NIST COLLEGE

Tribhuvan University

Institute of Science and Technology

SOCIAL NETWORKING

A PROJECT report

Submitted to

Department of Computer Science and Information Technology

NIST COLLEGE

In partial fulfillment of the requirement for the Bachelor Degree in Computer Science and Information Technology

Submitted by

Achyut Dahal (3652/070)

Aruna Shrestha (3655/070)

Rajan Prasad Bhattarai (3667/070)

Sabin Nepal (3672/70)

NIST COLLEGE

Tribhuvan University

Supervisor’s Recommendation

I hereby recommend that this project prepared under my supervision by Achyut Dahal, Aruna Shrestha, Rajan Prasad Bhattarai and Sabin Nepal entitled “SOCIAL NETWORKING” in partial fulfillment of the requirements for the degree of B.Sc.in Computer Science and Information Technology be processed for the evaluation.

…………………………

Mr. Indra Chaudhary

Project Supervisor

NIST College

TABLE OF CONTENTS

List of Figures

List of tables

ACKNOWLEDGEMENT

The success of this project would not have been possible without the support and guidelines of many individuals, and we are very thankful to those who helped us during our project. We are highly indebted to NIST College for constant guidance and supervision, as well as for providing all the necessary infrastructures and friendly environment for the successful completion of the project. First and foremost, we like to thank Mr. Chetan Bista, principal, NIST COLLEGE, for his moral support towards completing our project work.

We would also like to thanks for the efforts of Mr. Tara Bahadur Thapa B.Sc. CSIT coordinator for his support, advice and guidance. We would like to thank our project supervisor Mr. Indra Chaudhary who guide us throughout the project and provide us help and support and provide many ideas for the development of our project

ABSTRACT

The documentation on “Social networking for developers” overviews the social network development and potential application and intended community. With the idea in mind to focus for the creation of the common platform for the developers, the project is aimed to fulfill the intended objectives and ideas of uniting the developers under one platform either for sharing the knowledge or getting help from fellow developers in conclusion or both.

Chapter 1: Introduction

1.1 Background

Social networking has been the 21st century's new tool for socializing. Social networking is the grouping of individuals into specific groups, like small rural communities or a neighborhood subdivision. Social network is the mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs, and other connected information/knowledge entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. Social network provides both a visual and a mathematical analysis of human relationships.

People are almost used to with this type of communication system that it is easier for people to find them in social network rather than in real life. Social network is popular in every field not because it is effective means to communicate between people but also to share the knowledge between them. This case can be explained with possibly in the scenario like university, offices, school.

Social networks are major platform nowadays because of the wide availability of the internet since the past decade which explode like a wildfire. Even if we compare the internet user in the past decade, then based on the fact we can say that the user in the internet has been increased by almost 4 times in between 2000 and 2010. To be exact, there were 361Million user around 2000 which increased by 1967 Million in 2010 [1]. This figure even rose to 3835 Million in 2017 [2].

In the proposed system, it is planned to develop the social network for the developers/programmers. The project focuses on mainly two things, sharing the user content through the posts known as blogs, asking and getting answers for the questions.

1.2 Problem Definition

In today's context, there is group divided into two sections that either have online presence or wish to seem totally ghost. It is because people are searching from site to site for the information they need. For eg, One have to use a platform for posting his experience using site A, while for asking for some help he has to go through site B. Sites like medium.com, stackoverflow.com would provide these type of service but often provide specialized services only. This would make people very annoying because many people believe that even though being addicted to social network, it is waste of time to spend on too many social networks. Countries like ours have even worse problem scenario because the developers and the programmers who are city apart are unable to recognize the group of similar interest without actual being connected. With the idea of this in mind the project was born.

1.3 Objectives

The following are the objectives of the proposed social network:

- To make a platform for posting the user experience as blog stories
- To recommend similar user stories based on the interest of the user
- To ask and answer the questions for and by the users.

Chapter 2: Requirement

2.1 Literature Review

Even though there are a lot of social network for developers, some popular platform for developers are: Stack Overflow, medium.com which are quite popular in the foreign countries but in the local context, such system were not in the highlight.

2.1.1 Medium.com:

Medium.com offers the user to post the lengthy post on the user experience. “Interesting ideas that set your mind in motion[4]” is as set by medium.com Currently it offers this type of service to general user. Similar to the medium, it provides user the capacity to share their stories but is focused on the developer’s side.

