

Name: Junaid KHALID

Professor: Thomas Broussard

Session: 2016

Project: Identity and Access Management

The IAM Web Application

Subject: JAVA

Table of Contents

[Subject Description 3](#_Toc469086310)

[Overview 3](#_Toc469086311)

[Subject Analysis 3](#_Toc469086312)

[Main Features 3](#_Toc469086313)

[Application Feasibility 4](#_Toc469086314)

[Data Description 4](#_Toc469086315)

[Expected Results 5](#_Toc469086316)

[Scope and Limitations 5](#_Toc469086317)

[Conception 5](#_Toc469086318)

[Chosen Algorithm 5](#_Toc469086319)

[Data Structures 5](#_Toc469086320)

[Application Flow Diagram 6](#_Toc469086321)

[Configuration instructions 7](#_Toc469086322)

[Maven: 7](#_Toc469086323)

[Server: 7](#_Toc469086324)

[To access page 8](#_Toc469086325)

[Screenshots 8](#_Toc469086326)

# Subject Description

Overview  
  
Identity management (IdM) is the task of controlling information about users on computers. Such information includes information that authenticates the identity of a user, and information that describes information and actions they are authorized to access and/or perform. It also includes the management of descriptive information about the user and how and by whom that information can be accessed and modified Core Java concepts

A build management tool is very useful for this kind of project. **Maven** is a "build management tool", it is for defining how.java files get compiled to .class, packaged into .jar (or .war or. ear) files, (pre/post) processed with tools, managing CLASSPATH, and all others sorts of tasks that are required to build project.

For this project **Spring Framework** has been used it handles Dependency Injection. Spring does is wrap excellent Java libraries in a very elegant way to use in applications. Spring also provides a bunch of implementations of web stuff like REST, an MVC web framework and more.

**Hibernate** is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

# Subject Analysis

## Main Features

**Authentication:** First user entersusername and password to access the application. User can register itself and after to access the other features user need to authenticate himself which can be achieve by using login.

**Create Identity:** User can create identity by providing display name, e-mail address and Date of birth and these attributes will be added in the database.  
**Search an Identity:** User can search for an entity by giving name or email address. User enter display name and email address to fetch the related record in the database if it’s already present in the db.

**Update an Identity:** User can update an entity by selecting the searched entity. User can modify entity by utilizing an already present information form which is result of user’s search selection. User enter display name and email address to fetch the related record in the database if it’s already present in the db.

**Delete an Identity:** User can delete an entity by selecting the searched entity. User can delete the entity by selecting it using radio button which is generated by search.

## Application Feasibility

This application is operating in reduced cost, independent.

## Data Description

Modular approach is used for the development of this project.

There are three basic modules of this project:

1. **IAMCORE:**
   1. Users:
      1. This module is used to handle user related activities like user registration.
   2. DAO service:
      1. This module is handling all database related queries using HQL.
   3. Identity data module:
      1. This module is used to handle all identity related operations
   4. Test module:
      1. This module is handling Junit Tests.
2. **IAM-WEB:**
   1. Servlets:
      1. Servlets are used to handle all control all DAO and User related activities it’s acting like a controller.
   2. JSP:
      1. JSP is acting as a view. Jsp are used to build custom tags and perform java operations and it can directly call java beans.

## Expected Results

This application can authenticate the user and can create, delete, update and search for the identity without any issue. The communication between application and database is smooth. Highly sophisticated, user friendly and secure tool created for Identity and access management.

Junit Tests are utilized in this project for the testing purpose and to make sure that there are no loopholes in the application.

## Scope and Limitations

**Scope**

The password is storing in database without encryption.

Dependence of dependencies.

# Conception

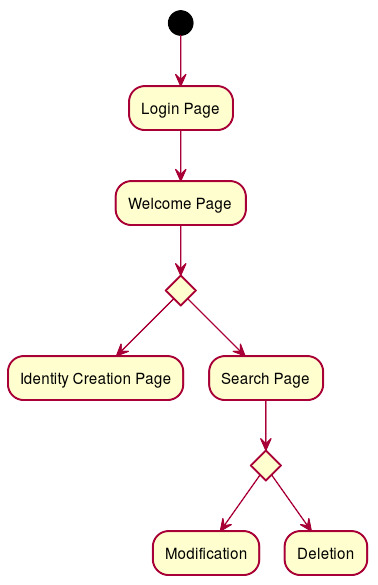
## Chosen Algorithm

The algorithms which is used for search of the identity is based on user name and user email which user can enter to search the specific identity if it already exists.

## Data Structures

Data structure which is used based on presentation, business and data access layer. Which is a good technique. We can modify our application without affecting its layers.

## Application Flow Diagram

****

# Configuration instructions

## Maven:

1. Open Eclipse and go to: Help / Eclipse Marketplace.
2. Insert on search: "Maven" and look for "Maven Integration for Eclipse WTP" and install it.
3. If the installation was successfully made in Window/Preferences, the Maven option will be visible.
4. Download the last version from here:
   1. http://maven.apache.org/download.html and put it on a folder in C partition (or other partitions) - just **remember** that the path to the downloaded maven shouldn't contain spaces (you will have errors) - for example, path is: "c:\Apache\_Maven\apache-maven-3.0.4...".
5. After downloading it, go in Eclipse -> Window -> Preferences -> Maven (from left menu and expand it) -> Installations -> and add path to the downloaded maven.
6. To create the project: New -> Other -> Maven -> Maven Project -> and search on filter the archetype you would like to use (for example: use maven-archetype-webapp for web application etc.). Click Next and put project details: **Group Id, Artifact Id, Version** etc.
7. After creating the project, the most important file is **pom.xml**. There add dependencies. Another important thing, maven local repository will be created here: "c:\Users\YOUR\_USERNAME\m2."
8. To perform maven, install/build/clean etc, right click on project and click "Run as" -> Select option.

