330660264 Srs for Tumblr

Software Engineering (Lovely Professional University)

Software Requirements Specification

for

Tumblr

Prepared by Pavan Chandraval

Registration number -11410543

Roll.no – B58 Section -K1429

Lovely Professional University

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of Tumblr. It describes the software requirements and specifications to Tumblr which is a microblogging and social networking website founded by David Karp in New York city in 2007, and is presently owned by Yahoo!

1.2 Document Conventions

Font: Times New Roman 12

1.3 Intended Audience and Reading Suggestions

The document is intended for all the stakeholder customer, new users to social networking sites and the developer (**designers, testers, maintainers**).

1.4 Product Scope

The page views of Tumblr to engineer ratio is 1 billion PV per month to 1 engineer. This means that every line of code that every developer writes has huge impact. The focus is on building tools and technologies that will advance the state of the art while dealing with massively scaled websites as it is growing quickly with over 30,000 requests in past and 1,300 posts per second. Tumblr aims to deliver an exciting range of new products that will enable the users to share their own creative content, discover content, and connect to one another in new ways.

The purpose of Tumblr is to have a way to put all the blogs, pictures, anything that one wants to share with other people in one single location. This is useful for personal interests, such as putting writing compositions together or photography compositions together but also for educational reasons.

1.5 References

https://www.tumblr.com/policy/en/terms-of-service#dmca

https://www.**tumblr**.com/

2. Overall Description

2.1 Product Perspective

Tumblr is a new self-contained product which was released in 2007 and is presently owned by Yahoo! Tumblr allows one to express himself freely and use it to reflect who you are, and what you love, think, and stand for. Tumblr today hosts over 45 million blogs with over 19 billion posts to date.
The website tumblr.com works 24 hours. The website identifies a user by a username and a password. Tumblr celebrates creativity.

2.2 Product Functions

Blog management

Dashboard: The dashboard is the primary tool for the typical Tumblr user. It is a live feed of recent posts from blogs that one has followed. Through the dashboard, users are able to comment, reblog, and like posts from other blogs that appear on their dashboard. The dashboard allows the user to upload text posts, images, video, quotes, or links to their blog with a click of a button displayed at the top of the dashboard. Users are also able to connect their blogs to their other networking sites such as Twitter and Facebook accounts, so whenever they make a post, it will also be sent as a tweet and a status update.

Queue: Users are able to set up a schedule to delay posts that they make. They can spread their posts over several hours or even days
.
Tags: For each post a user creates, they are able to help their audience find posts about certain topics by adding tags. If someone were to upload a picture to their blog and wanted their viewers to find pictures, they would add the tag #picture, and their viewers could use that word to search up posts with the tag #picture.

HTML editing: Tumblr allows users to edit their blog's theme HTML coding to control the appearance of their blog. Users are also able to use a custom domain name for their blog.

Inbox and messaging
Tumblr blogs may optionally allow users to submit questions, either as themselves or anonymously, to the blog for a response. Tumblr also offered a "fan mail" function, allowing users to send messages to blogs that they follow.
On November 10, 2015, Tumblr introduced an integrated instant messaging function, allowing users to chat between other Tumblr users. The feature is being rolled out in a "viral" manner; it was initially made available to a group of 1500 users; other users may receive access to the messaging system if they are sent a message by any user that has received access to the system itself. The messaging system only supports text-based conversations, although other features (such as group chat and image embeds) will be added in the future. The messaging platform will also replace the fan mail system, which has been deprecated.

Editorial content
In May 2012, Tumblr launched Storyboard, a blog managed by an in-house editorial team which features stories and videos about noteworthy blogs and users on Tumblr. In April 2013, Storyboard was shut down.

Smash the cache
When the Tumblr iOS app shows a message that one knows he had answered in the, one can smash the cache and the inbox returns to normal on the app. When the Tumblr app is taking up too much space on iPhone, we smash the cache. The Tumblr app sheds hundreds of megabytes without having to uninstall and reinstall.

2.3 User Classes and Characteristics

This product is used as a social networking site where a user i.e. the person who has an account on Tumblr can put all the blogs, pictures, anything that he/she wants to share with other people in one single location. This is useful for personal interests, such as putting writing compositions together or photography compositions together but also for educational reasons.

The users of this site do not require any technical expertise or skills. Anyone above the age of 13 years can use Tumblr to posts his/her original content. Users with different age group and interests use this site. Users who are interested in blogging, they use this site for blogging and other use Tumblr according to their interest.

Operating Environment

The hardware, software and technology used should have following specifications:
• Greatly user friendly
• Ability to run for long period of time.
• Resource requirement from the device must be minimum.
• Must support touch and keypad inputs from device.
• Ability to connect to server side database and Softwares.
• Ability to validate user and get input from user.
• Ability to provide output in minimum amount of time.
.

2.4 Design and Implementation Constraints

The Services change frequently, and their form and functionality may change without prior notice to you. Tumblr retains the right to create limits on and related to use of the Services in its sole discretion at any time with or without notice. Tumblr may also impose limits on certain services or aspects of those services or restrict your access to parts or all the services without notice or liability. Tumblr may change, suspend, or discontinue any or all the services at any time, including the availability of any product, feature, database, or content. Tumblr may also terminate or suspend Accounts at any time, in its sole discretion.

