SOCIAL MEDIA APPLICATION

A PROJECT REPORT SUBMITTED

by

Mudit Rastogi (2000290140072)

Lokesh Gangwar (2000290140062)

Rashika Garg (2000290140100)

Prabhat Chaudhry (2000290140087)

Submitted in partial fulfillment of the

Requirements for the Degree of

Master of Computer Application

Under the Supervision of

Dr. Sangeeta Arora

ASSOCIATE PROFESSOR

Submitted to

Faculty of MCA

DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY
LUCKNOW
(Formerly Uttar Pradesh Technical University, Lucknow)

DECLARATION

I hereby declare that the work presented in this report entitled “Social Media Application”, was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

(Candidate Signature)

CERTIFICATE

Certified that Mudit Rastogi ,Lokesh Gangawar , Rashika Garg , Prabhat Chaudhry has carried out the project work presented in this report entitled "Social Media Application" for the award of Master of Computer Application from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

Dr. Sangeeta Arora
Associate Professor
Dept. of Computer Applications
KIET Group of Institutions, Ghaziabad

External Examiner

Dr. Ajay Kumar Srivastava
Professor & Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Date:

Table of Contents

INTRODUCTION

Teleconferencing or chatting, is a method of using technology to bring people and ideas “together” despite of the geographical barriers. The Technology has been available for years but the acceptance it was quit recent. Our project is an example of a chat server.

It is made up of 2 applications the client application, which runs on the user’s PC and server application, which runs on the PC in the network. To start chatting client should get connected to server where they can practice two kind of chatting, public one(Message is broadcasted to all connected users) and private one (between any 2 users only).

The “Online Chat Application” has been developed to override the problem prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faces by this existing system. Moreover this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

This application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal Knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly.

Online chat application, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping . Thus it will help organization in better utilization of resources.

1.1 Project Details

The purpose of Online Chatting Application is to automate the existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Online Chat Application, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information.

The aim is to automate its existing manual system by the help of computerized equipments and full-fledge computer software, fulfilling their requirements so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the client.

1.2Objective of Project on Online Chat Application

The main objective of the Project on Online Chat Application is to manage the details of chat Profile, Chat History, Group Chat, Smilies Chat. It manages all the information about Chat Profile, Multi Chat, Chat Profile. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the chat Profile, chat user, multi chat. It tracks all the details about the Chat history, Group Chat, Smilies chat.

1.3 Functionalities provided by Online Chat Application are as follows:

- Provides the searching facilities based on various factors. Such as Chat History, Group Chat, Smilies Chat.
- Online Chat Application also manage the Multi Chat details online or Group Chat details, Smile Chat details,Chat Profile, Uploding Post.
- It Manage the information of chat user.
- It can do audio, video calls.
- Show the information and description of the Chat Profile Chat History
- Integration of all records of Smiles Chat.
- Manage the information of Group Chat.

1.4 Scope of the project Online Chat Application

It may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to Online Chat Application. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

Our project aims at Business process automation, i.e we have tried to computerize various processes of online Chat Application.

- In computer system the person has to fill the various forms & number of copies of the form can be easily generated at a time.
- In computer system it is not necessary to create the manifest but we can directly print it.

1.5 Hardware requirements

In Hardware requirement we require all those components which will provide us the platform for the development of the project. The minimum hardware requirement for the development of this project is as follows-

- **Ram – minimum 128 Mb**
- **Hard Disk – minimum 5 Gb**

- **Processor- Pentium 3**
- **Floppy drive 1.44 Inch**
- **CD Drive**

These all are the minimum hardware requirement for our project. We want to make our project to be used in any, Type of computer therefore we have taken minimum configuration to a large extent 128 MB ram is used so that we can execute our project in a least possible RAM 5 GB Hard disk is used because project takes less space to be executed or stored. Therefore minimum hard disk is used. Others enhancements are according to the needs.

1.6 Software requirement

Software's can be defines as program which run on our computer. It act as petrol in the vehicle. It provides the relationship between the human and a computer. It is very important to run software to function the computer. Various software's are needed in this project for its development.

Which are as follows--??

- **Operating System- windows 7**
- **Others- Visual Studio**

We will be using visual basic as our front hand because it is easier to use and provides feature to the users which is used for the development of the project.

2.0 Feasible Study

After doing the project Online Chat Application, study and analying all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

2.1 Economical Feasibility

This is a very important aspect to be considered while developing a project. We decided the technology based on possible cost factor.

- All hardware and software cost has to be borne by the organization.
- Overall we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for system.

2.2 Technical Feasibility

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the system requirement specification(SRS), and checked if everything was possible using different type of frontend and backend platform.

2.3 Operational Feasibility

No doubt the proposed system is fully GUI based that is very user friendly and all inputs to be taken all self-explanatory even to a layman , Besides, a proper training has been conducted to let know the essence of the system to the users so that they feel comfortable with new system. As far our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

1.1 Purpose

- There are numerous product available that allow for real time "Chatting" over the Internet.
- The purpose of this project is to implement a Mern based chat application that will allow users with an internet connection to engage in private and public conversation.
- The development of this project centered on the development of a message protocol that would allow the application to properly log in users , send message, and perform system maintenance.

