

EAST WEST UNIVERSITY



Project Report

Project Title: Socify-A Web Based social Networking system

Prepared by

Name: Junan Chakma

ID: 2009-3-60-009

Supervised by

Dr. Taskeed Jabid

Assistant Professor

Department of Computer Science & Engineering

East West university

September 16, 2015

Chapter 1

Introduction

1 Introduction

Socify is a social networking web application. This application has mainly three section, user section and admin section. In user section a user can login/registration, follow other user, see his/her followers, see his/her followed users , see other users profile, post tweet ,delete tweets, post picture, update own profile, find out other users. An admin can do all the things that a user can do as well as admin can delete any other user. If any user deleted by admin, all of his/her information(tweets, pictures, email, name) will be deleted and updated all users followers/followings list. There are also many features in this system such as email verification, forgot password, login with remember me check box, hashed password, automated testing, strong security.

1.2 Motivation

Now days web application is getting more popular and powerful. Most of the popular desktop software like Photoshop, MS Office has web version now. Unlike desktop application, web application can access from any part of the world by using Internet and it is also platform independent that means cross platform - apps can be easily ported to virtually any platform with a web browser. There are many advantages of web application over desktop application such as desktop applications needs to be updated per desktop, web applications are updated once at the server, The application will run on the browser regardless the platform, no installation required, accessible anywhere.

So these are the key reasons of my motivation to build a web based system.

1.3 Technology Stack

I used Ruby programming language and it's framework Rails to build Socify web application. For database system, I used Sqlite. For client side, I used html, css , javascript, jquery, bootstrap. I also

used Rubymine IDE and Ubuntu operating system.

Programming languages: Ruby, Javascript

Database system: Sqlite

Frameworks: Rails, JQuery, Bootstrap

Markup languages: Html, Css

IDE/Editor: Rubymine, Sublime text

Operating system: Ubuntu

1.4 Project Requirement

There are many tools I had to install to build this systems.

- Ruby interpreter
- Rails
- Sqlite
- JQuery
- Bootstrap
- Rubymine
- Subllime text
- Ubuntu

Chapter 2

Ruby and Rails Platform

2.1 Ruby:

Ruby is a dynamic, object-oriented, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write. It was designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto, a computer scientist in Japan.

Early concept

Ruby was conceived on February 24, 1993. In a 1999 post to the *ruby-talk* mailing list, Ruby author Yukihiro Matsumoto describes some of his early ideas about the language:

“I was talking with my colleague about the possibility of an object-oriented scripting language. I knew Perl (Perl4, not Perl5), but I didn't like it really, because it had the smell of a toy language (it still has). The object-oriented language seemed very promising. I knew Python then. But I didn't like it, because I didn't think it was a true object-oriented language — OO features appeared to be add-on to the language. As a language maniac and OO fan for 15 years, I really wanted a genuine object-oriented, easy-to-use scripting language. I looked for but couldn't find one. So I decided to make it.”

Matsumoto describes the design of Ruby as being like a simple Lisp language at its core, with an object system like that of Smalltalk, blocks inspired by higher-order functions, and practical utility like that of Perl.

The name "Ruby"

The name "Ruby" originated during an online chat session between Matsumoto and Keiju Ishitsuka on February 24, 1993, before any code had been written for the language. Initially two names were

proposed: "Coral" and "Ruby". Matsumoto chose the latter in a later e-mail to Ishitsuka. Matsumoto later noted a factor in choosing the name "Ruby" – it was the birthstone of one of his colleagues.

First publication

The first public release of Ruby 0.95 was announced on Japanese domestic newsgroups on December 21, 1995. Subsequently three more versions of Ruby were released in two days. The release coincided with the launch of the Japanese-language *ruby-list* mailing list, which was the first mailing list for the new language.

Already present at this stage of development were many of the features familiar in later releases of Ruby, including object-oriented design, classes with inheritance, mixins, iterators, closures, exception handling and garbage collection.

2.2 Rails

Rails, is a web application framework written in Ruby under MIT License. Rails is a model–view–controller (MVC) framework, providing default structures for a database, a web_service, and web pages. It encourages and facilitates the use of web standards such as JSON or XML for data transfer, and HTML, CSS and JavaScript for display and user interfacing. In addition to MVC, Rails emphasizes the use of other well-known software engineering patterns and paradigms, including convention over configuration (CoC), don't repeat yourself (DRY), and the active record pattern.

History

David Heinemeier Hansson, a Danish programmer created Ruby on Rails from his work on the project management tool Basecamp at the web application company also called Basecamp. Hansson first released Rails as open source in July 2004, but did not share commit rights to the project until February 2005. In August 2006, the framework reached a milestone when Apple announced that it would ship Ruby on Rails with Mac OS X v10.5 "Leopard", which was released in October 2007.

Rails version 2.3 was released on March 15, 2009 with major new developments in templates, engines, Rack and nested model forms. Templates enable the developer to generate a skeleton application with custom gems and configurations. Engines give developers the ability to reuse application pieces complete with routes, view paths and models. The Rack web server interface and Metal allow one to write optimized pieces of code that route around Action Controller.

Rails 3.1 was released on August 31, 2011, featuring Reversible Database Migrations, Asset Pipeline, Streaming, jQuery as default JavaScript library and newly introduced CoffeeScript and Sass into the stack.

Rails 3.2 was released on January 20, 2012 with a faster development mode and routing engine (also known as Journey engine), Automatic Query Explain and Tagged Logging. Rails 3.2.x is the last version that supports Ruby 1.8.7. Rails 3.2.12 supports Ruby 2.0.

Rails 4.0 was released on June 25, 2013, introducing Russian Doll Caching, Turbolinks, Live Streaming as well as making Active Resource, Active Record Observer and other components optional by splitting them as gems.

Rails 4.1 was released on April 8, 2014, introducing Spring, Variants, Enums, Mailer previews, and secrets.yml.

Rails 4.2 was released on December 19, 2014, introducing Active Job, asynchronous emails, Adequate Record, Web Console, and foreign keys.

Chapter 3

Features

3.1 Users In this web application, there are two users. 1) Normal user 2) Admin

Normal user:

- An user need to register befor login to the system.
- After successful registration he/she can follow other user.
- He/she can also unfollow the user.
- He/she can post any tweet
- He/she can delete own tweet
- He/she can post any photo.
- He/she can edit own profiles
- H

e/
sh
e
ca
n
vi
sit
ot
he
r
us
er
s
pr
of
ile
.

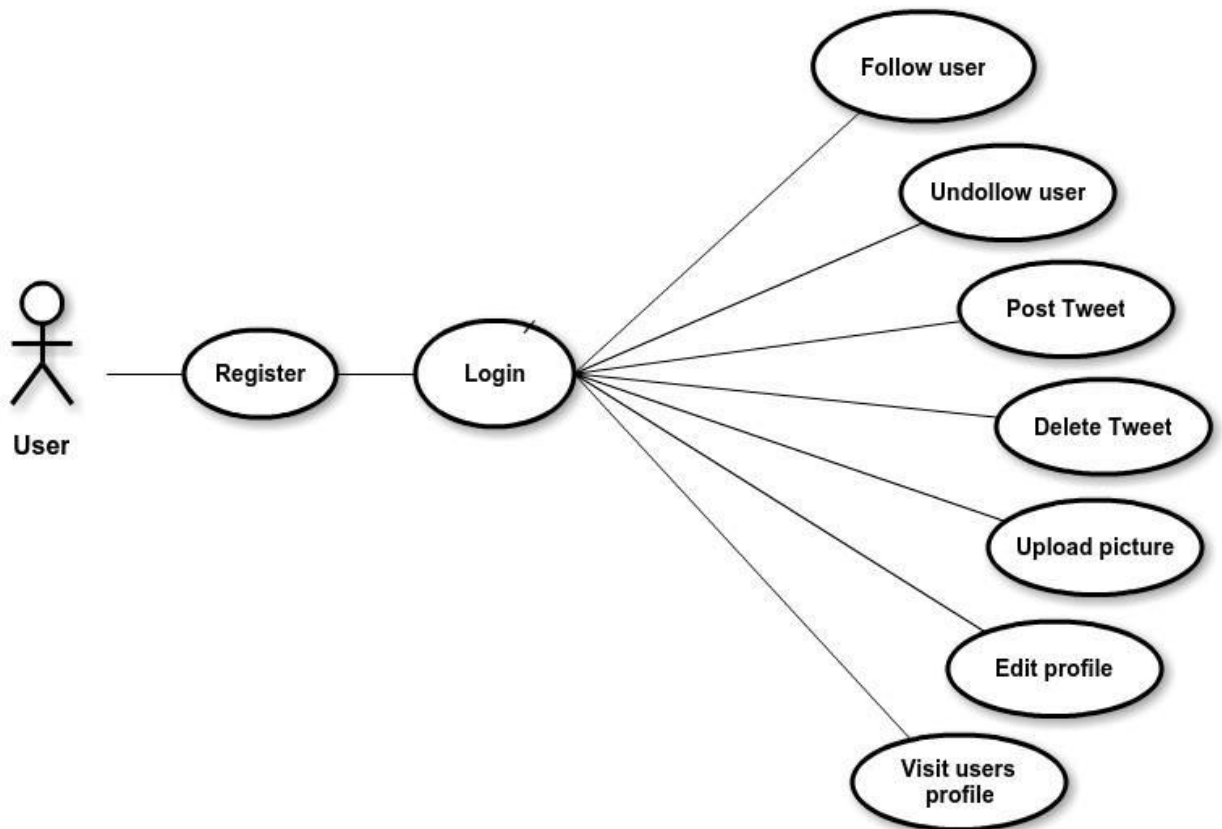


Figure 1: Use case Diagram for user

Admin:

- An admin does not need to register before login to the system.
- An admin can delete any user.
- After successful registration he/she can follow other user.
- He/she can also unfollow the user.
- He/she can post any tweet
- He/she can delete own tweet
- He/she can post any photo.
- He/she can edit own profiles
- He/she can visit other users profile.