## Server:

**Add Server**

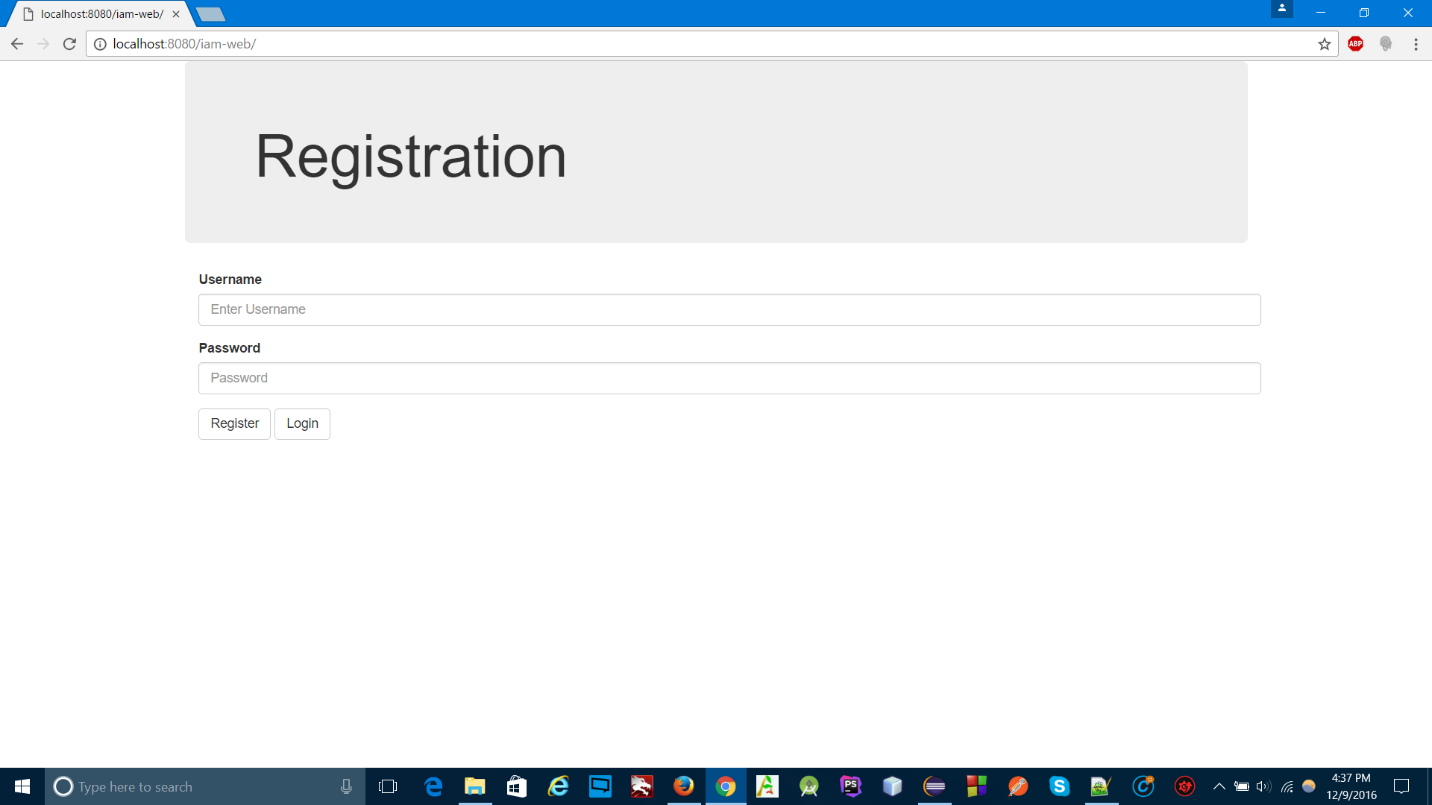
1. Windows -> Show View -> Servers.
2. Then in the servers view, right click and add new.
3. It will show a pop up containing many server vendors.
4. Under Apache select Tomcat v7.0 (Depending upon your downloaded server version).
5. And in the run time configuration point it to the Tomcat folder downloaded.