1.2 Document Conventions

a) OCA: Online Chatting Application
b) SRS: System Requirement Specifications
c) GUI: Graphical User Interface
d) UC: Use Case
e) Approx: Approximately

1.3 Intended Audience and Reading Suggestions

The intended audience of this SRS:

Developers: Project developers can go through this document and understand the requirements presented by the client for the development of the product and start to work on it accordingly. If the developer wants to change, modify or add new requirements/features into the existing system, he/she must first consult this document, update the requirements in an appropriate manner and pass the information correctly to the other phases of the development process.

Users: Users, mainly students, can refer this document to get a better idea of the functioning of the OCA and determine whether the requirements listed here have been satisfactorily implemented.

Testers: Testers are required to go through the requirements very carefully to design test cases and test the end product to ensure that the requirements presented by the client have been implemented and the software is working efficiently.

1.4 Product Scope

1.5 References

IEEE Software Requirement Specification standard format

1. Overall Description

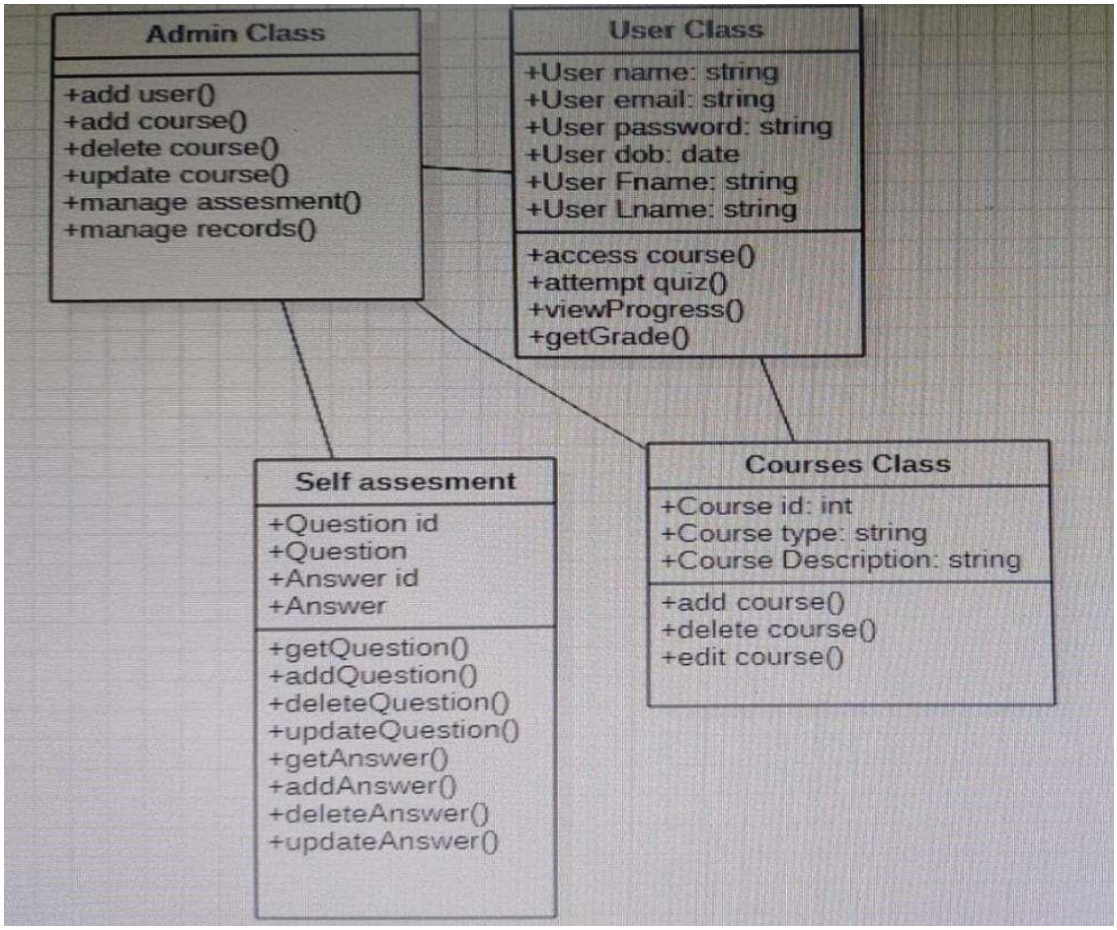
2.1 Product Perspective

There is a two way communication between different client and server. This chat application can be used for group discussion. It allows users to find other logged in users.

2.2 Product Functions

- OCA enables users to perform following functions:
    1. Register
    2. Email verification
    3. Change password
    4. Login
    5. Chat bot
    6. Message broadcasting
    7. Encryption
    8. Cloud Synchronization
    9. Push Notification
- It will enable admins to perform:
    1. Keep records of users
    2. Ad/delete/update Feature

2.3 User Classes and Characteristics



2.4 Operating Environment

- Web browser: Any web browser
- Operating system: Windows 7 and above, ios12 and above, android 6.0 and above
- Internet speed: Minimum speed of 2Mbps
- OCA is free to browse and can be accessed by anyone

2.5 Design and Implementation Constraints

- The user must have individual ID for creating an account
- OCA shall be developed using HTML5/CSS and MERN on Visual Studio Code and MongoDB for database.
- Bootstrap will be used to make OCA responsive.

