Shantanu Kamble

EPITA | Msc spring 2018

kambleshan04@gmail.com

Identity Access Management

Technical specification Documentation

Table of Contents

[**1.** **Subject Description** 1](#_Toc519268420)

[**2.** **Subject Analysis** 2](#_Toc519268421)

[**2.1** **Major Features** 2](#_Toc519268422)

[**2.2** **Application Feasibility** 3](#_Toc519268423)

[**2.3** **Data Description** 3](#_Toc519268424)

[**2.4** **Expected Results** 4](#_Toc519268425)

[**2.5** **Algorithm Study** 4](#_Toc519268426)

[**2.6** **Scope of Application** 4](#_Toc519268427)

[**3.** **Conception** 5](#_Toc519268428)

[**3.1** **Chosen Algorithm** 5](#_Toc519268429)

[**3.2** **Data Structures** 6](#_Toc519268430)

[**3.3** **Global Application Flow** 6](#_Toc519268431)

[**3.4** **Global Schema and Major Features Schema** 6](#_Toc519268432)

[**4.** **Console Operations Description** 7](#_Toc519268433)

[**4.1** **Authentication** 7](#_Toc519268434)

[**4.2** **Creating an Identity** 8](#_Toc519268435)

[**4.3** **Updating an Identity** 8](#_Toc519268436)

[**4.4** **Searching an Identity** 8](#_Toc519268437)

[**4.5** **Deleting an Identity** 8](#_Toc519268438)

[**5.** **Configuration Instructions** 8](#_Toc519268439)

[**5.1** **Development Tools** 8](#_Toc519268440)

[**5.2** **Schema and Database Configuration** 9](#_Toc519268441)

[**6.** **Commented Screenshots** 9](#_Toc519268442)

[**7.** **Bibliography** 10](#_Toc519268443)

## **Subject Description**

The identity and access management project implements basic features of CRUD system. It consists of two major elements, authentication of user and identity management. The project defines the user identity life cycle within an organization. It creates, or establishes the user identity, deals with user identity related operations, and finally the destruction of the user identity within the organization.

The project is implemented using JAVA language and allows the user to communicate with the database using a Console. The user will be presented with a series of inputs to login into the system and then is shown a list of options to create, update, search or delete an identity.

## **Subject Analysis**

### **Major Features**

The major advantage of this project is that it is reusable and configurable. This project can be used to cater the needs of other projects that involve database interaction. Some of the features of this project can be listed below:

* User Authentication
* Creating an Identity
* Updating an Identity
* Searching an Identity
* Deleting an Identity

### **Application Feasibility**

* The console input allows the user to freely interact with the system and provides an easy way to maintain identities of users.
* The IAM project is developed using Java, Derby database and Eclipse IDE which are easily available to all.
* This application follows a strategic approach, easy configuration and self-explanatory error messages.

### **Data Description**

The data stored by the application is related to identity and information about the user. The structure of data stored in the database is as below.

**Users table**

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | Data type | Length | Description |
| USER\_ID | Integer | 10 | This Id is a unique identifier for a user |
| USER\_USERNAME | Varchar | 55 | This is the username of the user |
| USER\_PWD | Varchar | 55 | This is the password of the user |

**Identities table**

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | Data type | Length | Description |
| IDENTITY\_ID | Integer | 10 | This Id is a unique identifier for an identity |
| IDENTITY \_DISPLAYNAME | Varchar | 55 | This is the display name for an identity |
| IDENTITY \_EMAIL | Varchar | 55 | This is the email for an identity |
| IDENTITY\_UID | Varchar | 55 | This is the unique id for an identity |

### **Expected Results**

* User authentication: The system/application would prompt the user for login credentials. When the user enters the credentials, it will validate the credentials against the data stored in the Users table.
* Identity management: When the user is successfully authenticated, he/she is shown different options for identity management. The options are: Create, Search, Update, Delete and Exit.

### **Algorithm Study**

The application is a simple console-based one. We have not used any special algorithm here. For searching an identity, we have implemented a LIKE clause feature which can compare a part of search string entered by the user, to search for a particular thing. This makes the search easier and flexible.

### **Scope of Application**

The primary focus of this application is on identity management. So, it does not consider the user management functionality. As the identities are stored in a database, the user needs to install a database to use this application.

Limitation:

* The system is based on console format, a User Interface would be more appealing for new users.
* The password of the user is currently stored in plain text format in the database. So, there is a lack of password encryption.
* Maintaining one identity at a time would be time-consuming job considering there are numerous number to be maintained as a part of this system.

Evolution:

* Encryption of the passwords stored in database.
* Concept of user and identity can be integrated by adding functionality to assign roles to the user based on the operation.
* User Interface can be developed with the functionality to add multiple identities at same time.

## **Conception**

### **Chosen Algorithm**

Here we can discuss about the design of the application.

* The application follows factory pattern for managing the identities. The advantage of using this design is that we can have multiple implementations for the identity.

For example, instead of database we can have XML based approach, or a File based approach to store the identities.

* The application is made configurable by using a Configuration Service which takes database properties from a properties file.
* The application uses log4j library for logging purpose. The logging configuration is stored in the file named “log4j2.properties”.

### **Data Structures**

Data structure used in this project is LIST, which is a collection of elements. List is used in the project to store the collection of identities. String is also a widely used data structure in this project.

### **Global Application Flow**

It is a three-tier application in which the user requests are captured by the launcher with help of the Console operations. The Configuration service provides the required properties to the factory. The DAO is responsible for data retrieval and its manipulation. The DAO interacts with the database and sends back the required information. The retrieved information is then displayed to the user.