2.1.2 Stack overflow:

Stack overflow offers the complete forum alike site for the developers and the general people on various topics[4]. User can follow such topics and problems in the field of computer programming.

2.1.3. Forrst:

Forrst is a forum for Web designers and developers where they can share designs and code to get feedback from other designers and developers, or write posts about designtopics and ultimately getting better at their craft. It also offers an About.me-like profile page for designers called Forrst.me. Users can make public posts and share them with non-users. Forrst is an Invite only community [5]

2.2 Tools

Front-End

As it is a web-based application, the front-end of the application is going to implemented using HTML, CSS, JavaScript and Bootstrap.

Back-End

The server side programming language of the application is Ruby and MySQL as database. The Ruby programming language is rich in various libraries that can be used within the application.

Chapter 3 System Analysis

3.1 Software Requirement Specification

Before the development of our system, following requirements are taken into consideration

3.1.1 Functional Requirements

Functional requirements are the main functions which the system is supposed to have. The functional requirements of our system are

a) Messaging

In our system, private messaging feature is available, users can send and receive messages.

b) Blog stories

User can post blogs and also comment in the blog posts.

c) Forum

Users can get help from forum where they can ask question and get answers from other users of the related field.

3.1.2 Non Functional Requirements

a) user friendly

The proposed system is user friendly with easy to use nav and other features

b) responsive

Our system is responsive; it uses Bootstrap, which improves the responsiveness of the web application. Since it uses Bootstrap, it also concur the mobile-first technology which would enhance the functionality of web application in mobile devices. This nature could prove to be extremely beneficial to people living in areas with limited access to computers.

c) reliability

With a reliable source and tools, the outcome would be reliable as well.

d) speed

The response time of the system is fast.

e) performance

The performance of the system is determine by various factor like response time, throughput, resource utilizations etc.

3.2 Feasibility Analysis

Feasibility Analysis is an assessment of the practicality of a proposed project. It provides the degree of viability of a proposed project. A feasibility analysis helps us determine the value of the proposed project, determine whether or not there is a market for the proposed project, determine if the proposed project is financially viable, and eventually, decide whether or not it is worth investing time and money into the proposed project.

In short, a feasibility analysis evaluates the project‟s potential for success. Following Feasibility Analysis was performed prior to working on the project:

3.2.1 Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Does it can be accessible by both large and small devices?

The web app will have responsive design. In the development perspective, the tools and frameworks are easily usable. Website can be hosted on the Rails supporting platform. Because of the use of the bootstrapping technique, any modern browser can be used to access the site.

3.2.2 Operational Feasibility:

The proposed project is beneficial only if it meet operating requirements. Operational feasibility aspects of the project are to be taken as an important issues raised are to test the operational feasibility of the project include

- Will the system be used and work properly if it is being developed and implemented?

The user needs an active internet connection and the browser to have access to the social network. After onwards, user can go the site, follow other developers, browse the topic of their interest and get support.

3.2.3 Economical Feasibility:

The social network is completely free to sign up/login beside the cost of accessing which depends on the user device.

Chapter 4.System Design

4.1 System Architecture

System design is simply the design of systems. It is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. Our system makes use of internet and computer peripherals. This system is supported by any different devices that can access the internet. The system architecture is shown below:

Fig4.1.1 System Diagram

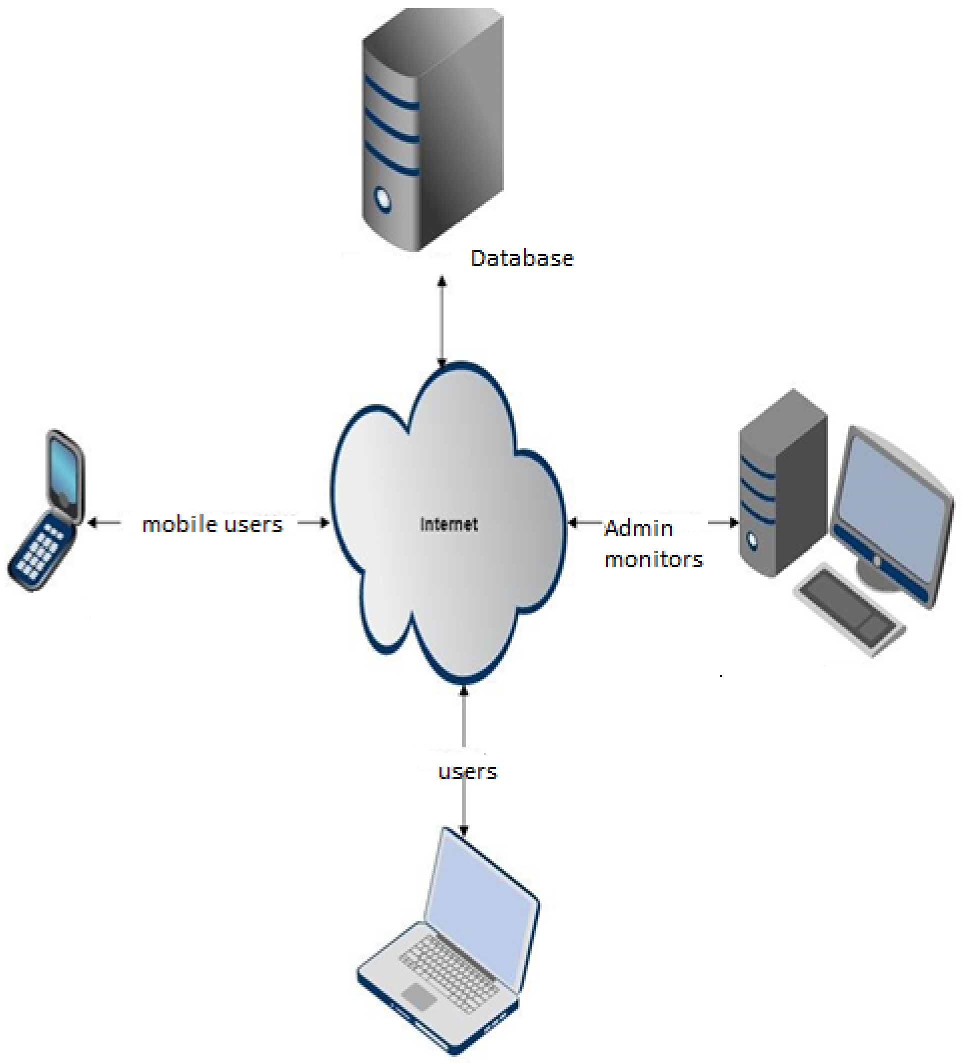


Fig: 4.1.2 Context free diagram

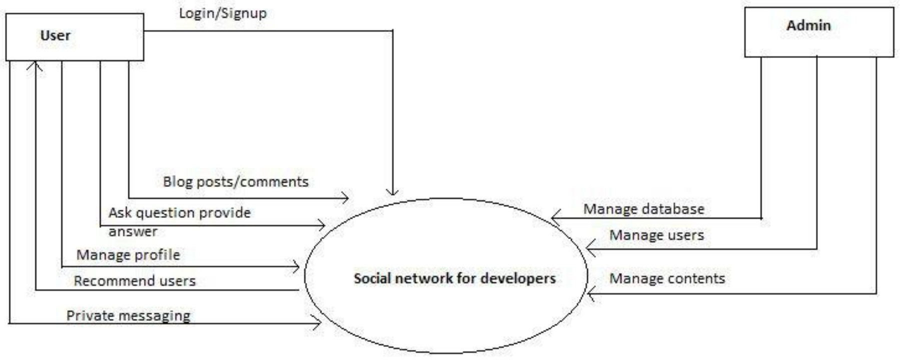


Fig:4.1.3 Use case Diagram