2.6 User Documentation

- Operating procedures

- Codes and test cases

2.7 Assumptions and Dependencies

- Database used for OCA is MongoDB
- Operation of the OCA depends on the number of concurrent users
- Each user must have a User ID and password
- There must be an Internet connection
- There should only be one account per email address

2. External Interface Requirements

3.1 User Interfaces

Login Screen: This feature is inbuilt in the software for the verification of authenticated user over the unregistered ones.

User Profile: This enables the user to search people, view profile and people.

3.2 Hardware Interfaces

- Laptop, Personal Computer, Mobile phones, Tablets

3.3 Software Interfaces

OCA is a multi-user environment, it uses HTML/CSS for the web pages and MongoDB as the backend application tool.

3.4 Communications Interfaces

User can communicate with the people send message , photos ,video ,audio.

3.5 Registration page

4.1.1 Description and Priority

Provides the user with a page to register for the OCA

Priority = 10

4.1.2 Stimulus/Response Sequences:

Stimulus: User clicks on Signup Link.

Response: Signup Page is displayed

Stimulus: User enters email address

Response: The system checks if the email already exists in the database or not. If not then, a verification code sent to user's email address and the user will be redirected to a page to enter the verification code. Else, and error message pops up stating "This email already exists."

Stimulus: User enters verification code

Response: The system verifies the code, if true then the user is asked to create password.

Stimulus: User enters password and click on create account button.

Response: New account is created and user is redirected to the home page.

4.1.3 Functional Requirements

REQ-1: The user should be able to view and click on login link

REQ-2: The user should be able to enter and validate the username and Password.

REQ-3: There should be only one account per email else and error pop-up should appear.

3.6 Login/Signup Page

4.2.1 Description and Priority

Provides the user with a page to login for the OCA.

Priority = 10

4.2.2 Stimulus/Response Sequences

Stimulus: User clicks on Login Link.

Response: Login Page is displayed

Stimulus: User Enters email and Password

Response: email and Password are validated from the database.

Stimulus: User Clicks on Login Button

Response: Home Page will be displayed, if the login credentials are correct else Error Message will be displayed and the user will be told to enter email and password again.

4.2.3 Functional Requirements

REQ-1: The user should be able to view and click on login link

REQ-2: The user should be able to enter and validate the username and Password.

REQ-3: There should be only one account per email else and error pop-up should appear.

3.7 View Course

4.3.1 Description and Priority

Provides the user with a page to view different user profiles with an option to Add friend.

Priority = 9

4.3.2 Stimulus/Response Sequences

Stimulus: User clicks on Profile.

Response: Profile is displayed

Stimulus: User clicks on Search people

Response: Different users is displayed.

Stimulus: User Clicks on add friend.

Response: it will send friend request to user.

Stimulus: User Clicks on send message.

Response: it will send the message.

4.3.3 Functional Requirements

REQ-1: The user should be able to view people and add friend.

REQ-2: The user should be able to send message audio video make calls. REQ-3: The user should be able to block user

4.4.1 Description and Priority

Provides the user to send the message ,audio ,video ,make calls block user.

Priority = 8

4.4.2 Stimulus/Response Sequences

Stimulus: User Clicks on send message.

Response: it will send the message

Stimulus: User clicks on block.

Response: The user will blocked.

View User Profile

4.5.1 Description and Priority

Provides the user with a page to view his/her profile

Priority = 9

4.5.2 Stimulus/Response Sequences

Stimulus: User clicks on Settings -> User Profile

Response: User profile is displayed with user details and courses enrolled.

4.5.3 Functional Requirements

REQ-1: The user should be able to click and view his/her profile.

REQ-2: The user should be able to view the list of enrolled courses.

3.8 Logout Page

4.6.1 Description and Priority

Provides the user with a logout option.

Priority = 10

4.6.2 Stimulus/Response Sequences

Stimulus: User clicks on Logout link

Response: User is logged out and index page is displayed.

4.6.3 Functional Requirements

REQ-1: The user should be able to logout from the system.

REQ-2: If no activity is performed within 10 minutes, then the user will be logged out.

3. Other Nonfunctional Requirements

4.1 Performance Requirements

PE-1: Responses to activities should not take very long time onto the screen (approx.

2-5s, depending upon user's Internet speed).

PE-2: The system should display confirmation message to the user very quickly after the user submits information to the system (depending upon user's Internet

speed).

PE-3: OCA should work fine even with a number of users using concurrently.

Pe-4: It should support both 32 and 64 bits.

4.2 Safety Requirements

SR-1: Consistency: Checking the fact that all clients must be attached to one server, so there is an appropriate control of the information.

SR-2: Users' data should be kept confidential.

4.3 Software Quality Attributes

SQA-1: ELP should be available to the users all the time.

SQA-2: The code should be neat and it should contain all the necessary comments for the ease of maintenance.

4.4 Business Rules

(cc, contents created by admin, about us module)

5. Other Requirements

All the requirements have been specified.

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: To Be Determined List

Practical-1

Aim: Prepare an SRS document in line with the IEEE recommended standards for the specified Case Study. (Functional Requirements).