2.5 User Documentation

Tumblr provide its user some documents components such as user -manual, online help, and tutorials for the new users so that they will be able to use this site easily and efficiently. With the help of these documentations the user will be able to explore this site in a better way. Video tutorials will be provided to the new users.

2.7 Assumptions and Dependencies

- Hardware never fails
- Website is bugless
- Limited number of activity per day.

3. External Interface Requirements

3.1 User Interfaces

The user interface should be interactive, such that the user of Tumblr can use this site easily and effectively. the interface needs to be in such a way that it contains images, graphical user interface needs to be good, buttons functions need to be used that make this site interactive.

3.2 Hardware Interfaces

The devices (pc, tablets, mobiles etc.) should have following features:

- Ability to store, transfer and process big data about users.
- Optimal RAM
- Network Connectivity
- As tumbler is a 24-hour working site, so a robust hardware is required.

3.3 Software Interfaces

The software requirements for Tumblr is such that it should meet the specifications required for any social networking site. It should support audio, video, text and image files with different extensions.

3.4 Communications Interfaces

- There should be good internet connection.
- The client system can be any system with normal Operating System and Internet Explorer.
- The system uses HTTP and FTP.

4. System Features

Functional Requirements 1: validate user

- **Introduction:** This function is used when a new user joins in.
- **Inputs:** when a new user joins in he is asked whether he is an existing user or already exists. He is asked to enter his email id or contact number.
- **Processing:** A confirmation message or link is send to the new user.
- **Outputs:** A new user has been validated.

Functional Requirements 2: create account

- **Introduction:** This function is used to create a new account for a user.
- **Inputs:** the new user is supposed to enter his details which include his name, date of birth, email id.
- **Processing:** the eligibility criteria is checked in database and a new account is successfully created.
- **Outputs:** A new user account has been created.

Functional Requirements 3: creation of text post

- **Introduction:** This function is used to create a new text post. this can be either your own text post or reclogged one.
- **Inputs:** the user is supposed to enter manually the text he/she wants to post.
- **Processing:** the text post is displayed on the dashboard.
- **Outputs:** A new text post has been created which is visible to all the followers of the user.

Functional Requirements 4: creation of Photo post

- **Introduction:** This function is used to create a new photo post. this can be either your own photo post or reclogged one.
- **Inputs:** the user is supposed to upload the photo post he/she wants to post.
- **Processing:** the photo post is displayed on the dashboard.
- **Outputs:** A new photo post has been created which is visible to all the followers of the user.

Functional Requirements 5: creation of Quote post

- **Introduction:** This function is used to create a new quote post. this can be either your own quote post or reclogged one.
- **Inputs:** the user is supposed to enter manually the quote post he/she wants to post.
- **Processing:** the quote post is displayed on the dashboard.
- **Outputs:** A new quote post has been created which is visible to all the followers of the user.

Functional Requirements 6: creation of Chat post

- **Introduction:** This function is used to create a new chat post. this can be either your own photo post or reclogged one.
- **Inputs:** the user is supposed to enter chat post he/she wants to post.
- **Processing:** the chat post is displayed on the dashboard.
- **Outputs:** A new chat post has been created which is visible to all the followers of the user.

Functional Requirements 7: creation of Audio post

- **Introduction:** This function is used to create a new audio post. this can be either your own audio post or reclogged one.
- **Inputs:** the user is supposed to upload the audio post he/she wants to post.
- **Processing:** the audio post is displayed on the dashboard.
- **Outputs:** A new audio post has been created which is visible to all the followers of the user.

Functional Requirements 8: creation of Video post

- **Introduction:** This function is used to create a new video post. this can be either your own video post or reclogged one.
- **Inputs:** the user is supposed to upload the video post he/she wants to post.
- **Processing:** the video post is displayed on the dashboard.
- **Outputs:** A new video post has been created which is visible to all the followers of the user.

Functional Requirements 9: creation of Link post

- **Introduction:** This function is used to create a new link post. this can be either your own link post or reclogged one.
- **Inputs:** the user is supposed to enter the link post he/she wants to post.
- **Processing:** the link post is displayed on the dashboard.
- **Outputs:** A new link post has been created which is visible to all the followers of the user.

Functional Requirements 10: customization of information

- **Introduction:** This function is used to change the information which is provided by user at the starting. User can change his/her information at any time.
- **Inputs:** the user is supposed to change the information.
- **Processing:** the information of the user has been changed.
- **Outputs:** the changed information of the user is visible to all the followers of the user.

Functional Requirements 11: customization of themes

- **Introduction:** This function is used to change the themes which are present on the user account, he/she can change the theme according to their wish.
- **Inputs:** the user is supposed to change the themes.
- **Processing:** the themes present in the user account has been changed.
- **Outputs:** the changed theme of the user is visible to all the followers of the user.

Functional Requirements 12: customization of appearance

- **Introduction:** This function is used to change the appearance of the home page of the Tumblr profile of the user.
- **Inputs:** the user is supposed to change the appearance.
- **Processing:** the appearance of the user's home page has been changed.
- **Outputs:** the changed appearance of the user's home page is visible to all the followers of the user.

Functional Requirements 13: customization of service

- **Introduction:** This function is used to change the services needed to the user,
- **Inputs:** the user is supposed to change the services
- **Processing:** the services required to the user has been changed.
- **Outputs:** the services have been changed.

