POLITECNICO
MILANO 1863

Software Engineering 2 project

Academic Year 2021-2022

DREAM - Data-dRiven PrEdictive FArMing in Telengana

Implementation and Test Deliverable

Version 0.1

Authors:

Xu Qiongjie

Zhang Rui

Hu Zhijun

Professor:

Di Nitto Elisabetta

Content

1.INTRODUCTION

1.1 Revision History

Vision	Date	Content
1.0	2022-02-06	ITD

1.2 Purpose and Scope

1.2.1 Purpose

This document is the Implementation and Testing Document for the Customers Line-Up system. Here will be listed the implemented requirements, the used frameworks and the testing done

1.2.2 Scope

Its aim is to define and report about the software implementation and test procedures defined for all the released project. Precisely, it verifies that the Dream as a whole satisfies its functional requirements. To this end, we define a set of tests covering the different uses that have been defined in the RASD and DD. The main goal of these tests is to specify for each tested demonstrator scenario a series of actions leading to an expected result.

1.3 Definitions and Abbreviations

1.3.1 Definitions

- **Telengana**: Telengana is the 11th largest state in India with a geographical area of 112,077 km2 and 35,193,978 residents (data from 2011)
- **DREAM System (or "The System")**: refers to the whole system to be developed.
- **DREAM Services (or "Services")**: refers to the functionalities offered by the DREAM System, such as daily plan management mechanism and the visualization service.
- **DREAM Application (or "The Application")**: refers to the application that makes DREAM Services available everywhere. In this document, this term is intentionally used in a generic way. How the application will actually be delivered (e.g., as a web app or as a mobile app) will be defined later in the Design Document of DREAM.
- **DD**: Design Document.

1.3.2 Abbreviations

- **CMD:** Command Prompt
- **API:** Application Programming Interface

1.4 Reference Documents

- Course slides on WeBeeP
- Design Document for DREAM

1.5 Document structure

1.Introduction: Introduction and purpose of this document.

2.Implemented Requirements: The requirements and functions that are actually implemented in the software.

3. Adopted Development Frameworks: includes adopted development frameworks with references to sections in the DD. It presents adopted frameworks and programming languages with their advantages and disadvantages.

4. Source Code Structure: explains how the source code is structured both in the front end and in the back end.

5. Testing: provides the main test cases applied to the the application.

6. Installation: explains how to install and deploy the application.

7. Effort Spent: keeps track of the time spent to complete this document.

8. Reference: lists the references used in this document.

2.Implemented Requirements

The following subsections list the requirements implemented in the prototype. Since for a group of three people, only features offered to two of the three stakeholders are required. Therefore, the features offered to policy makers and farmers are implemented. Features for agronomists are partially implemented.

2.1 Policy maker

Requirement ID	Requirement Description	Implementation
R.1	The System must allow Policy Maker to register DREAM by filling in a form containing a set of fields	Implemented.
R.2	The System must store the personal data relating to Policy Maker	Implemented.
R.3	The System must allow Policy Maker to login into DREAM by entering his email and password	Implemented.
R.4	The System must allow Policy Maker to visualize the data about his/her interested area in Telengana.	Implemented.
R.5	The System must be able to store Farmer's performance	Implemented
R.6	The System must allow Policy Maker to input Farmer's performance	Implemented
R.7	The System must allow Policy Maker to visualize daily plans of Agronomist	Implemented

2.2 Farmer

Requirement ID	Requirement Description	Implementation
R.8	The System must allow Farmer to register DREAM by filling in a form containing a set of fields	Implemented.

R.9	The System must store the personal data relating to Farmer	Implemented.
R.10	The System must allow Farmer to login into DREAM by entering his phone number and password.	Implemented.
R.11	The System must allow Farmer to search the data	Implemented.
R.12	The System must allow Farmer to ask for help	Implemented.
R.13	The System must store the problem data	Implemented.
R.14	The System must allow Farmer to report his/her production	Implemented.
R.15	The System must store the production report data	Implemented.
R.16	The System must allow Farmer to create a post	Implemented.
R.17	The System must store the new post	Implemented.
R.18	The System must allow Farmer to visualize posts list	Implemented.
R.19	The System must allow Farmer to visualize post details	Implemented.
R.20	The System must allow Farmer to leave a comment	Implemented.
R.21	The System must store the new comment	Implemented.

2.3 Agronomist

Requirement ID	Requirement Description	Implementation
R.22	The System must allow agronomist to register to	Implemented.