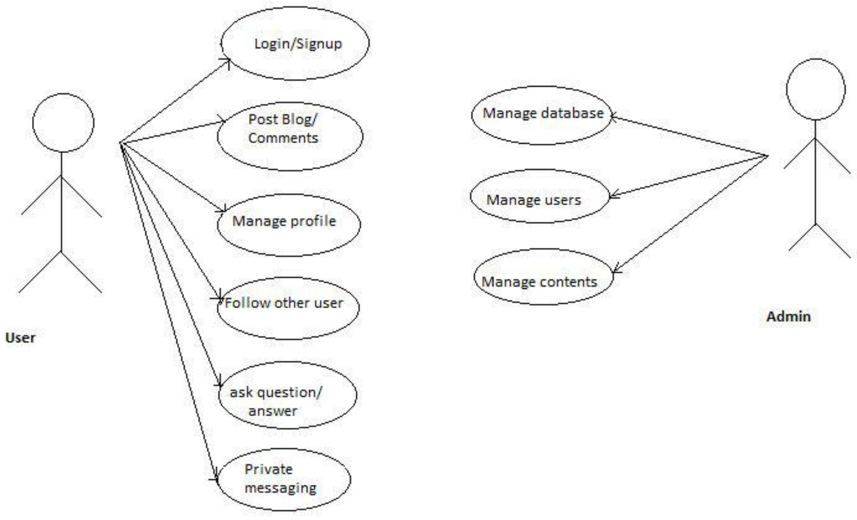


Fig 4.1.4 Activity Diagram

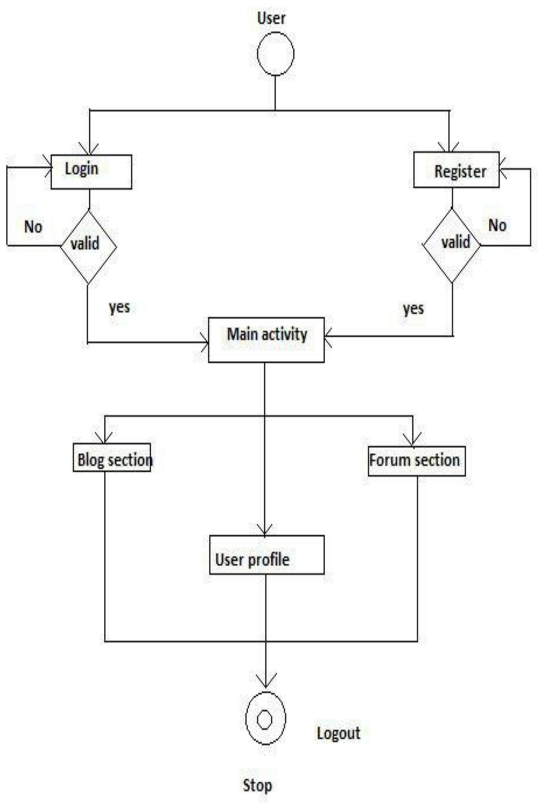
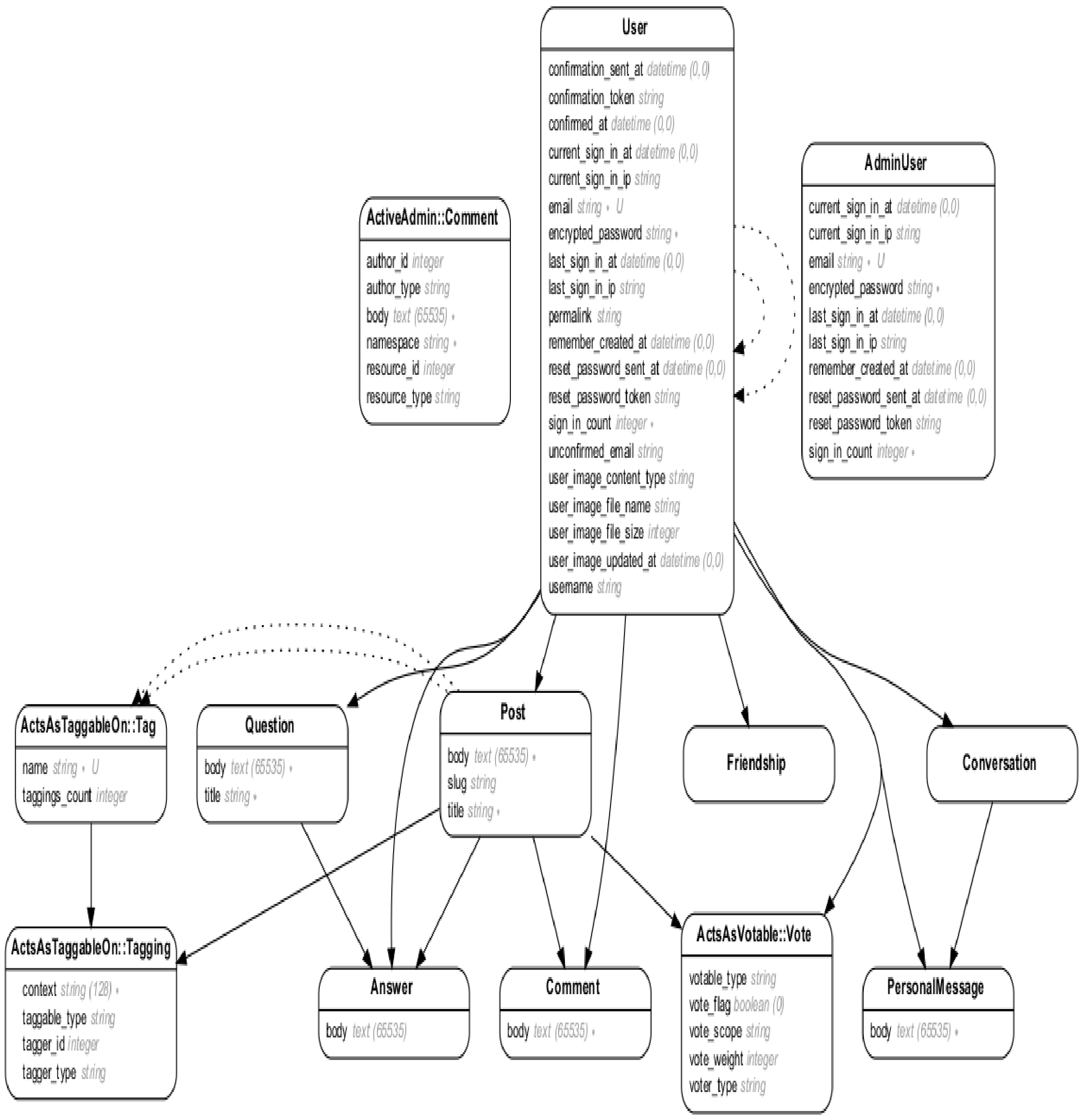