A functional requirement document helps you to define the functionality of a system or one of its subsystems. Functional requirements along with requirement analysis help identify missing requirements. They help clearly define the expected system service and behavior.

The functional requirement of Project discussed as follows:

REQ-1: The user should be able to view and click on login link .

REQ-2: The user should be able to enter and validate the username and Password.

REQ-3: There should be only one account per email else and error pop-up should appear.

REQ-4: The user should be able to view people and add friend.

REQ-5: The user should be able to send message audio video make calls.

REQ-6: The user should be able to block user.

Practical-2

Aim: Prepare an SRS document in line with the IEEE recommended standards for the specified Case Study. (Non-Functional Requirements).

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. They ensure the usability and effectiveness of the entire system

The Non-functional requirement of Project discussed as follows:

Performance Requirements

PE-1: Responses to activities should not take very long time onto the screen.

PE-2: The system should display confirmation message to the user very quickly after the user submits information to the system .

PE-3: OCA should work fine even with a number of users using concurrently. PE-4: It should support both 32 and 64 bits.

Safety Requirements

SR-1: Consistency: Checking the fact that all clients must be attached to one server, so there is an appropriate control of the information.

SR-2: Users' data should be kept confidential.

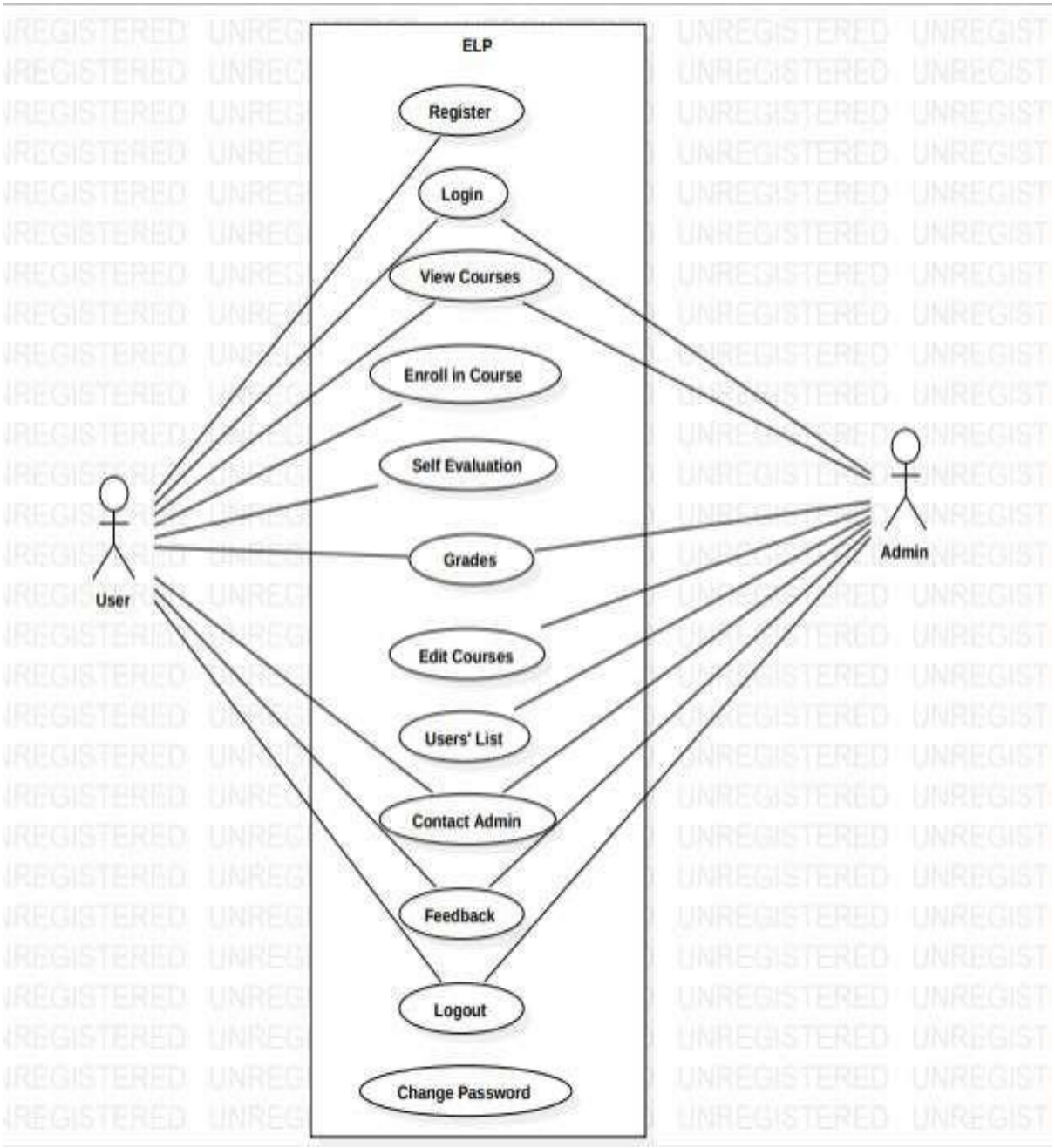
Software Quality Attributes

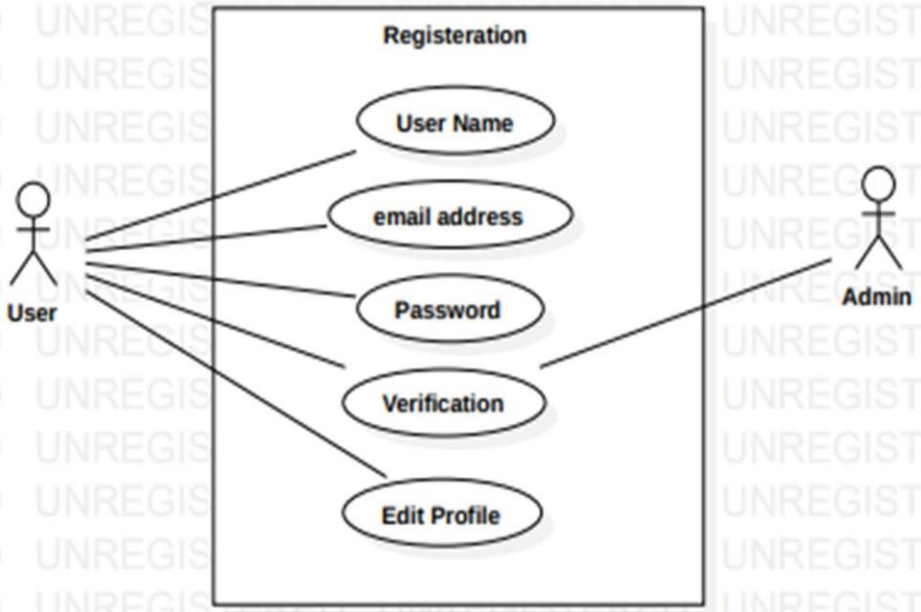
SQA-1: ELP should be available to the users all the time.

