PROJECT: **IAM CORE**

Project owners: Donatien Adjé KOUAKEY & Ebaneck ATUH

Professor Name : Thomas Broussard

Course name : Introduction to Java

February 2018

Table of Contents

Introduction 3

Subject description 3

Subject analysis 4
- Major features 4
- Application Feasibility 4
- Data description 4
- Expected results 6
- Algorithms study 6
- Scope of the application (limits, evolutions) 7

Conception 7
- Chosen algorithm 7
- Data structures 8
- Global application flow 9
- Global schema and major features schema 10

Console operations description 14

Configuration instructions 15

Commented Screenshots 27

Future Prospects 32

Conclusion 32

Work Cited 32

Introduction

The goal of this project is to allow us realise a typical java project after completing the theoretical course Introduction to Java programming. This project has a good covering in the Java language usage nowadays, We have learned things, like creating and using the Java Developer Kit APIs, but also to persist data in databases. The subject of Identity Management has been chosen because it will put us in the Business application world, which is what is needed at this point in time in our careers.

1. Subject description

The subject of Identity Management has been chosen because it put us (the students) in the Business application world, which is what is needed at this point in time in our careers.

The main goal of the project is to design and build an information system to manage users. The overall concept is generic but can be improved with security optimizations. Our application will be able to accomplish the following task;

- Connect to a derby database instance
- Authenticate login requests against a database table
- Create user/password records in the database
- Store all user password using md5 encryption
- Access, create and modify user specific information
- Continually persist user data within a database

2. Subject analysis

2.1. Major features

As mentioned above, the project obviously has interesting features which are: create an identity, update an identity, delete an identity and view the list of identities.

Our application is still generic and hence contains no user interface so the main form of interaction will be through the command line.

2.2. Application Feasibility

Our design application is generic and can be used, extended or adapted to other applications to manage and handle user creation, identity search/find and finally authentication.

2.3. Data description

Our application is built with the following schema/dataset in mind with a set of actions to be performed.

- Display name
- UID   All as string elements
- Email

Authentication

- Username
- Password

Database Description

Database Name - DATABASE_NAME=”iam-core”

Table Name- ROOT Schema=”identities” “üsers”

Schema for Identities

Field name	**Data type**	**Description**	**Example**
Display name	VARCHAR	Typical name of the user	User demo
email	VARCHAR	Users email address	user@gmail.com
UID	VARCHAR	Numeric id	FR1010
Identity_id	INTEGER	Auto generated and primary key	002

Schema for Admin authentication(users)

Field name	**Data type**	**Description**	**Example**
username	VARCHAR	Typical alphabetical characters	admin
pass_salt	VARCHAR	a string of 16 random bytes	ldlkajdkajdkajkdjada
pass_md5	VARCHAR	binary MD5 hash of pass_salt concatenated with the password	cccccccccccccaka09a39jj33903
userid	INTEGER	Auto generated as primary key	001

2.4. Expected results

The expected results of our application are that an administrator should be able to use our application to manage, perform some actions on identities(create, update, delete, etc) and more especially view the current list of identities.

2.5. Algorithms study

This section describes all the methods, procedures or techniques used in designing and building the application.

2. 5. 1 Selection

The selection approach is performed by the user and only the user once he is signed in into the application. Each of the operations (create, update, delete and view) involved in the application are done according to the choice/selection and action of the user.

2.5.2 Iteration

The iteration is applied to the set of instructions which we want to get repeatedly executed**.** The authentication process of the user follows an iterative approach. When the user inputs an incorrect username and password, he is asked the retry. This is what explains the iteration/repetition. Also, the Menu is repeatedly displayed, giving the chance to the user to perform another action if he wants to.

2.5.3 Recursive:

The recursive approach is applied to all the view operations. The table is loaded with new entries when a creation operation has been performed.

2.5.4 Random

The approach is applied to the passwords provided by the users thanks to MD5 encryption. The Md5 encryption transforms the passwords into a random characters.

2.6. Scope of the application (limits, evolutions)

The Application, although having interesting features has some limitations. It is restricted only to one user; an administrator whose task is to manage identities. The application does not give the option to create an account and be part of all the identity management process. The Fields involved in the process are only 4 and no more. These fields are Name, the User id, the Email and the Address.

The current state of our application makes it impossible to reset an admin account if this is forgotten. We will continue to improve this part of the application

Also, our application is currently only available from the terminal, we plan to build a proper GUI for our application in future.

3. Conception

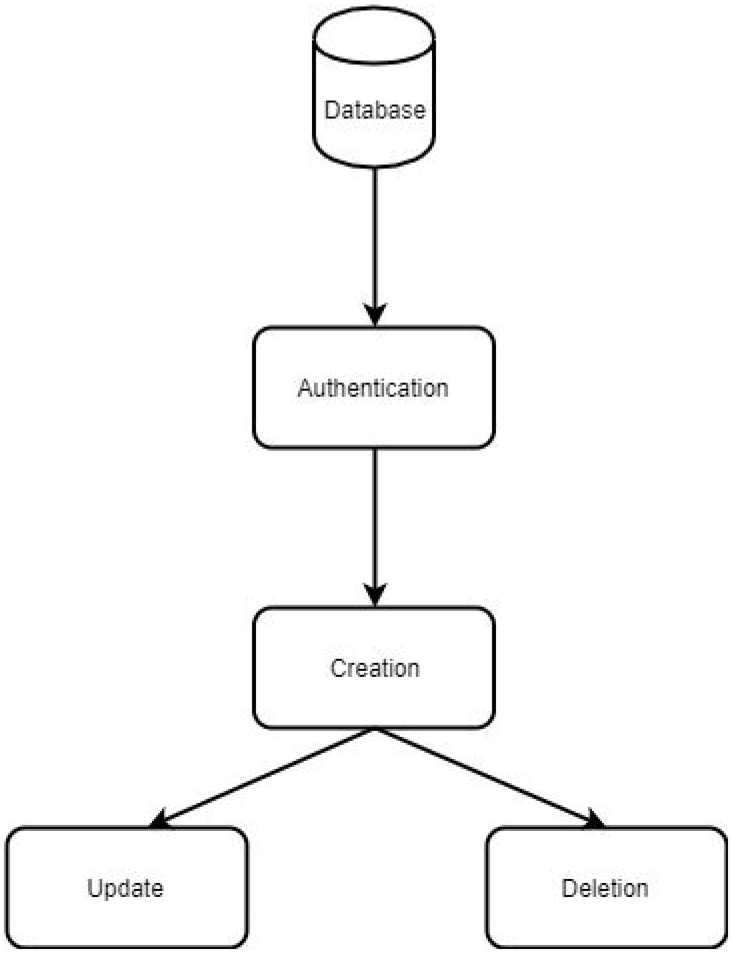
3.1. Chosen algorithm

```
{Start}
{Authentication: login}
    {Data}
        Entry of the username
        Entry of the password
    {Checking}
```

```
            The  credentials are checked. If they exist, the Menu is displayed to the user.
            When he/she  fails to input the correct credentials, he is asked to try again.
            The application closes if he fails again.
        { Menu: Only if authentication successful}
            Entry of the number corresponding to the option wanted
            {Checking}
            The entry is checked against the options numbers. If the entry corresponds to one
            of the options, the operation for the option is performed.
{End}
```

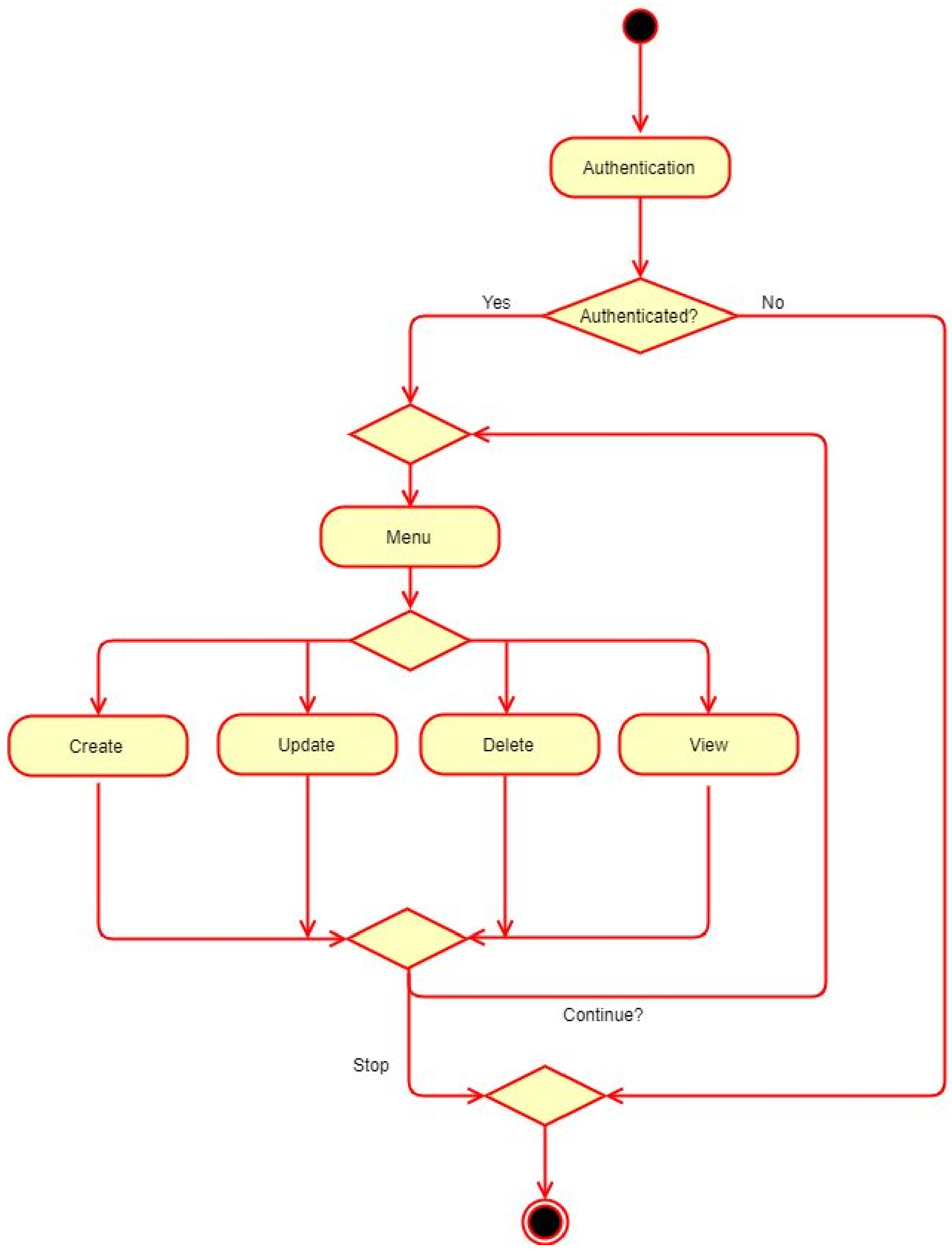
3.2. Data structures

In designing and creating the iam core application, we opted for the **Tree-like structure**. This is justified because**,** the application comports a scenario which authenticates a user, and makes him/her use the Identity management through predefined methods such as Create, Update and Delete. Before one will be able perform any action or operation, he/she must have an account. Also, before being able to update or delete, there has to be first of all an identity created in the database (there must be something to update or delete).The Tree structure could be represented as shown below:



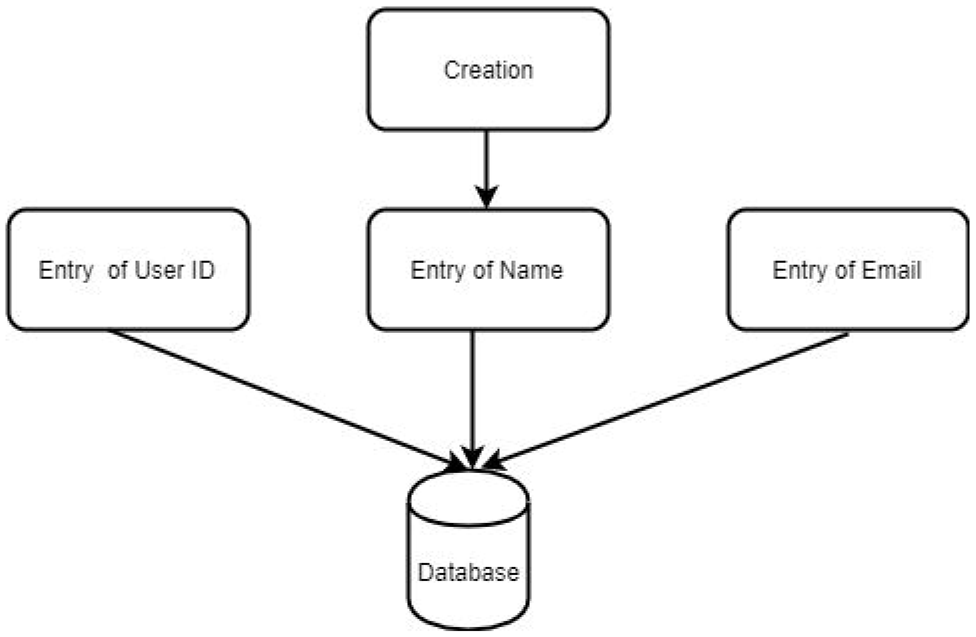
3.3. Global application flow

The sequence diagram below shows the global flow of events within the Iam core application from start to finish.

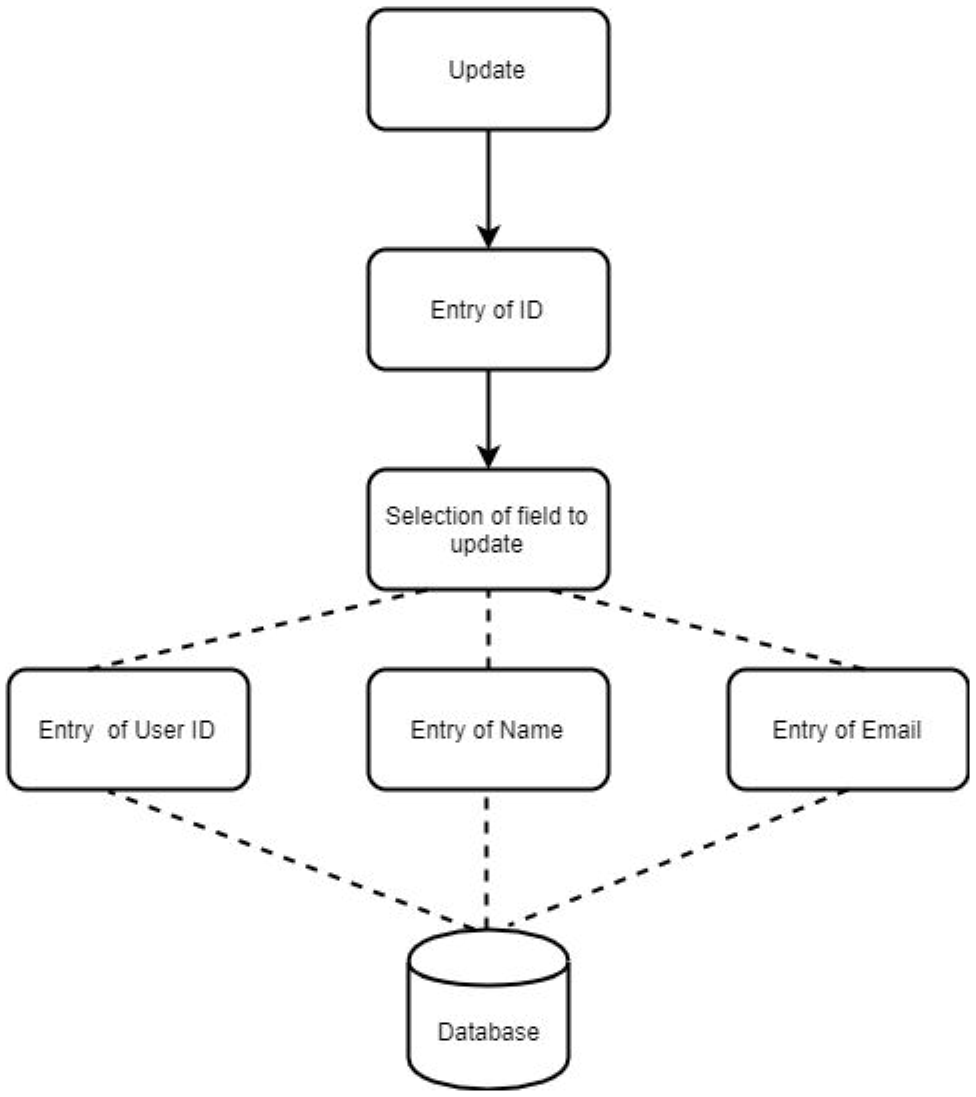


3.4. Global schema and major features schema

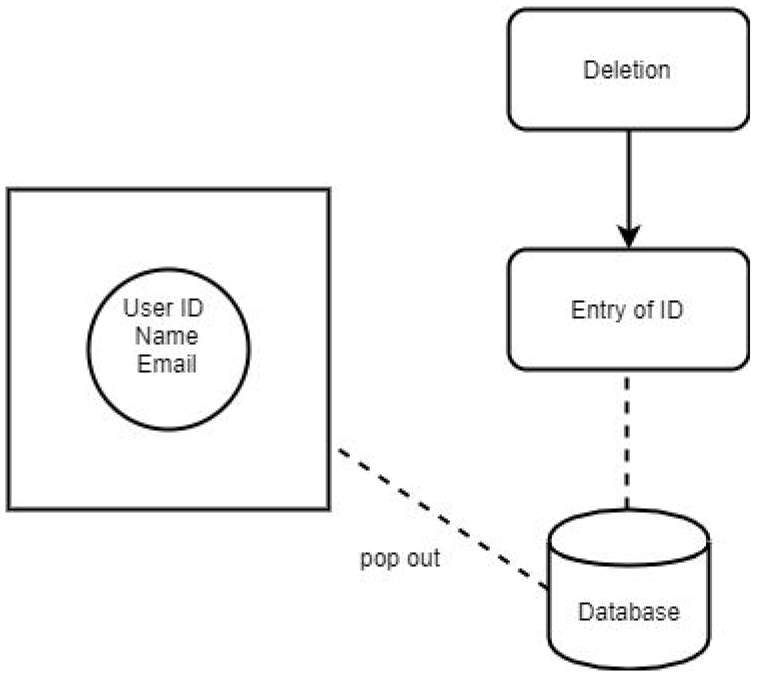
Create feature schema



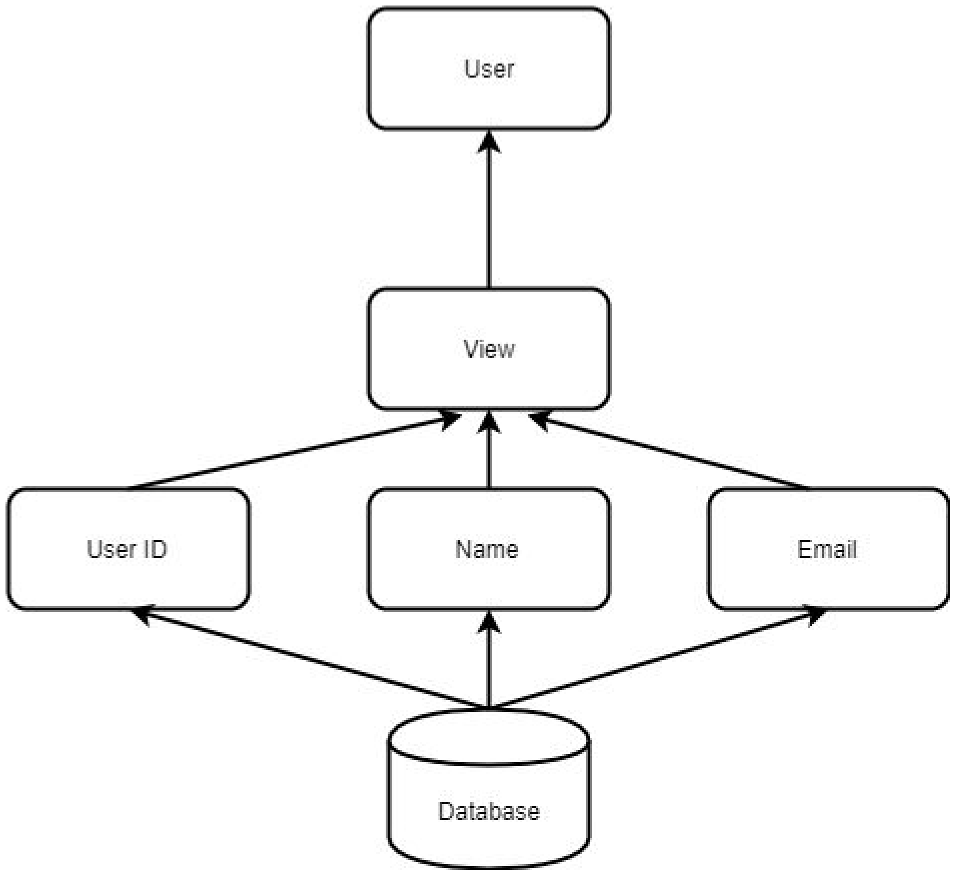
Update feature schema



<u>Delete feature schema</u>



View feature schema



4. Console operations description

The console operation consist of the following synchronous activities that will run one after the other:

I. Authentication

- Application is launched by the user using the created bat file(windows users only)
- Application prompts the user to create an administrator account
- User creates an admin account
- User is asked to login with the admin account just created(username and password)
- Welcome menu is displayed upon successfully authenticating or a failure message if provide credentials are not valid

II. Create operation

- User is given the option to choose the creation option by entering (1.)
- Application prompts user to enter identity details like (user ID, display name and email)
- User enters the information and an Identity is created.
- User is prompted to save the created credentials or start over the creation process
- Upon accepting, an identity is saved in the database as entered by the user
- The saved information is also immediately printed to show the user the operation has been executed.