Functional Requirements 14: customization of community

- **Introduction:**
- **Inputs:** the user is supposed to change the information.
- **Processing:** the information of the user has been changed.
- **Outputs:** the changed information of the user is visible to all the followers of the user

Functional Requirements 15: customization of page

- **Introduction:** This function is used to change the page settings.
- **Inputs:** the user is supposed to change the page setting.
- **Processing:** the page setting has been changed.
- **Outputs:** the page in the user account has been changed.

Functional Requirements 16: Advanced customization

Introduction: This function is used to include other options, including time zone selection, custom CSS, and how many posts you wish to display per page on your blog.

- **Inputs:** the user is supposed to include time zone selection, how many posts to display.
- **Processing:** the time selection, custom CSS has been changed.

- **Outputs:** the new changes have been displayed in the users account.

Functional Requirements 17: Like post

- **Introduction:** This function is used if you or someone likes your post so much that they want to share it, they can reclog it or if they don't want to share, they can simply like that.
- **Inputs:** the user is supposed to like the post of the other person.
- **Processing:** the like on the post is displayed on the post.
- **Outputs:** each post is noted with the like they get, with each like post is incremented by one note.

Functional Requirements 18: Reclog post

- **Introduction:** This function is used if you or someone likes your post so much that they want to share it, they can reclog it.
- **Inputs:** the user is supposed to reclog the post of the other person.
- **Processing:** liked post is reclogged
- **Outputs:** each post is noted with the like they get, with each like or reclog post is incremented by one note.

Functional Requirements 19: Queuing

- **Introduction:** This function is used when you are offline, you can add the posts to 1f2our queue and choose at which interval of time they will be published.
- **Inputs:** the user add posts to the queue according to his/her wish.
- **Processing:** after certain time interval, the post will be published.
- **Outputs:** when the time interval of the post is completed, the post get automatically published.

Functional Requirements 20: Drafts

- **Introduction:** This function is used when you are offline, you can add the posts to our queue and choose at which interval of time they will be published.
- **Inputs:** the user add posts to the queue according to his/her wish.
- **Processing:** after certain time interval, the post will be published.
- **Outputs:** when the time interval of the post is completed, the post get automatically published.

Functional Requirements 21: Queuing

- **Introduction:** This function is used to save the posts online.
- **Inputs:** the user can save the post online and can then finish the post when user next log onto.

- **Processing:** The post is saved online.
- **Outputs:** post will be finished when user next log onto.

Functional Requirements 22: Sending Message

- **Introduction:** This function is used to send the message to other users.
- **Inputs:** the user is supposed to enter the message whatever he/she wants.
- **Processing:** The message has been created.
- **Outputs:** the message is visible to the user to whom it is written.

Functional Requirements 23: Comment

- **Introduction:** This function is used to do comment on the post that person likes.
- **Inputs:** the user is supposed to write comment on the post.
- **Processing:** The comment on the post has been written.
- **Outputs:** comment on the post is displayed.

Functional Requirements 24: Follow

- **Introduction:** This function is used to follow other users.
- **Inputs:** The user is supposed to follow the person whom he/she wants to follow.
- **Processing:** people's you will follow will displayed on dashboard.
- **Outputs:** people the user follows has been displayed to the followers of that users.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Algorithm used for searching must be highly efficient.
- Lite version of website should run on minimum requirements without any burden on device
- Algorithm used for downloading and uploading data must be highly efficient.

5.2 Safety Requirements

Tumblr, Inc. takes the private nature of information that is very seriously. Note that Tumblr was acquired by, and is now a wholly-owned subsidiary of, Yahoo! Inc. As of June 19, 2013, the privacy policy describes how the information collected is treated when you visit and use tumblr.com) and/or Tumblr's other domains, products, advertising products, services, and/or content, including our iOS and Android mobile applications.

5.3 Security Requirements

Account Information: When you create an account on the Services ("**Account**"), information such as your username, password, age, and email address ("**Account Information**") is inputted. The account Information, alone or together with other information, is used to enhance and improve the Services, such as by personalization. Age is used to verify that you can lawfully use the Services. Email address to verify your Account and to communicate with you. Users can also look for their friends by email address; you can, however, opt out of email lookup through your Account Settings.
`

Email Communications with Us: As part of the Services, one may occasionally receive email and other communications from the site. Administrative communications relating to your Account (e.g., for purposes of Account recovery or password reset) are considered part of the Services and your Account, which you cannot opt-out from receiving

