The test part should not be done only by developers but also by stakeholder and shareholders. We can split the testing phase in different parts:

- **Functional testing:** To make sure that all the functionalities of the project are working according to the requirements.
- **Performance testing:** To make sure that there aren't any bottlenecks which affects the operation time of the functionalities of the system.
- **Load testing:** To make sure to avoid bugs as memory leaks or buffer overflow. It highlights also the upper limits of the components. There is also the possibility to test the system in order to understand how long it can resist with the maximum of the load.
- **Stress testing:** To make sure that the system recovers carefully from a failure.
- **Network testing:** To make sure that the system can maintain its behaviour in case of a down's node.

6 | EFFORT SPENT

The time here reported is an estimation.

Student 1: Brunello Simone

Topic	Hours
General Reasoning	9
Purpose	1
Scope	2
Overview	3
Component View	4
Deployment View	5
Component Interfaces	3
Selected Architectural Style And Patterns	3
Requirements Traceability	7
Implementation, Integration and Test Plan	3
Revision	2

Student 2: Nicolis Nicholas

Topic	Hours
General Reasoning	9
Purpose	1
Scope	2
Overview	3
Component View	5
Deployment View	5
Component Interfaces	3
Selected Architectural Style And Patterns	3
User Interface Design	7
Implementation, Integration and Test Plan	3
Revision	2

7 | REFERENCES

7.1. Reference Documents

- Specification document: Assignment RDD A.Y. 2021-2022
- Course slides
- wikipedia

7.2. Reference Sites

- https://www.accuweather.com/
- https://www.diagram.net/
- https://www.canva.com/
- https://www.figma.com/
- https://www.overleaf.com/