|  |
| --- |
|  |
| TwitterClone |
| A JAVA EE Application Specification for Wipro  Prepared By: **S Praveen Reddy** |
|  |
|  |
|  |

|  |
| --- |
| [The document details a requirements specification for a TwitterClone application to be implemented by Wipro as part of their training/coursework. It includes a functional specification, design and some general recommendations] |

Contents

[1. Introduction 2](#_Toc392092342)

[1.1 Purpose 2](#_Toc392092343)

[1.2 Scope 2](#_Toc392092344)

[1.3 Technologies/framework(s) to use 2](#_Toc392092345)

[2. Overall Description 3](#_Toc392092346)

[2.1 Use-case Model 3](#_Toc392092347)

[2.2 Architecture Diagram 4](#_Toc392092348)

[2.3 Database Design 5](#_Toc392092349)

[2.3.1 Tables 5](#_Toc392092350)

[3. Detailed Design 6](#_Toc392092351)

[3.1 Presentation 6](#_Toc392092352)

[3.1.1. Login/Sign up page 6](#_Toc392092353)

[3.1.1a. Sign up page 7](#_Toc392092354)

[3.1.2. Home page (Where users can tweet, search others) 7](#_Toc392092355)

[3.1.3. Profile page 8](#_Toc392092356)

[3.2 Domain/Model 9](#_Toc392092357)

[3.3 Business tier & Data Access tier 9](#_Toc392092358)

[3.4 Project Structure 10](#_Toc392092359)

[4. Additional Specifications 10](#_Toc392092360)

[4.1 Recommendations 10](#_Toc392092361)

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to define a specification for a typical Java EE application that can be used as part of training course work/project work for Wipro. This specification defines the requirements and design for a Twitter application clone.

### 1.2 Scope

Develop a web application using Java EE. The application should behave like a basic clone of the web-based twitter ( <http://twitter.com> )

The functional requirements for the TwitterClone application are as follows

* Only a basic clone of twitter is required. i.e only features like signup, tweet, follow are required
* Users should be able to visit the application and Sign up
* Registered users, hereinafter referred to as “Users”, should be able to tweet from their home page (If you’re not familiar with twitter or tweet, please visit http://www.twitter.com)
* Users should be able to edit or delete their own *tweet*
* Users should be able to edit their profile (change password, update name,etc)
* Users should not be allowed to change their *username* (Twitter allows you to change your *username*, but we’d like to keep it simple)
* Users should be able to search other registered users
* Users should be able to *follow* other users (if you don’t know what following is, please refer [this](http://support.twitter.com/groups/31-twitter-basics#topic_108))
* Users should be able to receive tweets from people they follow
* The application must enable users to view their followers
* *Additional features are up to your (the developer) discretion*

Initial non functional requirements will be

* Secure the web application (User must signup!)
* The application should display and behave uniform across all major browsers(IE, FF, Chrome, Safari etc).
* The home page of the application must load in < 3 secs
* Application should cater to more users in the future (Scalable)
* Easy installation to any web/app server

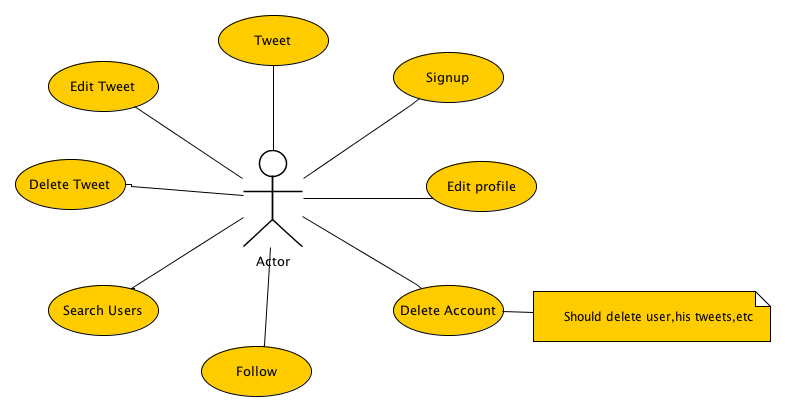
### 1.3 Technologies/framework(s) to use

Technology: Spring 4x, HTML5, CSS3, Hibernate, Restful web services

Database: MY SQL.