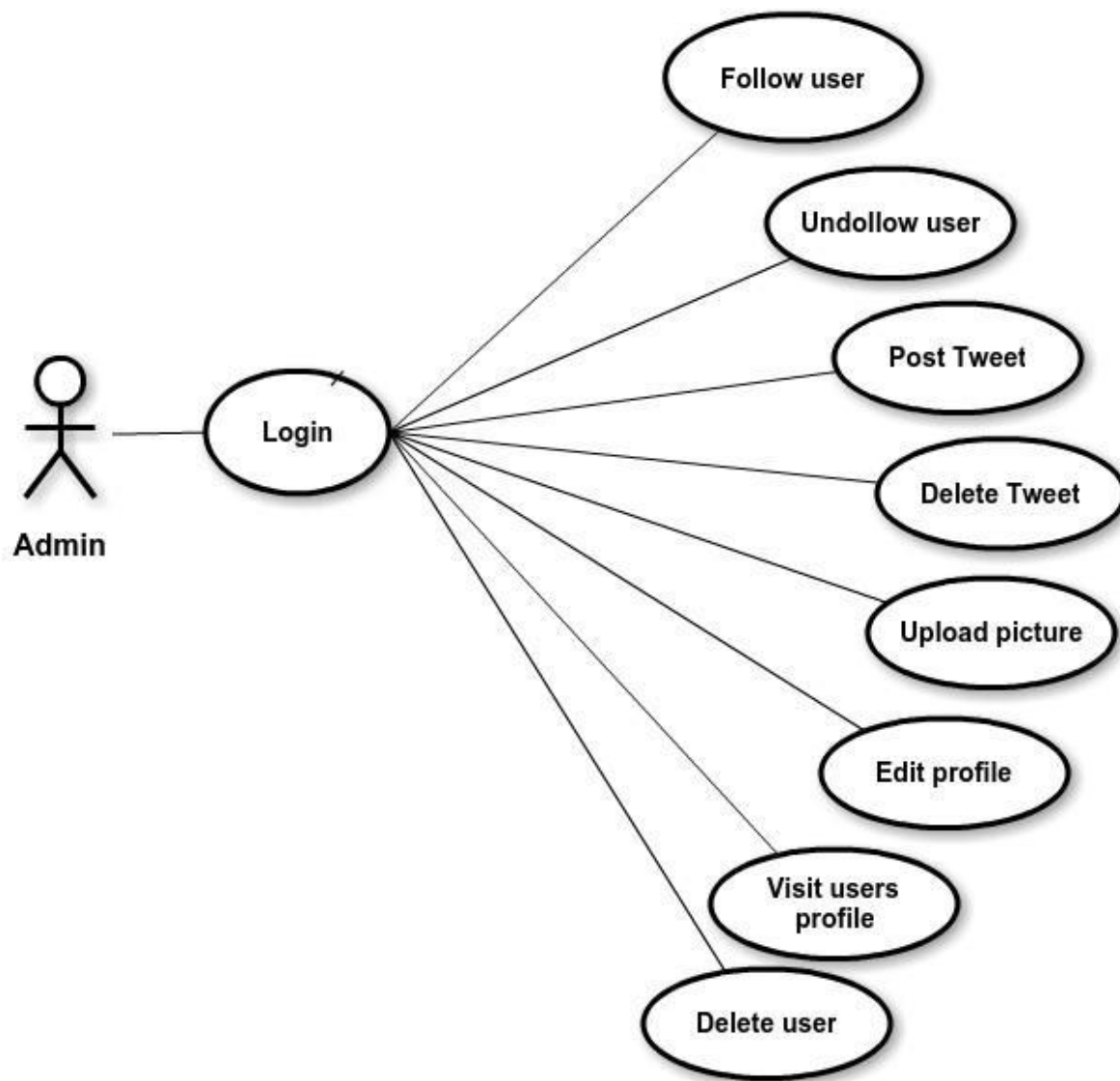


Figure 2: Use case diagram for admin

Starting page: My starting page of my application has six navigation link.

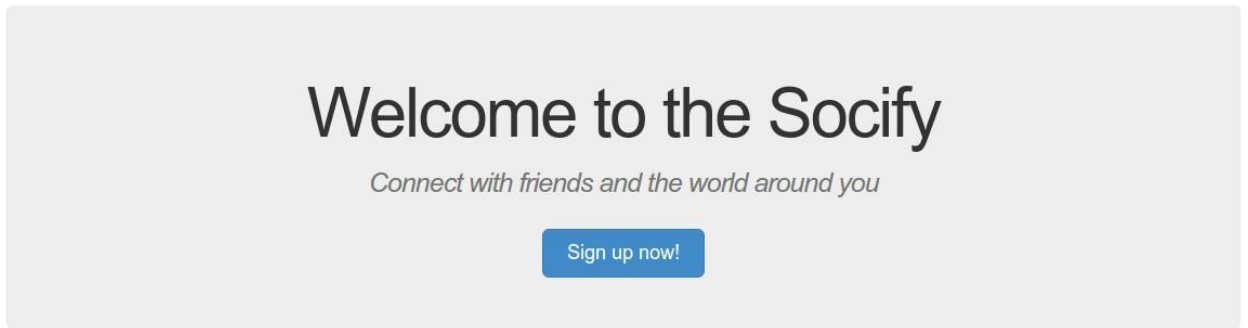
- Sign up now!
- Home
- Log in
- Socify
- About
- Contact

Figure:

Home

Home Log in

3
Startin
g page
of
Socify
app



3.2 Login System: To login Socify app, we have to login with correct credentials. That means it needs both, an correct email and correct password corresponding to the email id. Both login field must be filled and email address has to be a valid email otherwise it will show errors with the explanation about the error.

Log in

Email

Password ([forgot password](#))

Remember me on this computer

New user? [Sign up now!](#)

Figure

4: Login page of Socify app

Invalid email/password combination

Figure
5:
Login
validati
on of
Socify
App

Log in

Email
2www@gmail.com

Password (forgot password)
.....

Remember me on this computer

Log in

New user? Sign up now!

Successful Login: After successful login, it will go through the user profile page. In user profile page, user profile will show in the upper left side. Down the profile picture it show the following and followers numbers.


Figure
5:
Users
profile
page

Socify Home Users Account

Junan Chakma
48 following 39 followers

Tweets (52)

Junan Chakma
Good Night Everyone.
Posted about 1 hour ago. [delete](#)

Junan Chakma
Having dinner

Posted about 1 hour ago. [delete](#)

Junan Chakma
Consequatur minus id quis sequi occaecati autem.
Posted about 4 hours ago. [delete](#)

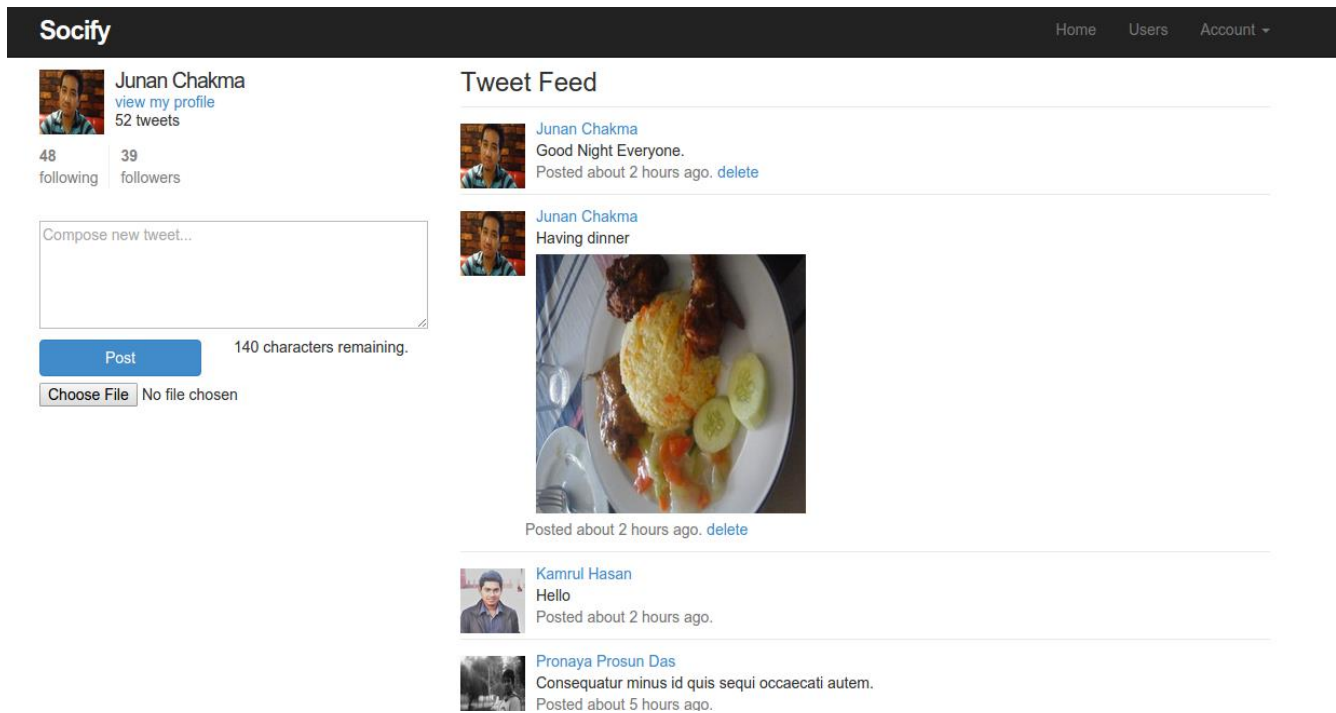
Junan Chakma
Culpa minima fugiat consequuntur possimus odio laudantium ut incidunt quidem.
Posted about 4 hours ago. [delete](#)

In the right side it shows all the

tweets of the users he/she posted. There is a delete button, user can delete any post by clicking it. It will also show picture the user posted and shows the time when the post posted. From here, I can go to the home page by clicking to the Home navigation link.