Fig 4.1.5 ERD


User
confirmation_sent_at datetime (0,0)
confirmation_token string
confirmed_at datetime (0,0)
current_sign_in_at datetime (0,0)
current_sign_in_ip string
email string ∘ U
encrypted_password string ∘
last_sign_in_at datetime (0,0)
last_sign_in_ip string
permalink string
remember_created_at datetime (0,0)
reset_password_sent_at datetime (0,0)
reset_password_token string
sign_in_count integer ∘
unconfirmed_email string
user_image_content_type string
user_image_file_name string
user_image_file_size integer
user_image_updated_at datetime (0,0)
username string
AdminUser
current_sign_in_at datetime (0,0)
current_sign_in_ip string
email string ∘ U
encrypted_password string ∘
last_sign_in_at datetime (0,0)
last_sign_in_ip string
remember_created_at datetime (0,0)
reset_password_sent_at datetime (0,0)
reset_password_token string
sign_in_count integer ∘
ActiveAdmin::Comment
author_id integer
author_type string
body text (65535) ∘
namespace string ∘
resource_id integer
resource_type string
ActsAsTaggableOn::Tag
name string ∘ U
taggings_count integer
Question
body text (65535) ∘
title string ∘
Post
body text (65535) ∘
slug string
title string ∘
Friendship
Conversation
ActsAsTaggableOn::Tagging
context string (128) ∘
taggable_type string
tagger_id integer
tagger_type string
Answer
body text (65535)
Comment
body text (65535) ∘
ActsAsVotable::Vote
votable_type string
vote_flag boolean (0)
vote_scope string
vote_weight integer
voter_type string
PersonalMessage
body text (65535) ∘

Chapter 5. Implementation

5.1 Recommendation System Algorithm

Cosine similarity as it's name suggests identifies the similarity between two(or more) vectors .The dot product and norm has been identified, then the cosine similarity of two vectors is simply the dot product of two unit vector [6].

1. Term Frequency(TF)

Term Frequency also known as TF measures the number of times a term(word) occurs in a document. Given below are the terms and their frequency on each of the document [7].

For example:

document = "I live in Nepal and Nepal is the best country in the world"

term frequencies can be listed as:

I = 1, live = 1 , in = 2, Nepal = 2, and = 1, is = 1, the = 2, best = 1, country = 1, world = 1

2. Normalized Term Frequency

I = 1/13, live = 1/13, in = 2/13, Nepal = 2/13, and = 1/13, is = 1/13, the = 2/13, best = 1/13, country = 1/13, world = 1/13

3. Inversed Document Frequency(IDF)

In the above document for IDF(Nepal):

For example:

IDF(Nepal) = 1 + log(Total Number of Documents / Number of Documents with term "Nepal" in it)

4. TF*IDF

We are trying to find out the relevant document that are similar to a given document. We should find tf*idf for all the terms in the given document.