5.4 Software Quality Attributes

4.4.1 Availability
The Tumblr Website has to be available 24 hours a day.

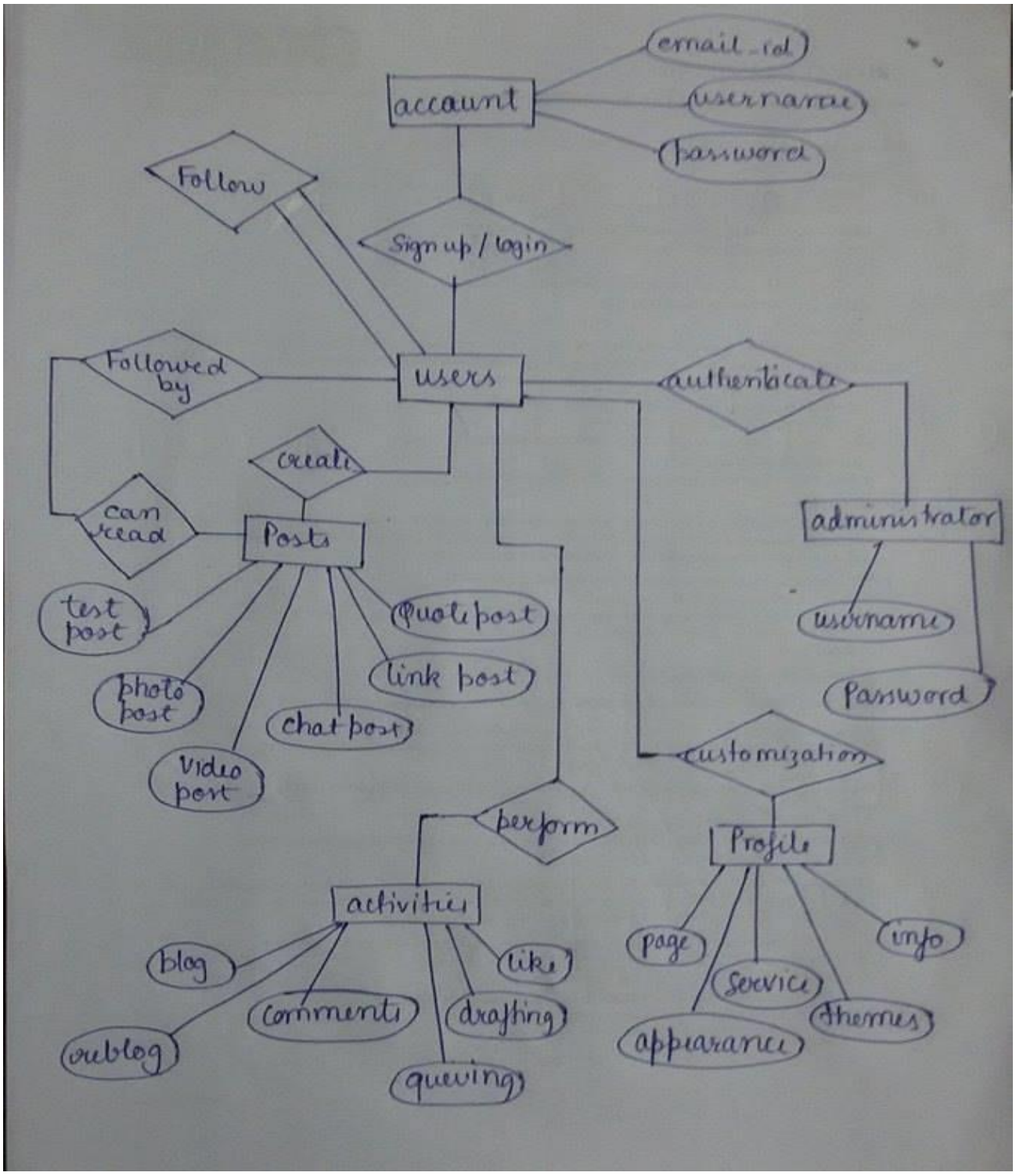
4.4.2 Security
The Tumblr Website's connection must be highly secured.

4.4.3 Maintainability
Maintain by highly professional maintainers, only maintainers are allowed to update and add new features to the network.