## 2. Overall Description

### 2.1 Use-case Model



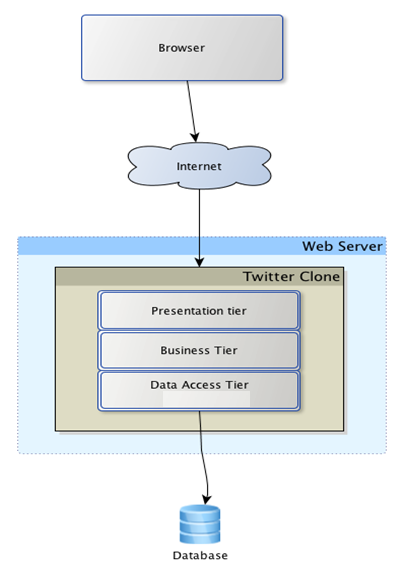
The following are the identified use-cases for the TwitterClone application

1. Signup : User should be able to sign-up before being able to post
2. Tweet : User, after registration, can tweet (post a message) and all his followers should receive the message
3. Edit Tweet: User can edit his tweet
4. Delete Tweet: User can delete his tweet
5. Search Users: User can search for other registered users
6. Follow: User (after a search) follows another user (to be able to receive his tweet)
7. Edit profile: User updates his profile (email, name,etc)
8. Delete Account: User decides to leave the application by deleting his account

For the sake of simplicity, other use-cases like re-tweeting, replying, mentions have not been included. But you’re free to expand the scope of the application.

### 2.2 Architecture Diagram

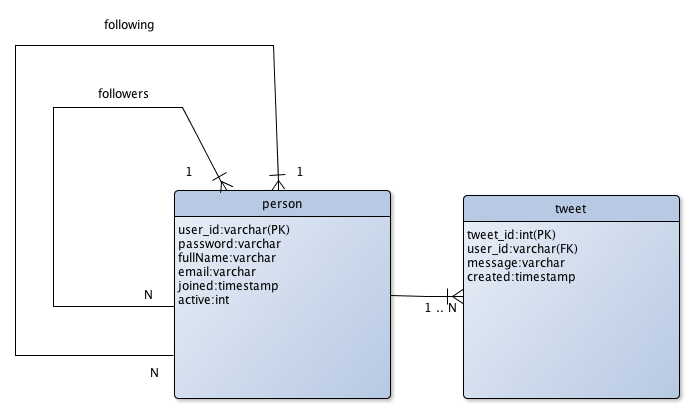
Design the application ideally with a typical 3-tier architecture with a presentation layer (jsp), business tier and the data access tier (where you’d use Hibernate)



Twitter Clone represents the deployment artefact (.jsp, CSS) which is deployed in a web server like Tomcat. The different tiers are nothing but terms used to segregate the presentation(jsp,CSS), business tier(Spring MVC code which orchestrates between these 2 layers) and data tier (Spring ORM code which handles database calls through Hibernate). Each of these sections will be described in Section 3, in detail.

### 2.3 Database Design

The database schema for the TwitterClone application is straightforward with only two entities namely Person and Tweet as shown below



A person has a one-to-many relation with a tweet (1 person -> many tweets). Similarly, a person has a 1...N relation with others in the form of following and followers. i.e,

* + - A person can have many followers
    - A person can follow other persons

### 2.3.1 Tables

The tables have only preliminary specification. You are free to modify as per your need.

**PERSON**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Size** | **Not Null** |
| user\_id (PK) | varchar | 25 | Y |
| password | varchar | 50 | Y |
| fullName | varchar | 30 | Y |
| email | varchar | 50 | Y |
| joined | DateTime | - | Y |
| active | bit | 1 | Y |

*password* - Store it encrypted using MD5 or SHA (The DB will provide these options)

**FOLLOWING (table representing the relation between persons ie. Following & Followers)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Size** | **Not Null** |
| user\_id (FK) | varchar | 25 | Y |
| following\_id(FK) | varchar | 25 | Y |

**TWEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Type** | **Size** | **Not Null** |
| tweet\_id(PK) | int (Auto increment) | 10 | Y |
| user\_id(FK) | varchar | 25 | Y |
| message | varchar | 140 | Y |
| created | DateTime | - | Y |

PK *- Primary Key,* FK *- Foreign Key*

## 3. Detailed Design

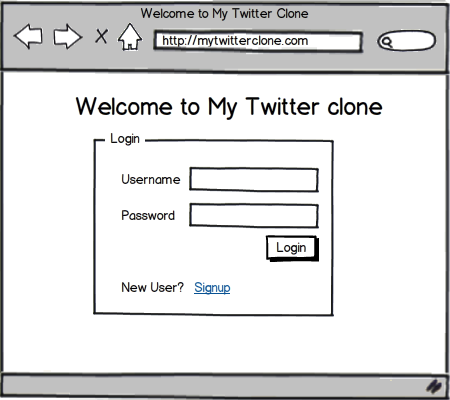
### 3.1 Presentation

Design the presentation (with jsp/CSS). The ideal requirement is to have the following

### 3.1.1. Login/Sign up page

The user should be presented with the home page/login page (when unauthenticated). A sample

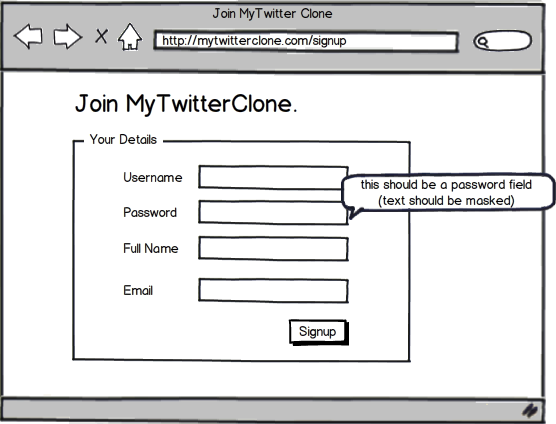
home/login page mock-up is as below. You’re free to design your own page.



### 3.1.1a. Sign up page

It is up to your discretion to either have a separate page for sign-up or use the home page itself.

A UI wireframe for sign-up is given below



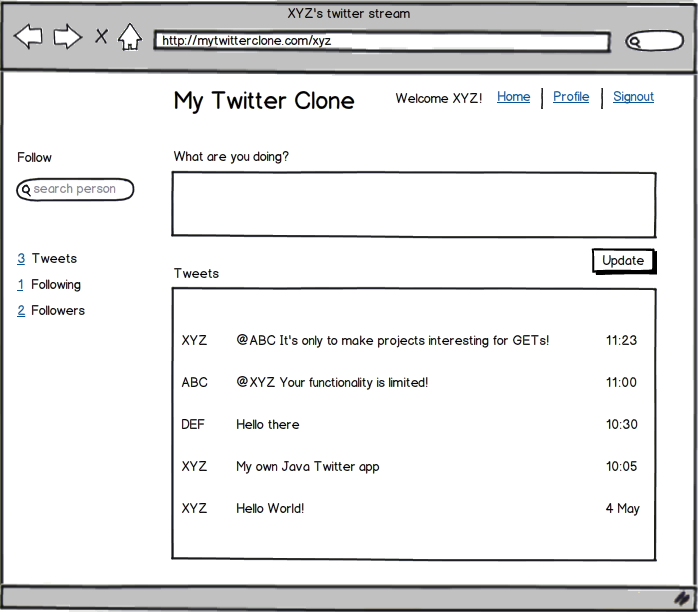
Any error messages (during a sign-up) should be presented to the user on the same page itself.

### 3.1.2. Home page (Where users can tweet, search others)

The home page, let’s call it the twitter stream, is where most of the action takes place. Users can tweet, edit or delete tweets, search for other registered users, follow/unfollow them if needed, update profile and delete the account.

When the user opts to sign out, automatic redirection to the login page should take place.

