# ClubUMLSpring2014

# Sprint 3

# Validation Team Documentation

## Table of Contents

1…………………………………………………………………………………………………Need for Validation

2…………………………………………………………………………………………………….Existing scenario

3……………………………………………………………………………………………………………Modifications

4………………………………………………………………………………………………….Files altered/added

5…………………………………………………………………………………………………….Database changes

6………………………………………………………………………………………………………………Screenshots

**Team members:**

Ameya Joshi

Sahil Patil

1. **Need for Validation:**

Validation is considered an essential part of any application that deals with databases (data-driven applications). This is because the data that is entered by the user need not be completely accurate each time and is prone to human error.

Any application which deals with sensitive data should not be built in a manner which would rely upon just good fortune and the developer shouldn’t have blind faith that there would not be any mistake while data entry. Hence, validation of data at entry is a very important part of an application.

Furthermore, any data-driven application which might contain sensitive data which can be harmful if leaked such as customer data in banks and credit card companies, patient records of hospitals, etc. should be secure. There are many ways in which a database can be hacked. One of them is SQL Injection.

SQL injection is a code injection technique, used to attack data driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

Our project, ClubUML, contains sensitive data. Data related to projects and its details of a company, the procedures followed by the company and sensitive discussions are things a company would not want to go in wrong hands. And hence, validation of the data being entered is very important for our project.

1. **Existing scenario:**

The unmodified ClubUML project from the previous semester does not have any validations present in it. Validation is a concept being added in this semester as we feel that it is an important part of any data-driven application especially when the data is sensitive and a hacker getting into the database is most undesirable.

The important parts where validation was agreed upon to be important by the team are login, registering a new account, uploading diagrams and updating policies. Login is an essential part of an application. To make sure that the data being entered is safe to be used in a query and would not create any data loss is very important.

Similarly, while creating a new account, there are certain things to cross check. We need to make sure that the username being selected is not already present in the database; we need to make sure that the email id being used is not already used to create an account. Apart from these we need to imply some constraints on the username and password selection.

These are some scopes where we need to implement validation in our project. We need validation when we need accurate inputs from user, when we need to imply some constraints on the input and when we need to prevent SQL Injection attacks.

1. **Modifications/Alterations:**

Modifications that are needed in the current system to implement validation checks are as follows:

During login, we need to check that the input is safe to be used in a query and would not harm the data or the database itself. We need to make sure that it is free from escape characters and some special characters that have greater possibility of occurring in a query. In short, we need to make sure that the input does not have a query hidden in it. To do this, we can set some constraints on the username and password; and directly check these strings for the acceptable inputs. This will make sure that it is valid and safe.

Similarly, while creating an account, we implement the same constraints on username and password as implemented during login. This makes sure that the username and password fit a certain rule (constraint) that we imply. Apart from this, we need to make sure that the username is not already in use by any other user, if so we need to alert the user that the username is already taken. Similar function for email will check if the email address being entered was already used to create an account. We do not want multiple accounts with the same email id.

In the display page, the user can upload diagrams that he can later compare with one another. Currently, the project supports only ‘ecore’ and ‘xmi’ files. Hence, here we need to make sure that the user is uploading the right type of file. We need to alert the user if he selects a mismatched file.

In the update policy page, the user can update existing policies and change their metrics. In doing so, the user can by error set the fields to some unexpected input types. Hence to make sure that the user is using correct type of input, we put few validation checks that make this sure for us.

1. **Files altered/added:**

*Files altered:*

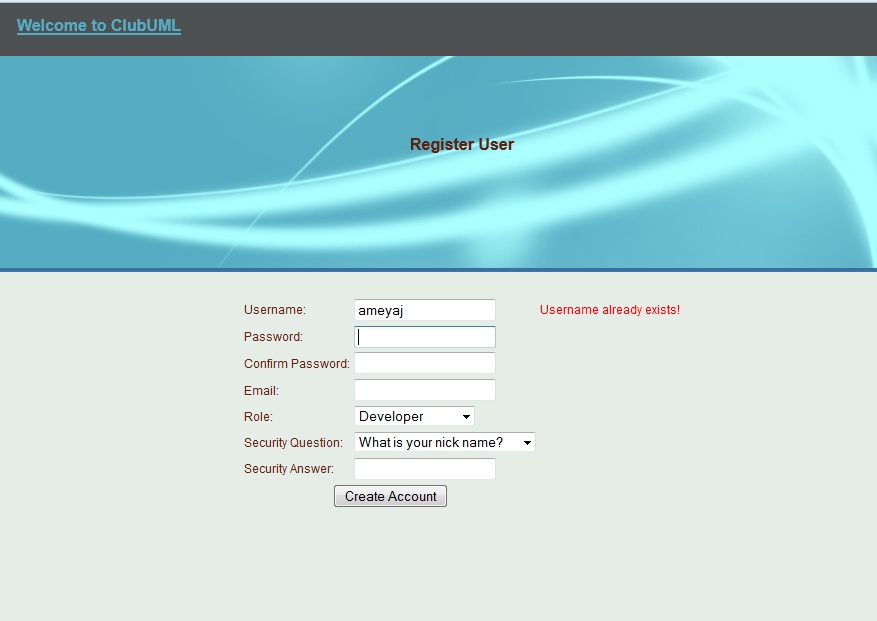
1. index.jsp
2. register.jsp
3. display.jsp
4. managePolicy.jsp
5. register.js
6. UserDAO.java

