|  |  |
| --- | --- |
| Project  **JAVA FUNDAMENTALS** | Abstract  Final Project (Identity Information Management) documentation of JAVA Fundamentals.  Bhrigu Mahajan  M.E (SDM) |

Table of Contents

[Subject description - 3 -](#_Toc507093259)

[Subject analysis - 3 -](#_Toc507093260)

[Major features - 3 -](#_Toc507093261)

Expected results - 3 -

Scope of the application……………………………………………………………………………………………………………..- 3 -

**Limitations…………………………………………………………………………………………………………………………………- 3 -**

[Conception.](#_Toc507093264).........................................................................................................................- 4 -

[Identities.](#_Toc507093266).................................................................................................................................- 4 -

[Users.](#_Toc507093267).......................................................................................................................................- 4 -

[**Some Important URL’s…………………………………………………………………………………………………**………**.**](#_Toc507093267).- 4 -

**Design and Description……………………………………………………………………………………………………. - 5 -**

**User Interface Workflow Design……………………………………………………………………………………………..- 5 -**

**UI Description….…….…………………………………………………………………………………………………………….. - 6 -**

**Business Logic Design Workflow……….…………………………………………………………………………………… - 7 -**

**Business Logic Description………………………………………………………………………………………………………- 8 -**

**Database Diagram………………………………………………………………………………………………………………….- 9 -**

**Database Description……………………………………………………………………………………………………………..- 9 -**

[Global Application Flow - 10 -](#_Toc507093268)

[Option: Create Identity - 12 -](#_Toc507093269)

[Option: Login Authentication - 13 -](#_Toc507093270)

[Option: Home Screen - 13 -](#_Toc507093271)

[Option: Update - 14 -](#_Toc507093272)

**Option: Delete………………………………………………………………………………………………………………………**- 13 -

**Option: Search………………………………………………………………………………………………………………………**- 13 -

**Option: Logout…………………………………………………………………………………………………………………….** - 14 -

[Database Configuration - 15 -](#_Toc507093273)

[Prerequisites - 15 -](#_Toc507093274)

**Bibliography…………………………………………………………………………………………………………………….- 15 -**

**Website……………………………………………………………………………………………………………………………………**- 15 -

# **Subject description**

This project is the implementation of an Identity Management Software. The project was developed during the semester’s class and has been completed during the last week during the end of the semester.

The implementation of this project has been done by using Frontend (HTML, Angular JS), Backend (REST, Java 8), Database (Derby) and Server (Tomcat).

# **Subject analysis**

## **Major features**

The IAMProject has 4 major features.

* *Login Authentication*: A User should be authenticated before been able to work and manage the Identities.
* *Manage Identities*: the user should be able to
  + Register
  + Update
  + Delete
  + Search.

The Identities and the users are stored as application data. *DERBY* used as the Database engine for managing the entities. *Rest API* is used as a service between UI and Java. *Angular JS* and *HTML* is used in User Interface. *Tomcat* is used as Application server (Port 9191).

**Expected results**

Users with administrative privileges can create, delete and modify identities.

User provided information can be stored in database.

**Scope of the application**

The application is limited to only manage Identities and not to manage the Users. This is of course one of the first steps to be improved in a next revision.

Also, the Identities are only going to be managed on a database, for now file storage mode for Identities is not implemented, so the user must have an installed Database.

**Limitations**

* User roles are not defined.
* Services are not secured.
* Password is exposed in payload sent for authentication.
* Need to refresh page after update user details.

**Conception**

**Data structures**

**Identities**

Identities are basically the entities that are going to be managed, within the application we should be able to create, update and delete these. All identities are going to be stored in a Derby database.

As per the requirements an identity can have:

* UId: User Id, a unique id combination of username and email as a Primary Key in the Identity table.
* Display Name: represented as a string.
* Email: represented as a string also.

**Users**

Users are not supposed to be managed by the application, for this reason a decision was made that user entities will be stored in a text file.

A user will have a name and a password; both are stored as string files.