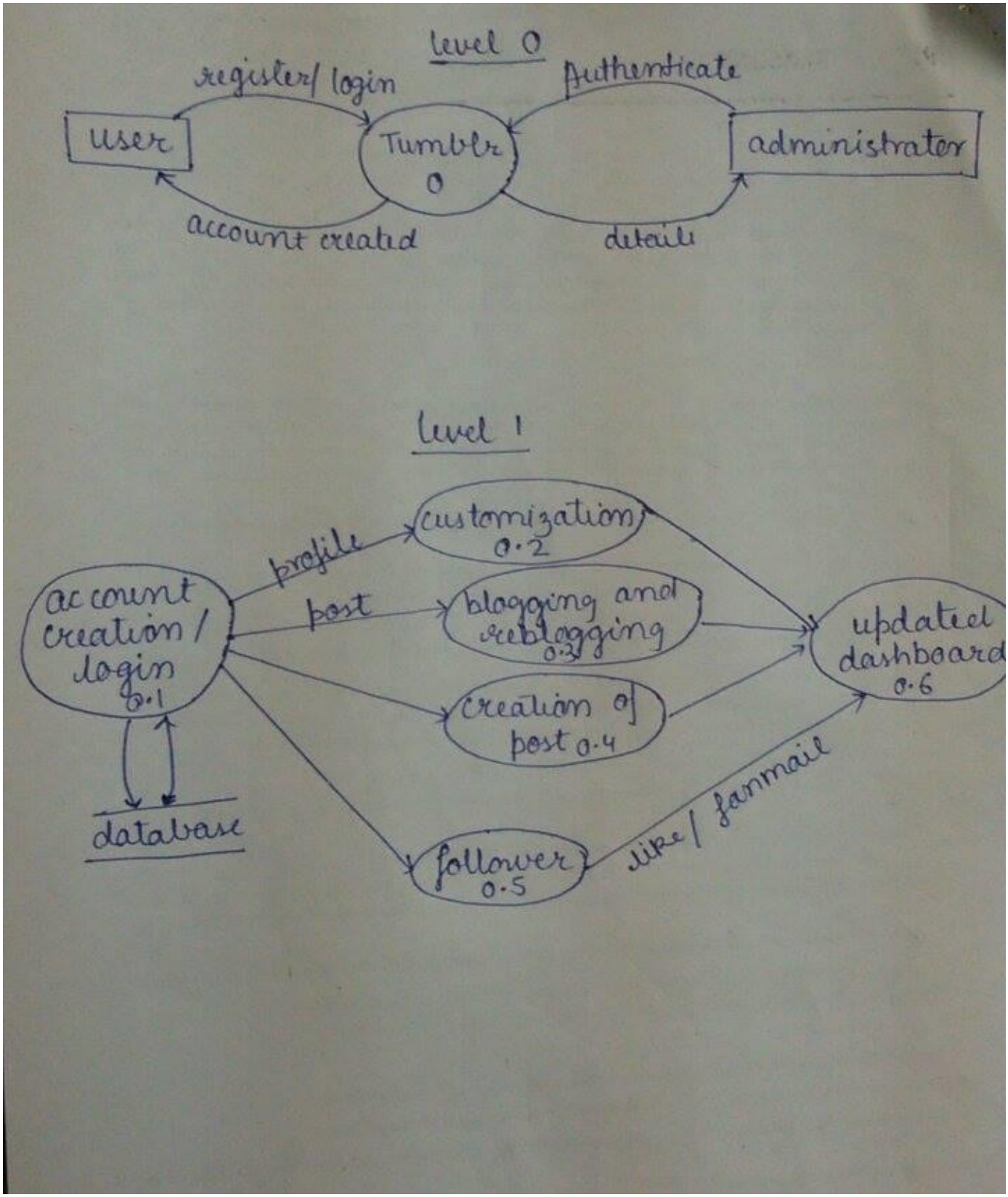
6. Other Requirements

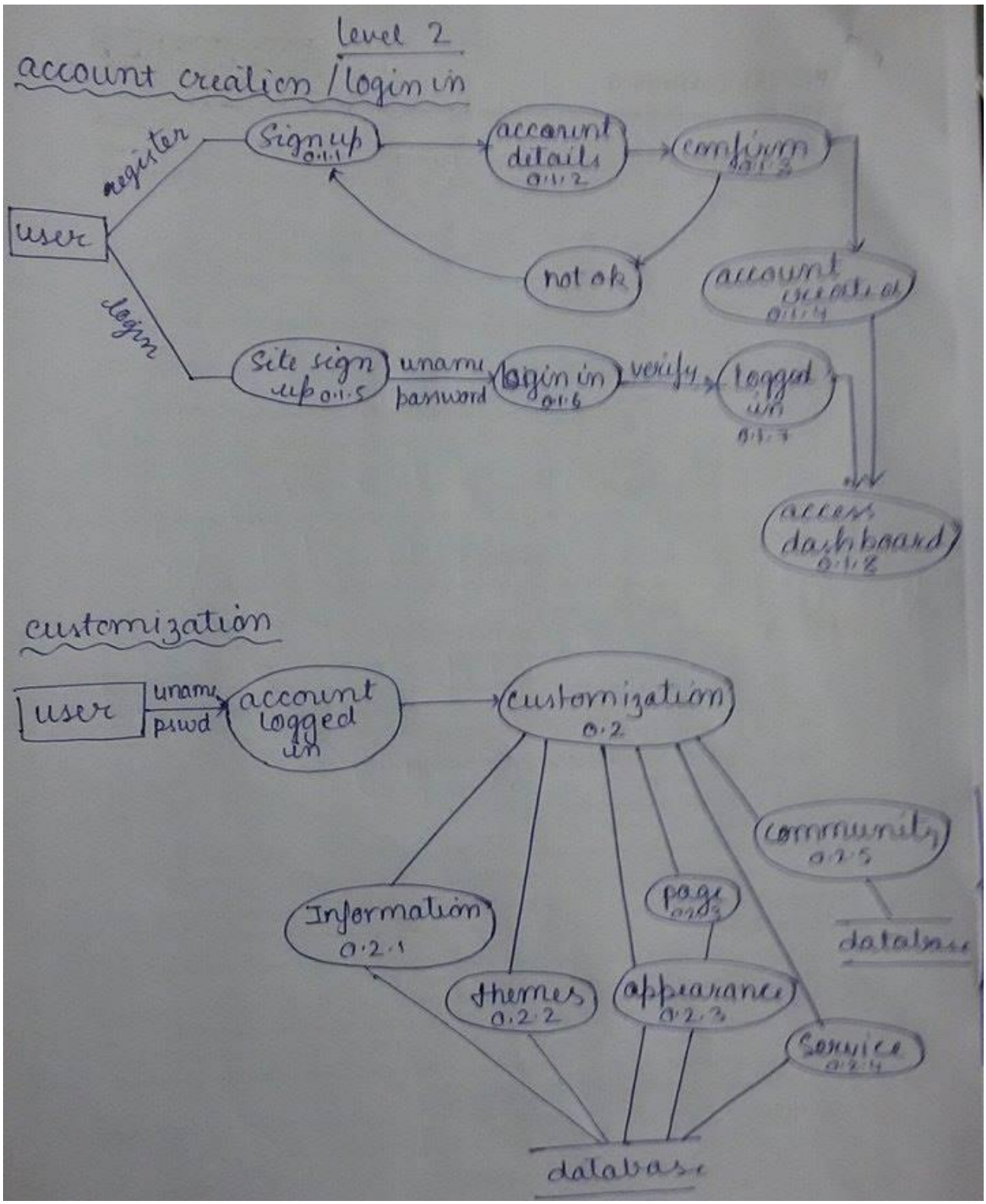
- Video and Audio calling.
- Security of Tumblr.
- Database for tumbler.