III. Update operation

- The Update option is made available to the User by entering (2)

- The Application displays the list of all the existing identities and then prompts the user to choose the one he/she wants to update by entering the ID of the corresponding identity.
- Next, the application prompts him to enter the field he/she wants to update by entering the corresponding number.
- Finally, he/she enters something new to replace the old one.

IV. Delete operation

- The user is given the option to delete by entering (3)
- The existing identities are displayed to the user.
- The application prompts the user to enter the User ID referring to the row or record he wants to delete.
- He enters the user ID and the record is then deleted.

V. View all Identities

- The Option 4 is the one corresponding to (view) the list of all identities. Once the user enters (4), the list of identities are displayed.

VI. Exit

- The user enters 5 to close the application when he wants to quit.

5. Configuration instructions

This section contains information on how to configure and run the application on a new windows computer having java 8 installed and running. We assume that the user has setup java jdk1.7 and this installation is present with the environmental path.

Users could verify this by running **java -version** in the terminal window

This section will be based on database configuration and finally the **.ba**t file execution.

1. **Database setup and integration step.**

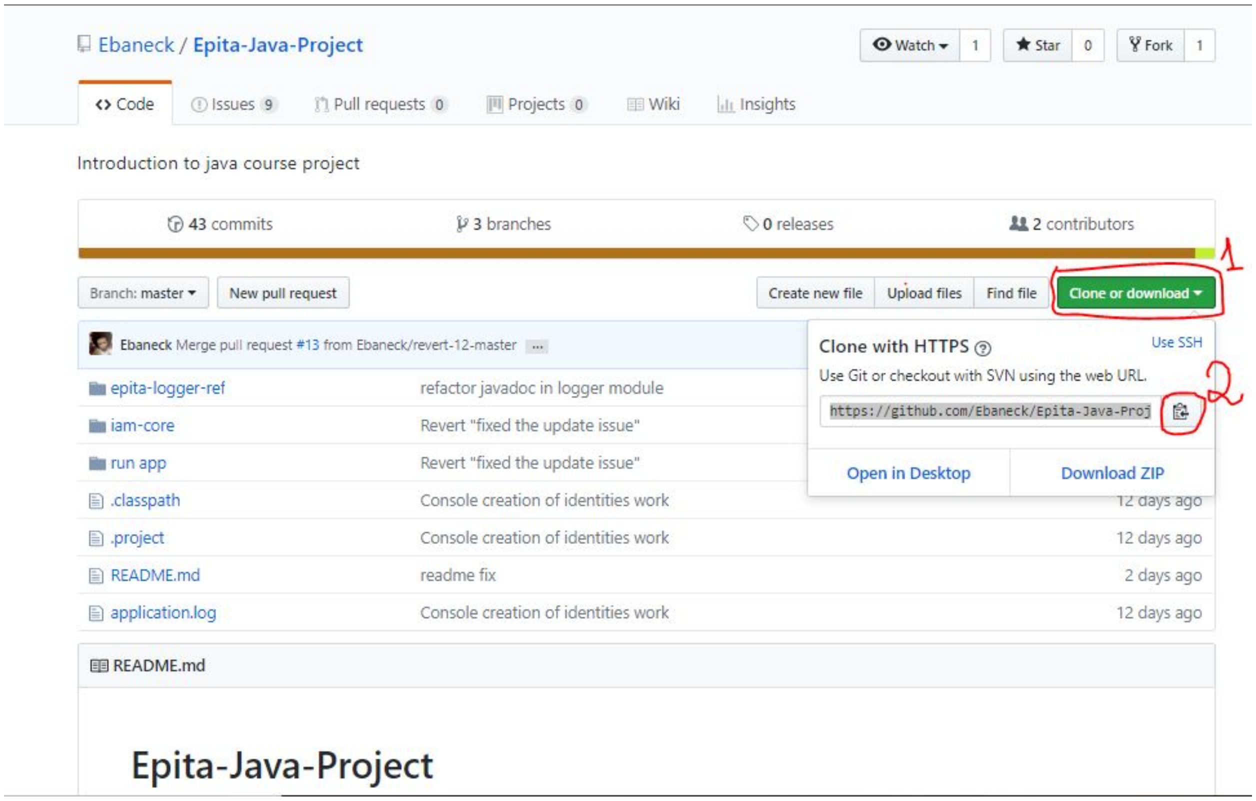
- We are going to add derby database as a dependency to the github repo(easy to download with the entire project).
- The provided executable bat file is configured to Run the derby Networksetup.bat file to start the derby instance
- The project is packaged with a database already, hence no special need to Create a database, user and password as follows (iam-core, root, password)

2. **Running the windows .bat**

- On a windows based pc, run the file named **execute.bat** in the project dir
- The application should run normally until the exit button is touched

Eclipse Setup instructions (Video presentation might be included)

01. Import the project into eclipse by cloning the entire project from this link : **https://github.com/Ebaneck/Epita-Java-Project.git**



Introduction to java course project

Ebaneck Merge pull request #13 from Ebaneck/revert-12-master

Use Git or checkout with SVN using the web URL.

https://github.com/Ebaneck/Epita-Java-Proj

epita-logger-ref	refactor javadoc in logger module	
iam-core	Revert "fixed the update issue"	
run app	Revert "fixed the update issue"	
.classpath	Console creation of identities work	12 days ago
.project	Console creation of identities work	12 days ago
README.md	readme fix	2 days ago
application.log	Console creation of identities work	12 days ago

Epita-Java-Project

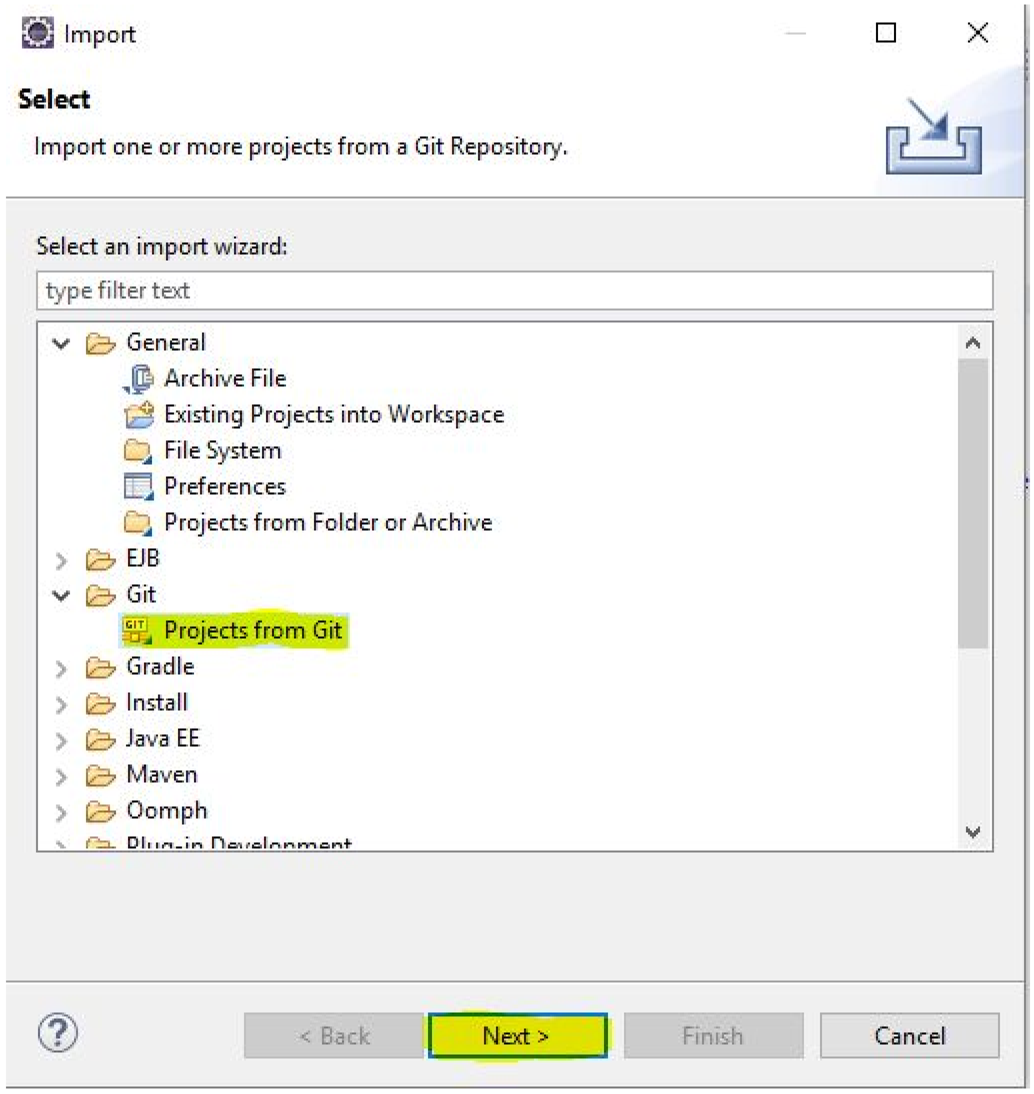
Fig 1.

The steps required in cloning the project is in the above screenshot (fig1.). Once the user has clicked on the link, he will have access to the above page. Then he needs to click on “Clone or Download” and finally copy to clipboard by clicking on the icon referred as 2 on fig 1.

After this, he will need to go into Eclipse and follow these steps:

a-) Click on Import in the File Tab

b-)Select “Project from Git” in the Git folder and click on Next



Import

Select

Import one or more projects from a Git Repository.

Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
  - Projects from Folder or Archive
- EJB
- Git
  - Projects from Git
- Gradle
- Install
- Java EE
- Maven
- Oomph
- Plug-in Development

Fig 2.

c-) Select "Clone URI" and Click on Next

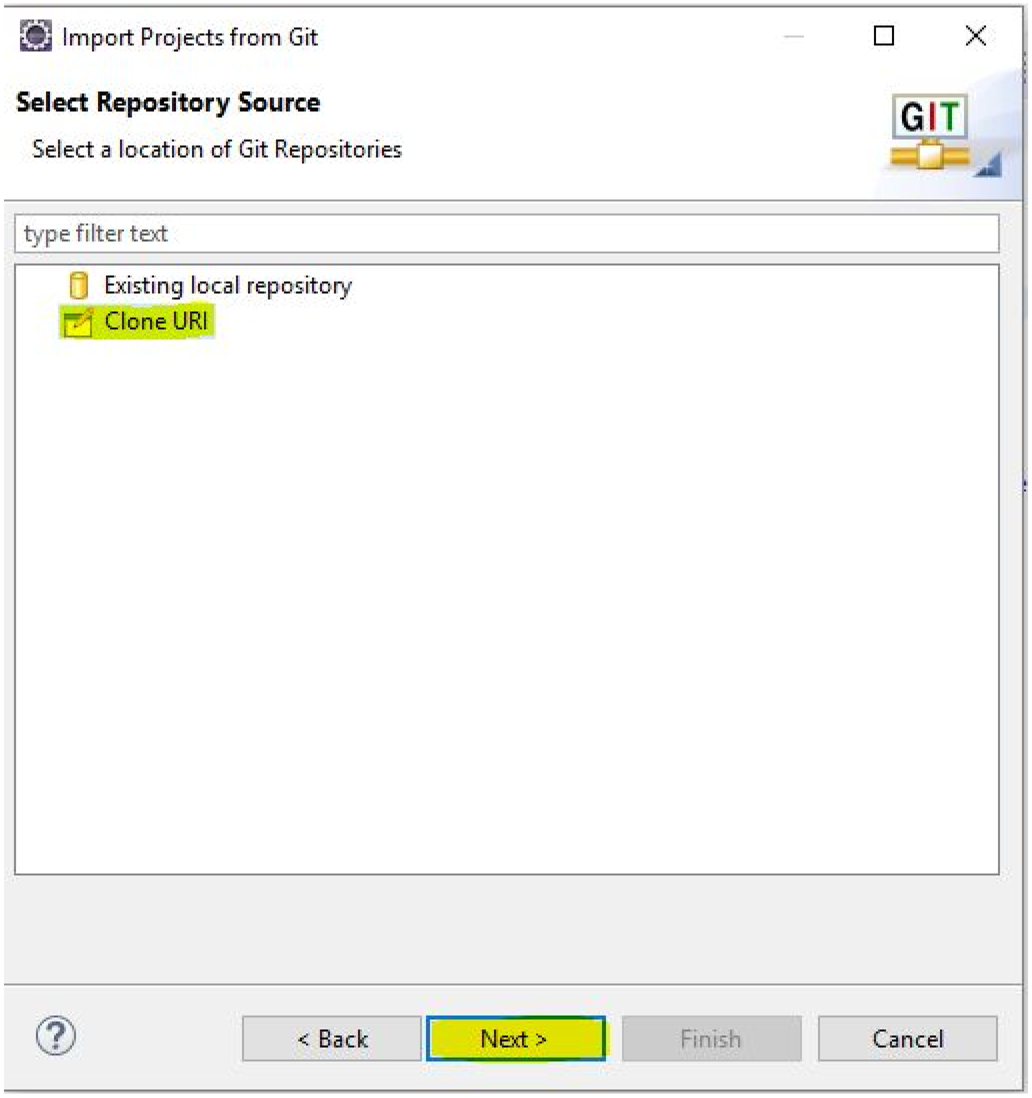
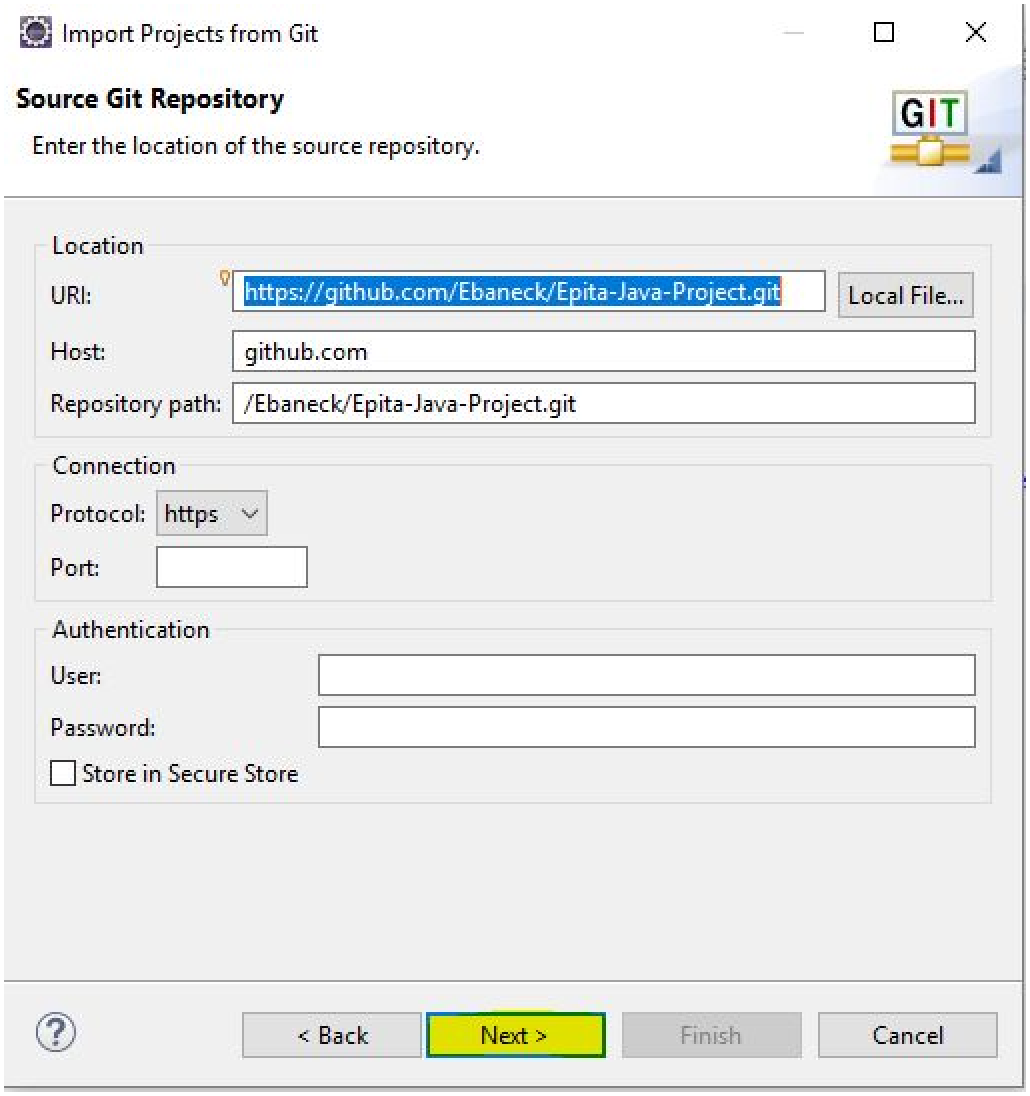


Fig 3.

d-)After copying to clipboard, the URI, Host and Repository Path are automatically field up and then Click on Next and the Project will appear inside your eclipse



URI: https://github.com/Ebaneck/Epita-Java-Project.git
Host: github.com
Repository path: /Ebaneck/Epita-Java-Project.git
Protocol: https

Fig 4.

02. Move into the derby client folder located inside the main project directory and run this file found in bin directory **startNetworkServer.bat**

03. In eclipse, create a new derby database instance, specifying the database name, username and password. These are the steps required:

a-) In the Window tab, Click on Show view and then Other; a search bar will be provided and type Data Source Explorer and then click it.

b-) Right Click on DataBase Connections, Click on New.

c-) Select Derby and then Click on Next

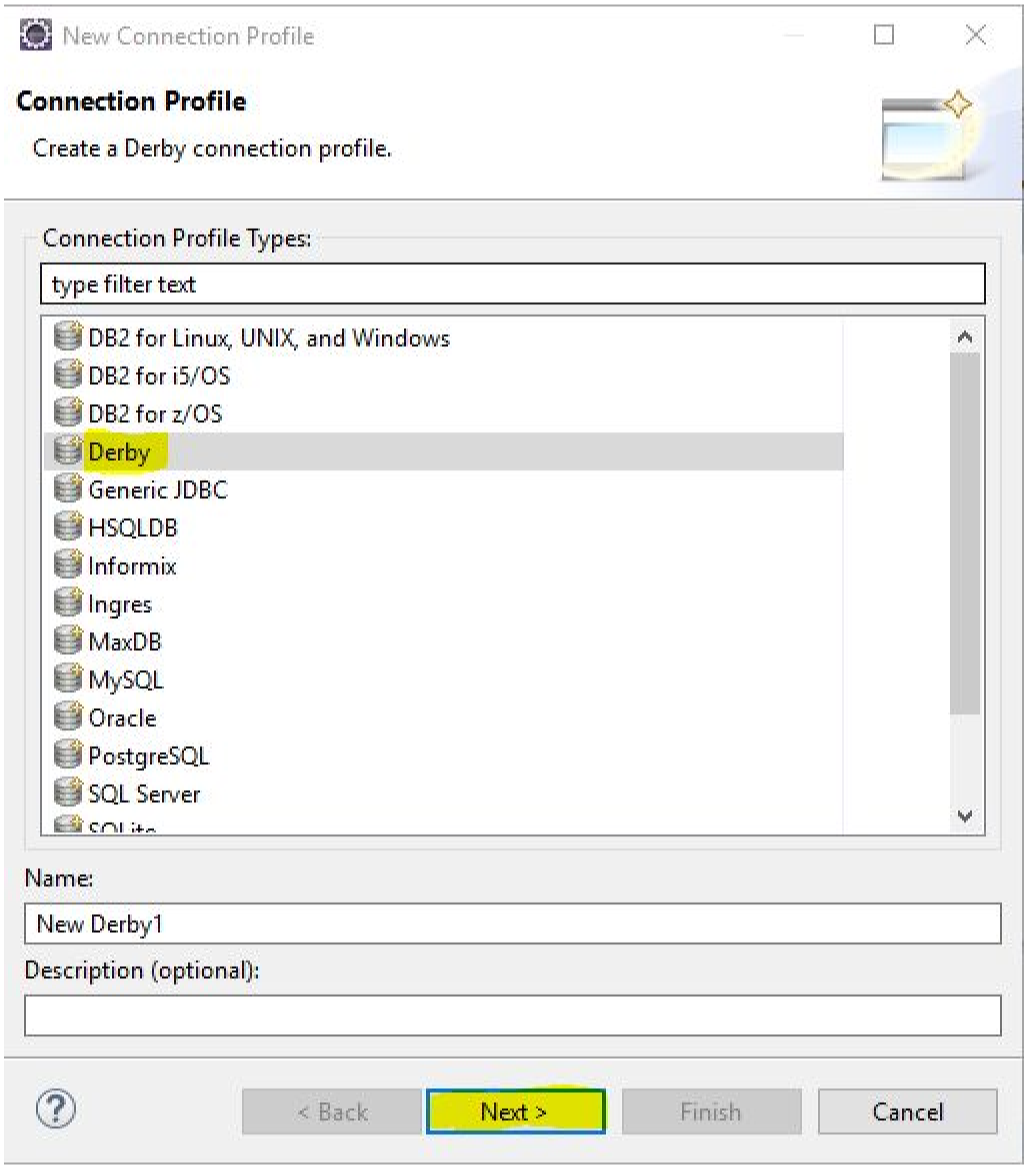
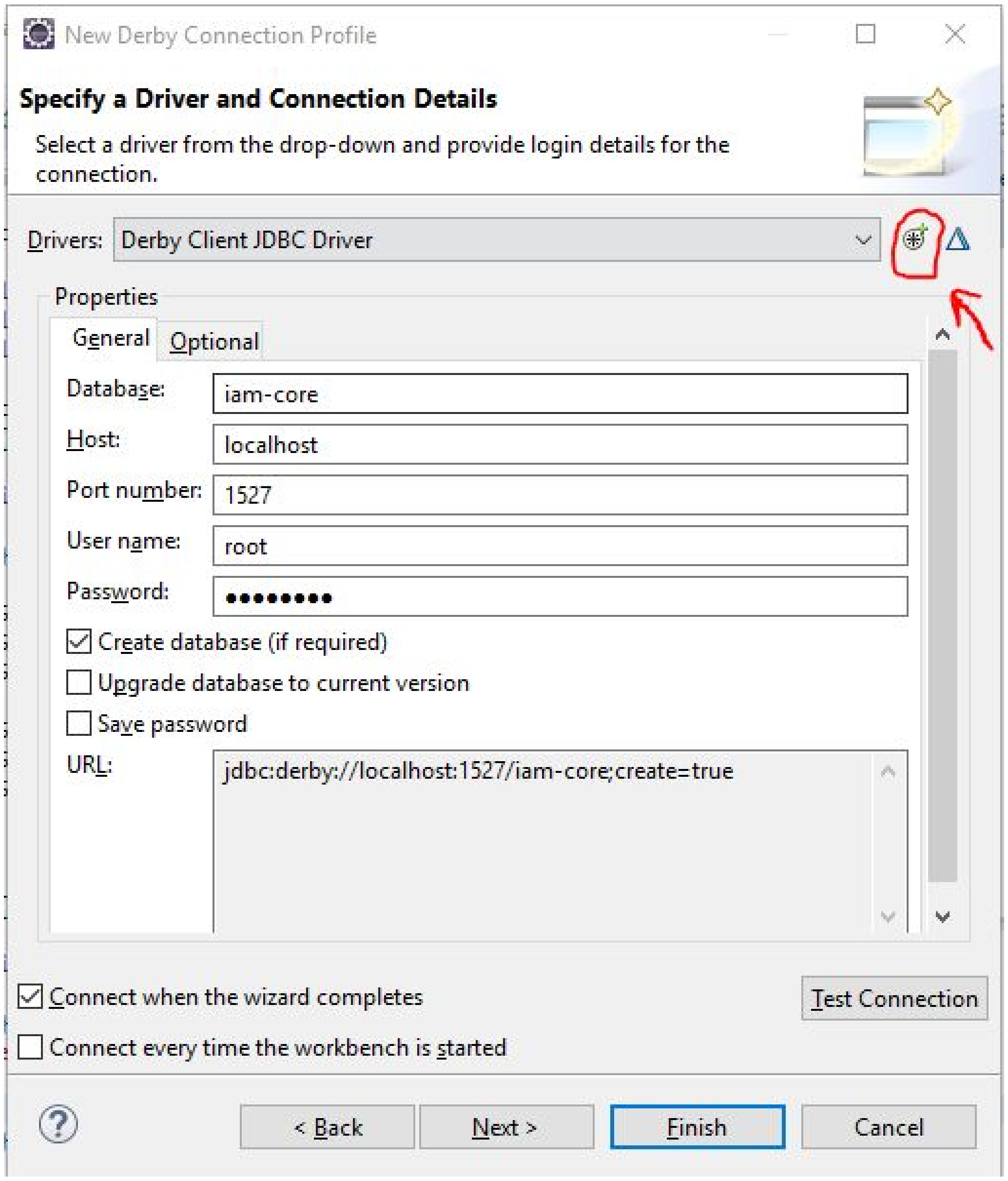


Fig 5.

d-) Update the Database, Username and password fields with what is found in the **testConfiguration.properties** inside Iamcore folder and choose the Driver (Derby Client JDBC Driver) by clicking on the icon shown in the fig 7. and


New Derby Connection Profile

Specify a Driver and Connection Details

Select a driver from the drop-down and provide login details for the connection.

Drivers: Derby Client JDBC Driver

Properties

General | Optional

Database:	iam-core
Host:	localhost
Port number:	1527
User name:	root
Password:	●●●●●●●●●

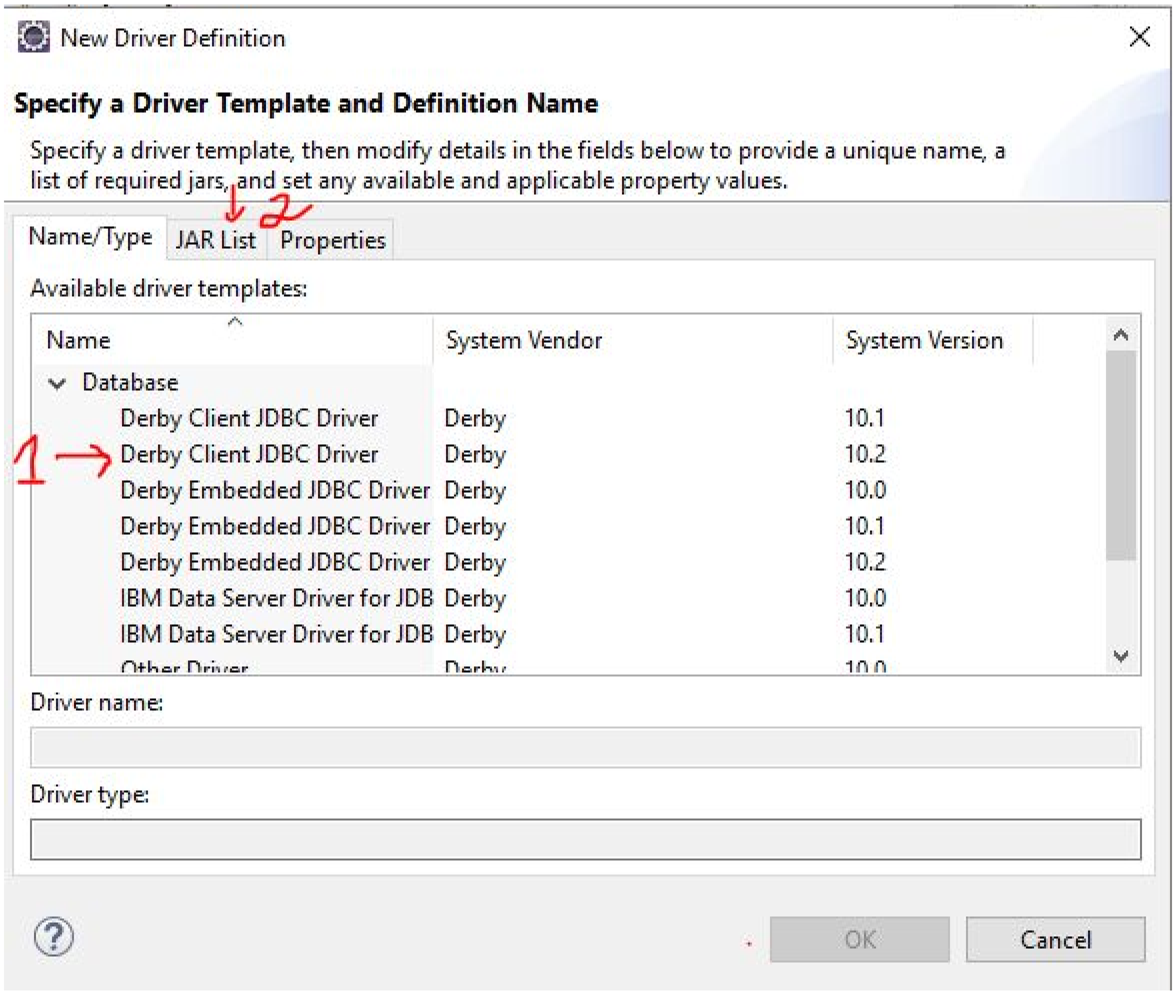
☑ Create database (if required)
☐ Upgrade database to current version
☐ Save password

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

☑ Connect when the wizard completes
☐ Connect every time the workbench is started

Fig 6.

e-) Select Derby Client JDBC Driver and move to the JAR List tab as shown in the fig 7.


New Driver Definition

Specify a Driver Template and Definition Name

Specify a driver template, then modify details in the fields below to provide a unique name, a list of required jars, and set any available and applicable property values.

Available driver templates:

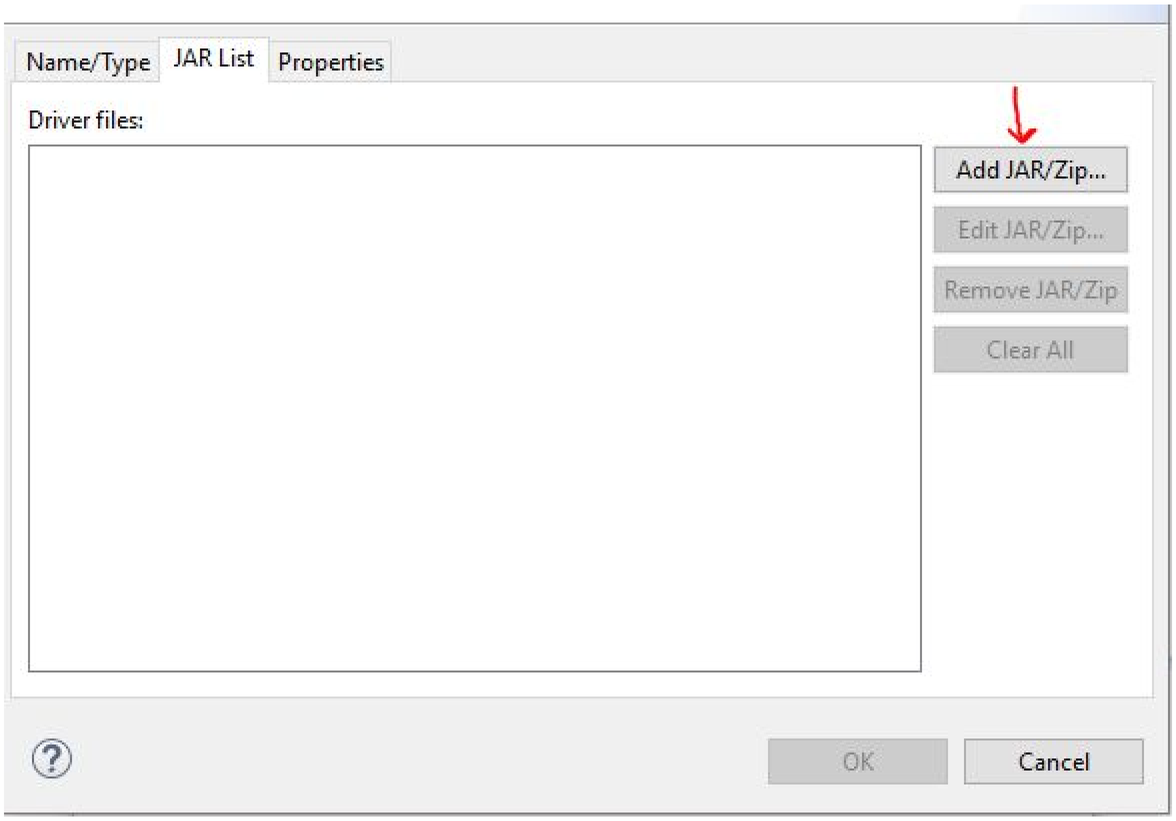
Name	System Vendor	System Version
Database		
Derby Client JDBC Driver	Derby	10.1
Derby Client JDBC Driver	Derby	10.2
Derby Embedded JDBC Driver	Derby	10.0
Derby Embedded JDBC Driver	Derby	10.1
Derby Embedded JDBC Driver	Derby	10.2
IBM Data Server Driver for JDB	Derby	10.0
IBM Data Server Driver for JDB	Derby	10.1
Other Driver	Derby	10.0

Driver name:

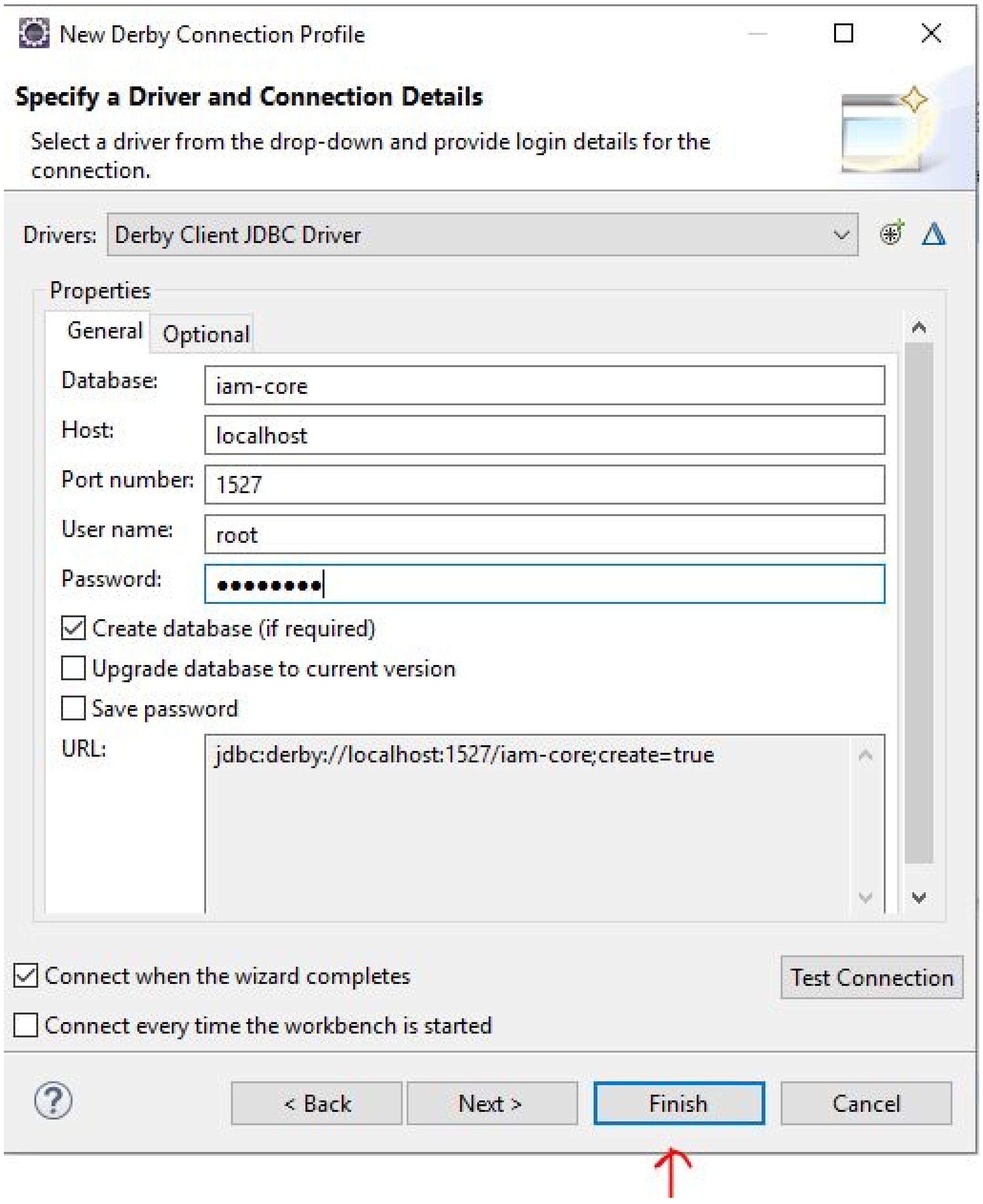
Driver type:

Fig 7.

f-) Click on "Add JAR/Zip", locate and add your derbyclient.jar



g-) Click on finish


New Derby Connection Profile

Specify a Driver and Connection Details

Select a driver from the drop-down and provide login details for the connection.

Drivers: Derby Client JDBC Driver

Properties

General | Optional

Database: iam-core

Host: localhost

Port number: 1527

User name: root

Password: ●●●●●●●●●

☑ Create database (if required)

☐ Upgrade database to current version

☐ Save password

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

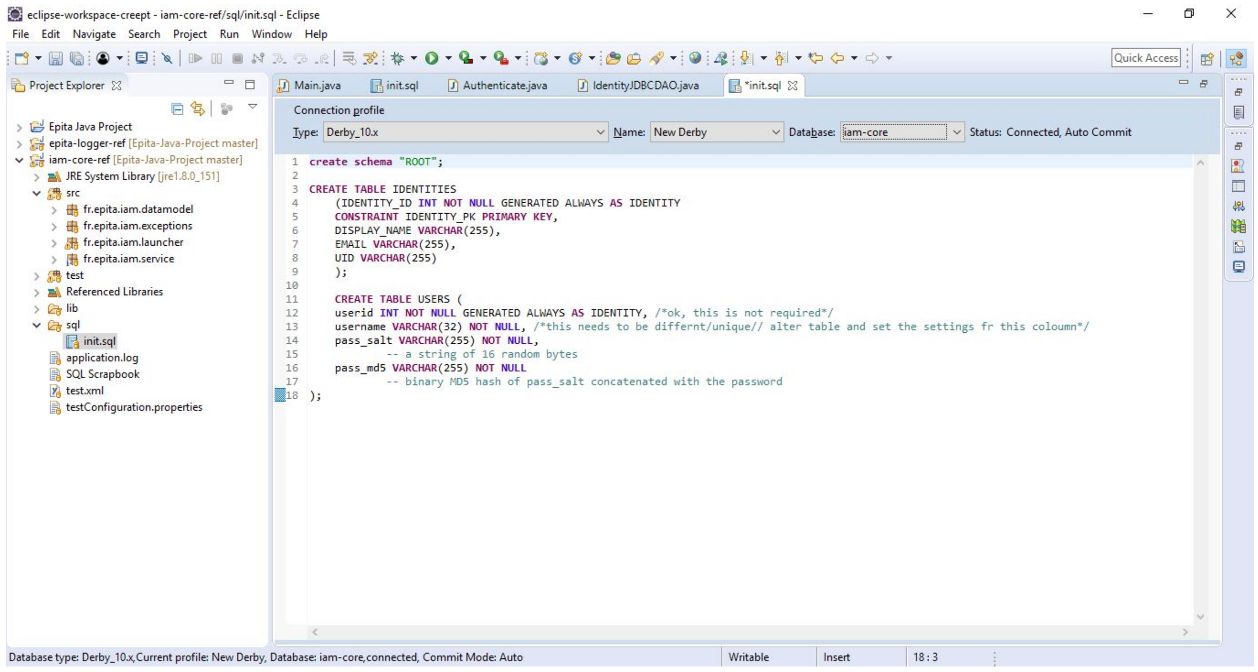
☑ Connect when the wizard completes

☐ Connect every time the workbench is started

Fig 8.

04. Move to the sql folder and import the sql schema into your specified database.

a-) Double Click the file located in the sql folder (init.sql), field up the type, name and Database field as in the Fig 9.



Type: Derby_10.x Name: New Derby Database: iam-core Status: Connected, Auto Commit

```sql
create schema "ROOT";

CREATE TABLE IDENTITIES
    (IDENTITY_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 USERS (
    userid INT NOT NULL GENERATED ALWAYS AS IDENTITY, /*ok, this is not required*/
    username VARCHAR(32) NOT NULL, /*this needs to be differnt/unique// alter table and set the settings fr this coloumn*/
    pass_salt VARCHAR(255) NOT NULL,
            -- a string of 16 random bytes
    pass_md5 VARCHAR(255) NOT NULL
            -- binary MD5 hash of pass_salt concatenated with the password
);
```

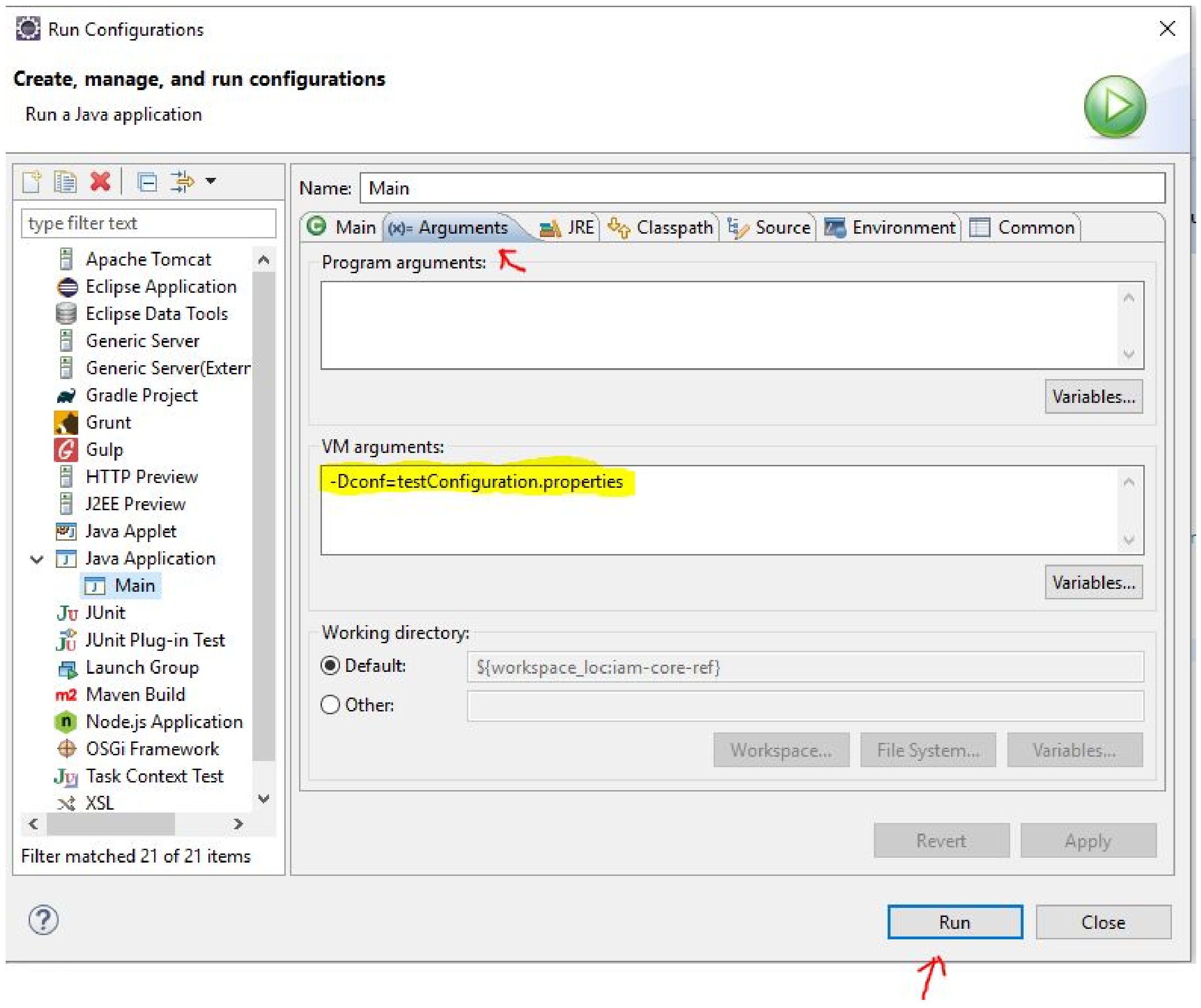
Database type: Derby_10.x,Current profile: New Derby, Database: iam-core,connected, Commit Mode: Auto

Fig 9.

b-) Select all the code, right click on it and click on “Execute Selected Text” to create a schema, and tables

05. Compile and run the application.

a-) Click on the arrow shown in the Fig 10. , select Run configurations. Under Arguments Tab Write down the same thing under VM arguments as shown in the Fig 10 and Run the application.