5. Vector Space Model(Cosine Similarity)

Cosine Similarity(d1,d2) = Dot product(d1, d2) / ||d1|| * ||d2||

Dot product (d1,d2) = d1[0] * d2[0] + d1[1] * d2[1] * … * d1[n] * d2[n]

||d1|| = square root(d1[0]2 + d1[1]2 + ... + d1[n]2)

||d2|| = square root(d2[0]2 + d2[1]2 + ... + d2[n]2)

6. In this way similarity between posts are determined.

5.2 TESTING

At the completion time of this system, entire testing has been done and system has been made free as far as possible. For testing purposes, following testing methods has been used:

5.2.1 Unit Testing

Each of the modules of the system are tested separately in unit testing. The functionality of a specific section of code, usually at the function level are verified. Blogs, users are some modules functionality has been tested and compared with desired output. The test explorer has been used to check and manage each tests. User with their level to the access when logging and registering has been checked and compared with the desired output.

S.N	Email	Password	Expected result	Actual result	Remarks	
1.	rajan@gmail.com	123456	Blog posts	Blog posts	No error	
2	sabin@gmail.com	9865421	Blog posts	Blog posts	No error	
3.	ach@gmail.com	ach985689	Blog posts	Blog posts	No error	
4.	As@gmail.com	aruna98	Blog posts	Blog posts	No error	

Table 5.2.1 Unit Testing

5.2.2 Integration Testing

After unit testing the modules are integrated one by one and then tested the system for problems arising from component interaction. In integration testing, modules are combined and tested as a group to make sure that one unit does not cause problems to other unit. During integration testing phase, recipe add and user add module are tested to ensure proper functionality. Similarly, all other modules are also integrated and tested. Test cases for integration test between displaying user details:

Test	User	Expected result	Actual result	Remarks
1	Cdrrazan	Profile, blog, forum	Profile blog forum	No errpor
2.	Neymarsabin	Profile, blog,	Profile, blog,	No error

		forum	forum	
3.	Achyutd	Profile, blog, forum	Profile, blog, forum	No error

Table 5.2.2: Integration testing table

5.2.3 System Testing

System testing is done after all the units and integration testing are successfully completed,. Entire system test has been done to verify that it meets its requirements and if any problems are found, they are fixed and the system is tested again.

SN	Test case	Expecte d	Actual	Remarks
1.	Registration	Register user and send confirmation link	Register user and send confirmation link	No error
2.	Login	Access the blog	Access the blog	No error
3.	Create Blog	Write new blog	Write new blog	No error
4.	Forum	View and ask question	View and ask question	No error
5.	Profile	View user blog and forum question	View user blog and forum question	No error
6.	Conversation	Message sent and received	Message sent and received	No error
7.	Edit profile	Change user’s details	Change user’s details	No error
8.	Commmenting	Post comments and replies to the thread	Post comments and replies to the thread	No error
9.	Settings	Delete user account	Delete user account	No error

Table 5.2.3: System testing table

Chapter 6. Limitation and Future Enhancement

6.1 Limitation

Currently there is no user online viewing system. So, for the messaging platform users will have to use private messaging features. Sharing features will also not be available for now. Also the system has no question searching feature.

6.2 Future Enhancements

1. Real time Chatting system
2. Notification system
3. Jobs for developer and recommendation system,

7.Project-references

[1] *The incredible growth of the Internet since 2000*[online]. Available http://royal.pingdom.com/2010/10/22/incredible-graowth-of-the-internet-since-2000/

[2] *World Internet Users and 2017 Population Stats*[online]. Available http://www.internetworldstats.com/stats.htm

[3] *Medium homepage*[online]. Available https://medium.com/

[4] *Welcome to Stack Overflow*[online]. Available https://stackoverflow.com/tour

[5] *10 Social Networks for Developers*[online]. Available https://www.awwwards.com/10-social-networks-for-developers.html

[6] *Cosine similarity*[online]. Available http://www.thefactmachine.com/cosine-similarity/

[7]*Tf-Idf cosine similarity* [online]. Available https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