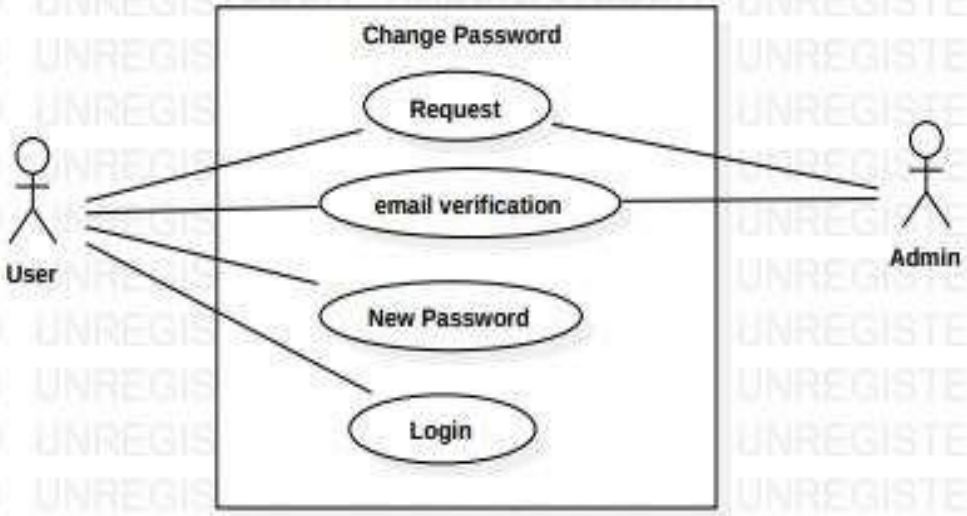
SQA-2: The code should be neat and it should contain all the necessary comments for the ease of maintenance.

PRACTICLE :- 3

Aim :- Draw the use case diagram and specify the role of each of the actors for the specified Case Study.




Registeration
User Name
email address
Password
Verification
Edit Profile
User
Admin


Change Password
Request
email verification
New Password
Login
User
Admin

Practicle :- 4

Aim :- Prepare state the precondition, post condition and function of each use case for the specified Case Study.

Precondition and postconditions for use case diagram

o Preconditions:

- Registration page

  - REQ-1: The user should be able to view and click on login link
  - REQ-2: The user should be able to enter and validate the username and Password.
  - REQ-3: There should be only one account per email else and error pop-up should appear.

- Login/Signup Page

  - User clicks on Login Link.
  - Login Page is displayed
  - User Enters email and Password
  - email and Password are validated from the database.
  - User Clicks on Login Button

□□ Home Page will be displayed, if the login credentials are correct

- else Error Message will be displayed and the user will be told to enter email and password again.

- View Course

□□Provides the user with a page to view different user profiles with an option to Add friend.

□□ User clicks on Profile. □□ Profile is displayed

□□ User clicks on Search people □□ Different users is displayed. □□User Clicks on add friend.

□□ it will send friend request to user. □□User Clicks on send message.

□□it will send the message.

- View User Profile

□□□□□□□□□ User clicks on Settings -> User Profile

- User profile is displayed with user details and courses enrolled.