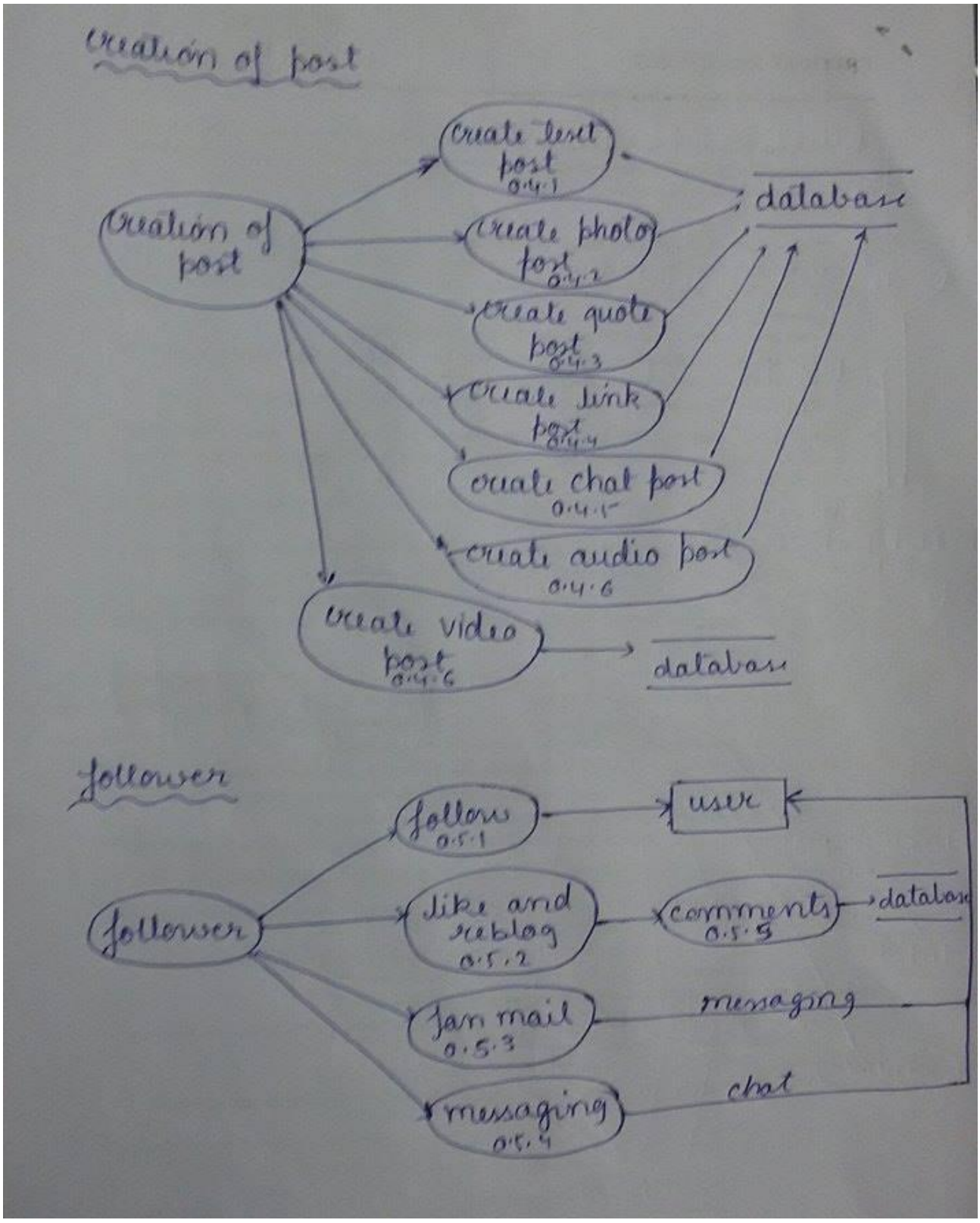
7. ER Diagram



8. DFD FOR TUMBLR







9.TESTING:

Test Cases TC 1:

Login Purpose: To check user authorization into Tumblr
Pre-Requirement: User must have entered correct information
Test Data: Login id = Email address stored in database.
Test-Methods:
1)Password = String (Numerals, alphabets, Symbols)
2)Enter Invalid Email Id and Correct password.
3)Enter Correct Email Id and Invalid password.
4)Leave Email Id and Password Fields Empty.
5)Enter Email ID and leave password
6)Enter Email ID and Password.

TC 2:

Control Panel Purpose: To check if admin can change and modify database.
Pre-Requirement: User must be admin.
Test Data: MySQL query and pup Code.
Test-Methods:
1) Admin can execute new modified PHP code.
2) Admin must successfully execute MySQL queries.

TC 3:

Search Result: To check if entered data is searched from database in right manner.
Pre-Requirement: User must be registered user.
Test Data: Other user's name, Post, Pictures, songs etc.
Test-Methods:
1)Don't enter anything in box and search.
2)Enter Post name and search.
3)Enter song name and search.
4)Enter Picture name and search.