## To access page

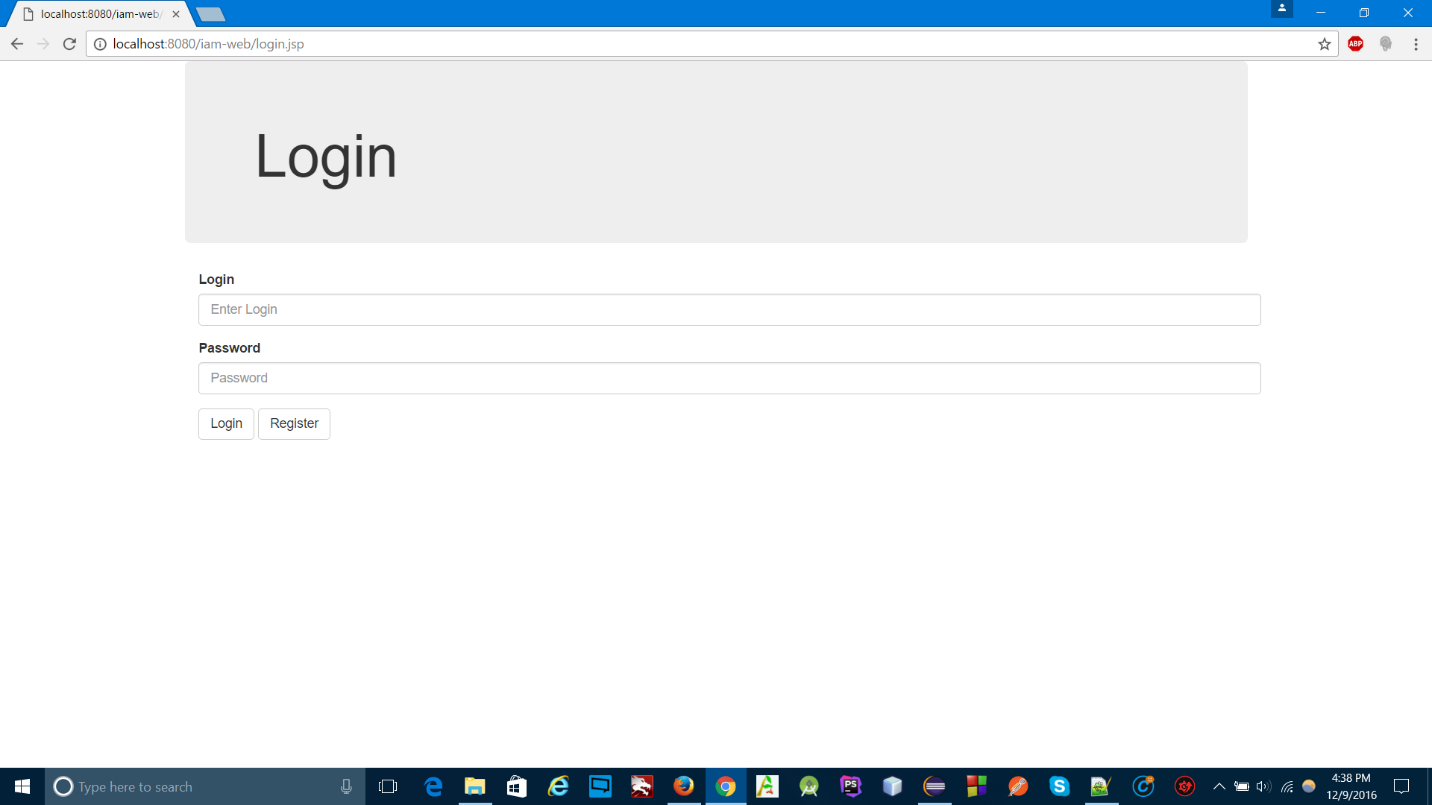
<http://localhost:8080/iam-web/>

# Screenshots