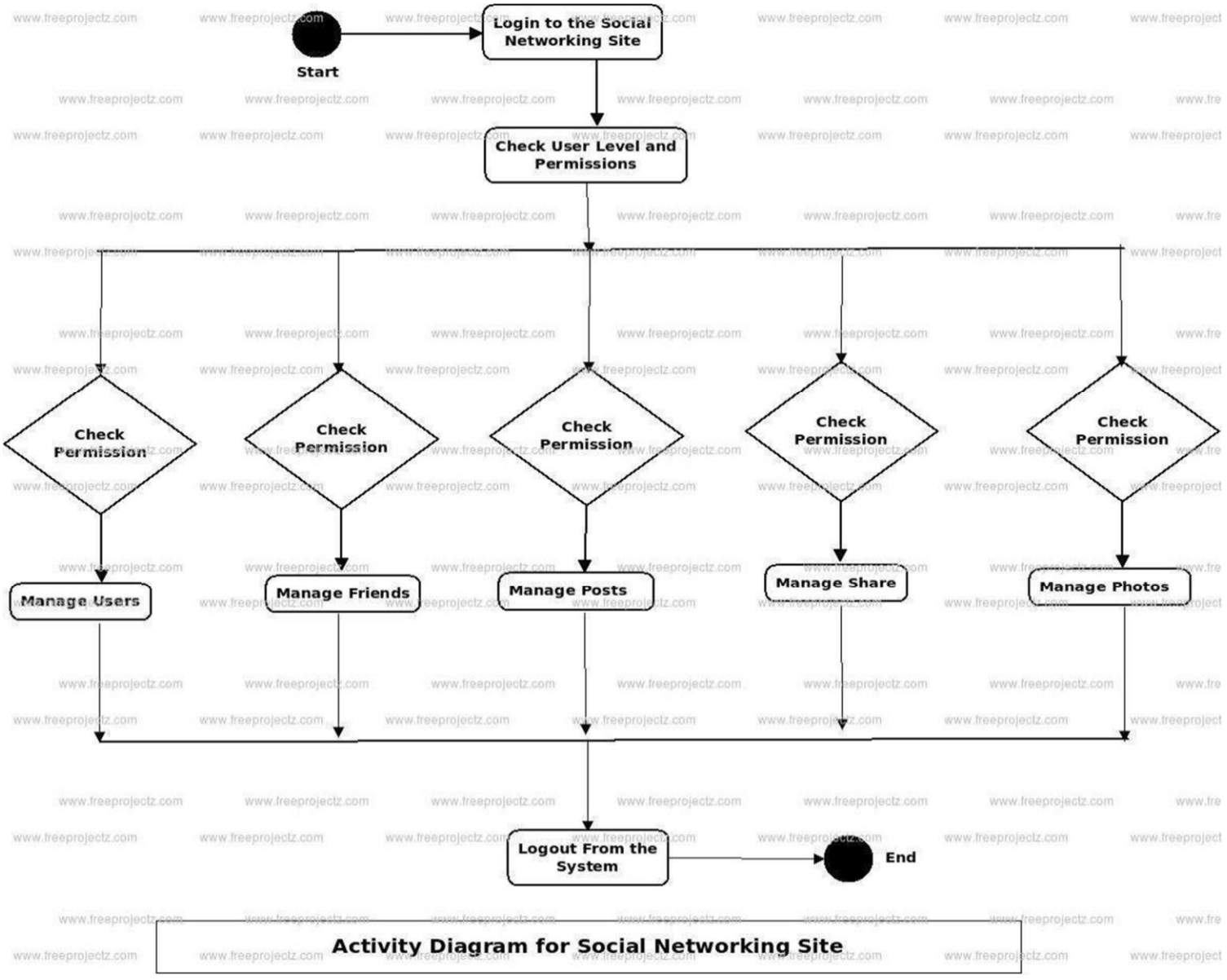
o Postconditions:

- o Logout Page

- User clicks on Logout link
- User is logged out and index page is displayed.
- The user should be able to logout from the system.
- If no activity is performed within 10 minutes, then the user will be logged out.