**Some important URL’s**

***Login URL:***  *http://localhost:9191/iamProject/*

***Get All Records URL :*** *(/iamProject/rest/UserService/users)*

***Get All Records by Id URL :*** *(‘/iamProject/rest/UserService/id/' + id)*

***Search Records URL :*** *(‘/iamProject/rest/UserService/search/' + value )*

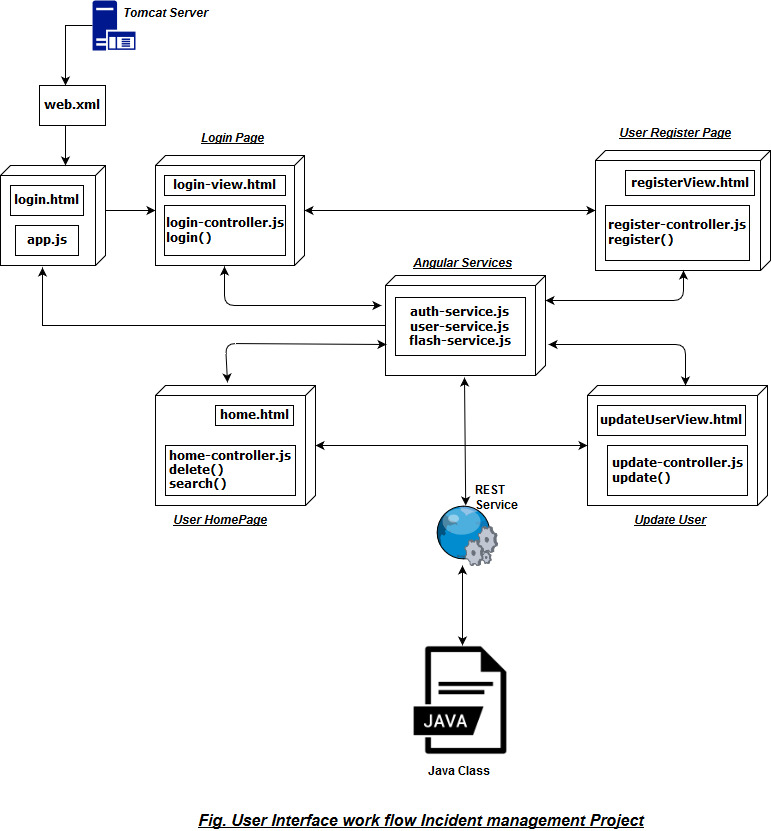
***Create User Object URL :*** *(‘/iamProject/rest/UserService/create', user)*

***Update User object URL :*** *(‘/iamProject/rest/UserService/update' , user)*

***Delete login user Record URL :*** *(‘/iamProject/rest/UserService/delete/' + id)*

**Design and Description**

**User Interface Workflow Design**

****

**UI Description**

**Tomcat** is an application server. To launch application, run application server. *Server.xml* is the configuration file for server, we can change ports on server.xml and set connection Timeout value. We can also add *VMArguments* to Tomcat.

My VMArguments are:

*-Dcatalina.base="C:\java\_Class\.metadata\.plugins\org.eclipse.wst.server.core\tmp0" -Dcatalina.home="C:\Users\Bhrigu Mahajan\Downloads\apache-tomcat-8.5.20" -Dwtp.deploy="C:\java\_Class\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps" -Djava.endorsed.dirs="C:\Users\Bhrigu Mahajan\Downloads\apache-tomcat-8.5.20\endorsed" -Dconf="C:\java\_Class\iamProject\incidentManagement.properties"*

**Web.xml** is used to instantiate servlet class and provide the page to be display on launch of application. Servlet class is instantiated by adding servlet class name and map that servlet class with the name of servlet and URL Pattern of servlet.

**Login.html**  act as controller. All controllers are added on this page so when this application is started control navigate through this page.

***app.js*** is used to provide route navigation path to application based on the URL provided to controller from application.

**login-view.html** is used to provide login page to user where username and password is authenticated.

**login-controller.js** contains login function and authentication service layer to check for successful authentication of user.

**registerView.html** is user registration screen for new user.

**register-controller.js**  contains service layer for creation new user.

**home.html** is the homepage of the application where all records will be loaded and operations like search, update and delete performed through this screen.