	DREAM by filling in a form containing a set of fields	
R.23	The System must store the personal data relating to the Agronomist	Implemented.
R.24	The System must allow Agronomist to login to DREAM by entering his/her email and password	Implemented.
R.25	The System must allow Agronomist to visualize the data about his/her interested area in Telengana.	Implemented.
R.26	The System must allow Agronomist to create the daily plan	Not implemented.
R.27	Agronomist must be able to visualize the detail of the daily plan	Not implemented.
R.28	Agronomist must be able to visualize the list of the daily plan	Implemented.
R.29	The System must store the daily plan data	Not implemented.
R.30	The System must allow Agronomist to modify the daily plan	Not implemented.
R.31	The System must allow Agronomist to confirm the daily plan	Not implemented.
R.32	Agronomist must be able to visualize the status of daily plan	Not implemented.
R.33	The System must allow Agronomist to answer the requests	Not implemented.
R.34	The System must allow Agronomist to visualize the list of the requests	Not implemented.

R.35	The System must allow Agronomist to visualize the details of the requests	Not implemented.
R.36	The System must store the answer to the request	Not implemented.

3. Adopted Development Frameworks

The backend-end is implemented in **Java EE** with TomEE application/web server.

Java EE is a Java based software framework used for development of Enterprise Application. It consists of a powerful set of specifications and APIS, such as EJB and JPA. Besides, the component-based and platform-independent Java EE makes the four-tier architecture (described in chapter 2 of the DD) easy to implement.

TomEE is the Java Enterprise Edition of Apache Tomcat, which includes all the additional libraries required to provide a Java EE environment. It is very lightweight, and easy to install and set up.

In addition, some libraries and frameworks have been adopted such as:

- **EclipseLink**: It provides an extensible framework that allows Java developers to interact with a variety of data services, including databases, Web services, object XML mapping, and enterprise information systems.
- **Maven**: a project management and integration tool. It provides a complete build lifecycle framework which allows development teams to automate the basic build configuration of a project.
- **Bootstrap**: a set of open-source front-end frameworks for website and web application development, including HTML, CSS and JavaScript frameworks, designed to make dynamic web pages and the development of web applications is easier.
- **JQuery**: a set of cross-browser JavaScript libraries for simplifying operations between HTML and JavaScript.
- **JSP**: Java Server Pages (JSP) is a server-side programming technology that allows developers to create dynamic, platform-independent method for building Web-based applications.

3.1 Programming languages

3.1.1 Java

Java is a high-level object-oriented programming language based on classes. Since Java EE is adopted, it has been used for server-side development. Below are its advantages and disadvantages.

- Advantage
    - **Platform-independent**. Java is a “write once, runs anywhere” language (WORA) which can be converted into standard bytecode at the compile time. The bytecode can be run on any device equipped with a Java virtual machine (JVM).
    - **Object-Oriented Programming language**. Everything in Java is an object which takes care of both data and behavior.
    - **Robust**. Java uses strong memory management. It does not provide explicit pointers so that the developer cannot access the memory directly from the code as in C. Besides, Java provides the syntax which allow developer to handle the exceptions.
- Disadvantage
    - Learning curve.

    - **Verbose and complex codes**. Java code is verbose, which means there are many long and complex sentences that are difficult to read and understand. This reduces the readability of the code.

3.1.2 Python

Python is a general-purpose, interpreted, high-level dynamic programming language. It is adopted to create the stub to simulate the APIs provided by external systems/services. Its advantages and disadvantages are listed below.

- Advantage
    - **Easy to use and learn**. The syntax of Python is like the English language which make it very easy to learn and adapt to. On the other hand, the simple nature of Python helps the developers to concentrate on solving the task instead of wasting too much time to understand the syntax and behavior of the programming language.
    - **Extensive library**. Python has a large number of libraries and frameworks, which can boost the development process and increase the productivity.
- Disadvantage
    - **Performance**. The performance of Python is worse than other languages such as Java and C due to the nature of interpreted language. But as we used it only for stub, this disadvantage is acceptable.

4.Source Code Structure

This section describes the structure of the source code.

4.1 Server side

The server side includes two major modules: **ejb** and **web**.

4.1.1 ejb

This module encapsulates the business logic of the application.