Figure 6: Home page

3.3



Home Page: Every website has a home page and Socify has so. In this Home page, there is user's profile in the upper left side and there is a text area box for writing tweets down to this profile. This text area box only allow maximal 140 and minimum 1 characters. If I put more than 140 characters in it, then post button will be disabled automatically. There is a helper text bellow the text area box which dynamically shows how many characters are remaining. This text will be red color

when I put more than 140 character on it. Here we can post any photo by uploading

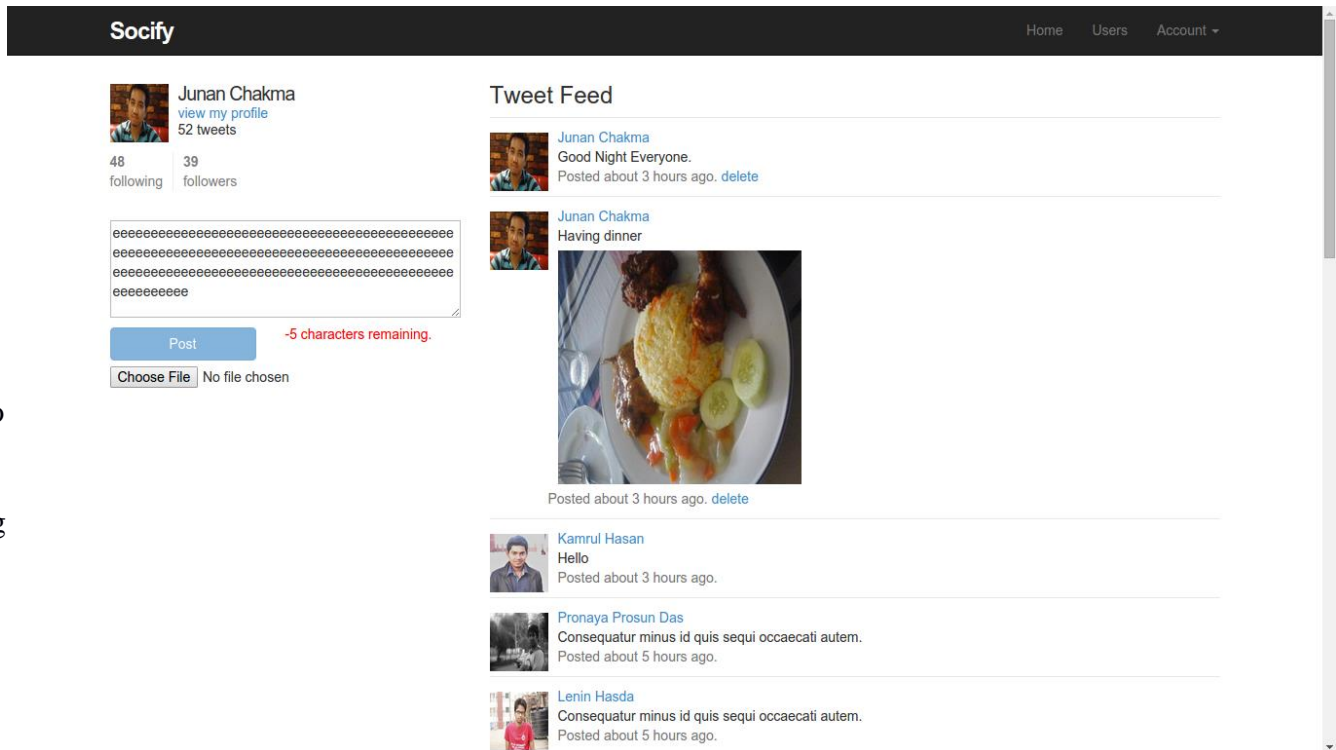


Figure 6: Textarea form validation

In the right side, it will show the tweets of following users and user himself tweets. We can go to user navigation link to see all the successfully registered user. If we logged in by admin login credentials, we can delete any users. If we delete any user by using admin power, all of the tweets of the deleted user will be delete automatically.

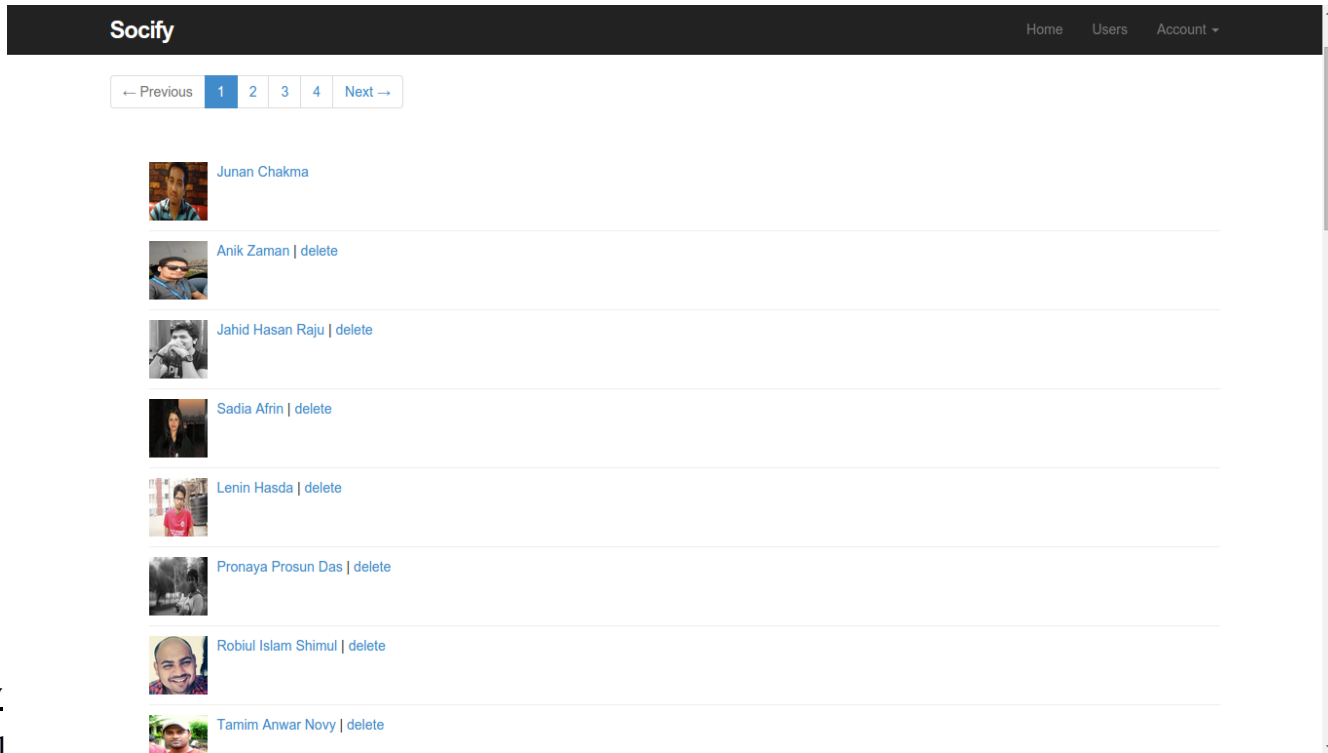


Figure 7 : All users page

3.4

Following and Unfollowing:

In all

users page, I can see many users in the list. I can go their profile page by clicking on their profile picture or name. Suppose I want to go lenin hasda's profile. I can see his profile after clicking on his profile

Figure 8:
Lenin Hasda Profile

In Lenin Hasda profile, I see his picture,

name, tweets. I also see his following and followers numbers. Here he followed one person and being followed by one person. Interestingly there is a unfollow button in his profile, this is because I already followed him. That means his one followers is me. I can unfollow him by clicking unfolow button. See the figure 9 where there is a follow button and Now he has zero followers because I ve unfollwed him.