Practicle :- 5

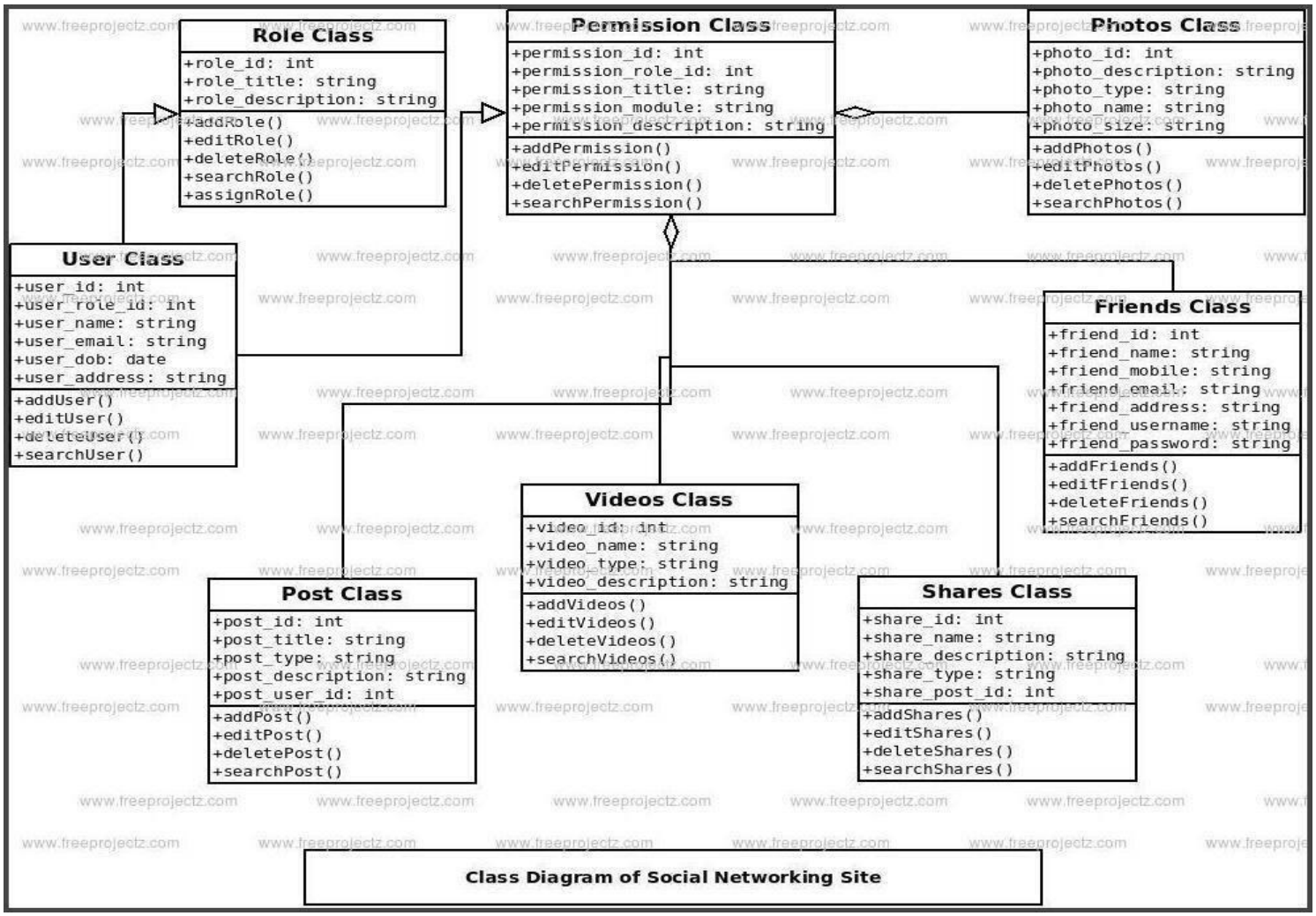
Aim :- Draw the activity diagram for the specified Case Study.