TC 4:

Reclog purpose: To check if user can reclog a post successfully.
Pre-Requirement: User must be registered user.
Test Data: pictures, posts, audio, video
Test-Methods:
1)Search through entire posts and reclog.
2)Enter Audio name and search then reclog song.
3)Enter video name and search then reclog video.

TC 5:

Log Out Purpose: To check if User Can Log Out from Tumblr successfully.
Pre-Requirement: User must have already logged in.
Test Data: NULL
Test-Methods:
1)Close the tab and check if it is automatically logged out or not.

2)Use log Out button to log out successfully.

TC 6:

Sign Up Purpose: To register into Tumblr database successfully.
Pre-Requirement: User must be ready with payment methods.
Test Data: Password, Email Id, and other user credentials.
Test-Methods:
1)Close the tab and check if it is automatically logged out or not.
2)Use log Out button to log out successfully.

TC 7:

Post Creation Purpose: To create a new post in Tumblr.
Pre-Requirement: User must have already be registered in Tumblr.
Test Data: NULL
Test-Methods:
1)Create post.
2)Create post==>logout==>login==>check if post is published or not.

TC 8:

Change theme Purpose: Change theme of user's account.
Pre-Requirement: User must be a registered user
Test Data: Themes
Test-Methods:
1) Click on change theme of Tumblr.
2) Select a theme from themes List.
3) Press ok.
4)Check if theme was changed

TC 9:

Compatibility Purpose: Compatible in browsers
Pre-Requirement: User must have a browser installed.
Test Data: Tumblr
Test-Methods:
1)Check compatibility on

TC 10:

Customization Purpose: Customization of user's data.
Pre-Requirement: User must be a registered user
Test Data: user's detail
Test-Methods:
1)check whether new details saved in the database.
2)check whether old details are removed from the database.

10.FRONT END

Front end for signup/login form

```
<!DOCTYPE html>
<html lang = "en">

  <head>
    <meta charset = "utf-8">
    <meta http-equiv = "X-UA-Compatible" content = "IE = edge">
    <meta name = "viewport" content = "width = device-width device, initial-scale = 1">

    <title>Tumblr</title>


    <link href = "bower_components/bootstrap/dist/css/bootstrap.min.css" rel =
"stylesheet">
     <script src="bower_components/bootstrap/dist/js/Myjs.js"
type="text/javascript"></script>

  </head>

  <body  >

    <script src =
"https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
     <script src = "//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>



  <div class="container" >
   <div class="col-md-4 col-md-offset-4">
    <form class="form-signin"  action="controllerServlet">
    <div class="panel-heading"><h2 class="form-signin-heading">Please sign in</h2>
     </div>
    <div class="panel-body">
     <label for="inputEmail" class="sr-only">Email address</label>
     <input type="email" name="email" id="inputEmail" class="form-control"
placeholder="Email address" required autofocus>
     <label for="inputPassword"  class="sr-only">Password</label>
     <input type="password" name="password" id="inputPassword" class="form-control"
placeholder="Password" required>
     <div class="checkbox">
      <label>
```

```
                <input type="checkbox" value="remember-me"> Remember me
              </label>
            </div>
            <button class="btn btn-primary btn-lg" type="submit" name="button"
        value="login">Log in</button>        
              <a href="forgotpassword.html" >Forgot Password </a>
          </div>
          </form>
          </div>
        </div>
       </body>
      </html>
```

Code for home.jsp

```
   <jsp:useBean id="user" class="model.user"></jsp:useBean>
<!DOCTYPE html>
<html lang="en">

<head>

   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <meta name="description" content="">
   <meta name="author" content="">

    <title>Theme</title>
      <link href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
      <link rel="stylesheet" href="//code.jquery.com/ui/1.12.0/themes/base/jquery-ui.css">
      <link rel="stylesheet" href="/resources/demos/style.css">

      <style>
      #draggable
      {
       width: 150px; height: 150px; padding: 1em;
        padding-top:5px
      background:buttonface;
      }
      #mygame
      {
      width: 200px;
      float: left;
      }
```

```
#startover
{
width: 100px;
height: 50px;
text-align: center;
padding-top: 5px;
float: right;
}
.hoverOut
{
color: red;
background-color: yellow;
}
.hoverIn
{
background-color: red;
color: yellow;

}
#logout
{
float: right;
padding-right: 10px;
padding-top: 10px;
}

}

</style>
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
<script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
 <script>
 $( function() {
 var score=0;

    $('#draggable').draggable();

 $('#startover').addClass('hoverOut');
 $('#success').hide();
 $('#doremon').click(function()
   {

    score++;
    $('#score').text(score);
    $('#success').show('slow').fadeOut(2000);
```

```
        }
        );

      $('#startover').hover(function()
                {
         score=0;
         $('#score').text(score);
         $('#startover').addClass('hoverIn').removeClass('hoverOut');
                },function()
                {
                        $('#startover').removeClass('hoverIn').addClass('hoverOut');

                }
      );

      } );
      </script>

  </head>
  <body>

   <div id="wrapper">

   <div  id="logout">
   <form class="form-group" action="usercontroller" method="post">
   <input type="submit" class="btn btn-primary"  name="button"  value="logout"/>
   </form>
   </div>
     <div id="draggable" class="ui-widget-content">
                        <p>Welcome TO Tumblr</p>
     </div>
   </div>
</body>

</html>
```

10.BACK END

Back End for Signup/login.

```
package controller;
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.channels.SeekableByteChannel;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import dao.userdao;
import model.user;

public class usercontroller  extends HttpServlet
{

        @Override
        protected void doPost(HttpServletRequest req, HttpServletResponse resp)
                        throws ServletException, IOException
                        {
                HttpSession session=req.getSession();

                resp.setContentType("text/html");
                   PrintWriter out=resp.getWriter();
            String button=req.getParameter("button");

            if(button.equals("logout"))
            {
                RequestDispatcher rd=req.getRequestDispatcher("index.html");
                rd.include(req, resp);
                 session.invalidate();

            }

            if(button.equals("login"))
              {
```

```
                String userid=req.getParameter("userid");
                 String password=req.getParameter("passwprd");
                 System.out.println();
                user user=new user();
                 user.setUserid(userid);
                user.setPassword(password);
                    userdao userdao=new userdao();
                 Boolean b=userdao.find(user);

                 if(b==true)
                 {

                 session.setMaxInactiveInterval(1);
               session.setAttribute("userid", userid);
                  RequestDispatcher rd=req.getRequestDispatcher("home.jsp");
              rd.include(req, resp);
                  }
                  else
                  {
                          out.println("Server problem or password is not correct");
                          RequestDispatcher rd=req.getRequestDispatcher("index.html");
                          rd.include(req, resp);
                  }

              }

        if(button.equals("signup"))
            {

                 String userid=req.getParameter("userid");
                 String password=req.getParameter("password");
                 String email=req.getParameter("email");
                 String name=req.getParameter("name");

                  user user=new  user();
                  user.setEmail(email);
                 user.setName(name);
                 user.setPassword(password);
                 user.setUserid(userid);

                 userdao userdao=new userdao();
                 Boolean b=userdao.create(user);
                 if(b==true)
                 {
                 out.println("Sucessfully account created");
```

```
                RequestDispatcher rd=req.getRequestDispatcher("index.html");
                rd.include(req, resp);
                    }
                  }
   }

}
```

Model User.java

```
package model;
public class user
{

        String name,userid,password,email;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getUserid() {
                return userid;
        }

        public void setUserid(String userid) {
                this.userid = userid;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
```

```
                this.email = email;
        }
}
```

Dao

```
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import util.ConnectionProvider;
import model.user;

public class userdao
{
        Connection con=ConnectionProvider.getconnection();
public Boolean find( user user)
 {
        Boolean flag = true;
        try {
                PreparedStatement ps=con.prepareStatement("select name from COLLAGESIGNIN
where userid=? and password=?");
                 ps.setString(1, user.getUserid());
                 ps.setString(2, user.getPassword());
                 ResultSet rs=ps.executeQuery();
                 if(rs.next())
                 {
                        flag=true;
                 }
          } catch (SQLException e) {

        }
         return flag;
}

public Boolean create(user user)
{
        Boolean flag=false;
        try {
                PreparedStatement  ps=con.prepareStatement("insert into COLLAGESIGNIN
values(?,?,?,?)");
                ps.setString(3, user.getEmail());
```

```
                ps.setString(1, user.getName());
                ps.setString(2, user.getUserid());
                ps.setString(4, user.getPassword());
            ResultSet rs=ps.executeQuery();
             if(rs.next())
             {
                  flag=true;
             }

        } catch (SQLException e) {

                flag=false;
                e.printStackTrace();
        }
   return flag;
}


}
```

ConnectionProvider.java

```
package util;

import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Properties;

public class ConnectionProvider
{

        static Properties prop;
        static
        {
                prop=new Properties();
                try
                {
                InputStream is=ConnectionProvider.class.getResourceAsStream("/db.properties");
                prop.load(is);
                }catch(Exception ex)
                {
                        ex.printStackTrace();
```

```
		}
	}

	public static Connection getconnection()
	{
		Connection con=null;
		try
		{
			Class.forName(prop.getProperty("driver"));
			con=DriverManager.getConnection(prop.getProperty("url"),
prop.getProperty("user"), prop.getProperty("password"));

		} catch (Exception e)
		{
			e.printStackTrace();
		}
		return con;
	}


}
```

10.Appendix A: Glossary

- **Account**

An individual has a single account on Tumblr which can be accessed using your login id and password. First time users need to sign up and create an account.

- **Sign-up**

The first-time users of Tumblr need to sign up that is register themselves by providing some basic contact information to the application. The application accesses the basic information of the user and stores it.

- **Username**

The username is the name which the user uses and is visible to other users using the application the username is not permanent and can be changed by going to the blog settings.

- **Posts**

What one puts or displays on his/her timeline or dashboard is called a post. Tumblr allows seven types of post that are text, photo, quote, link, chat, audio and video.

- **Microblogging**

Tumblr is a microblogging site to which the user makes short and frequent posts. The activity or practice of making short, frequent posts to a microblog is called microblogging. Tumblr is an exciting path for all bloggers who prefer to post frequently.

- **Reblogging**

It is the mechanism in blogging which allows users to repost the content of another user's post with an indication that the source is another user.

- **Follow**

This feature of Tumblr allows one to follow the posts and blogs of other users that is the follower gets a notification when the person being followed makes a post.

- **Dashboard**:

The dashboard is the primary tool for the typical Tumblr user. It gives live feed of recent posts from blogs that one follows. Through the dashboard, users are able to comment, reblog, and like posts from other blogs that appear on their dashboard