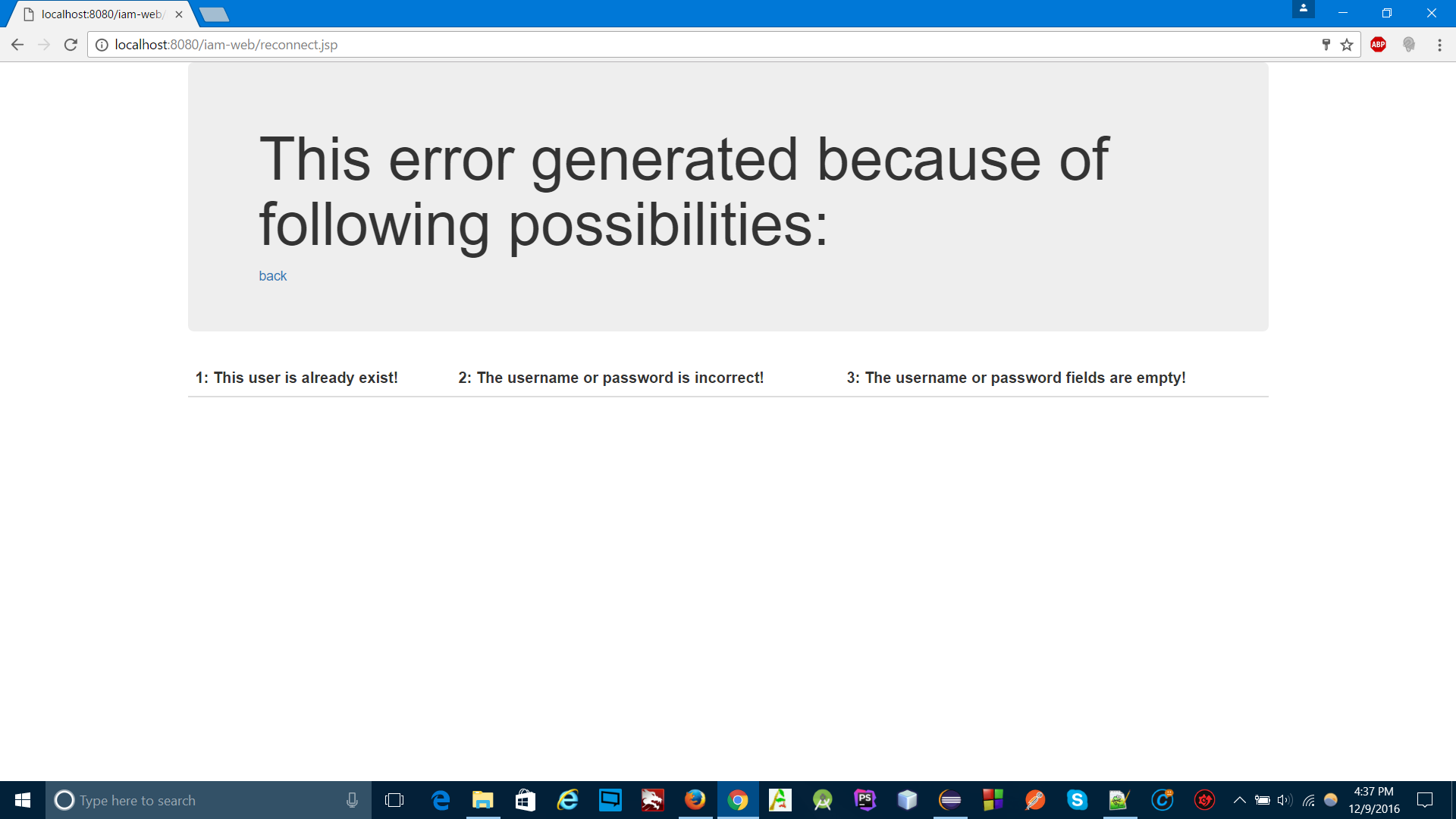
1: Registration



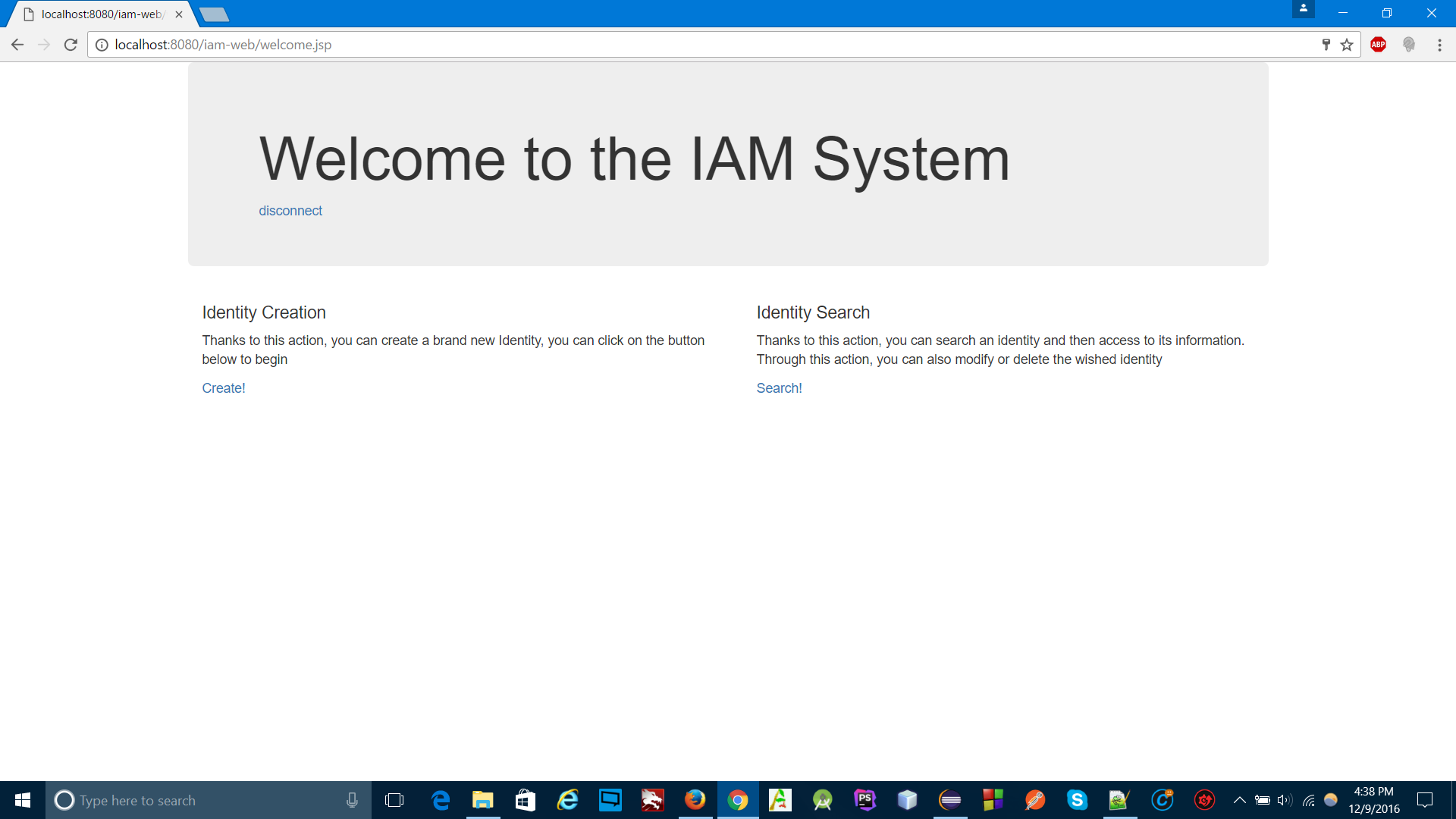
2: Login



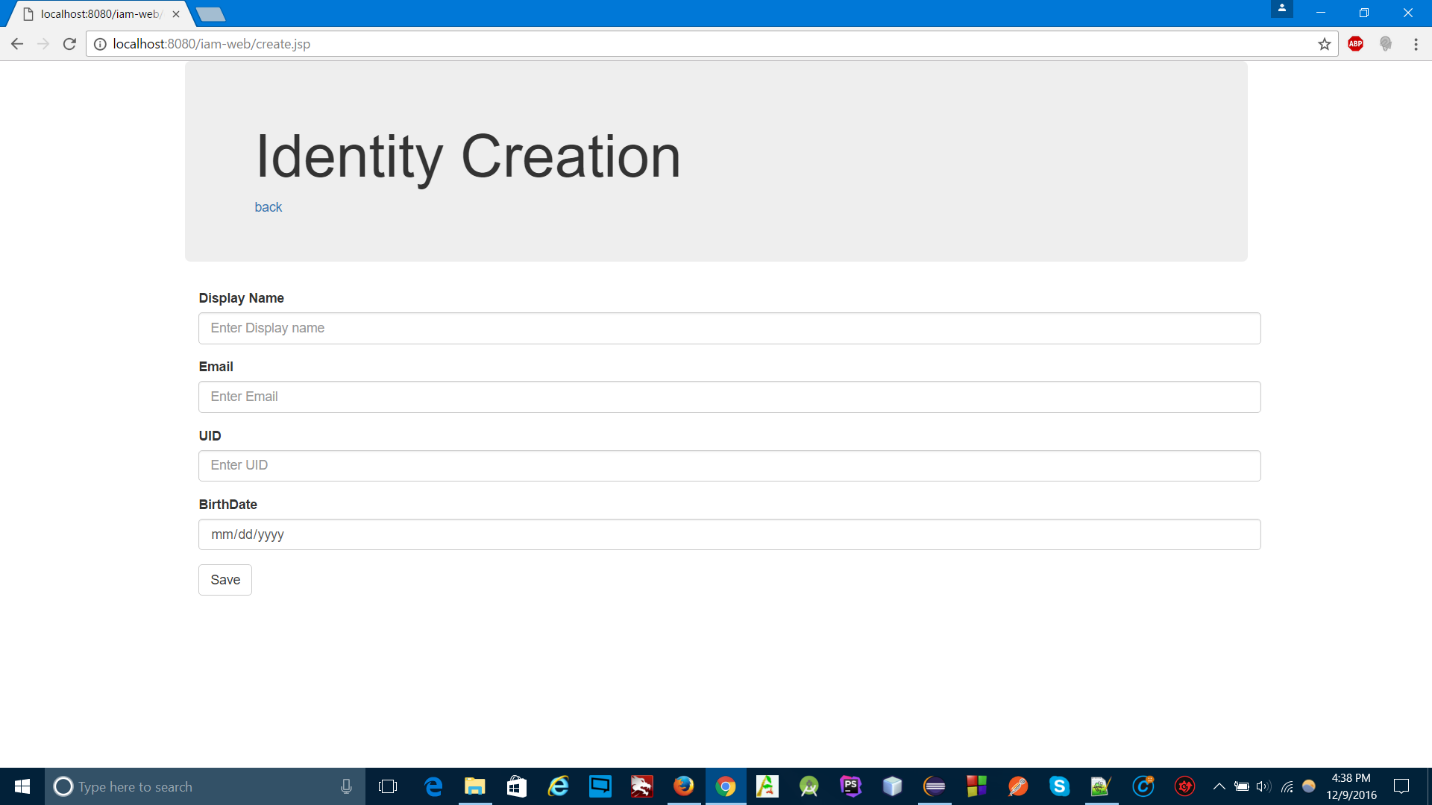