A UI wireframe of the main page is as below. (*Again, you’re free to design your own*)



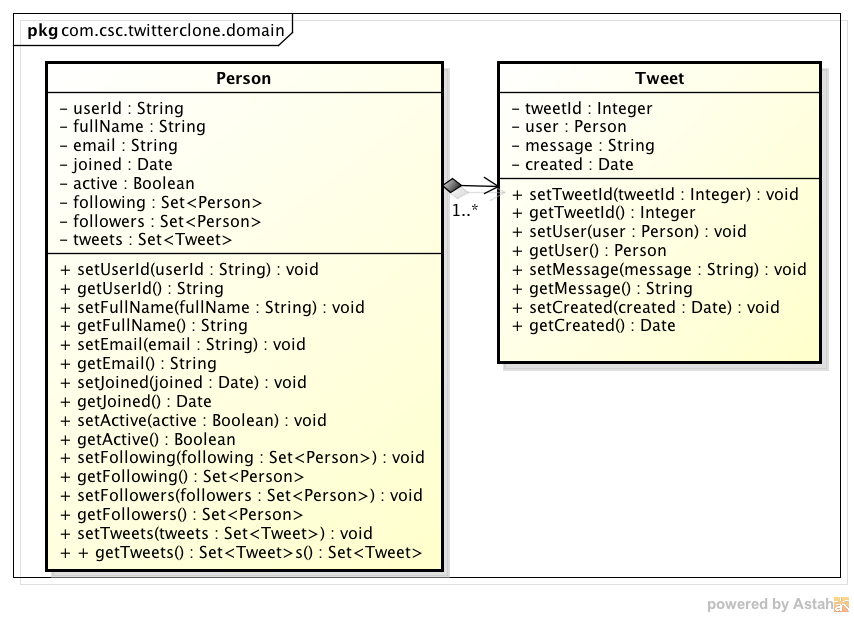
As soon as a user tweets, the tweet should appear on his/her follower’s pages. This can be done with Ajax or with automatic timed page refreshes.

### 3.1.3. Profile page

The profile page enables user to edit profile details like full name, update password, etc., It should also provide an option to delete the account itself. Design is left to your choice.

### 3.2 Domain/Model

As defined earlier, there are two domain objects in the TwitterClone application. The class diagram of the domain objects is given below



If you’re familiar with UML representation, it should be easy to note that the Person domain object has an instance variable called *“tweets”* that represents a “*has-a*” relationship with Tweet domain object(Composition). In other words, when a person is deleted, all the tweets are deleted too.

### 3.3 Business tier & Data Access tier

The business/middle tier holds business logic and validations, if any for user/tweet CRUD (Create, Read, Update and Delete) operation. This is where you will consider if the user can tweet, perform validations on the message being tweeted (if you choose to validate) and so on.

**Signup**The following is the sequence of operations for a sign-up:

1. User lands in the sign-up page and enters his details.
2. The validation of the fields should be done with the help of JQuery on client side.
3. Once validation is successful the data is submitted from the browser and reaches the code behind class of the page.
4. The requisite class for the user should be invoked and data should be persisted to database using Spring ORM .
5. Any exception encountered should be intimated to end user in a user-friendly manner.

The other operations like find, edit and delete also follow a similar sequence.

**Tweet operation**

The following is the sequence of operations for tweet operation:

1. User logs on and tweets.
2. The validation of the fields (like maximum 130 characters and number of characters remaining) should be done with the help of Java script on client side.
3. The data is submitted from the browser and reaches the code behind class of the page.
4. The object of tweet is invoked and the tweet is persisted in database using Spring ORM
5. Any exception encountered should be intimated to user in a user-friendly manner.

### 3.4 Project Structure

You need to create the project using Eclipse IDE. There should be separate folder for keeping script files (Script folder) and CSS files (Content folder). Master page and other pages can remain on the root folder.

## 4. Additional Specifications

### 4.1 Recommendations

1. Use. properties files to store your application constants or database values.
2. Avoid using tables for layout in your jsp design. Use <div> with the width attribute instead.
3. Separate your style(.css) and js from your markup (jsp) and move it to their own files.
4. Implement a good logging mechanism to log system errors/unhandled exception.(AOP in spring)
5. Test across multiple browsers. As a standard, use Firefox 26.0 & above, IE 9.0 and above& Google Chrome.