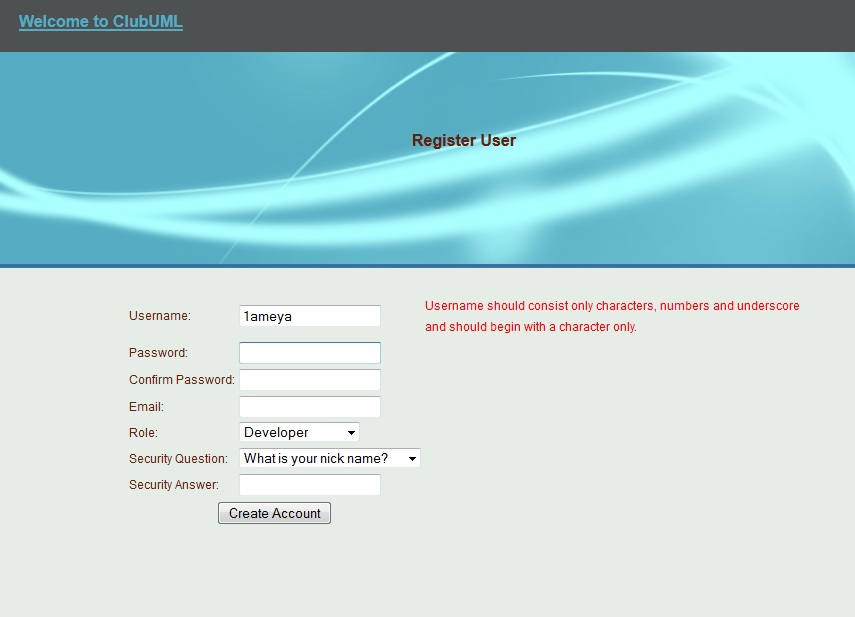
*Files added:*

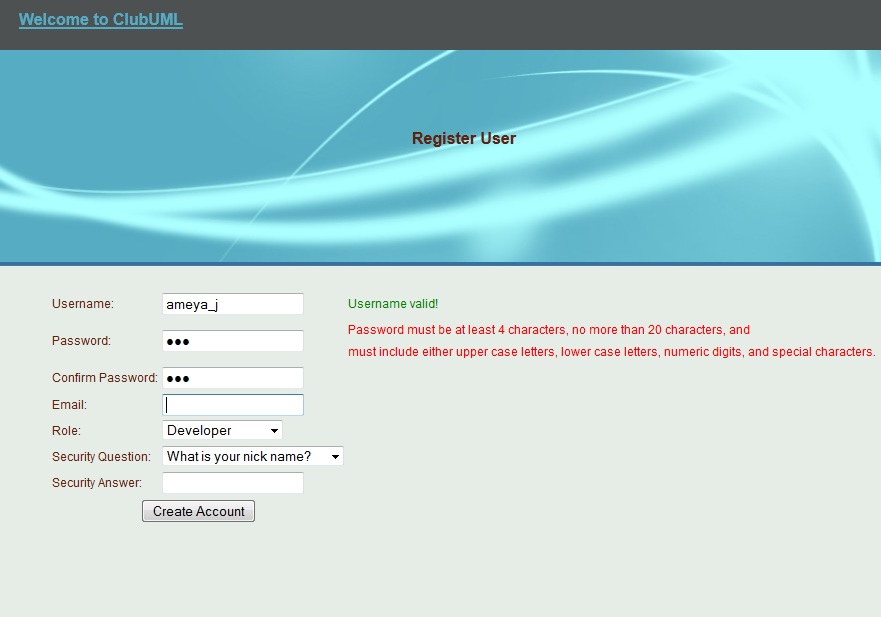
1. display.js
2. ValidateEmail.java
3. **Database changes:**