It includes:

- **models**: POJOs which represent data that can be persisted to the database. Each model is an entity, corresponding to a table in the database.
- **externals**: data types for external API.
- **services**: Enterprise Java Bean (EJB), core components of this module which interacts with the EntityManager to communicate with the database.

4.1.2 web

This module contains all Java servlets which handle the web HTTP requests and act as server-side servlet web API. It includes:

- **controller**: all web servlets which serve different pages and interact with the injected EJB module.

Besides, it includes all the static content like CSS, JS, images and web pages which will be processed by JSP to create dynamically generated web pages and served to the client web browser.

5.Testing

Before we release an application, it undergoes a thorough testing process to ensure that the system is working in the manner, in which it was intended. There are four main stages of testing that need to be completed before a program can be cleared for use: unit testing, integration testing, system testing, and acceptance testing. In this project, we focus on “unit testing”, “integration testing”, and “system testing”.

5.1 Unit Testing

During this first round of testing, the program is submitted to assessments that focus on specific units or components of the system to determine whether each one is fully functional. The main aim of this endeavor is to determine whether the application functions as designed.

Here, we focus on the unit test of the component interfaces in Application Server. White-box Testing method is used to get the job done.

In the following subsections, the Services refer to the manager components described in chapter 2.5 of DD.

For “Data Manager”, “Entity Manager” in Java is used directly, therefore, it is not listed here.

5.1.1 Account Service Test U1

Unit Testing case ID	Interface	Description	Expected Outcome
1	**createAgronomistAccount(agronomistData)**	Call with correct agronomistData	No exception thrown
2		call with nonexistent area	Exception is thrown
3		call with same email input	Exception is thrown
4		call with incorrect password format input	Exception is thrown
5	**authenticateAgronomist(email, pwd)**	call with correct email and password	Return specified agronomist
6		Call with incorrect password	Exception is thrown
7		call with no exist user input	Exception is thrown
8	**getFarmerListByAgronomist(agronomistID)**	Call with corrrect agronomistID	Return a list of farmers who belong to the area the

			specific agronomist is responsible for
9		Call with incorrrect agronomistID	Return a empty list
10	createFarmerAccount(farmerData)	Call with correct farmerData	No exception thrown
11		Call with incorrect farmerData	Exception is thrown
12		Call with same phone number input	Exception is thrown
13		call with incorrect password format input	Exception is thrown
14	**authenticateFarmer(phonenumber, pwd)**	call with correct phonenumber and password	Return the specific farmer
15		Call with incorrect email and password	Exception is thrown
16		call with no exist user input	Exception is thrown
17	**getAgronomistByFarmer(farmerID)**	Call with correct farmerID	Return the agronomist who is responsible for the area where the specific farmer belongs to
18		Call with incorrect farmerID	Return null
19	**createPolicyMakerAccount(policyMakerData)**	Call with correct policyMakerData	No exception thrown
21		Call with incorrect policyMakerData	Return errors accordingly
22		call with same email input	Exception is thrown
23		call with incorrect password format input	Exception is thrown
24	**authenticatePolicymaker(email, pwd)**	call with correct email and password	Return the specific policymaker
25		call with incorrect email and password	Exception is thrown

26		Call with no exist user input	Exception is thrown
27	**getAgronomistList()**	Call directly	Return a list of agronomists in Telengana.
28	**getAgronomist(agronomsitID)**	Call with correct agronomistID	Return a specific agronomist
29		Get specific agronomist with incorrect agronomistID	Return null
30	**getFarmerListByArea(areaID)**	call with correct AreaID	Return a list of farmers who belong to the specific area.
31		call with incorrect agronomistID	Return an empty list
32	**updatePerformance(performanceData)**	Update performance with Performancedata	No exception thrown

5.1.2 Daily Plan Service Test U2

Only the interface required by the features of Policymaker and Farmer are implemented. Thus, only those are tested.

Unit Testing case ID	interface	Description	Expected Outcome
1	**getDailyPlanDetail(dailyPlanID)**	Call with correct dailyPlanID	Return detail information for a specific daily plan.
2		Call with nonexistent dailyPlanID	Return null
3	**getDailyPlanList(agronomistID, page, count)**	Call with correct agronomistID	returns a list of daily plans of the given agronomist.
4		Call with nonexistent agronomistID	Return an empty list.

