Politecnico di Milano

Computer Science and Engineering

Implementation and Test Deliverable

DREAM

Software Engineering 2

Optional Project AY 2021-2022

Curated by: Francesco Mazzola and Alessio Ferrara

1 Introduction 3
1.1 Scope 3
1.2 Definitions, acronyms and abbreviations 3
1.2.1 Definitions 3
1.2.2 Acronyms 4
1.3 Reference documents 4
1.4 Document Structure 4
2 Product functions 5
2.1 Implemented functions 5
2.1.1 Evaluate Farmer 5
2.1.2 Assign Farmer to a Steering Initiative 5
2.1.3 Evaluate Steering Initiative 5
The Policy Maker can indeed evaluate the production rates of a specific steering initiative and to choose if the initiative has produced noteworthy results or not. 6
2.2 Discarded functions 6
2.2.1 Inserting Production 6
2.2.2 Checking personalized informations 6
2.2.3 Creating a new discussion 6
2.2.4 View an open discussion 6
2.2.5 Commenting an open discussion 6
3 System development frameworks 6
3.1 Adopted frameworks 6
3.1 Adopted programming languages 6
3.1.1 JavaScript 6
3.1.2 HTML, CSS 7
3.1.3 SQL 7
3.2 Adopted middlewares 7
3.2.1 Other Libraries 7
3.2 Adopted APIs 8
4 Source code structure 8
4.1 Back-end 8
4.1 Front-end 8
5 Testing activity 9
5.1 Testing environment 9
5.2 Test cases 9
5.2.1 Account Routes 9

5.2.1 Evaluate Routes 9
5.2.1 Evaluate Routes 10
5.2.1 Middlewares 10
6 Installation guide 11
7 Effort spent 11
7.2 Ferrara Alessio 11
7.2 Mazzola Francesco 11

1 Introduction

1.1 Scope

The purpose of this document is to present the work done during the implementation of the first prototype of this application.
This document presents a summary of the functions that are available in the current version of the software and an overview of the languages and frameworks used during the development.
Finally, it outlines the tests that have been carried out and presents a simple installation guide. Together with the RASD and the DD, this document has the purpose to inform about the realization of the DREAM software.

1.2 Definitions, acronyms and abbreviations

1.2.1 Definitions

Definition	Description
Type of Production	**The food that the farmer is producing**
Good Farmer	**A farmer that is performing particularly well**
Steering initiative	**A set of instructions that are provided by an Agronomist that is assigned to a farmer that is performing badly**
Agronomist	**A person that provides useful practices to those farmers that are having problems**

Discussion Forum	**A post in which a farmer asks for help or provides information on some problems to other farmers**

1.2.2 Acronyms

Acronyms	Description
RASD	**Requirement Analysis and Specification Document**
DD	**Design Document**
ITD	**Implementation and Testing Deliverable**

1.3 Reference documents

During the acceptance test the following documents were considered:

- Requirement Analysis and Specification Document – Dream, version "1.1.1" (referenced as "RASD")
- Design Document – Dream, version "1.0.1" (referenced as "DD")
- Implementation and Test Derivable - Dream, version "1" (referenced as "ITD")
- Assignment RDD AY 2021-2022 ("Requirement Engineering and Design Project: goal, schedule, and rules")
- Assignment IT AY 2021-2022 (" I&T assignment goal, schedule, and rules")

1.4 Document Structure

- 1 Introduction
    - The first section introduces the purpose of the document and the scope DREAM. Included here is a glossary with the definitions, acronyms and abbreviations used in this document.
- 2 Product functions
    - This section is a reference to the previous documents. Here are presented the functions that are implemented in the software and those that are not (together with the motivation for excluding them).
- 3 System development frameworks
    - In this section are explained the adopted development framework which also includes an overview of the adopted programming languages, middleware and APIs.
- 4 Source code and structure
    - This section contains the structure of the source code, with some information about the content of the main folders.

- 5 Testing activity
    - Here is specified how the system has been tested and are described the main test cases and their outcome.
- 6 Installation guide
    - This section includes the installation guide.

2 Product functions

2.1 Implemented functions

2.1.1 Evaluate Farmer

A Policy Maker can evaluate a farmer according to the productions that the farmer has inserted in DREAM. After the examination of the productions the Policy Maker can evaluate the farmer as a "Bad Farmer" or a "Good Farmer".

The first evaluation marks the farmer as a good farmer while the latter makes him a bad farmer in order to be able to assign him/her to a steering initiative aiming at improving his/her production.

2.1.2 Assign Farmer to a Steering Initiative

This functionality allows the Policy Maker to select a farmer that is performing badly and that is not currently in an ongoing steering initiative and choose one of the agronomists available to create a new steering initiative to help the badly performing farmer improve their production with the professional support of an agronomist.

2.1.3 Evaluate Steering Initiative

This functionality allows the Policy Maker to view the steering initiative info and the steering report, if available, in which are reported all the production rates before and after the steering (improvements are represented in green while lower or equal rates are represented in red to emphasize the lack of improvement).

The Policy Maker can indeed evaluate the production rates of a specific steering initiative and to choose if the initiative has produced noteworthy results or not.

2.2 Discarded functions

The implementation of the application focused on the functionalities of the Policy Maker so some functions have been discarded from the implementation. It’s worth emphasizing that due to the previous point the discarded functions are those concerning the farmer.

2.2.1 Inserting Production

2.2.2 Checking personalized informations

2.2.3 Creating a new discussion

2.2.4 View an open discussion

2.2.5 Commenting an open discussion

3 System development frameworks

3.1 Adopted frameworks

To facilitate the development process, the application has been built starting from already existing frameworks.

• Express: as already shown in the DD, the system business logic resides on a server running on NodeJS. ExpressJS is considered the de facto web framework for this type of execution environment.

• React: framework that provides all the functions and interfaces used for the front-end part of the software.

3.1 Adopted programming languages

3.1.1 JavaScript

The whole logic of the system has been written using JavaScript ES6, a high-level just-in-time compiled programming language that, together with HTML and CSS, is one of the core technologies of the WWW. In this application, the logic is almost completely on the web and application servers, with the workload on the client side only responsible for making the page interactive.

3.1.2 HTML, CSS

3.1.3 SQL

Language based on relational model used in the dump and to define the database schema and queries to get results from the database and show them to the client interface.

3.2 Adopted middlewares

The list below contains the middlewares used to add functionalities to the application server:

- Body Parser: to parse incoming request bodies in a middleware before handlers
- Cookie Parser: used to parse cookies contained in the incoming requests.

3.2.1 Other Libraries

The list below contains the external libraries used to add key functionalities to the application server during the implementation:

- Mysql2: used to perform queries to the DB.
- Superagent: testing simulates an HTTP client making requests.
- Moment: used to get current date useful for computations
- Bcrypt: to crypt and decrypt passwords
- Bootstrap: front-end toolkit for prebuilt components
- Semantic-UI: front-end toolkit for prebuilt components and JavaScript plugins
- SweetAlert: to give feedback trough customizable pop-ups
- Formik: to manage forms informations
- Yup: to manage forms informations
- DotEnv: to manage the .env file
- React-dom: library useful for routing
- React-router-dom: library useful for routing
- Jsonwebtoken: library to handle tokens in http requests
- Nodemon: automatically restarts the express app on changes
- Sequelize: Sequelize is a promise-based Node.js ORM for Mysql, Postgres, MariaDb and SQLite.

3.2 Adopted APIs

- Google-Maps: API useful to display informations referring to the location that is specified for a farmer. The API is not totally functional since using a generated key we cannot access all the functionalities available but just some for development purposes.

4 Source code structure

This is a simple directory tree that shows the structure of our source code

4.1 Back-end

```
.
├── app.js //Main method that runs the server
├── config //Folder that contains config files of the database
├── controllers //Services accessed by the routers to handle the application
logic.
├── middleware //Custom middlewares that handle authentication and
authorization
├── models //Each model represents a table stored in the database
├── node_modules
├── package-lock.json
├── package.json
├── routes //Routes (APIs) of the application.
└── test //List of tests performed on the system
```

4.1 Front-end

```
.
├── README.md
├── node_modules
├── package-lock.json
├── package.json
├── public. //Folder that contains logo, manifest and the index.html file
└── src  //Contains all the page of the application.
```

5 Testing activity

5.1 Testing environment

In order to stimulate the system, a mock HTTP client was introduced thanks to an external module. This client has the task of performing multiple actions on the available services and verifying that the response matches the expected result for each test.
Due to the small size of the public interface classes, to avoid trivial tests, the test cases focus specifically on the routes of the web server. These components directly interact with the public interfaces, in this way, a complete test suite is guaranteed.

5.2 Test cases

5.2.1 Account Routes

The testing of the account routes begins with the initialization of the DREAM application and the instantiation of a HTTP client to perform the testing.

The following tests are done:

1. Login:
    - Login with invalid email.
    - Login with invalid password.
    - Login of a PolicyMaker.

All the tests are passed.

5.2.1 Evaluate Routes

The testing of the evaluate routes proceeds in the same way, always starting from the login.

The following tests are done:

1. Retrieve farmers
2. Retrieve steering
3. Retrieve farmer information:
    - Retrieve inexistent farmer.
    - Retrieve information of a farmer.
4. Retrieve steering initiative information:
    - Retrieve inexistent steering initiative.
    - Retrieve information of a steering initiative
5. Evaluate steering initiative:
    - Evaluate a steering initiative that doesn’t exists.

    - Evaluate a steering initiative with a non-positive grade.
    - Evaluate a steering initiative positively.
6. Evaluate farmer:
    - Evaluate a farmer that doesn’t exists.
    - Evaluate a farmer with a non-Boolean grade.
    - Evaluate a farmer.

All tests are passed.

5.2.1 Evaluate Routes

The testing of the evaluate routes proceeds in the same way, always starting from the login.

The following tests are done:

1. Retrieve badly performing farmers
2. Retrieve all the available agronomists
3. Create a new steering initiative:
    - Create a steering initiative with a non-existing farmer
    - Create a steering initiative with a non-existing agronomist
    - Create a steering initiative with a farmer already in a steering initiative.
    - Create a steering initiative.

All tests are passed.

5.2.1 Middlewares

Additionally, some tests are also performed to ensure that the middlewares for the authentication works properly.

The following tests are done:

1. No login:
    - Retrieve agronomists without login.
    - Retrieve bad farmers without login.
    - Create a new steering initiative without login.
    - Retrieve farmers without login
    - Retrieve farmer info without login.
    - Retrieve steering initiatives without login
    - Retrieve steering initiative information without login.

- Evaluate a farmer without login.
- Evaluate a steering initiative without login

All tests are passed.

6 Installation guide

The installation guide can be found in the README file inside the DREAM folder. Inside the DD is present a schema referring to the database structure used in the application, for further details the database can be explored after importing it into the desired workbench.

7 Effort spent

7.1 Ferrara Alessio

Topic	Hours
Section 1,2,3 and 4	3,5 h
Section 5	1 h

7.2 Mazzola Francesco

Topic	Hours
Section 1,2,3 and 4	3,5 h
Installation guide	1 h