USER

Dao

Service

Controller

Database

### **Global Schema and Major Features Schema**

The application has two tables, USERS and IDENTITIES. The users table contains the list of users allowed to access the application and the IDENTITIES table contains information of user’s identity like uid, display name and email.

The below DDL’s indicate the schema information.

CREATE TABLE IDENTITIES

(IDENTITY\_ID INT NOT NULL GENERATED ALWAYS AS IDENTITY

CONSTRAINT IDENTITY\_PK PRIMARY KEY,

IDENTITY\_DISPLAYNAME VARCHAR(55),

IDENTITY\_EMAIL VARCHAR(55),

IDENTITY\_UID VARCHAR(55)

);

CREATE TABLE USERS

(USER\_ID INT NOT NULL GENERATED ALWAYS AS IDENTITY

CONSTRAINT USER\_PK PRIMARY KEY,

USER\_USERNAME VARCHAR(55),

USER\_PWD VARCHAR(55)

);

DAOs

Create

Search

Update

Delete

Identity

User

## **Console Operations Description**

### **Authentication**

* This operation is responsible for authenticating the user credentials (i.e. Username and Password). The authentication is done against the credentials stored in the database table USERS.
* The Launcher (controller class) makes a call to the JDBCUserDAO method named checkLogin(). This method is a boolean method which will return True if user credentials match else it will return false.

### **Creating an Identity**

* This operation is responsible for creating an identity and storing it in the database. It takes the uid, display name and email as input from the user.
* The Launcher class makes a call to the JDBCIdentityDAO method named create().
* This method throws EntityCreationException which is a custom exception, thrown in case of any error in identity creation.

### **Updating an Identity**

* This operation is responsible for updating an identity and storing it in the database. It takes the uid as a key, display name and email as input from the user.
* The Launcher class makes a call to the JDBCIdentityDAO method named update().
* This method throws EntityUpdateException which is a custom exception, thrown in case of any error in identity updation.

### **Searching an Identity**

* This operation is responsible for searching an identity from the database. It takes the display name and email as input from the user for search criteria.
* The Launcher class makes a call to the JDBCIdentityDAO method named search().
* This method throws EntitySearchException which is a custom exception, thrown in case of any error while searching for the identity.

### **Deleting an Identity**

* This operation is responsible for deleting an identity from the database. It takes the uid as input from the user.
* The Launcher class makes a call to the JDBCIdentityDAO method named delete().
* This method throws EntityDeletionException which is a custom exception, thrown in case of any error in identity deletion.

## **Configuration Instructions**

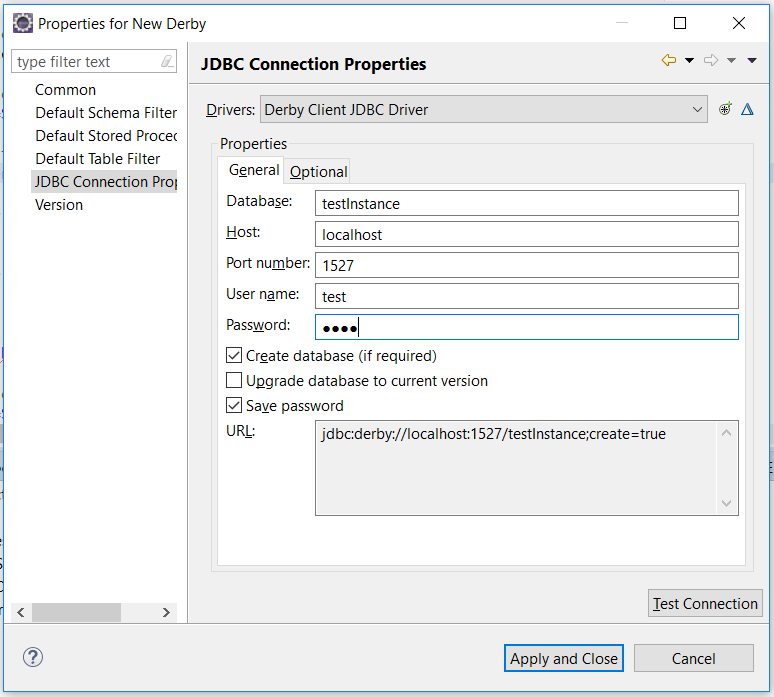
### **Development Tools**

* Eclipse IDE (Java EE IDE for Web Developers, version: Oxygen)
* Derby database (10.14.2)
* Java JDK 8
* Log4j libraries for logging
  1. Log4j-api-2.11.0
  2. Log4j-core-2.11.0

### **Schema and Database Configuration**

* The application mainly uses two database tables: “USERS” for user management and “IDENTITIES” for managing the identities.
* There is a “SQL” folder which contains all the required queries to create the tables.
* You need to manually create an entry for admin in the table “Users” for logging into the application.
* The configuration file “db-config.properties” file contains all the required database properties and different queries.
* Following are the database properties:

|  |  |
| --- | --- |
| Database properties | Values |
| Host | localhost |
| Port | 1527 |
| Username | test |
| Password | test |
| URL | jdbc:derby://localhost:1527/testInstance;create=true |



## **Commented Screenshots**

These screenshots are covered as part of user guide.

## **Bibliography**

* Object Oriented concepts and project scope: <http://thomas-broussard.fr/>
* For programming: <https://stackoverflow.com>