5.1.3 Forum Service Test U3

Unit Testing case ID	interface	Description	Expected Outcome
1	getForum()	Call directly	Return a list of posts, including comment number, title, time, post creator of each post
2	**createPost(postData)**	Call with correct postData	Return the new post.
3		Create a post with incorrect postdata	An exception is thown.
5	**getPost(postID)**	Call with an existing post ID.	Return the specific post
6		Call with nonexistent post ID	Return null
7	**createComment(commentData)**	Call with correct commentData	No exception is thrown.
8		Call with incorrect commentData	An exception is thown.

5.1.4 Geospatial Data Service Test u4

Unit Testing case ID	interface	Description	Expected Outcome
1	**getSoil(areaID)**	Call with an existing area ID	Return the data of soil humidity for a specific area.
2		Call with a nonexistent area ID	Return null.
3	**getWeather(areaID)**	Call with an existing area ID	Return the data of weather for a specific area.
4		Call with a nonexistent area ID	Return null.
5	**getWaterIrrigation(areaID)**	Call with an existing area ID	Return the data of water usage for a specific area.
6		Call with a nonexistent area ID	Return null.
7	**getTypeInfo(productType)**	Call with a supported product type.	Return the product suggestion for a specific product type.
8		Call with a non-supported product type.	Return null.

9	**getTypeList()**	Call directly	Return a list of product type
10	**getAreaList()**	Call directly	Return a list of areas in Telengana.
11	**getArea(areaID)**	Call with an existing area ID	Return a specific area in Telengana.
12		Call with a nonexistent area ID	Return null

5.1.5 Problem Service Test U5

Only the interface required by the features of Policymaker and Farmer are implemented. Thus, only those are tested.

Unit Testing case ID	interface	Description	Expected Outcome
1	getProblemByFarmer(farmerID)	Call with correct farmerID	Return a list of specific farmer's problems
2		call with incorrect farmerID	Return errors accordingly
3	createProblem(problemData)	Call with correct problemData	No exception is thrown.
4		call with incorrect problemData	An exception is thrown
5	feedbackProblem(feedbackData)	call with correct feedbackdata	No exception is thrown.
6		Call with incorrect feedbackdata	An exception is thrown

5.1.6 Production Report Service Test U6

Unit Testing case ID	interface	Description	Expected Outcome
1	reportProduction(reportData)	call with correct reportData	No exception is thrown.
2		call with incorrect reportData	An exception is thrown
3	getFarmerProductionList(areaID)	call with correct areaID	Return a list of production reports for the given area.
4		Call with incorrect areaID	Return null.

5.1.7 Search Service Test U7

Unit Testing case ID	interface	Description	Expected Outcome
1	getSearchInfo()	Call directly	Return a list of production types existing in external component and the area list in Telengana.
2	search(area, productType)	Call directly	returns weather forecast for the given area and suggestion for the specific production

5.2 Integration Testing

Integration testing allows individuals the opportunity to combine all the units within a program and test them as a group. This testing level is designed to find interface defects between the modules/functions.

5.2.1services I1

Account Service I1.1 T1

Test ID	I1.1 T1
Components	**Account Service, Data Manager, DBMS**
Input specification	**Register new account**
Output specification	Check if the new account is available from the database and if the email/phone number was correctly sent. Exception: if the method is invoked on a non-valid object, the account should not be created.
Description	Account Manager sends an update query to the DBMS in order to append the message to the list of existing accounts and uses the email/phone message for sending the confirmation message to the User.

Test ID	I1.1 T2
Components	**Account Service, Data Manager, DBMS**
Input specification	**Login with existing account**
Output specification	Check if the login was successful

	Exception: if the method is invoked on a non-valid object, the login request should be denied.
Description	Account Manager sends a select query to the DBMS in order to verify the account credentials.

Daily Plan Manager I1.2 T1

Test ID	**I1.2 T1**
Components	**Daily Plan Manager, Data Manager, DBMS**
Input specification	**Acquire the daily plan details**
Output specification	Check if the information was successfully loaded Exception: if the method is invoked on a non-valid object, the daily plan information should be displayed.
Description	Daily plan manager communicates with *Data Manager* to access, store and update information of daily plan.

Forum Manager I1.3 T1

Test ID	**I1.3 T1**
Components	**Forum Manager, Data Manager, DBMS**
Input specification	**Create a new post**
Output specification	Check if the post information was successfully submitted. Exception: if the method is invoked on a non-valid object, the post information should be uploaded.
Description	Forum manager accesses the *Data Manager* to retrieve and store the related information.