3: Error Page for incase of Login or Registration failed:



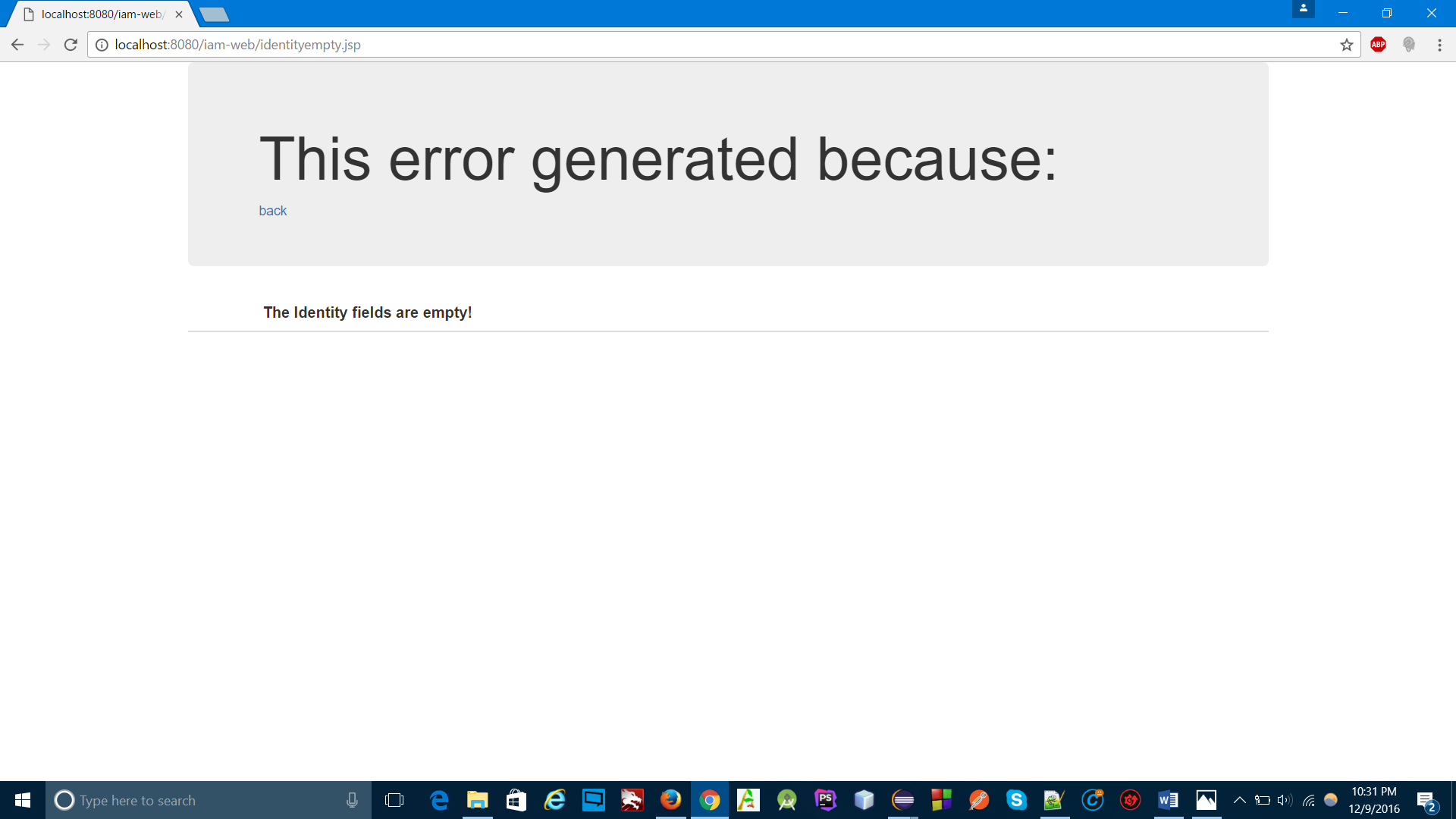
4: Welcome Page:



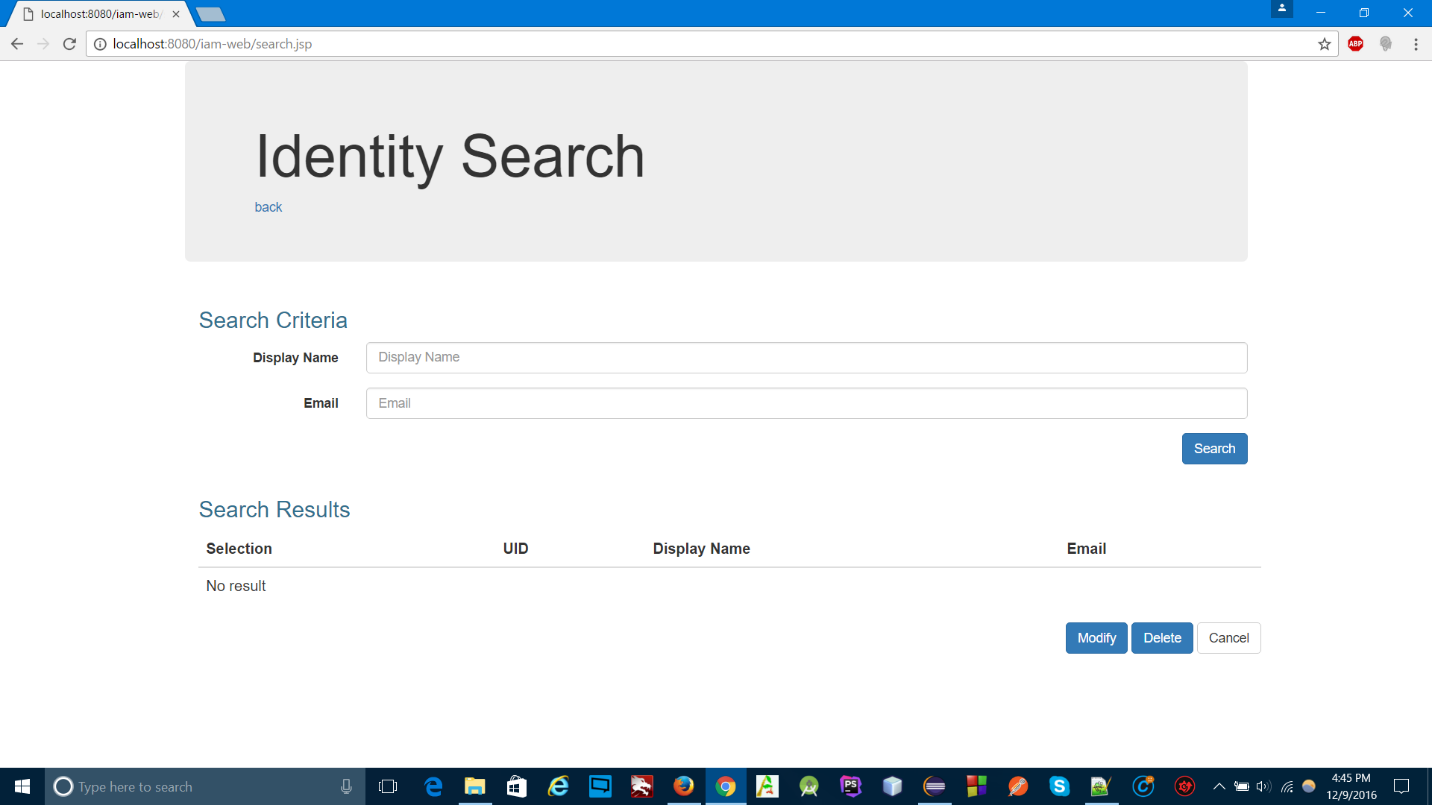
5: Identity creation page:



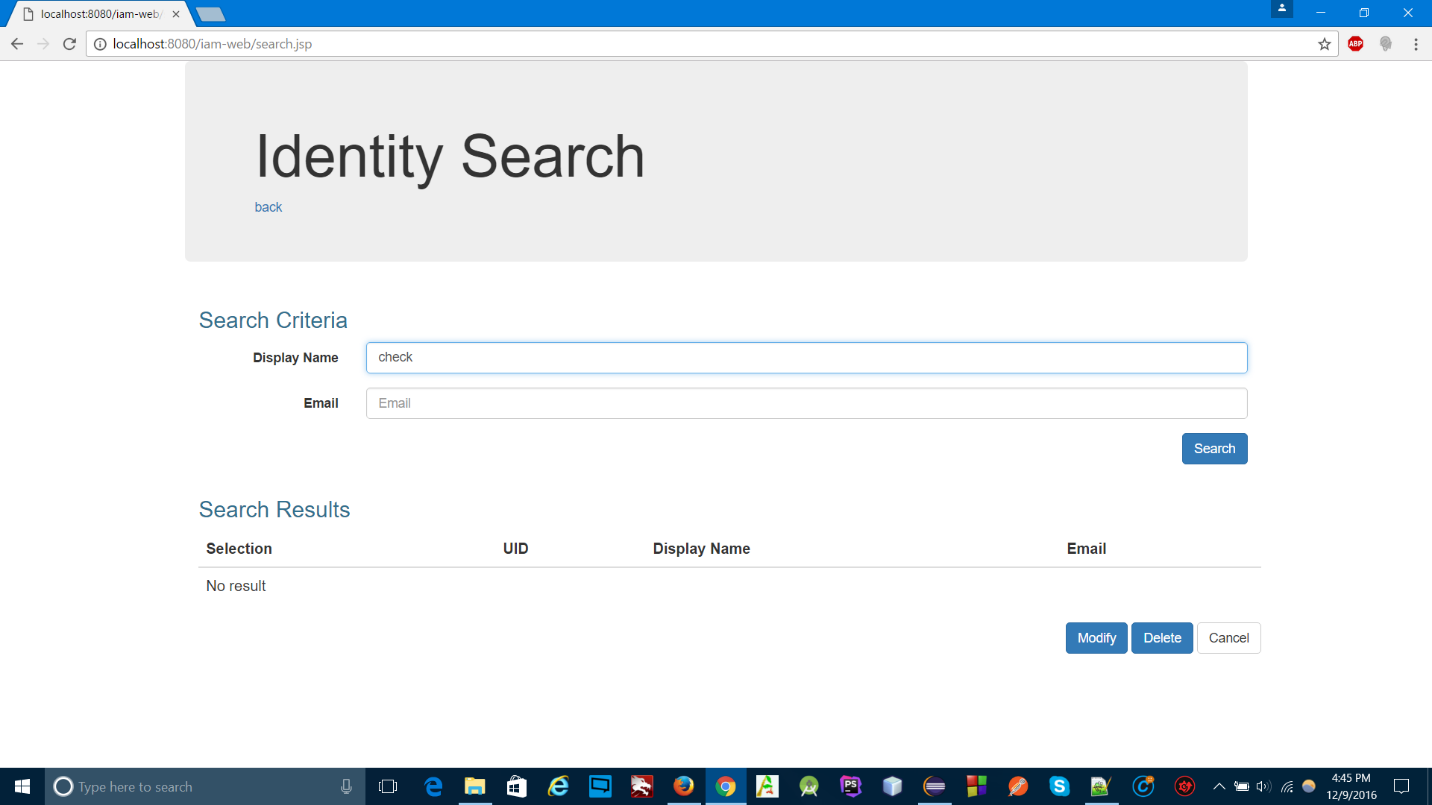
6: Identity creation error Page incase fields are empty:



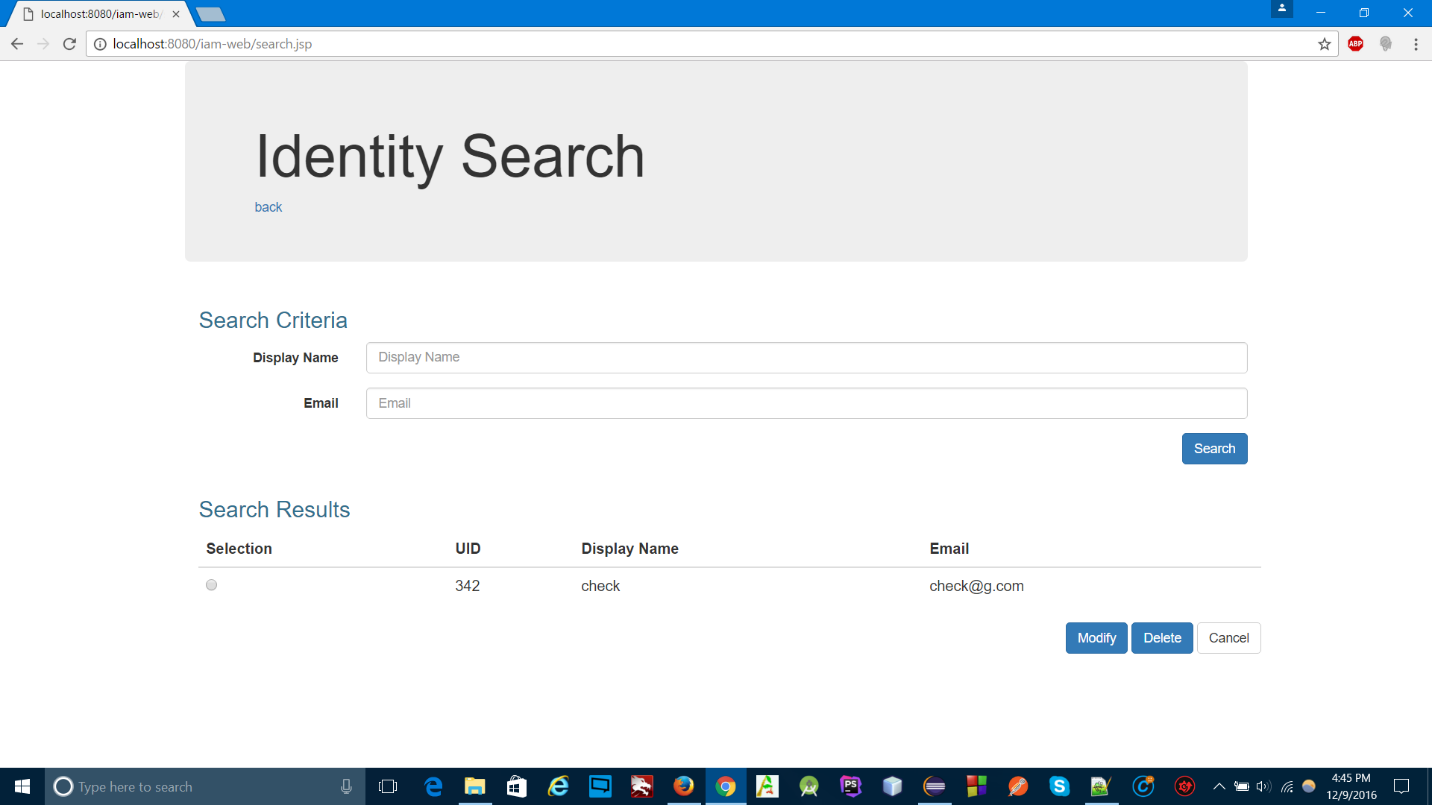
7: Identity Search Page:



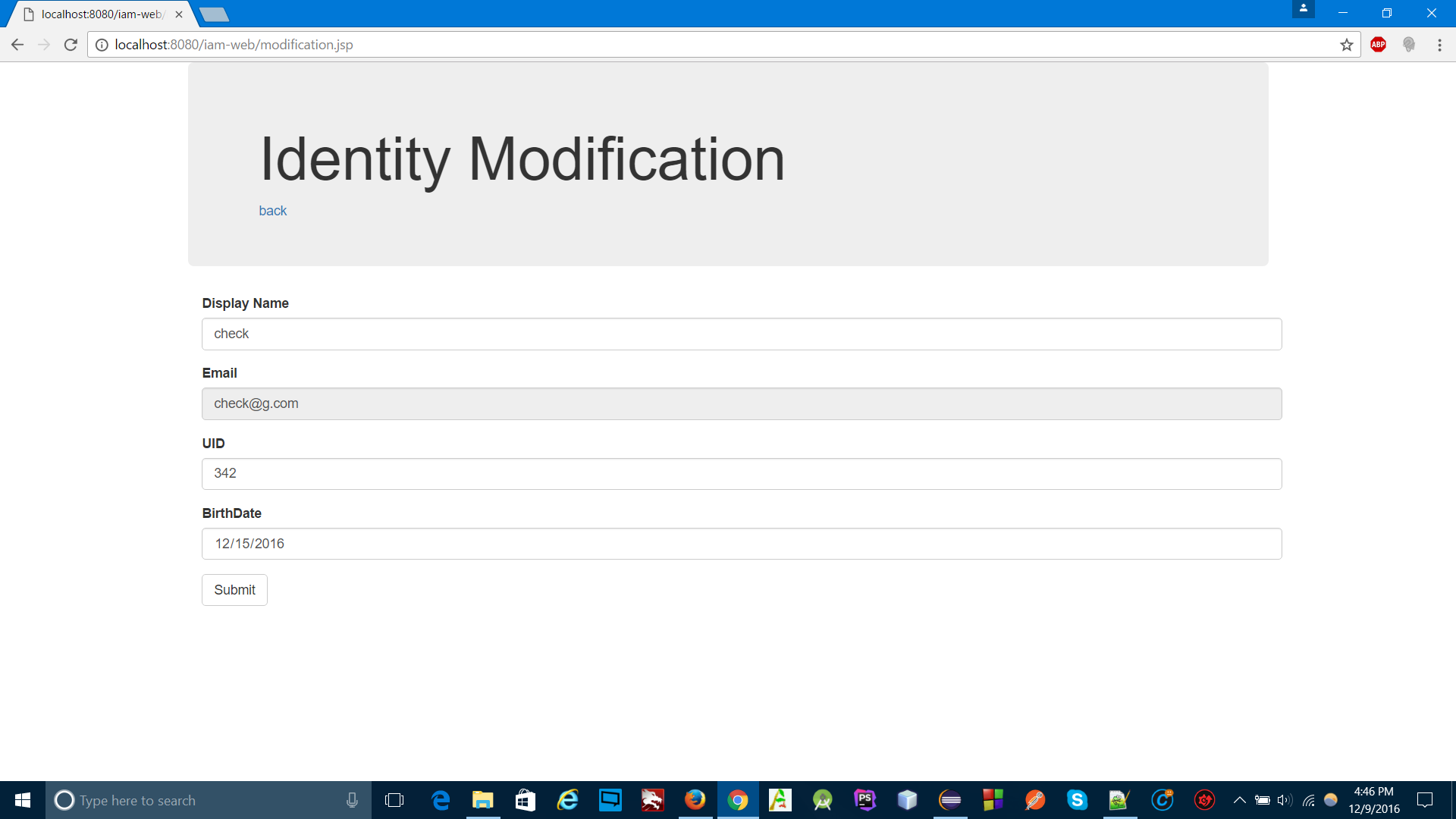
8: Identity can be searched by using Display name or Email:



9: Search result:



10: Identity modification page:



11: Success page in case of identity successfully saved, update or delete:

