Documentation:

1. Subject description

Built using servlets

The Project - The project focusses on managing identities and is built surrounding the core idea to handle all operations to manipulate the list of Identities. Javadoc will be built into the project to increase the understandability and reusability.

We have used the model, view and controller (MVC) model to structure the application. Possible operations to manipulate the Idenity are the create, read, update and delete. A seperate test class has been written in the test folder to test all the test methods.

#Users - The users are divided into two, The privileged users and the non-privileged users. The details of the both the types of users are maintained in different database tables.

#Flow - The privileged user needs to login to the application to be able to access, modify, delete or create new Identities. Hence the application is designed such that all operations are restricted only to the privileged users. The flow happens once the privileged user logs in, the control flows through the controller to the service and then to the dao layer to interact with the db and tip-toes back to the controller to make further judgements based on the value returned by the method.

#Logger -

The logger has been included into the project as a library in the name of logger-iam-core.

#Comments -

Every effort has been made to richly comment possibly everything in the project which is a boon and can be seen and confirmed in the javadocs.

#Things still left to be done-

Jmeter testing to see the performance of the application.

Making the web pages responsive.

1. Subject Analysis
   1. Major Features:

One of the major feature is the reusable API. The library can be packaged into a new project and using the LoginService and the IdentityService, the same project can be user to cater to other projects or developers to interact with the database.

* 1. Application Feasibility:

The application is feasible as it’s easily possible to create a login system and then permit access to modify certain records in the database with the help of java.

* 1. Data Description:

The data being manipulated on the identity object. The object contains a string displayName , string uid and a string email.

* 1. Expected Results:

Easily manipulate the list of records in the database without any problems. Login to the portal and check the state of the objects persisted in the database.

* 1. Algorithm study:

We have not used any special algorithm in the project. One design that we decided to have equalIgnorecase match for the user email while logging into the application while using case-senstive equals for the password. Therefore we have enforced a strict policy on the password while we are lenient on the email or user id.

* 1. Scope of the application (limits and evolution)

We would like to test the application using jmeter to check the limitations and efficiency of the application. We would also like to use qr code to have one more level of security while logging into the application and the POC for the QR is ready and tested. Incorporating this into the application and having a mobile version of the same application would be cool.

1. Conception
   1. Choosen algorithm:

The design is something we could discuss here. We have a login controller which logs in the user and once the login in successful we have prepared a documented identityService class which will house all the operations related to the identity. The service class in turn calls the dao class to interact with the db. Therefore the design is an example of the MVC pattern.

* 1. Data Structures:

The data structures used are List’s when the search operation is hit by the controller. And a list of Identity will be returned by the dao method to the controller via the service class.

* 1. Global Application Flow:

A three-tier application in which the user requests are captured by the controller and then forwarded to the service and then to the dao which is responsible to persist or retrieve the data. The control then flows backwards from the dao to the service and then to the controller and then the model or jsp is populated with appropriate information.

Service

Controller

Dao

Database

user

* 1. Global Schema

The application has two tables, The LOGIN and the IDENTITIES table. The login table contains the list of users with priveliged users and the IDENTITIES table contains

The below DDL’s indicate the schema information.

CREATE TABLE IDENTITIES

(ID INT NOT NULL GENERATED ALWAYS AS IDENTITY

CONSTRAINT IDENTITY\_PK PRIMARY KEY,

DISPLAY\_NAME VARCHAR(255),

EMAIL VARCHAR(255),

UID VARCHAR(255)

);

create table login(

login\_email varchar(50) NOT NULL UNIQUE,

password VARCHAR(50) NOT NULL

)

There for the identities table contain the id which is always generated and the display name and an email and the uid.

The login table contains just the login email and password.

1. Console operation description:

The application is not designed to work using the console.

1. Configuration instructions:

You would need to add an admin to the LOGIN Table before logging into the application and testing the operations.

You would need to configure the System property as conf and set it to a folder in your disk containing the db properties as a key value pair. Please be careful to use the keys in capital letters.

Sample :

DRIVER=org.apache.derby.jdbc.ClientDriver

USER=root

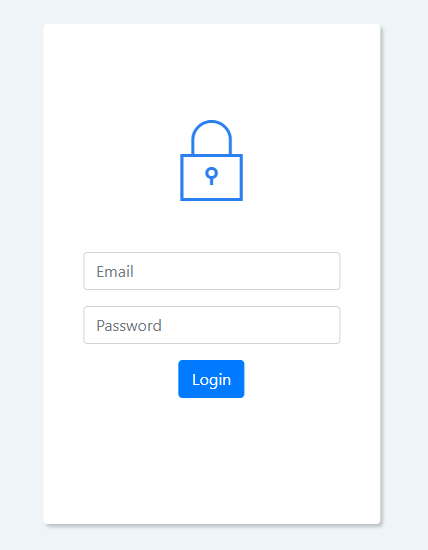
PASSWORD=root

URL=jdbc:derby://localhost:1527/iam-core;create=true

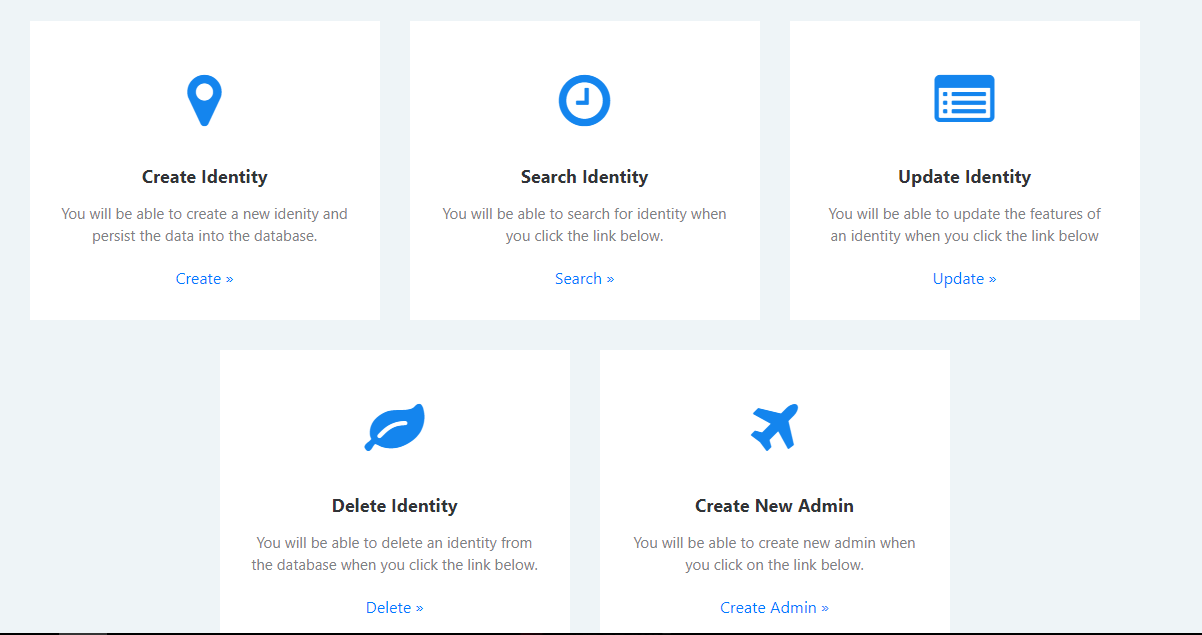
The database needs to be running to test the application.

1. Commented screenshots:

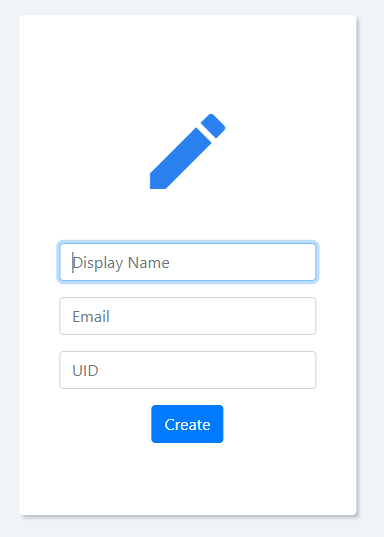
Login Screen



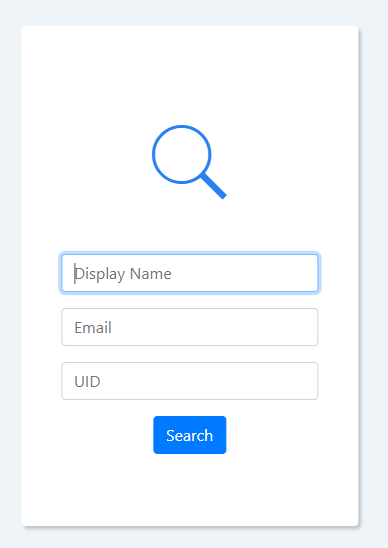
Admin Screen



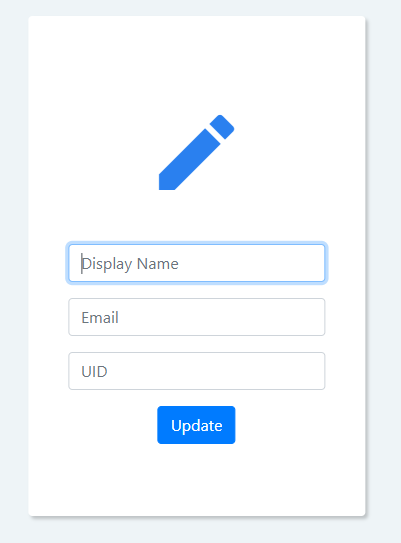
Create Screen



Search Screen



Update Screen



Delete Screen

Etc etc

1. Bibliography

Just referred this website <http://thomas-broussard.fr>