Name: Main

VM arguments: -Dconf=testConfiguration.properties

Default: ${workspace_loc:iam-core-ref}

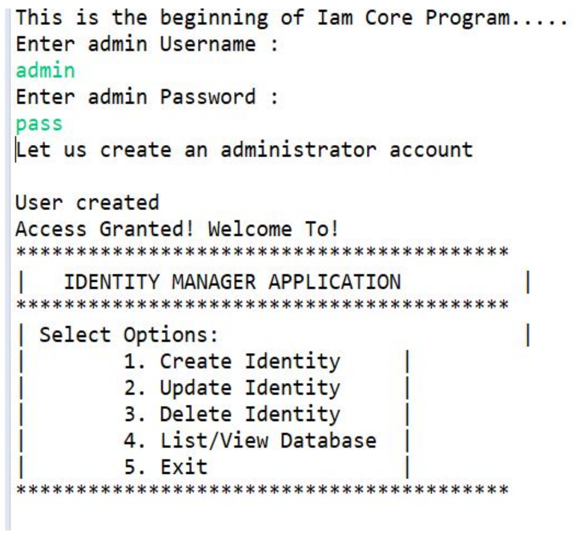
6. Commented Screenshots

6.0 Human User Interface Design

6.1 Setup Screen

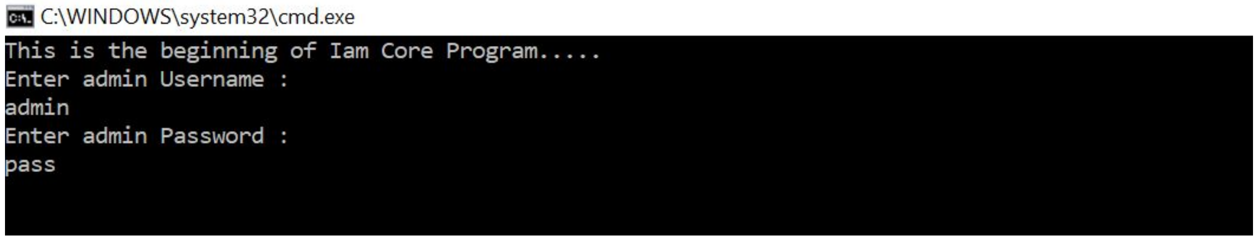
Here we will provide setup screen of the entire application and how each section of the application works as a real identity manager for a user.

1.1 Admin sign up



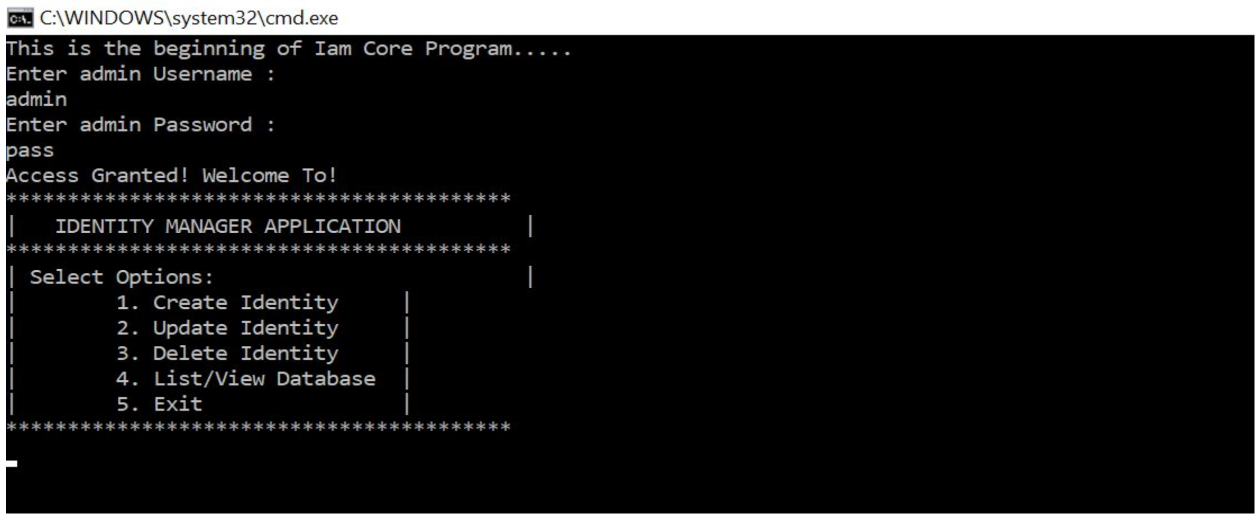
```
This is the beginning of Iam Core Program.....
Enter admin Username :
admin
Enter admin Password :
pass
Let us create an administrator account

User created
Access Granted! Welcome To!
*****************************************
|   IDENTITY MANAGER APPLICATION         |
*****************************************
| Select Options:                        |
|        1. Create Identity     |
|        2. Update Identity     |
|        3. Delete Identity     |
|        4. List/View Database  |
|        5. Exit                |
*****************************************
```

1.2 Admin sign in



```
C:\WINDOWS\system32\cmd.exe
This is the beginning of Iam Core Program.....
Enter admin Username :
admin
Enter admin Password :
pass
```

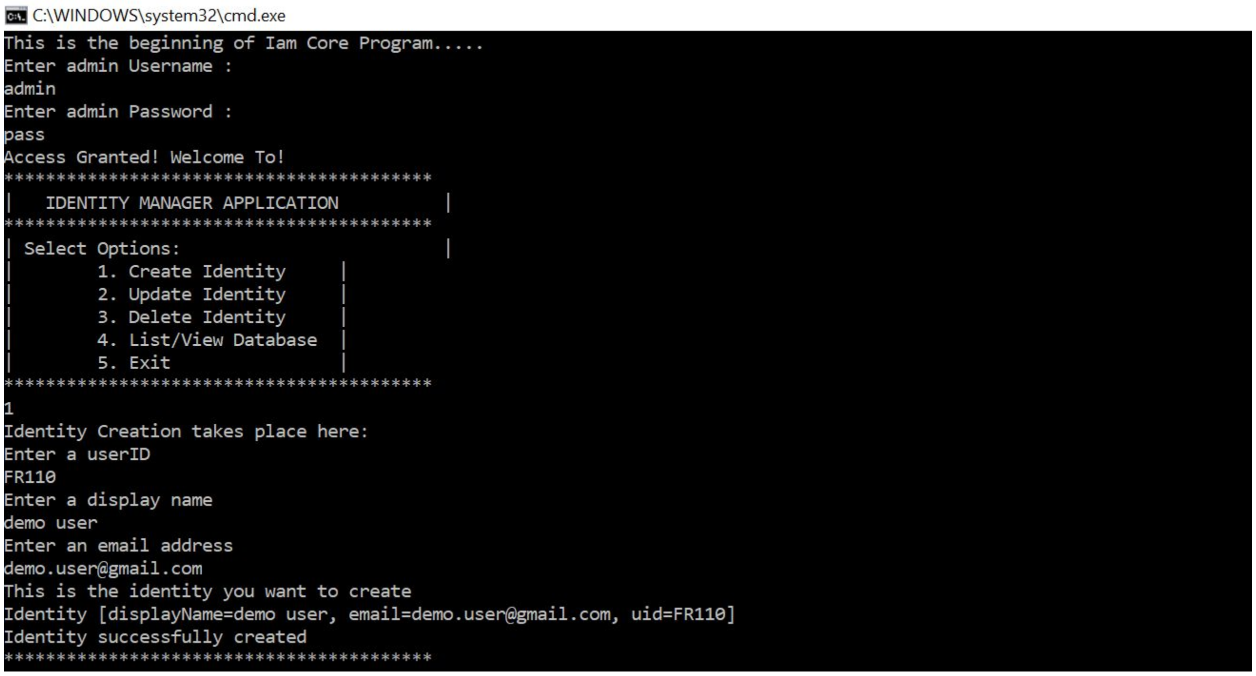
1.3 Welcome dashboard



C:\WINDOWS\system32\cmd.exe

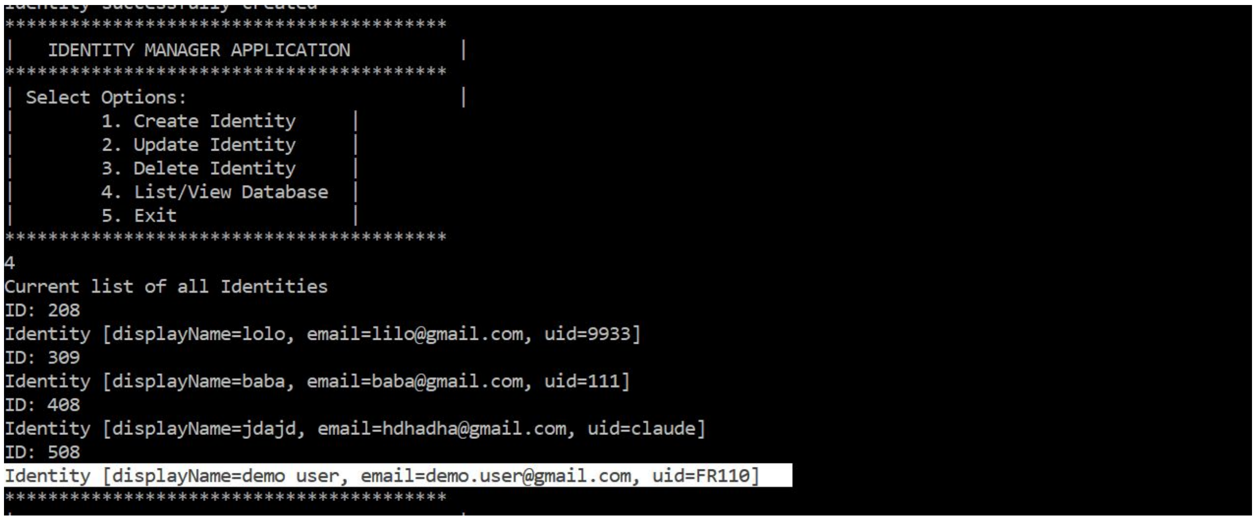
```
This is the beginning of Iam Core Program.....
Enter admin Username :
admin
Enter admin Password :
pass
Access Granted! Welcome To!
*****************************************
|   IDENTITY MANAGER APPLICATION          |
*****************************************
| Select Options:                         |
|        1. Create Identity     |
|        2. Update Identity     |
|        3. Delete Identity     |
|        4. List/View Database  |
|        5. Exit                |
*****************************************
```

1.4 Create Identity



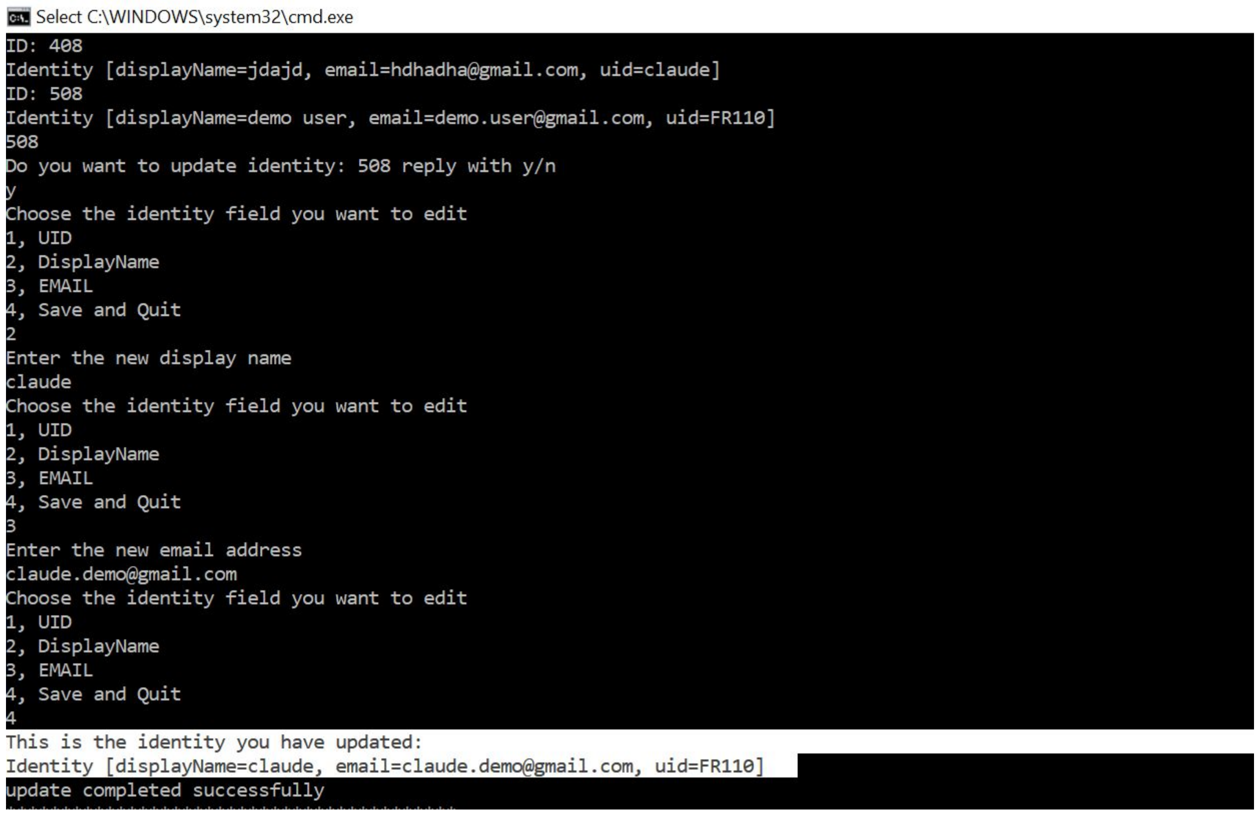
```
C:\WINDOWS\system32\cmd.exe
This is the beginning of Iam Core Program.....
Enter admin Username :
admin
Enter admin Password :
pass
Access Granted! Welcome To!
*****************************************
|   IDENTITY MANAGER APPLICATION         |
*****************************************
| Select Options:                        |
|        1. Create Identity     |
|        2. Update Identity     |
|        3. Delete Identity     |
|        4. List/View Database  |
|        5. Exit                |
*****************************************
1
Identity Creation takes place here:
Enter a userID
FR110
Enter a display name
demo user
Enter an email address
demo.user@gmail.com
This is the identity you want to create
Identity [displayName=demo user, email=demo.user@gmail.com, uid=FR110]
Identity successfully created
*****************************************
```

1.5 List Identities



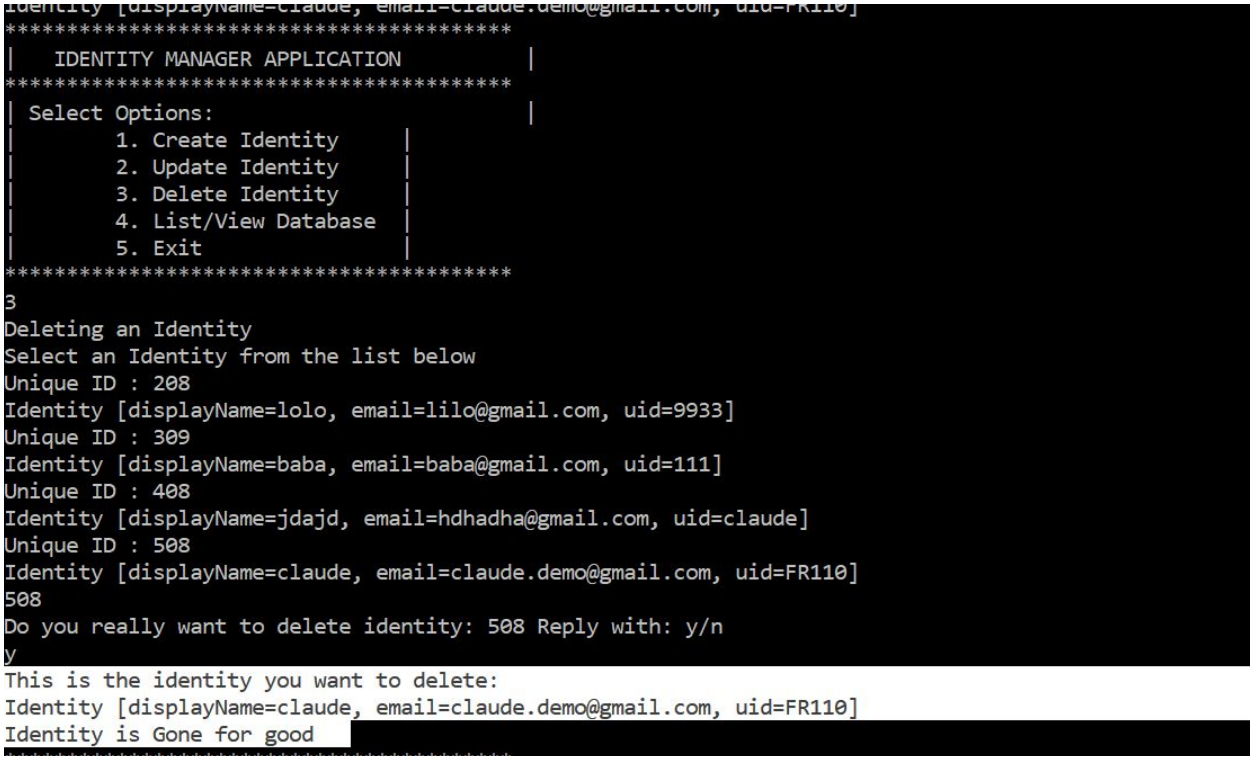
```
*****************************************
|   IDENTITY MANAGER APPLICATION         |
*****************************************
| Select Options:                        |
|        1. Create Identity     |
|        2. Update Identity     |
|        3. Delete Identity     |
|        4. List/View Database  |
|        5. Exit                |
*****************************************
4
Current list of all Identities
ID: 208
Identity [displayName=lolo, email=lilo@gmail.com, uid=9933]
ID: 309
Identity [displayName=baba, email=baba@gmail.com, uid=111]
ID: 408
Identity [displayName=jdajd, email=hdhadha@gmail.com, uid=claude]
ID: 508
Identity [displayName=demo user, email=demo.user@gmail.com, uid=FR110]
*****************************************
```

1.6 Update Identity



```
Select C:\WINDOWS\system32\cmd.exe
ID: 408
Identity [displayName=jdajd, email=hdhadha@gmail.com, uid=claude]
ID: 508
Identity [displayName=demo user, email=demo.user@gmail.com, uid=FR110]
508
Do you want to update identity: 508 reply with y/n
y
Choose the identity field you want to edit
1, UID
2, DisplayName
3, EMAIL
4, Save and Quit
2
Enter the new display name
claude
Choose the identity field you want to edit
1, UID
2, DisplayName
3, EMAIL
4, Save and Quit
3
Enter the new email address
claude.demo@gmail.com
Choose the identity field you want to edit
1, UID
2, DisplayName
3, EMAIL
4, Save and Quit
4
This is the identity you have updated:
Identity [displayName=claude, email=claude.demo@gmail.com, uid=FR110]
update completed successfully
```

1.7 Delete an Identity



```
Identity [displayName=claude, email=claude.demo@gmail.com, uid=FR110]
*****************************************
|   IDENTITY MANAGER APPLICATION          |
*****************************************
| Select Options:                         |
|        1. Create Identity     |
|        2. Update Identity     |
|        3. Delete Identity     |
|        4. List/View Database  |
|        5. Exit                |
*****************************************
3
Deleting an Identity
Select an Identity from the list below
Unique ID : 208
Identity [displayName=lolo, email=lilo@gmail.com, uid=9933]
Unique ID : 309
Identity [displayName=baba, email=baba@gmail.com, uid=111]
Unique ID : 408
Identity [displayName=jdajd, email=hdhadha@gmail.com, uid=claude]
Unique ID : 508
Identity [displayName=claude, email=claude.demo@gmail.com, uid=FR110]
508
Do you really want to delete identity: 508 Reply with: y/n
y
This is the identity you want to delete:
Identity [displayName=claude, email=claude.demo@gmail.com, uid=FR110]
Identity is Gone for good
```

7. Future Prospects

- Extend the identity management application to include address location data
- Make the api comprehensive and extendable to other applications
- Produce a gui for the application for easy interaction by users

8. Conclusion

Java is the most popular, mature and stable programming language. Its importance in today's world is very huge. This project has extremely been a tool in helping us learn what is called Java. Even though the Java language is a large field, we have learned the basics throughout the course and we are nonetheless prepared for the Business World.

9. Work Cited

- https://dzone.com/articles/static-code-analysis-and
- www.stackoverflow.com
- https://rosettacode.org/wiki/SQL-based_authentication