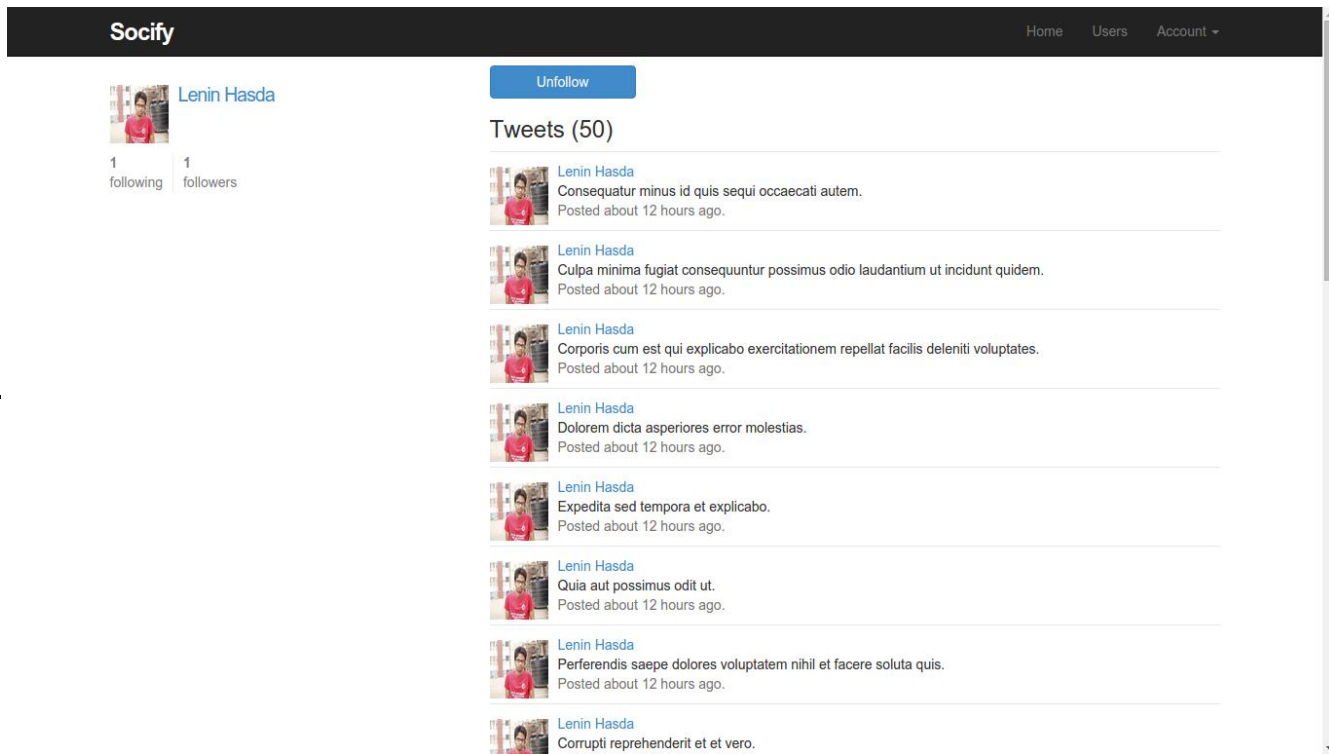
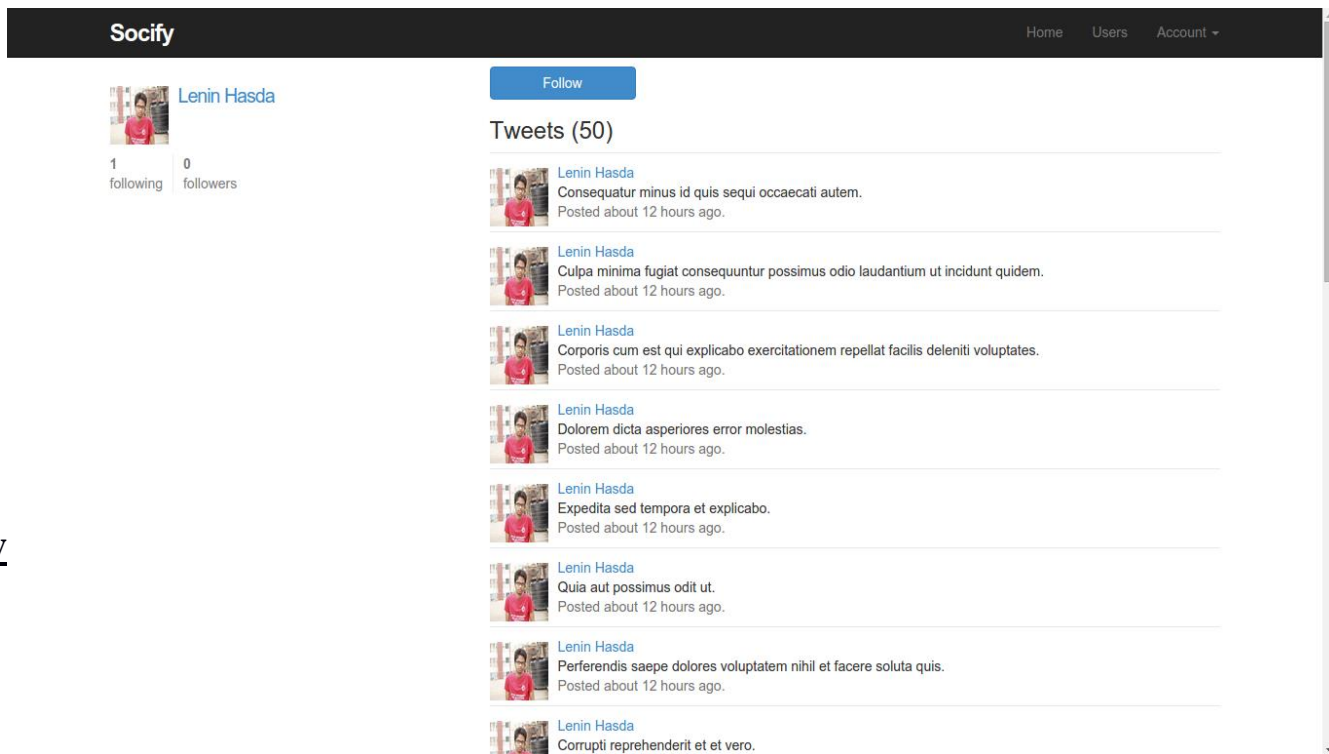


Figure 9:
Lenin hasda profile after hitting unfollow button



3.5 Update profile: I can update my own profile. To do this I have to go to account drop down box and have to clicking in setting link. Then I will get a editing page where I can change my name, email address, password and profile picture.

Fig
ure
10:
Upd
ate
prof
ile

3.6
Use
r
Sig
n
up:

Ever
y
user
has

to register before login to the site. To sign up, a user need to put his/her name, valid email address and minimum 6 charactered password. Every field must be filled currently otherwise it will show some errors messages back to the user and the sign up for can't be empty. After putting these things currently, he/she need to click the create button to successfully sign up.

Update your profile

Name

Email

Password

Confirmation

Upload Picture
 No file chosen

Figure

Socify

[Home](#) [Log in](#)

11: Sign up Form

Sign up

Name

Email

Password

Confirmation

[Create my account](#)

The Socify App by Junan Chakma

[About](#) [Contact](#)

Figure

12: Sign up form validati on

Password

Confirmation

[Create my account](#)

The Socify App by Junan Chakma

[About](#) [Contact](#)

3.7

Email

Verifica tion:

Socify app has email verification features. Without email verification no one can log in into this system. This is a very necessary feature for good security. If this feature were not exist, every user can register with fake or others email address. In noways most website has email verification feature. In Socify app, after successfully sign up we get a message “Please check your email to activate your account.”

Please check your email to activate your account.

Figure
13:
Email
verifica
tion
messag
e

Welcome to the Socify

Connect with friends and the world around you

Sign up now!

Account activation

Inbox x



socify.junan@gmail.com
to me

6:52 AM (31 minutes ago)



Socify App

Hi John Smith,

Welcome to the Socify App! Click on the link below to activate your account:

[Activate](#)



Click here to [Reply](#) or [Forward](#)

Figure
14:
Verifica
tion
messag
e link in
email

Socify

Home Users Account

Account activated!



John Smith

0 following | 0 followers

The Socify App by Junan Chakma

About Contact

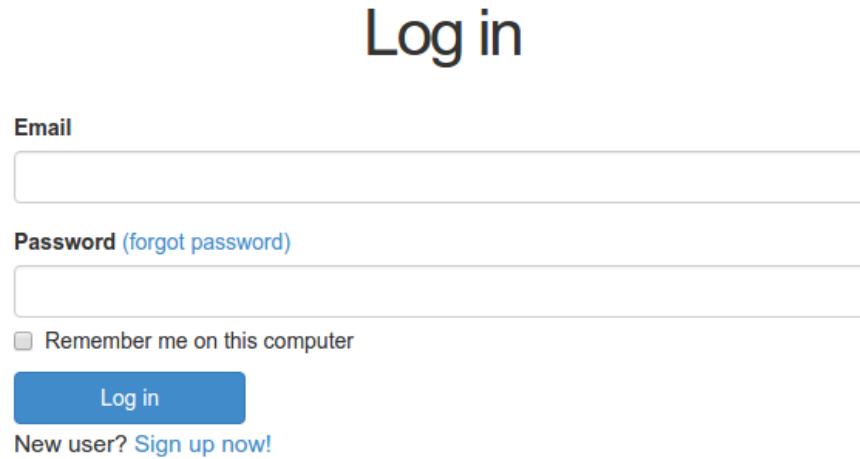
Figure
15:
After
clicking
verifica
tion

link

After successfully sign up, we will get a email message with verification link. After clicking this link we see that our account is activated and logged in.

3.8 Forgot password/reset password: We may forgot our password and we need to recover it. But how we do it? There is a feature called password resetting. In Socify app there is a link named “forgot password”.

Figure
16:
Forgot
passwo
rd



Log in

Email

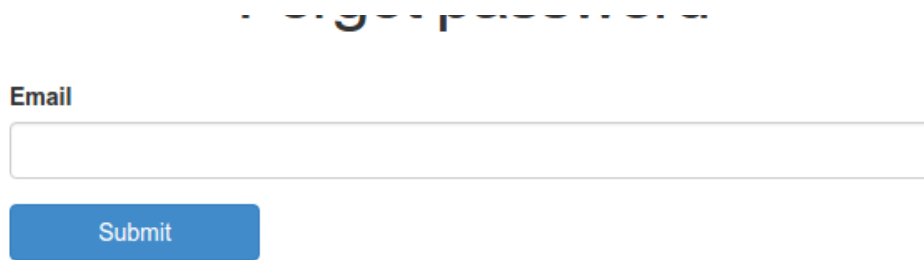
Password (forgot password)

Remember me on this computer

Log in

New user? [Sign up now!](#)

Figure
17:
Forgot
passwo
rd page



Forgot password

Email

Submit

I have to put my email address to this form and click submit button. Then I will get a page that saying “Email sent with password reset instructions”. After that I have to check my email address. In my email address I will see a email has come with password resetting link. After clicking that link I will get a form page for password resetting and I can change my password.

Figure
17:
Passwo
rd
resettin
g
messag
e

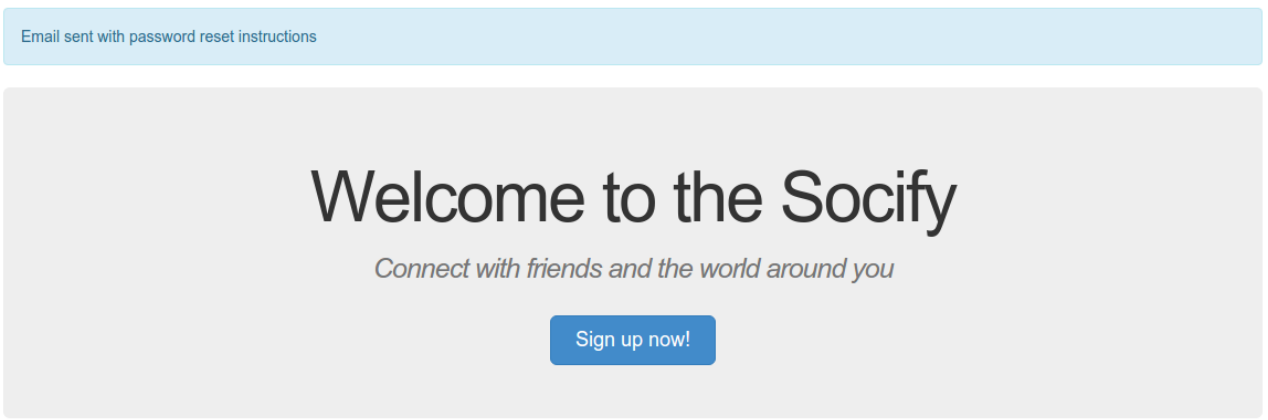


Figure
18:
Passwor
d reset
Email

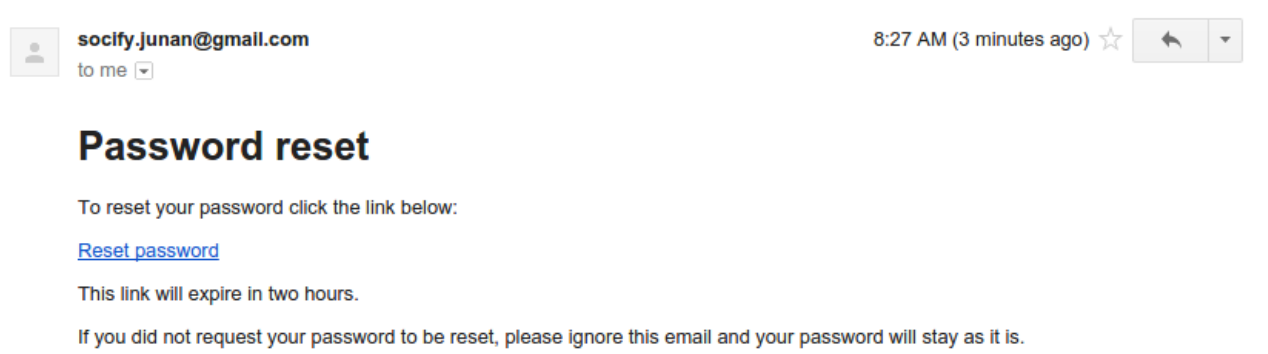


Figure
18
Passwo
rd
resettin
g form

Reset password

Password

Confirmation

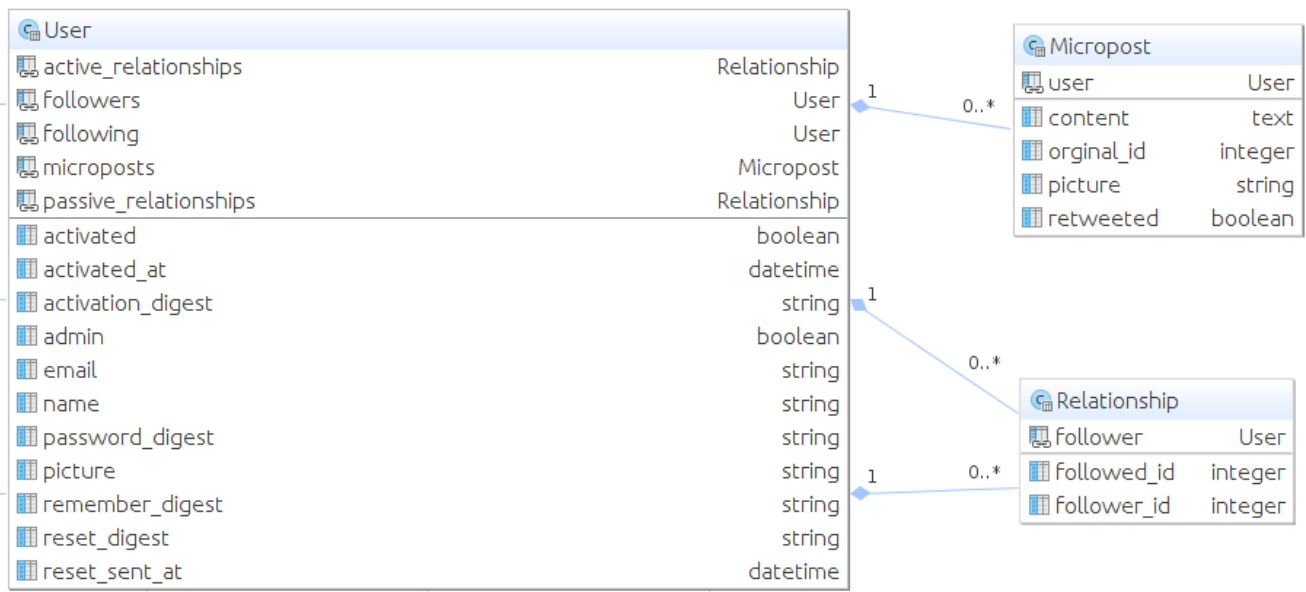
Update password

Chapter 4

Design and Implementation

4.1 Database Model: In my project database, I have three models(table) one for user (Users), two for tweets (Microposts), three for keep tracking following/followrs(Relationships). In users table, there are mainly 14 columns , in microposts table there are 8 columns, in relationships table there are 5 columns. Users table linked others two tables using a foreign key and I can easily access others table information from user table.

Figure 19:
Model Dependency Diagram



4.2 User table validation: In User table, email, name, password must present. I validate email with standard email pattern by using regular expression. Maximum length of user name is 50 characters. Password length will be minimum 6 characters. If there is a uppercase email,

the email will be saved automatically in lowercase in database.

Figure 19: User model

```

1 class User < ActiveRecord::Base
2
3   attr_accessor :remember_token, :activation_token, :reset_token
4   before_save :downcase_email
5   before_create :create_activation_digest
6
7   mount_uploader :picture, PictureUploader
8
9   has_many :microposts, dependent: :destroy
10  has_many :active_relationships, class_name: "Relationship",
11      foreign_key: "follower_id", dependent: :destroy
12  has_many :following, through: :active_relationships, source: :followed
13
14  has_many :passive_relationships, class_name: "Relationship",
15      foreign_key: "followed_id",
16      dependent: :destroy
17  has_many :following, through: :active_relationships, source: :followed
18  has_many :followers, through: :passive_relationships, source: :follower
19
20
21  validates :name, presence: true, length: {maximum: 50}
22  VALID_EMAIL_REGEX = /\A[\w+\-\.]+\@[a-z\d\-\-]+\([\.][a-z\d\-\-]+\)*\.[a-z]+\z/i
23  validates :email, presence: true, length: {maximum: 255},
24      format: {with: VALID_EMAIL_REGEX}, uniqueness: {case_sensitive: false}
25
26  has_secure_password
27  validates :password, presence: true, length: {minimum: 6}, allow_nil: true
28
29  # Returns the hash digest of the given string.
30  def self.digest(string)
31    cost = ActiveSupport::SecurePassword.min_cost ? BCrypt::Engine::MIN_COST :
32          BCrypt::Engine.cost
33    BCrypt::Password.create(string, cost: cost)
34  end
35
36  # Returns a random token.
37  def User.new_token

```

exist before post any tweet. The tweet maximum length is 140 characters. If I put 140+ characters the in the tweet form the submit button will be disabled.

Figure 20: Micropost Model

```

1 class Micropost < ActiveRecord::Base
2   belongs_to :user
3   default_scope -> { order(created_at: :desc) }
4   mount_uploader :picture, PictureUploader
5   validates :user_id, presence: true
6   validates :content, presence: true, length: { maximum: 140 }
7 end
8

```

4.4 Relationship model: This model(table) used for keep tracking following and followers and it is associated to user table.

```

class Relationship < ActiveRecord::Base
  belongs_to :follower, class_name: "User"
  belongs_to :followed, class_name: "User"
  validates :follower_id, presence: true
  validates :followed_id, presence: true
end

```

Figure 21: relationship model

4.5 Controller: Controller do the work of parsing user requests, data submissions, cookies, sessions and the “browser stuff”. A controller is the link between a user and the system. It provides the user with input by arranging for relevant views to present themselves in appropriate places on the screen. It provides means for user output by presenting the user with menus or other means of giving commands and data. The controller receives such user output, translates it into the appropriate messages and pass these messages on to one or more of the views.

4.6 User controller: My user controller is responsible for create new user, delete user, update user information, ensure correct user, ensure user is admin, show following/followers users and sharing status.

```
users_controller.rb x
1 class UsersController < ApplicationController
2   before_action :logged_in_user, only: [:index, :edit, :update, :destroy, :following, :followers]
3   before_action :correct_user, only: [:edit, :update]
4   before_action :admin_user, only: :destroy
5
6   def index
7     @users=User.paginate(page: params[:page])
8   end
9
10  def new
11    @user=User.new
12  end
13
14  def create
15    @user = User.new(user_params)
16    if @user.save
17      @user.send_activation_email
18      flash[:info] = "Please check your email to activate your account."
19      redirect_to root_url
20    else
21      render 'new'
22    end
23  end
24
25  def edit
26    @user = User.find(params[:id])
27  end
28
29  def update
30    @user = User.find(params[:id])
31    if @user.update_attributes(user_params)
32      flash[:success] = "Profile updated"
33      redirect_to @user
34    else
35      render 'edit'
36    end
37  end
38 end
```

Figure 22: User controller

4.7 Micropost controller

r: Micropost controller is responsible for creating status and destroy it. It also ensure the correct user and it

```
accept 1 class MicropostsController < ApplicationController
content 2   before_action :logged_in_user, only: [:create, :destroy]
and     3   before_action :correct_user, only: :destroy
picture 4
attribute. 5 def create
6   @micropost = current_user.microposts.build(micropost_params)
7   if @micropost.save
8     flash[:success] = "Tweet created!"
9     redirect_to root_url
10  else
11    @feed_items = []
12    render 'static_pages/home'
13  end
14 end
15
16 def destroy
17   @micropost.destroy
18   flash[:success] = "Tweet deleted"
19   redirect_to request.referrer || root_url
20 end
end
```

Figure

23:

Micropost

controller

code

4.8 Relationship controller: Relationship controller responsible for keep tracking for following and followers users.

```
class RelationshipsController < ApplicationController
  before_action :logged_in_user

  def create
    user = User.find(params[:followed_id])
    current_user.follow(user)
    redirect_to user
  end

  def destroy
    user = Relationship.find(params[:id]).followed
    current_user.unfollow(user)
    redirect_to user
  end
end
```

Figure 24 Relationship Controller

4.9 Session controller: Session controller is responsible for authenticating login user and it used permanent cookies for permanent cookies.

Figure 24:
Session Controller

**C
h**

```
class SessionsController < ApplicationController
  def new
  end
  def create
    user = User.find_by(email: params[:session][:email].downcase)
    if user && user.authenticate(params[:session][:password])
      if user.activated?
        log_in user
        params[:session][:remember_me] == '1' ? remember(user) : forget(user)
        redirect_back_or root_path
      else
        message = "Account not activated. "
        message += "Check your email for the activation link."
        flash[:warning] = message
        redirect_to root_url
      end
    else
      flash.now[:danger] = 'Invalid email/password combination'
      render 'new'
    end
  end
  def destroy
    log_out if logged_in?
    redirect_to root_path
  end
end
```

apter 5

Conclusion, Limitation and Future Work

Conclusion: Now days social networking web site are going very very popular. Social networking site changed our life style, our perception, communication system. Web-based social networking services make it possible to connect people who share interests and activities across political, economic, and geographic borders. In my Socify app, anyone can easily use it because it is very simple to use. After registration, a user can follow any user he/she like and gets their updates in his news feed.

Limitation: There are some limitation in my application. There are given bellow

- There is no real time chatting system.
- There is no option for comment
- There is no messaging option.

In my future work on this project, I will work on this area and hopefully I can implement these feature very efficiently.

References:

1. <http://stackoverflow.com/>
2. <https://gorails.com/>
3. <http://railscasts.com/>
4. <https://rubymonk.com/>
5. <http://www.sitepoint.com/>
6. <https://www.wikipedia.org/>
7. <http://www.w3schools.com/>
8. <http://www.youtube.com/>

APPENDIX

User controller:

```
class UsersController < ApplicationController
  before_action :logged_in_user, only: [:index, :edit, :update, :destroy,
  :following, :followers]
  before_action :correct_user, only: [:edit, :update]
  before_action :admin_user, only: :destroy
  def index
    @users=User.paginate(page: params[:page])
  end
  def new
    @user=User.new
  end
  def create
    @user = User.new(user_params)
    if @user.save
      @user.send_activation_email
      flash[:info] = "Please check your email to activate your account."
      redirect_to root_url
    else
      render 'new'
    end
  end
end
```

```

end
def edit
  @user = User.find(params[:id])
end
def update
  @user = User.find(params[:id])
  if @user.update_attributes(user_params)
    flash[:success] = "Profile updated"
    redirect_to @user
  else
    render 'edit'
  end
end
def show
  @user = User.find(params[:id])
  @microposts = @user.microposts.paginate(:page => params[:page], :per_page =>
20)
end
def destroy
  User.find(params[:id]).destroy
  flash[:success] = "User deleted"
  redirect_to users_url
end
def following
  @title = "Following"
  @user = User.find(params[:id])
  @users = @user.following.paginate(page: params[:page])
  render 'show_follow'
end
def followers
  @title = "Followers"
  @user = User.find(params[:id])
  @users = @user.followers.paginate(page: params[:page])
  render 'show_follow'
end
def retweet
  @micropost = current_user.microposts.create(content: params[:content],
retweeted: params[:retweeted], original_id: params[:original_id])
  flash[:success] = "Retweeted"
  redirect_to root_path
end
private
def user_params
  params.require(:user).permit(:name, :email, :password,
:password_confirmation, :picture)
end
end
# Confirms the correct user.
def correct_user
  @user = User.find(params[:id])
  redirect_to(root_url) unless current_user?(@user)
end
end
# Confirms an admin user.
def admin_user
  redirect_to(root_url) unless current_user.admin?
end
end

```

end

Micropost Controller

```
class MicropostsController < ApplicationController
  before_action :logged_in_user, only: [:create, :destroy]
  before_action :correct_user, only: :destroy
  def create
    @micropost = current_user.microposts.build(micropost_params)
    if @micropost.save
      flash[:success] = "Tweet created!"
      redirect_to root_url
    else
      @feed_items = []
      render 'static_pages/home'
    end
  end
  def destroy
    @micropost.destroy
    flash[:success] = "Tweet deleted"
    redirect_to request.referrer || root_url
  end
  private
  def micropost_params
    params.require(:micropost).permit(:content, :retweeted, :original_id, :picture)
  end
  def correct_user
    @micropost = current_user.microposts.find_by(id: params[:id])
    redirect_to root_url if @micropost.nil?
  end
end
```

Session Controller:

```
class SessionsController < ApplicationController
  def new
  end
  def create
    user = User.find_by(email: params[:session][:email].downcase)
    if user && user.authenticate(params[:session][:password])
      if user.activated?
        log_in user
        params[:session][:remember_me] == '1' ? remember(user) : forget(user)
        redirect_back_or root_path
      else
        message = "Account not activated. "
        message += "Check your email for the activation link."
        flash[:warning] = message
        redirect_to root_url
      end
    else
      flash.now[:danger] = 'Invalid email/password combination'
      render 'new'
    end
  end
end
```

```

end
def destroy
  log_out if logged_in?
  redirect_to root_path
end
end

```

password reset Controller:

```

class PasswordResetsController < ApplicationController
  before_action :get_user, only: [:edit, :update]
  before_action :valid_user, only: [:edit, :update]
  before_action :check_expiration, only: [:edit, :update]
  def new
  end
  def create
    @user = User.find_by(email: params[:password_reset][:email].downcase)
    if @user
      @user.create_reset_digest
      @user.send_password_reset_email
      flash[:info] = "Email sent with password reset instructions"
      redirect_to root_url
    else
      flash.now[:danger] = "Email address not found"
      render 'new'
    end
  end
  def edit
  end
  def update
    if params[:user][:password].empty?
      @user.errors.add(:password, "can't be empty")
      render 'edit'
    elsif @user.update_attributes(user_params)
      log_in @user
      flash[:success] = "Password has been reset."
      redirect_to @user
    else
      render 'edit'
    end
  end
  private
  def user_params
    params.require(:user).permit(:password, :password_confirmation)
  end
  def get_user
    @user = User.find_by(email: params[:email])
  end
  # Confirms a valid user.
  def valid_user
    unless (@user && @user.activated? &&
      @user.authenticated?(:reset, params[:id]))
      redirect_to root_url
    end
  end
end

```

```

# Checks expiration of reset token.
def check_expiration
  if @user.password_reset_expired?
    flash[:danger] = "Password reset has expired."
    redirect_to new_password_reset_url
  end
end
end

```

Account activation Controller:

```

class AccountActivationsController < ApplicationController

  def edit
    user = User.find_by(email: params[:email])
    if user && !user.activated? && user.authenticated?(:activation, params[:id])
      user.activate
      log_in user
      flash[:success] = "Account activated!"
      redirect_to user
    else
      flash[:danger] = "Invalid activation link"
      redirect_to root_url
    end
  end
end

```

Relationship controller:

```

class RelationshipsController < ApplicationController

```

```

  before_action :logged_in_user
  def create
    user = User.find(params[:followed_id])
    current_user.follow(user)
    redirect_to user
  end
  def destroy
    user = Relationship.find(params[:id]).followed
    current_user.unfollow(user)
    redirect_to user
  end
end

```

Session helper:

```

module SessionsHelper

```

```

  # Logs in the given user.
  def log_in(user)
    session[:user_id] = user.id
  end
end

```

```

end
# Remembers a user in a persistent session.
def remember(user)
  user.remember
  cookies.permanent.signed[:user_id] = user.id
  cookies.permanent[:remember_token] = user.remember_token
end
# Returns the user corresponding to the remember token cookie.
def current_user
  if (user_id = session[:user_id])
    @current_user ||= User.find_by(id: user_id)
  elsif (user_id = cookies.signed[:user_id])
    user = User.find_by(id: user_id)
    if user && user.authenticated?(:remember, cookies[:remember_token])
      log_in user
      @current_user = user
    end
  end
end
# Returns true if the given user is the current user.
def current_user?(user)
  user == current_user
end
# Returns true if the user is logged in, false otherwise.
def logged_in?
  !current_user.nil?
end
# Forgets a persistent session.
def forget(user)
  user.forget
  cookies.delete(:user_id)
  cookies.delete(:remember_token)
end
# Logs out the current user.
def log_out
  forget(current_user)
  session.delete(:user_id)
  @current_user = nil
end
# Redirects to stored location (or to the default).
def redirect_back_or(default)
  redirect_to(session[:forwarding_url] || default)
  session.delete(:forwarding_url)
end
# Stores the URL trying to be accessed.
def store_location
  session[:forwarding_url] = request.url if request.get?
end
end

```

Model

User Model:

```
class User < ActiveRecord::Base
  attr_accessor :remember_token, :activation_token, :reset_token
  before_save :downcase_email
  before_create :create_activation_digest
  mount_uploader :picture, PictureUploader
  has_many :microposts, dependent: :destroy
  has_many :active_relationships, class_name: "Relationship",
    foreign_key: "follower_id", dependent: :destroy
  has_many :following, through: :active_relationships, source: :followed
  has_many :passive_relationships, class_name: "Relationship",
    foreign_key: "followed_id",
    dependent: :destroy
  has_many :following, through: :active_relationships, source: :followed
  has_many :followers, through: :passive_relationships, source: :follower
  validates :name, presence: true, length: {maximum: 50}
  VALID_EMAIL_REGEX = /\A[\w+\-\.]+\@[a-z\d\-\-]+\([\. [a-z\d\-\-]+\)*\.[a-z]+\z/i
  validates :email, presence: true, length: {maximum: 255},
    format: {with: VALID_EMAIL_REGEX}, uniqueness: {case_sensitive: false}
  has_secure_password
  validates :password, presence: true, length: {minimum: 6}, allow_nil: true
  # Returns the hash digest of the given string.
  def self.digest(string)
    cost = ActiveModel::SecurePassword.min_cost ? BCrypt::Engine::MIN_COST :
      BCrypt::Engine.cost
    BCrypt::Password.create(string, cost: cost)
  end
  # Returns a random token.
  def User.new_token
    SecureRandom.urlsafe_base64
  end
  # Remembers a user in the database for use in persistent sessions.
  def remember
    self.remember_token = User.new_token
    update_attribute(:remember_digest, User.digest(remember_token))
  end
  # Returns true if the given token matches the digest.
  def authenticated?(attribute, token)
    digest = send("#{attribute}_digest")
    return false if digest.nil?
    BCrypt::Password.new(digest).is_password?(token)
  end
  # Forgets a user.
  def forget
    update_attribute(:remember_digest, nil)
  end
  # Activates an account.
  def activate
    update_attribute(:activated, true)
    update_attribute(:activated_at, Time.zone.now)
  end
  # Sends activation email.
  def send_activation_email
    UserMailer.account_activation(self).deliver_now
  end
end
```



```

end
# Sets the password reset attributes.
def create_reset_digest
  self.reset_token = User.new_token
  update_attribute(:reset_digest, User.digest(reset_token))
  update_attribute(:reset_sent_at, Time.zone.now)
end
# Sends password reset email.
def send_password_reset_email
  UserMailer.password_reset(self).deliver_now
end
def password_reset_expired?
  reset_sent_at < 2.hours.ago
end
def feed
  Micropost.where("user_id IN (?) OR user_id = ?", following_ids, id)
end
# Follow a user.
def follow(other_user)
  active_relationships.create(followed_id: other_user.id)
end
# Unfollow a user.
def unfollow(other_user)
  active_relationships.find_by(followed_id: other_user.id).destroy
end
# Returns true if the current user is following the other user.
def following?(other_user)
  following.include?(other_user)
end
private
# Converts email to all lower-case.
def downcase_email
  self.email = email.downcase
end
# Creates and assigns the activation token and digest.
def create_activation_digest
  self.activation_token = User.new_token
  self.activation_digest = User.digest(activation_token)
end
end
end

```

Micropost model:

```

class Micropost < ActiveRecord::Base
  belongs_to :user
  default_scope -> { order(created_at: :desc) }
  mount_uploader :picture, PictureUploader
  validates :user_id, presence: true
  validates :content, presence: true, length: { maximum: 140 }
end

```

Relationship model:

```

class Relationship < ActiveRecord::Base
  belongs_to :follower, class_name: "User"

```

```

belongs_to :followed, class_name: "User"
validates :follower_id, presence: true
validates :followed_id, presence: true
end

```

Views

application:

```
<!DOCTYPE html>
```

```

<html>
<head>
  <title><%= full_title(yield(:title)) %></title>
  <%= stylesheet_link_tag 'application', media: 'all',
    'data-turbolinks-track' => true %>
  <%= javascript_include_tag 'application', 'data-turbolinks-track' => true %>
  <%= csrf_meta_tags %>
  <%= render 'layouts/shim' %>
</head>
<body>
  <%= render 'layouts/headers' %>
  <div class="container">
    <% flash.each do |message_type, message| %>
      <%= content_tag(:div, message, class: "alert alert-#{message_type}") %>
    <% end %>
    <%= yield %>
    <%= render 'layouts/footer' %>
    <%= debug params if Rails.env.test? %>
  </div>
</body>
</html>

```

header:

```
<header class="navbar navbar-fixed-top navbar-inverse">
```

```

<div class="container">
  <%= link_to "Socify", root_path, id: "logo" %>
  <nav>
    <ul class="nav navbar-nav navbar-right">
      <li><%= link_to "Home", root_path %></li>
      <% if logged_in? %>
        <li><%= link_to "Users", users_path %></li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown">
            Account <b class="caret"></b>
          </a>

```

```

    <ul class="dropdown-menu">
      <li><%= link_to "Profile", current_user %></li>
      <li><%= link_to "Settings", edit_user_path(current_user) %></li>
      <li class="divider"></li>
      <li>
        <%= link_to "Log out", logout_path, method: "delete" %>
      </li>
    </ul>
  </li>
<% else %>
  <li><%= link_to "Log in", login_path %></li>
<% end %>
</ul>
</nav>
</div>
</header>

```

footer:

```

<footer class="footer">
  <small>
    The <%= link_to 'Socify App', root_path %>
    by Junan Chakma
  </small>
</footer>

```

Home:

```

<% if logged_in? %>
  <div class="row">
    <aside class="col-md-4">
      <section class="user_info">
        <%= render 'shared/user_info' %>
      </section>
      <section class="stats">
        <%= render 'shared/stats' %>
      </section>
      <section class="micropost_form">
        <%= render 'shared/micropost_form' %>
      </section>
    </aside>
    <div class="col-md-8">
      <h3>Status Feed</h3>
      <%= render 'shared/feed' %>
    </div>
  </div>
<% else %>
  <div class="center_jumbotron">
    <h1>Welcome to the Socify</h1>
  </div>
</%>

```

```

<h2>
  Connect with friends and the
  world around you
</h2>
<%= link_to "Sign up now!", signup_path, class: "btn btn-lg btn-primary" %>
</div>
<% end %>

```

Micropost:

```

<% current_user=@user || current_user %>

```

```

<li id="micropost-<%= micropost.id %>">
  <% if current_user.picture? %>
    <%= link_to (image_tag((current_user).picture.url, class: "gravatar")),
micropost.user %>
    <span class="user"> <%= link_to micropost.user.name, micropost.user
%></span>
  <% else %>
    <%= link_to (image_tag("default-icon.png", class: "gravatar")),
micropost.user %>
    <span class="user"> <%= link_to micropost.user.name, micropost.user
%></span>
  <% end %>
  <span class="content">
    <%= micropost.content %>
    <%= image_tag micropost.picture.url, class: "img-height" if
micropost.picture? %>
  </span>
  <span class="timestamp">
    Posted <%= time_ago_in_words(micropost.created_at) %> ago.
    <% if current_user?(micropost.user) %>
      <%= link_to "delete", micropost, method: :delete,
        data: {confirm: "You sure?"} %>
    <% end %>
  </span>
</li>

```

pass reset edit:

```

<% provide(:title, 'Reset password') %>
<h1>Reset password</h1>
<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(@user, url: password_reset_path(params[:id])) do |f| %>
      <%= render 'shared/error_messages', object: f.object %>
      <%= hidden_field_tag :email, @user.email %>
      <%= f.label :password %>
      <%= f.password_field :password, class: 'form-control' %>
      <%= f.label :password_confirmation, "Confirmation" %>
    </div>
  </div>
</div>

```

```

    <%= f.password_field :password_confirmation, class: 'form-control' %>
    <%= f.submit "Update password", class: "btn btn-primary" %>
  <% end %>
</div>
</div>

```

password reset new

```

<% provide(:title, "Forgot password") %>

<h1>Forgot password</h1>
<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(:password_reset, url: password_resets_path) do |f| %>
      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>
      <%= f.submit "Submit", class: "btn btn-primary" %>
    <% end %>
  </div>
</div>

```

Session new:

```

<% provide(:title, "Log in") %>

<h1>Log in</h1>
<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(:session, url: login_path) do |f| %>
      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>
      <%= f.label :password %>
      <%= link_to "(forgot password)", new_password_reset_path %>
      <%= f.password_field :password, class: 'form-control' %>
      <%= f.label :remember_me, class: "checkbox inline" do %>
        <%= f.check_box :remember_me %>
        <span>Remember me on this computer</span>
      <% end %>
      <%= f.submit "Log in", class: "btn btn-primary" %>
    <% end %>
  <p>New user? <%= link_to "Sign up now!", signup_path %></p>
</div>
</div>

```

error messages:

```

<% if object.errors.any? %>

  <div id="error_explanation">
    <div class="alert alert-danger">
      The form contains <%= pluralize(object.errors.count, "error") %>.
    </div>
    <ul>

```

```

    <% object.errors.full_messages.each do |msg| %>
      <li><%= msg %></li>
    <% end %>
  </ul>
</div>
<% end %>

```

feed:

```

<% if @feed_items.any? %>
  <ol class="microposts">
    <%= render 'shared/feed_items' %>
  </ol>
  <%= will_paginate @feed_items %>
<% end %>

```

Feed items:

```

<% user=@user||=current_user %>

<% @feed_items.each do |micropost| %>
  <li id="micropost-<%= micropost.id %>">
    <% if micropost.user.picture? %>
      <%= link_to (image_tag(micropost.user.picture.url, class: "gravatar")),
micropost.user %>
      <span class="user"><%= link_to micropost.user.name, micropost.user
%></span>
    <% else %>
      <%= link_to (image_tag("default-icon.png", class: "gravatar")),
user_path(micropost.user) %>
      <span class="user"><%= link_to micropost.user.name, micropost.user
%></span>
    <% end %>
    <% if micropost.retweeted %>
      <span class="content-retweet">
        <% @user=User.find_by(id: micropost.original_id) %>
        <%= link_to (image_tag(@user.picture.url, class: "gravatar")), @user %>
        <!--<span class="retweet">[ Retweeted from <%= @user.name %>]</span>-->
        <%= image_tag(micropost.picture.url, class: "img-height") if
micropost.picture? %>
        <span id="content-top"><%= micropost.content %></span>
      </span>
    <% else %>
      <span class="content">
        <%= micropost.content %>
        <%= image_tag(micropost.picture.url, class: "img-height") if
micropost.picture? %>
      </span>
    <% end %>
  </span>

  <span class="timestamp">

```

```

Posted <%= time_ago_in_words(micropost.created_at) %> ago.
<%= if current_user?(micropost.user) %>
  <%= link_to "delete", micropost, method: :delete,
    data: {confirm: "You sure?"} %>
<%= end %>
<%= form_for(:user, url: share_path) do |f| %>
  <%= hidden_field_tag :content, micropost.content %>
  <%= hidden_field_tag :retweeted, true %>
  <%= hidden_field_tag :original_id, micropost.user.id %>
  <%= if !micropost.retweeted and !micropost.picture.url %>
    <%= f.submit "Share", class: "btn btn-link" %>
  <%= else %>
    <%= f.submit ".", disabled: true ,class: "btn btn-link" %>
  <%= end %>
<%= end %>
</span>
</li>
<%= end %>

```

Micropost form:

```

<%= form_for(@micropost, html: {multipart: true}) do |f| %>

  <%= render 'shared/error_messages', object: f.object %>
  <div class="field">
    <%= f.text_area :content, class: "message", placeholder: "Post a status..."
  %>
  </div>
  <%= f.submit "Post", class: "btn btn-primary", id: "sub" %>
  <span class="countdown"> </span>
  <span class="picture">
    <%= f.file_field :picture, accept: 'image/jpeg,image/gif,image/png' %>
  </span>
<%= end %>
<script type="text/javascript">
  $(document).ready(function ($) {
    updateCountdown();
    $(' .message').change(updateCountdown);
    $(' .message').keyup(updateCountdown);
  });
  function updateCountdown() {
    // 140 is the max message length
    var remaining = 140 - $(' .message').val().length;
    $('#sub').removeAttr('disabled');
    $('span.countdown').removeClass('err_red');
    if(remaining<0){
      $('span.countdown').addClass('err_red');
      $('#sub').attr('disabled', 'disabled');
    }
    $(' .countdown').text(remaining + ' characters remaining. ');
  }
</script>

```

stats:

```
<% @user ||= current_user %>
```

```
<div class="stats">  
  <a href="<%= following_user_path(@user) %>">  
    <strong id="following" class="stat">  
      <%= @user.following.count %>  
    </strong>  
    following  
  </a>  
  <a href="<%= followers_user_path(@user) %>">  
    <strong id="followers" class="stat">  
      <%= @user.followers.count %>  
    </strong>  
    followers  
  </a>  
</div>
```

user info:

```
<% user=@user ||= current_user %>
```

```
<% if user.picture? %>  
  <h1>  
    <%= link_to (image_tag(user.picture.url, class: "gravatar")),  
user_path(user) %>  
    <%= user.name %>  
  </h1>  
<% else %>  
  <h1>  
    <%= link_to (image_tag("default-icon.png", class: "gravatar", alt: 'profile  
photo')), user_path(user) %>  
    <%= @user.name %>  
  </h1>  
<% end %>  
<span><%= link_to "view my profile", current_user %></span>  
<span><%= pluralize(current_user.microposts.count, "Status") %></span>
```

follow:

```
<%= form_for(current_user.active_relationships.build) do |f| %>
```

```
  <div><%= hidden_field_tag :followed_id, @user.id %></div>  
  <%= f.submit "Follow", class: "btn btn-primary" %>  
<% end %>
```

follow form:

```
<% unless current_user?(@user) %>
```

```
  <div id="follow_form">  
    <% if current_user.following?(@user) %>  
      <%= render 'unfollow' %>  
    <% else %>
```



```

    <%= render 'follow' %>
  <% end %>
</div>
<% end %>

```

unfollow:

```

<%= form_for(current_user.active_relationships.find_by(followed_id: @user.id),
  html: { method: :delete }) do |f| %>
  <%= f.submit "Unfollow", class: "btn btn-primary" %>
<% end %>

```

user:

```

<li class="index-img">
  <% if user.picture? %>
    <%= link_to (image_tag(user.picture.url, class: "gravatar")),
user_path(user) %>
    <%= link_to user.name, user %>
  <% else %>
    <%= link_to (image_tag("default-icon.png", class: "gravatar")),
user_path(user) %>
    <%= link_to user.name, user %>
  <% end %>
  <% if current_user.admin? && !current_user?(user) %>
    | <%= link_to "delete", user, method: :delete,
      data: {confirm: "You sure?"} %>
  <% end %>
</li>

```

edit:

```

<% provide(:title, "Edit user") %>
<h1>Update your profile</h1>
<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(@user, html: {multipart: true}) do |f| %>
      <%= render 'shared/error_messages', object: f.object %>
      <%= f.label :name %>
      <%= f.text_field :name, class: 'form-control' %>
      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>
      <%= f.label :password %>
      <%= f.password_field :password, class: 'form-control' %>
      <%= f.label :password_confirmation, "Confirmation" %>
      <%= f.password_field :password_confirmation, class: 'form-control' %>
      <span class="picture">
        <strong class="up_pic">Upload Picture</strong>
        <%= f.file_field :picture %>
      </span>
      <%= f.submit "Save changes", class: "btn btn-primary" %>
    </div>
  </div>
</div>

```

```
<% end %>
</div>
</div>
```

index:

```
<% provide(:title, 'All users') %>
```

```
<h1>All users</h1>
<%= will_paginate %>
<ul class="users">
  <%= render @users %>
</ul>
<%= will_paginate %>
```

new:

```
<% provide(:title, 'Sign up') %>
```

```
<h1>Sign up</h1>
<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(@user) do |f| %>
      <%= render 'shared/error_messages', object: f.object %>
      <%= f.label :name %>
      <%= f.text_field :name, class: 'form-control' %>
      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>
      <%= f.label :password %>
      <%= f.password_field :password, class: 'form-control' %>
      <%= f.label :password_confirmation, "Confirmation" %>
      <%= f.password_field :password_confirmation, class: 'form-control' %>
      <%= f.submit "Create my account", class: "btn btn-primary" %>
    <% end %>
  </div>
</div>
```

show:

```
<% provide(:title, @user.name) %>
```

```
<div class="row">
  <aside class="col-md-4">
    <section class="user_info">
      <% if @user.picture? %>
        <h1>
          <%= link_to (image_tag(@user.picture.url, class: "gravatar")), user_path
%>
          <%= link_to @user.name, @user %>
        </h1>
      <% else %>
        <h1>
          <%= link_to (image_tag("default-icon.png", class: "gravatar")), user_path
%>
```

```

    <%= link_to @user.name, @user %>
  </h1>
<% end %>
</section>
<section class="stats">
  <%= render 'shared/stats' %>
</section>
</aside>
<div class="col-md-8">
  <%= render 'follow_form' if logged_in? %>
  <% if @user.microposts.any? %>
    <h3>Statuses (<%= @user.microposts.count %>)</h3>
    <ol class="microposts">
      <%= render @microposts %>
    </ol>
    <%= will_paginate @microposts %>
  <% end %>
</div>
</div>

```

show follow:

```
<%= provide(:title, @title) %>
```

```

<div class="row">
  <aside class="col-md-4">
    <section class="user_info">
      <% if @user.picture? %>
        <%= link_to (image_tag(@user.picture.url, class: "gravatar")), @user %>
        <h1> <%= link_to @user.name, @user %></h1>
      <% else %>
        <%= link_to (image_tag("default-icon.png", class: "gravatar")), @user %>
        <h1> <%= link_to @user.name, @user %></h1>
      <% end %>
      <span><%= link_to "view my profile", @user %></span>
      <span><b>Tweets:</b> <%= @user.microposts.count %></span>
    </section>
    <section class="stats">
      <%= render 'shared/stats' %>
      <% if @users.any? %>
        <div class="user_avatars">
          <% @users.each do |user| %>
            <% if user.picture? %>
              <%= link_to (image_tag(user.picture.url)), user_path(user) %>
            <% else %>
              <%= link_to (image_tag("default-icon.png")), user_path(user) %>
            <% end %>
          <% end %>
        </div>
      <% end %>
    </section>
  </aside>
  <div class="col-md-8">
    <h3><%= @title %></h3>
    <% if @users.any? %>

```

```
<ul class="users follow">
  <%= render @users %>
</ul>
<%= will_paginate %>
<% end %>
</div>
</div>
```