Geospatial data service and Search Manager I1.4 T12

Test ID	**I1.4 T1**
Components	**Geospatial data service, Search manager, Data Manager, DBMS**
Input specification	**Search the farming information**
Output specification	Check if the farming information was successfully submitted. Exception: if the method is invoked on a non-valid object, the farming information should be uploaded.
Description	Search manager communicates with *Geospatial Data Manager* to access weather information and production suggestions.

Problem Manager I1.5 T1

Test ID	I1.5 T1
Components	**Problem Manager, Data Manager, DBMS**
Input specification	**Create a request**
Output specification	Check if the probelm was successfully submitted. Exception: if the method is invoked on a non-valid object, the problem should be uploaded.
Description	Problem manager access through *Data Manager* to retrieve, store and update information related to problem. Besides, once a new problem is created, it adds a notification to Agronomist. On the other hand, if a new problem is read, it removes the notification.

Production Report Manager I1.6 T1

Test ID	I1.6 T1
Components	**Production Report Manager, Data Manager, DBMS**
Input specification	**Submit a reportation**
Output specification	Check if the reportation information was successfully submitted. Exception: if the method is invoked on a non-valid object, the reportation information should be uploaded.
Description	Production report manager interacts with *Data Manger* to store and access production data.

5.3 System Testing

System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards.

This testing process is carried out manually with the help of a web browser. In particular, the following test cases and their expected outcome are considered.

All the login pages can be accessed by clicking the icon respectively in the home page of DREAM.

5.3.1 Policy Maker

System Test Case ID	Description	Expected Outcome

1	In the policy maker register page, Register with correct input	register successfully and redirect to Login page
2	In the policy maker register page, egister with incorrect input	register fail and the error message shown in Register page
3	In the policy maker login page, Login with Policy Maker credentials	login successfully and redirect to Policy Maker homepage
4	In the policy maker login page, Login with nonexist Policy Maker credentials	login fail and error message shown in Login page
5	In the home page, Check "Adilabad" on the map	Display basic information of Adilabad and “update performance” botton in Adilabad page
6	In the home page, Check "Adilabad" with nonexist information on the map	display fail and error message shown in home page
7	In the Adilabad page, Check "Weather"	Display weather information in Adilabad page
8	In the Adilabad page, Check "Weather" with nonexist information	display weather information fail and error message shown in home page
9	In the Adilabad page, Check "Water Irrigation"	Display water irrigation information in Adilabad page
10	In the Adilabad page, Check " Water Irrigation "with nonexist information .	display water irrigation information fail and error message shown in home page
11	In the Adilabad page, Check "Soil Humidity"	Display soil humidity information in Adilabad page
12	In the Adilabad page, Check " Water Soil Humidity "with nonexist information	display Water Soil Humidity information fail and error message shown in home page
13	In the Adilabad page, Check "Farmer Production"	Display farmer production information in Adilabad page
14	In the Adilabad page, Check " Farmer Production "with nonexist information	display farmer production information fail and error message shown in home page
15	In the Adilabad page, check " update performance "	Display list of farmer performance in performance page

16	In the Adilabad page, Check " update performance " with nonexist information	Display list of farmer performance fail in performance page
17	In the Performance page, Submit performance information	Display a interface to allow policy makers input performance of farmer in perfomance page
18	In the Performance page, Submit nonexist performance information	Submit fail in performance page
19	In the home page, Check “zhang”on the agronomist list	Display daily plan information of zhang in Zhang page
20	In the home page, Check agronomist with nonexist information	display fail and error message shown in home page

5.3.2 Farmer

System Test Case ID	Description	Expected Outcome
1	In the farmer register page, Register with correct input	register successfully and redirect to Login page
2	In the farmer register page, Register with incorrect input	register fail and the error message shown in Register page
3	In the farmer login page, Login with Farmer credentials	login successfully and redirect to Farmer homepage
4	In the farmer login page, Login with nonexist Farmer credentials	login fail and error message shown in Login page
5	In home page , Click "Search"	Display search page
6	In the search page , Submit correct location and production type	Submit successfully and redirect to suggestion page
7	In home page , Click "report"	Display report page
8	In the report page , Submit correct production information	Submit successfully.
9	In home page , Click "request"	Display request page
10	In the request page, Click “Create a request”	Diaplay Create a new Request page
11	In the Create a new Request page page, Submit correct title and content information	Submit successfully and redirect to request page, and the information was shown in the request page

12	In the home page , Click "forum"	Display forum page
13	In the forum page ,Submit correct post information	Submit successfully and redirect to post page, and the information was shown in the post page
14	In the forum page, submit the comment information.	Submit successfully and redirect to post page, and the information was shown in the post page

5.3.3 Agronomist

System Test Case ID	Description	Expected Outcome
1	In the agronomist register page, Register with correct input	register successfully and redirect to Login page
2	In the agronomist register page, Register with incorrect input	register fail and the error message shown in Register page
3	In the agronomist login page, Login with agronomist credentials	login successfully and redirect to agronomist homepage
4	In the agronomist login page, Login with nonexist agronomist credentials	login fail and error message shown in Login page
5	In the home page, Click "Location"	Display Location maps in home page
6	In the home page, Click "Weather"	Display weather information in home page
7	In the home page, Click "Water Irrigation"	Display water irrigation information in home page
8	In the home page, Click "Soil Humidity"	Display soil humidity information in home page
9	In the home page, Click "Farmer Information"	Display farmer performance in home page
10	In the home page, click "Daily Plan"	Display list of daily plans in home page

6.Installation

6.1 Server

6.1.1 MySQL

6.1.1.1 Download MySQL

OS X:

- MySQL Community server 8.0.26: http://dev.mysql.com/downloads/mysql/
- MySQL Workbench 8.0.26: http://dev.mysql.com/downloads/workbench/

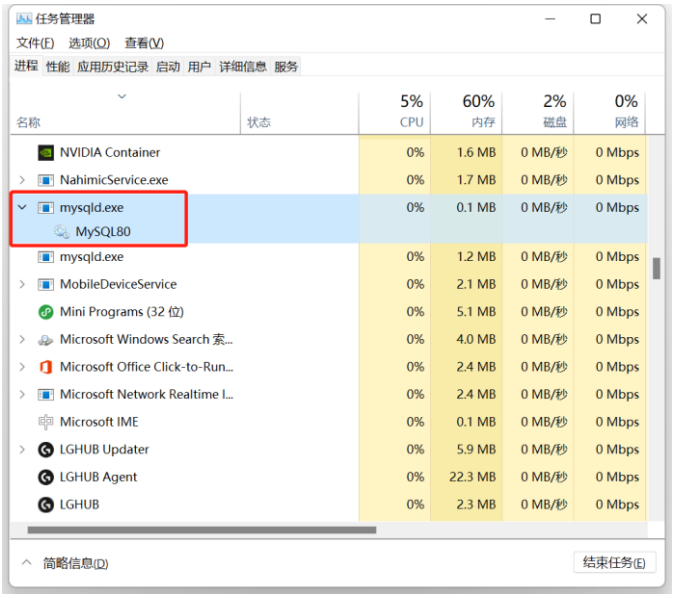
Windows:

- MySQL Installer 8.0.26: https://dev.mysql.com/downloads/installer/

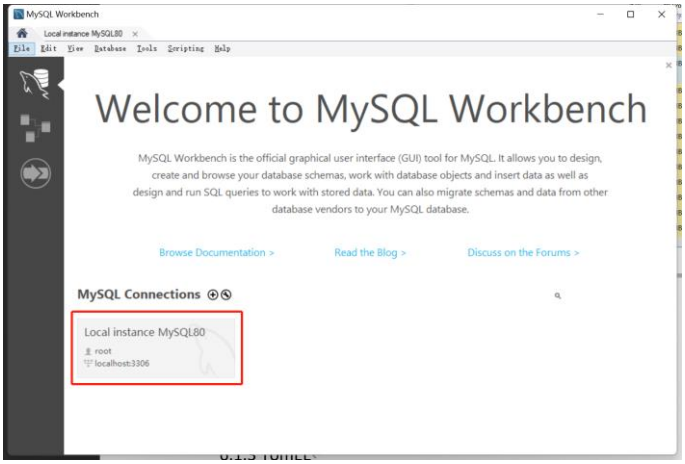
Note: MySQL Installer is 32-bit but will install both 32-bit and 64-bit binaries.

6.1.1.2 Install and start MySQL

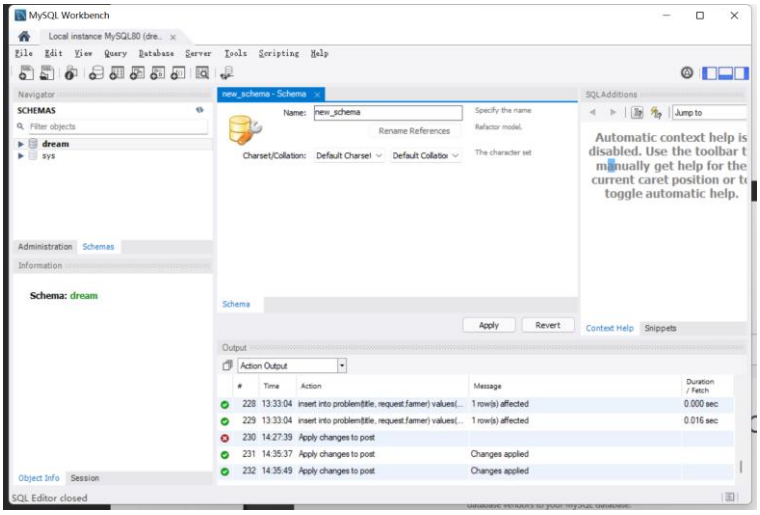
Start the MySQL Installer and install MySQL Community Server and MySQL Workbench, accepting all the default configurations and executing the steps necessary for installing possibly missing prerequisite packages. Be sure to store the root password in a safe and protected place. By default, MySQL is launched as a service at startup.



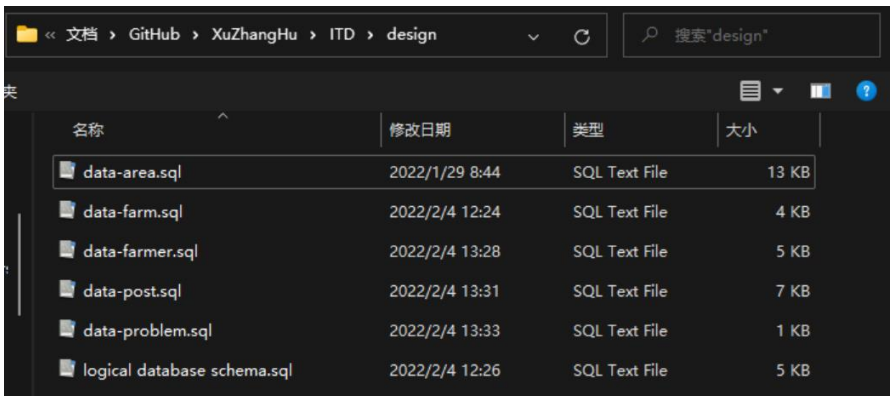
Open the workbench and check the status of the server.



Create a new schema named ***dream*** and set it as a default schema.



Open and run SQL script named '***logical database schema.sql'***, then run '***data-area.sql***'. If for test, run '***data-dailyplan,farmer,farm,forum&problem.sql***'.



6.1.2 Java JDK

6.1.2.1 Download Java JDK

Use the link to download the right version:

Oracle JDK and JRE, last version Java 16: https://www.oracle.com/java/technologies/javase/jdk16-archive-downloads.html

The full Java installation guide for all platforms can be found here:

https://docs.oracle.com/en/java/javase/index.html

Go to the Java SE download page, download the installer, and run it

6.1.2.2 Change JAVA environment variables

Set the windows environment variables, as explained e.g., here: https://stackoverflow.com/questions/1672281/environment-variables-for-java-installation When using Java SE 16 you don't need to set the JRE_HOME variable.

An example of environment variables is:

JAVA_HOME: C:\Program Files\Java\jdk-16.0.2

JDK_HOME: %JAVA_HOME%

CLASSPATH: .;%JAVA_HOME%\lib

Modify the PATH variable by adding the %JAVA_HOME%\bin to the already existing ones in your PC
PATH: your-unique-entries;%JAVA_HOME%\bin

6.1.3 TomEE

6.1.3.1 Download TomeEE

Use the link to download the right version.

Apache TomEE 8.0.9 (plume): http://tomee.apache.org/download.html

Note: do not use TomEE 9.0

- After unzipping, copy the TomEE folder into your favorite drive (e.g., C: or D:)

- Launch the TomEE Java Web server by executing the “TomEE.exe” file in the extracted TomEE folder, e.g: C:\Program Files\Apache Software Foundation\apache-tomee-plume-8.0.8\bin
- Alternatively, you can execute the file “startup.bat” on the same folder using the command line

6.1.3.2 Edit TomEE configuration

Because TomEE set hsql as Database, we have to change type and connection of Database. First , open ***{TomEE installation path}/conf*** folder, find ***tomee.xml*** file and add following code:

<Resource id="DREAMResource" type="DataSource">

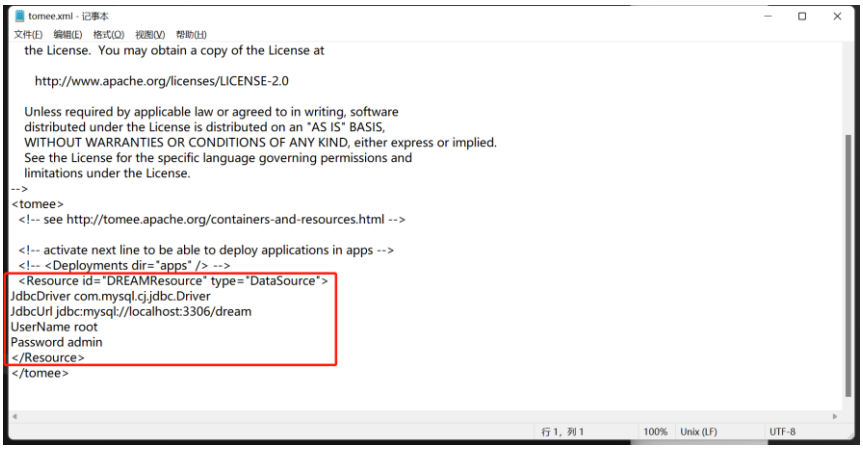
JdbcDriver com.mysql.cj.jdbc.Driver

JdbcUrl jdbc:mysql://localhost:3306/dream

UserName XXXXXX

Password XXXXXX

</Resource>



6.1.3.3 Add MySQL Connector

Download MySQL Connector/J, the official JDBC driver for MySQL. (check it is for the same version of your mysql version)

https://dev.mysql.com/downloads/connector/j/

Choose the platform independent version.

Copy the ***mysql-connector-java-{version}.jar*** into tomee by adding it to ***{TomEE installation path}/lib*** folder.

6.1.4 Server configuration

Copy ***{project path}/ITD/dream*** folder to ***{Tommee installation path}/webapps***

6.1.5 Stub

Simple installation guide is in ***{project path}/ITD/stub/readme.md***,

6.1.5.1 Install Python

Download and install python by following link:

https://www.python.org/downloads/release/python-3102/

6.1.5.2 Install pip

Pip is the standard package manager for Python. It enables the installation and management of third party packages that provide features and functionality not contained in the Python standard library. Newer versions of Python (Python 2 >= v2.7.9 or Python 3 >= v3.4) come prepackaged with pip by default. Pip is also included in the virtual environments created by virtualenv and pyvenv.

But if you're using an older version of Python, pip will need to be manually installed.

Check if pip is already installed by running the following on the command line:

```
pip --version
```

If pip is installed, you'll see something similar to the following output:

```
Pip 19.2.3 from c:usersjdoeappdatalocalprogramspythonpython38-32libsite-packagespip (python
3.8)
```

If not, Python comes with an ensurepip module, which can install pip in a Python environment.

```
py -m ensurepip --upgrade
```

Or download get-pip.py and install in command:

https://bootstrap.pypa.io/get-pip.py

6.1.5.3 Install Flask

Install Flask by running the following on the command line:

```
pip install -U Flask
```

If it doesn't work, run this command in ***{python installation path}/Python{version}/Scripts.*** Usually it is ***C:\Users\Username\AppData\Local\Programs\Python\Python310\Scripts***

6.1.5.4 Run geospatial.py

Run geospatial.py on the command line in ***{project path}/ITD/stub*** folder:

Then Stub is successfully running.

6.1.6 Deploy

In order to deploy the project to TomEE,

1. Start the server executing ***startup.bat*** in ***{TomEE installation path}/bin***.
2. Start ***geospatial.py*** in the ***{project path}/ITD/stub***
3. u can find the deployed server to http://localhost:8080/dream.

The preregistered users are:

Farmer: Dream

Phonenumber: 000001

Password: 000001

Agronomist: ag01

Email: ag01@email.com

Password: 12345678

7.Effort Spent

Task	Xu	Zhang	HU
development	96h	15H	40H
testing	7h	8H	2H
Document writing	8h	8H	8H

8.Reference

https://www.python.org/

https://dev.mysql.com/

https://tomee.apache.org/