Source Code

1. Post

```
class Post < ActiveRecord::Base
  # include PublicActivity::Model
  # tracked owner: ->(controller,model ) { controller && controller.current_user },
  #         title: ->(controller, model ) { controller && model.title }

  validates :title,:body, presence: true
  belongs_to :user
  delegate :email, to: :user
  delegate :username, to: :user
  has_many :comments,dependent: :destroy

  acts_as_votable

  # extend FriendlyId
  # friendly_id :title, use: :slugged



  def to_param
    "#{id} #{title}".parameterize
  end

  acts_as_taggable

end
```

2. Forum

```
class Question < ApplicationRecord
```

```
  validates :title, presence: true
  validates :body, presence: true

  has_many :answers
  belongs_to :user


  def to_param
    "#{id} #{title}".parameterize
  end

end
```

3. Admin

```
class AdminUser < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable,
         :recoverable, :rememberable, :trackable, :validatable
End
```

Controller

1. User

```
class UsersController < ApplicationController
  def index
    @users = User.search(params[:search])
  end
  def show
    #@user = User.find_by_permalink(params[:id])
    find_user
    @posts = @user.posts.all
  end
```

```
 def my_friends
   @friendships = current_user.friends
   @friendships_inverse = current_user.inverse_friends
 end

 def add_friends
   @friend = User.find(params[:friend])
   current_user.friendships.build(friend_id: @friend.id)
   if current_user.save
     redirect_to my_friends_path, notice:  "Friend was successfully followed"
   else
     redirect_to my_friends_path, flash[:error] = "There was an error with adding user as friend"
   end
 end

 private
 def find_user
   @user = User.find(params[:id])
 end

end
```

2. Questions

```
class QuestionsController < ApplicationController
  before_action :authenticate_user!, except: [:index,:show]
  before_action :set_question, only: [:show, :edit, :update, :destroy]

  # GET /questions
  # GET /questions.json
  def index
    @questions = Question.all
  end

  # GET /questions/1
  # GET /questions/1.json
  def show
  end

  # GET /questions/new
  def new
    @question = current_user.questions.build
  end

  # GET /questions/1/edit
  def edit
    authorize! :update, @question
  end

  # POST /questions
  # POST /questions.json
  def create
    @question = current_user.questions.build(question_params)

    respond_to do |format|
      if @question.save
            format.html { redirect_to @question, notice: 'Question was successfully
created.' }
        format.json { render :show, status: :created, location: @question }
      else
        format.html { render :new }
        format.json { render json: @question.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /questions/1
  # PATCH/PUT /questions/1.json
  def update
    if current_user == @question.user
      @question.update(question_params)
      @question.save
      flash[:notice] = "Edit sucessful: #{@question.title}"
      redirect_to @question
    # respond_to do |format|
    #   if @question.update(question_params)
      #     format.html { redirect_to @question, notice: 'Question was successfully updated.' }
    #     format.json { render :show, status: :ok, location: @question }
    #   else
    #     format.html { render :edit }
    #     format.json { render json: @question.errors, status: :unprocessable_entity }
    #   end
    end
  end

  # DELETE /questions/1
  # DELETE /questions/1.json
  def destroy
    authorize! :destroy, @question
    @question.destroy
    respond_to do |format|
        format.html { redirect_to questions_url, notice: 'Question was successfully deleted.' }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_question
      @question = Question.find(params[:id])
    end

    # white listing params
    def question_params
      params.require(:question).permit(:title, :body)
    end
end
```

3. Application

```
class ApplicationController < ActionController::Base
  include Pundit
  include PublicActivity::StoreController



  protect_from_forgery with: :exception
  before_action :configure_permitted_paramaters, if: :devise_controller?
        before_action    :configure_permitted_paramaters_for_registration,    if:
:devise_controller?
  #rescue_from ActiveRecord::RecordNotFound, :with => {:render => "404"}

  rescue_from CanCan::AccessDenied do |exception|
    redirect_to question_path, :alert => exception.message
  end

  def get_current_user_email
    current_user.email
  end
  protected

  def configure_permitted_paramaters
                                  devise_parameter_sanitizer.permit(:account_update,         keys:
[:user_image,:username])
  end

  def configure_permitted_paramaters_for_registration
    devise_parameter_sanitizer.permit(:sign_up, keys: [:username])
  end
end
```