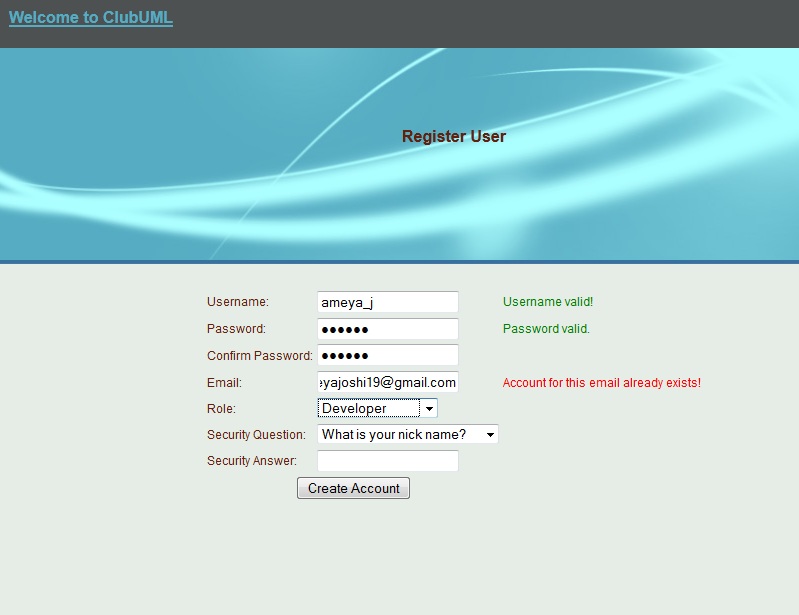
No changes to the database schema were made by the validation team.

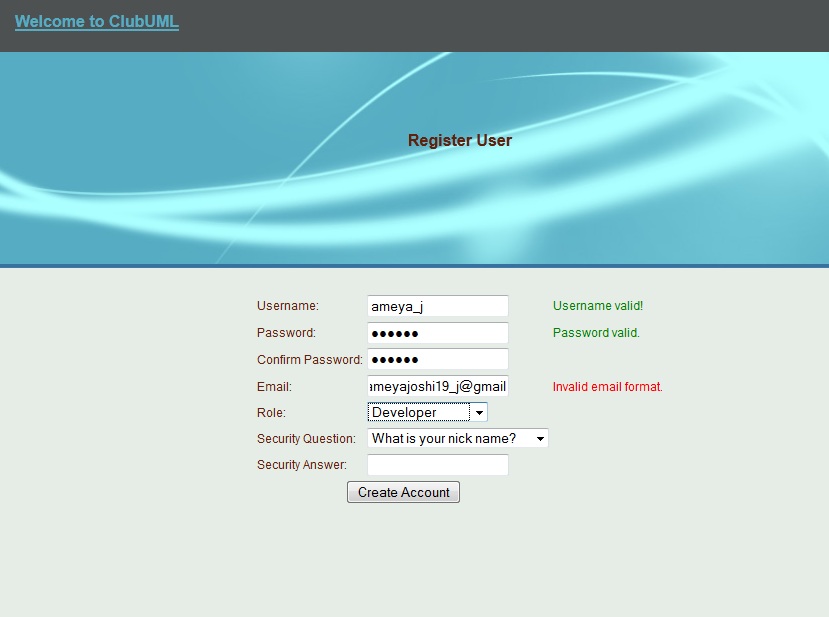
1. **Screenshots:**

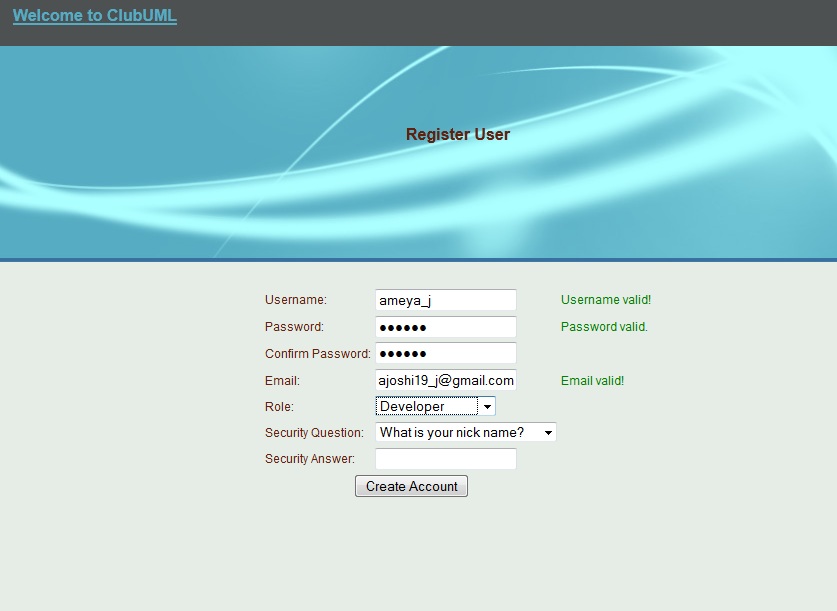


****

****

****

****

****

****

****