**home-controller.js** provide services to home.html to perform all operations.

**updateUserView.html** is page loaded when user try to update its personal details.

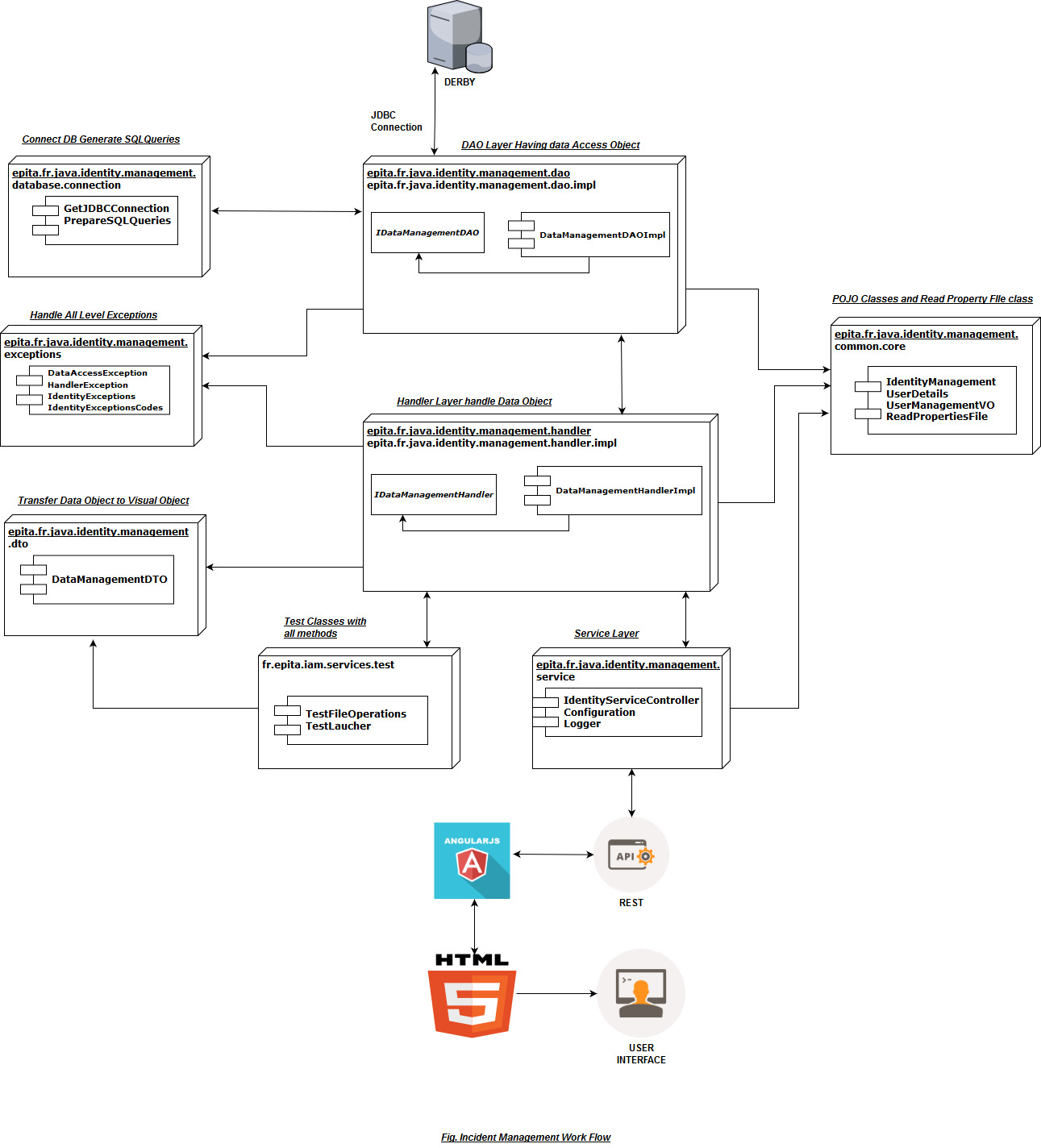
**update-controller.js** contains services to update user profile.

**auth-service** is invoked when user try to login in application.

**user-service** contains all the services which are used on the application. It is the center controller java script file for the application.

**flash-service** used to display success and error messages.

**Business Logic Design Flow**

****

**Business Logic Description**

**POJO Classes**

* **IdentityManagement** - Class used for Identity table
* **UserDetails** - Class used for user table
* **UserManagementVO** - Class used to wrap IdentityManagement and UserDetails class and send object to UI.

**ReadPropertiesFile** - Class used to read Property file objects and store them In Map.

**Auth** - Class used for login authentication.

**IdentityConstants** - Class contains constants used in application.

**DataManagementDAOImpl** - Class get Database object.

**GetJDBCConnection** - Class used to connect database.

**PrepareSQLQueries** - Class contain functions used to prepare SQL queries.

**DataManagementHandlerImpl** - Class used to handle DAO object.

**DataManagementDTO** - Class used to transform data object to UI object and vice versa.

**IdentityServiceController** – Class is used take request and response of REST API calls.

**Configuration** – Class is used to read *IncidentManagement.properties* file.

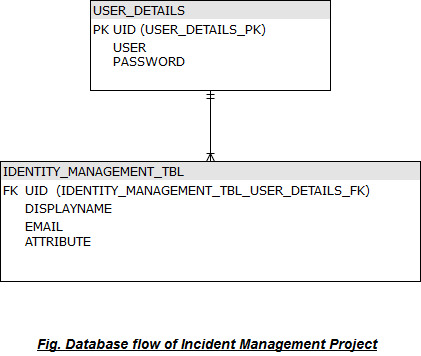
**Logger** - Class is used to generate logfile name *log4j-application.log .*

**IdentityException** - Class is main exception class of application and it extend Exception class.

**TestFileOperations** – Class contains all test methods of application.

**Testlaucher** – Class is used to launch TestFileOperations class.

**Database Diagram**

****

**Database description**

1. **IDENTITY\_MANAGEMENT\_TBL**

It is the table which contains all the information related to the Identity. The details of the table are described below.

**DISPLAYNAME** (**varchar** (100)) It contains the display name of the user.

**EMAIL** (**varchar** (100)) It contains the birthdate of the user.

**ATTRIBUTES** (**varchar** (100)) It contains the email address of the user.

**UID (varchar (**100**))** It is the primary key of the table. *It is combination of username and email.*

1. **USER\_DETAILS**

It is table which contains the names of the person having their user id, username and password.

**UID** (**varchar** (100)) It contains the name of the user having a valid user id.

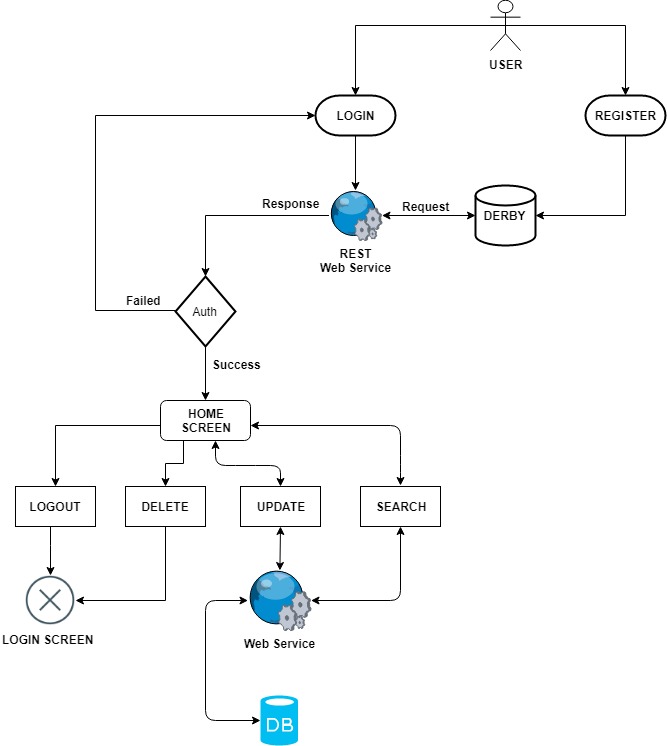
**USER (varchar (**50**))** It is user name of user.

**PASSWORD** (**varchar** (30)) It is password set by the user to login into account.

*Location of SQL file in project for reference*

**Folder Name :** iamProjectSQL/initProject.sql

## **Global Application Flow**

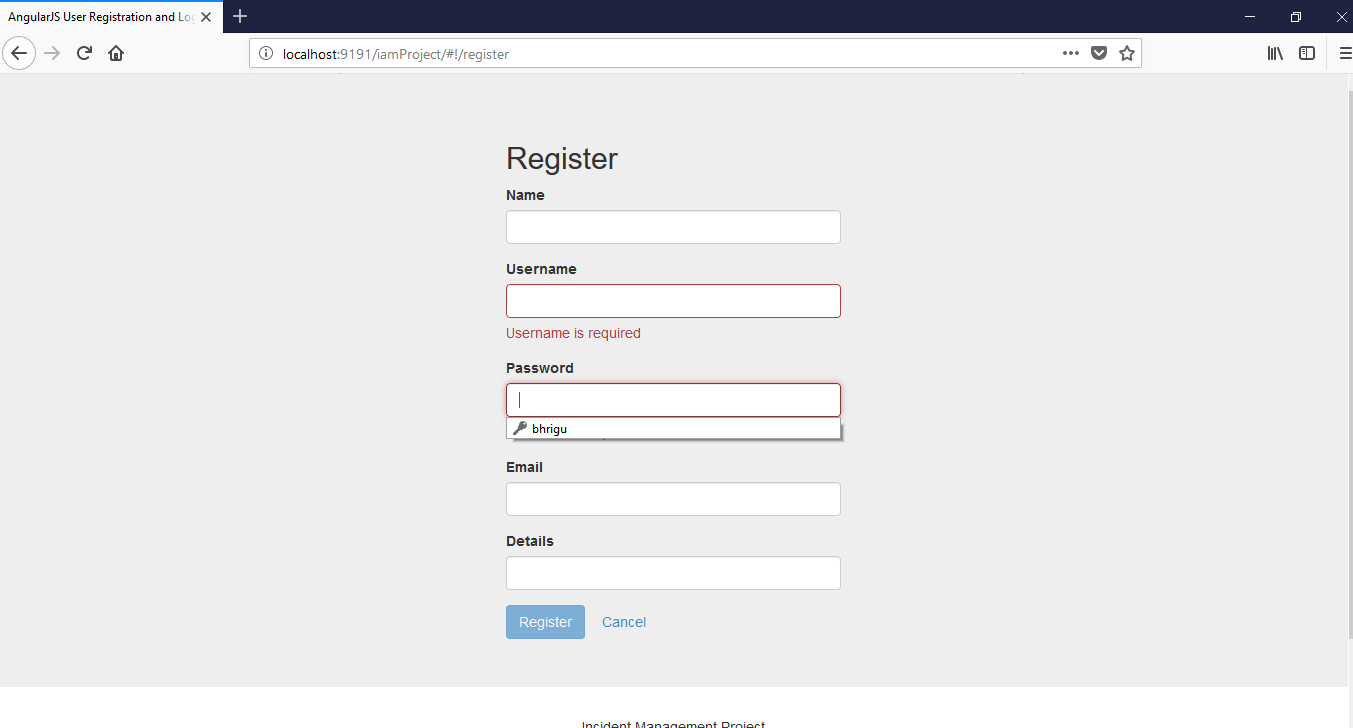
  
*Fig. Application flow was presented in the requirements of the project*

So, the implementation respected this diagram to fulfill the requirements. A description of this flow is done with screenshots in the following pages:

**Option: Create identity**

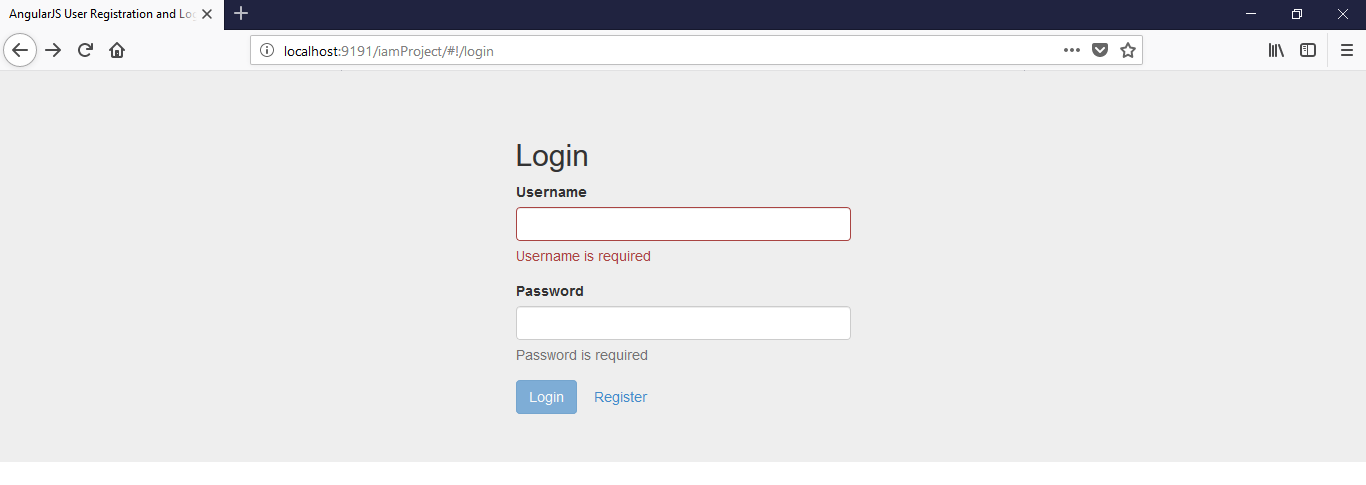
When register button is clicked, the user is required to input the Display Name, email, attributes, username and password for creating new user.

Once the register button is clicked with user data saved and created user can login in Login screen.



**Option: Login Authentication**

Enter user name and password for successful authentication.



**Option: Home Screen**

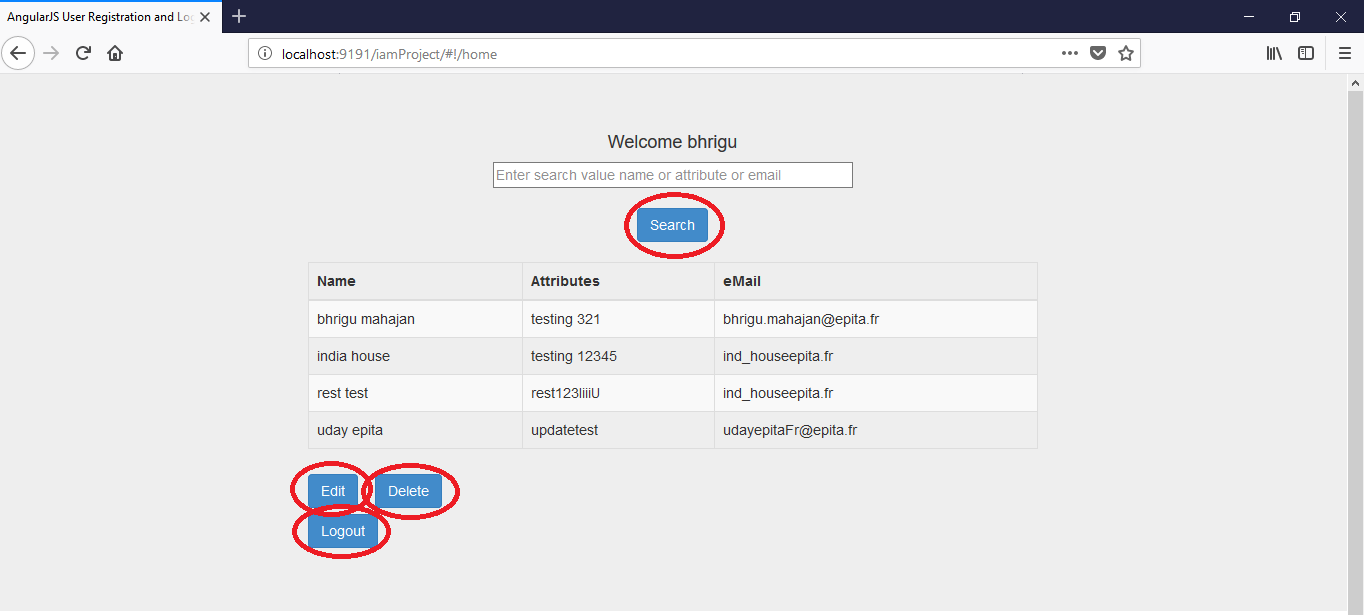
Home Screen page is redirect o successful authentication. On home page user have options to:

1.Search - user can search by name, attribute and email of any other user.

2.Edit - user can update his name, attribute and email.

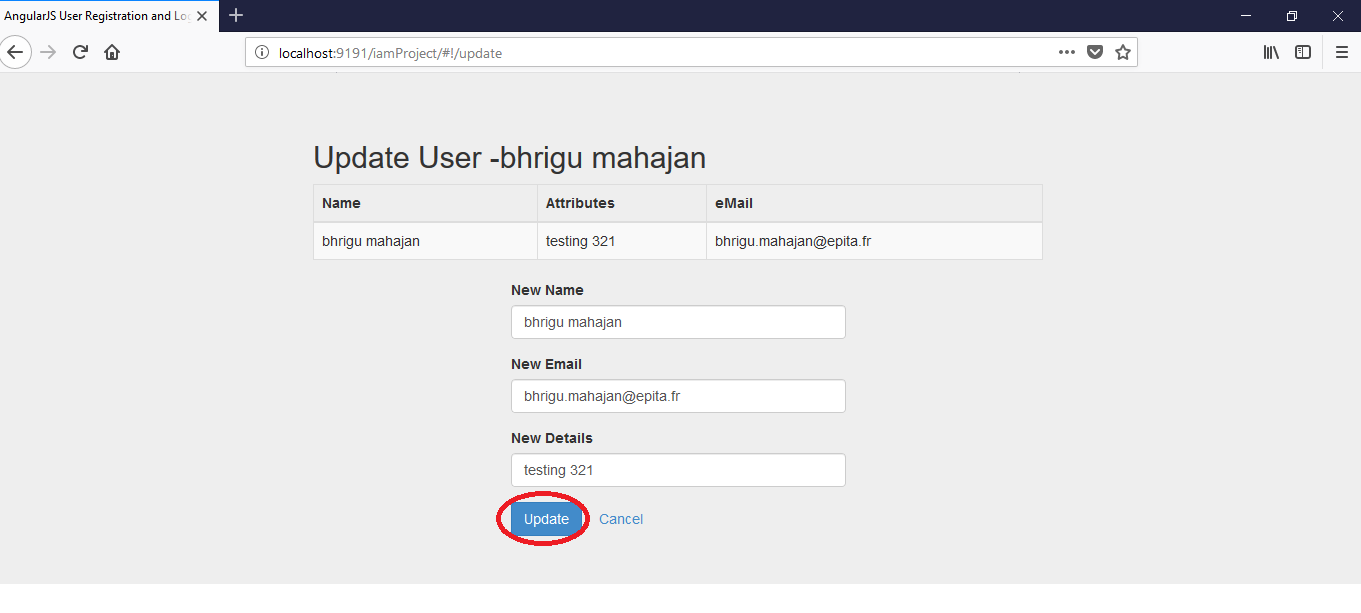
3.Delete – user can delete his own profile.

4.Logout – user exit to login screen.



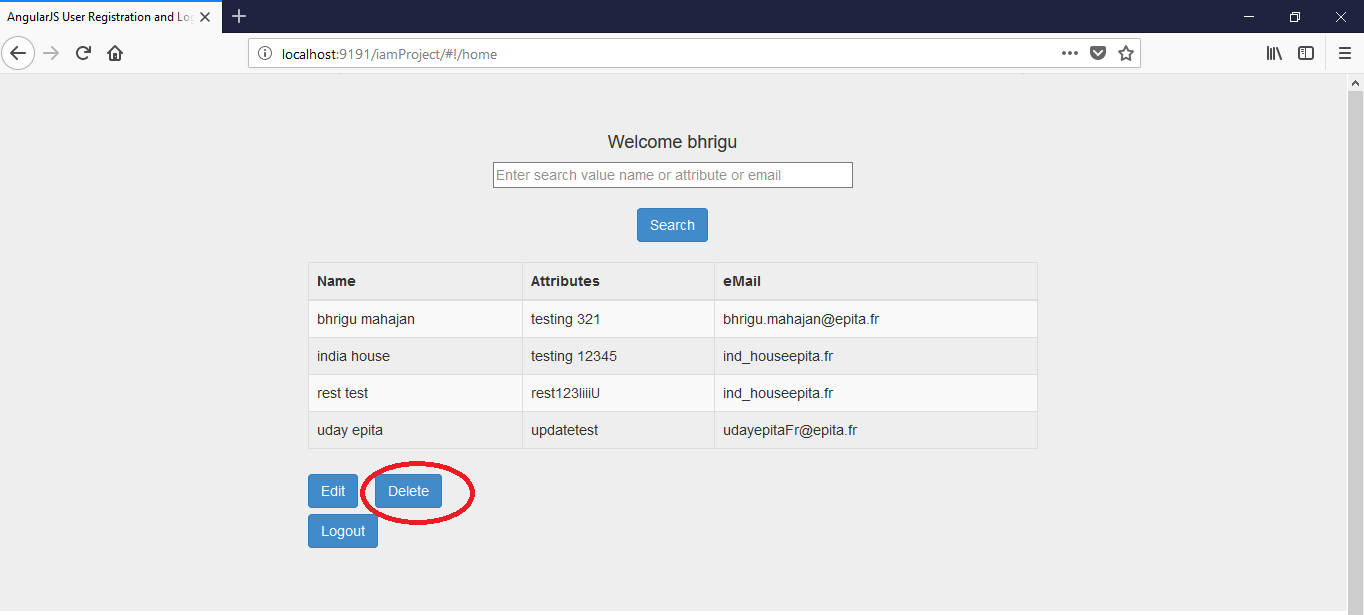
### **Option: Update**

The functionality is to update an existing user. Every user can change only his personal records and after clicking on update button redirected to home screen.



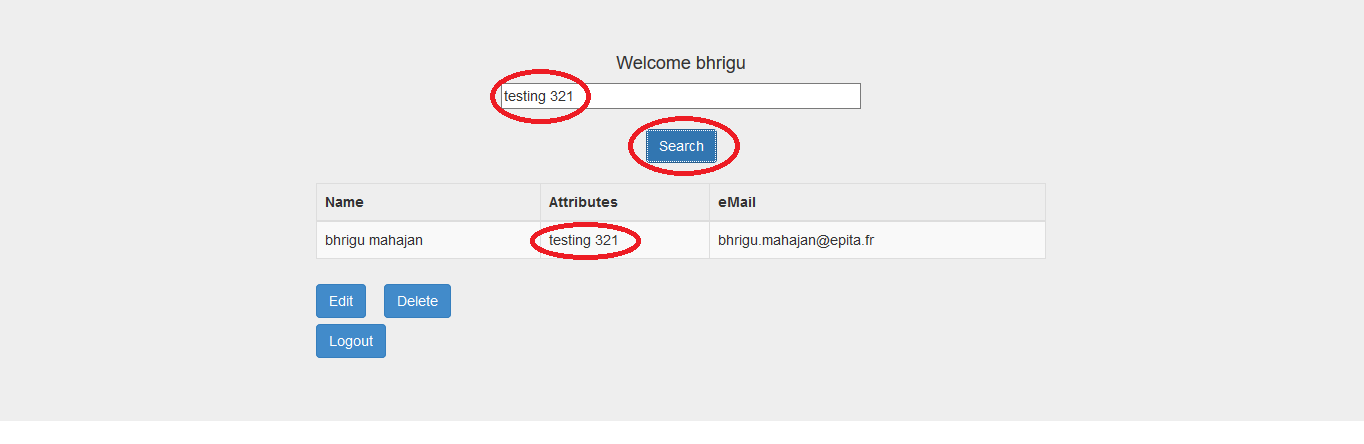
### **Option: Delete Identity**

Delete button will delete all records of user from database and redirect to login screen.



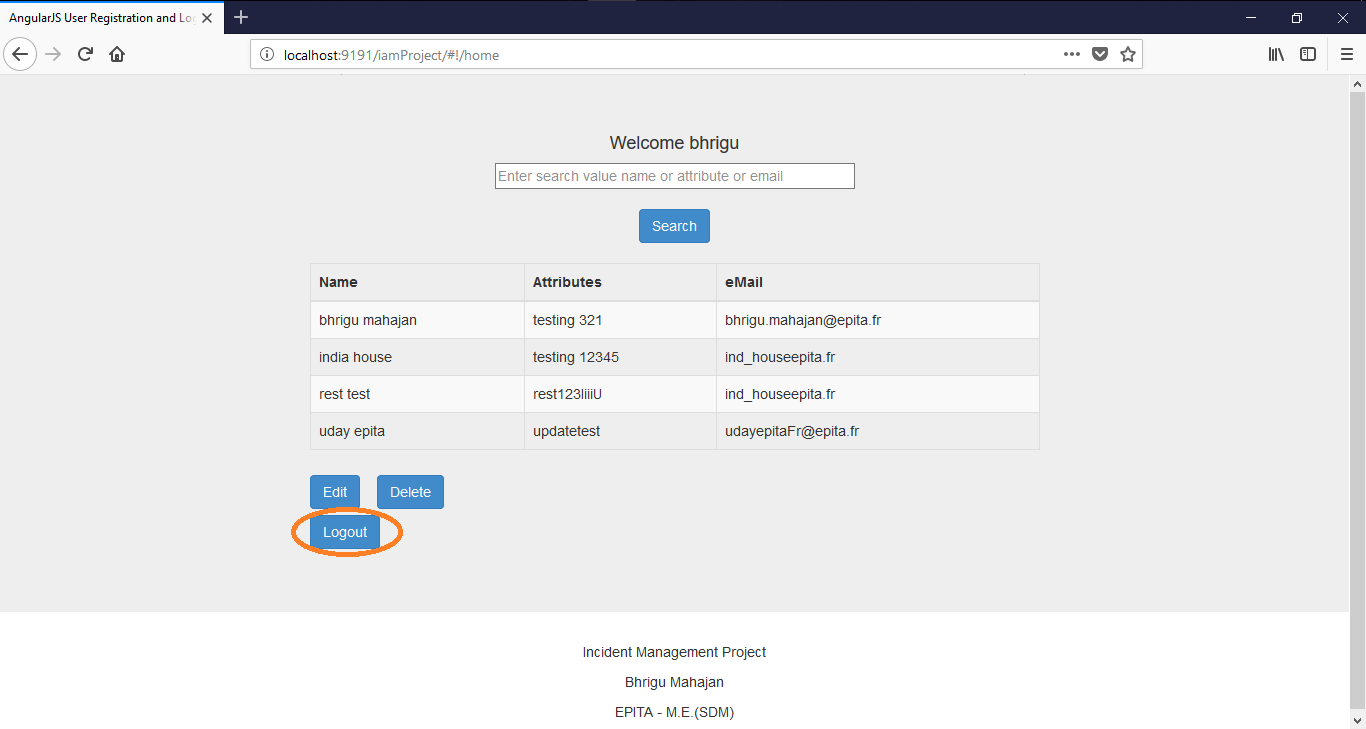
### **Option: Search**

This option is to search anything Name, Attribute, email by search button.



### **Option: Logout**

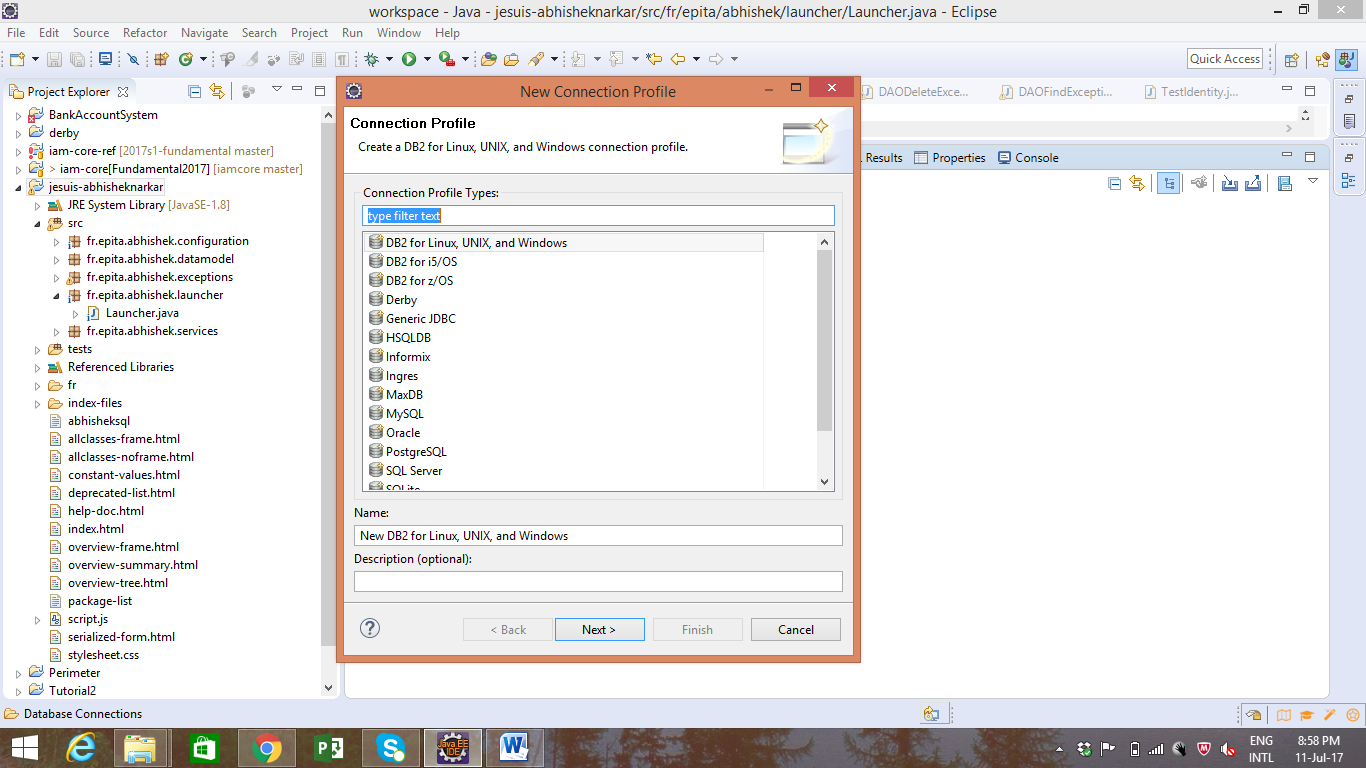
Exit to login screen.



# **Database Configuration**

## **Prerequisites**

Please make sure you have installed the Java JDK 8, Eclipse oxygen Java EE and the Derby 10.1.



1. In the Data Source Explorer window select Database Connections tab right click the select new database select Derby. Click Next.
2. In the new window click on the “New Driver definition button”
3. Select any Definition on the Name/Type tab and then move to the JAR List tab.
4. On the JAR List tab click on Clear all button, so no files are listed. Then click on the Add and navigate to the lib folder inside your Derby database. Once there select the derbyclient.jar file.
5. Click Open, then the “New Derby Connection Profile”.
6. Click on Test Connection button, if succeed then connection is successful.
7. Click on Finish button.

**Bibliography**

**Websites:**

<http://thomas-broussard.fr/work/java/courses/project/fundamental.xhtml>