Activity Diagram for Social Networking Site

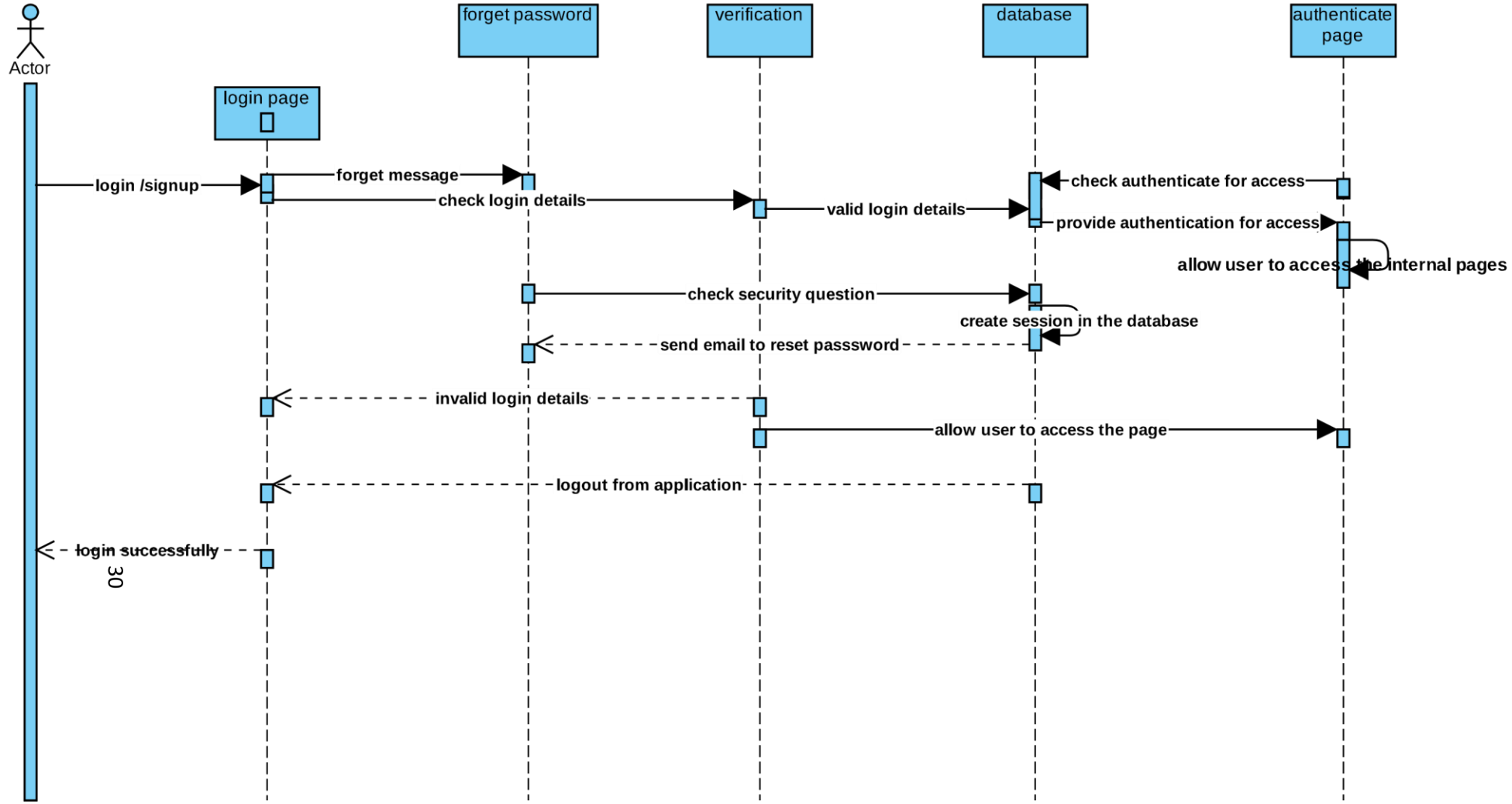
Practicle :- 6

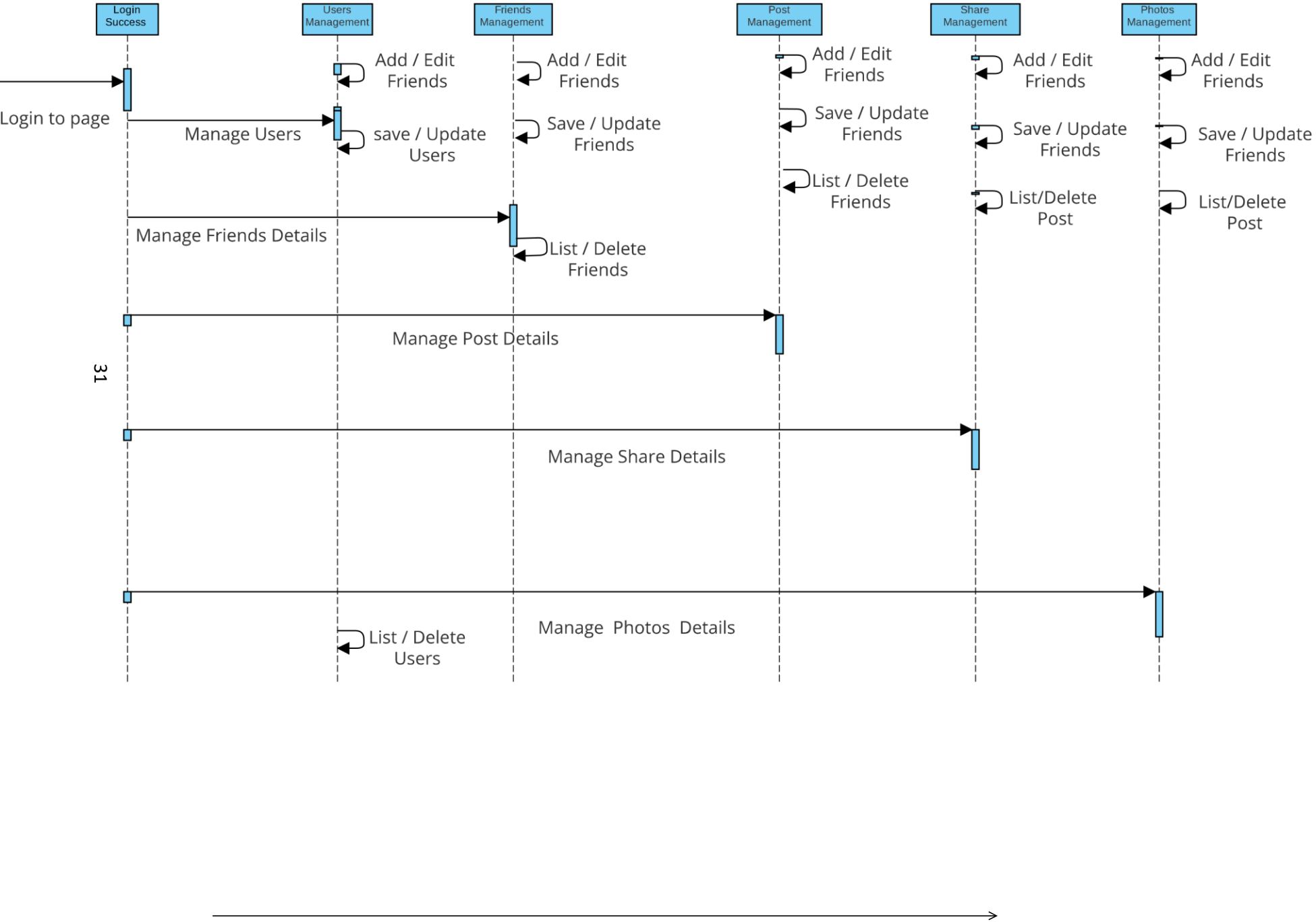
Aim :- Identify the classes. Classify them as weak and strong classes and draw the class diagram for the specified Case Study.



Practicle :- 7

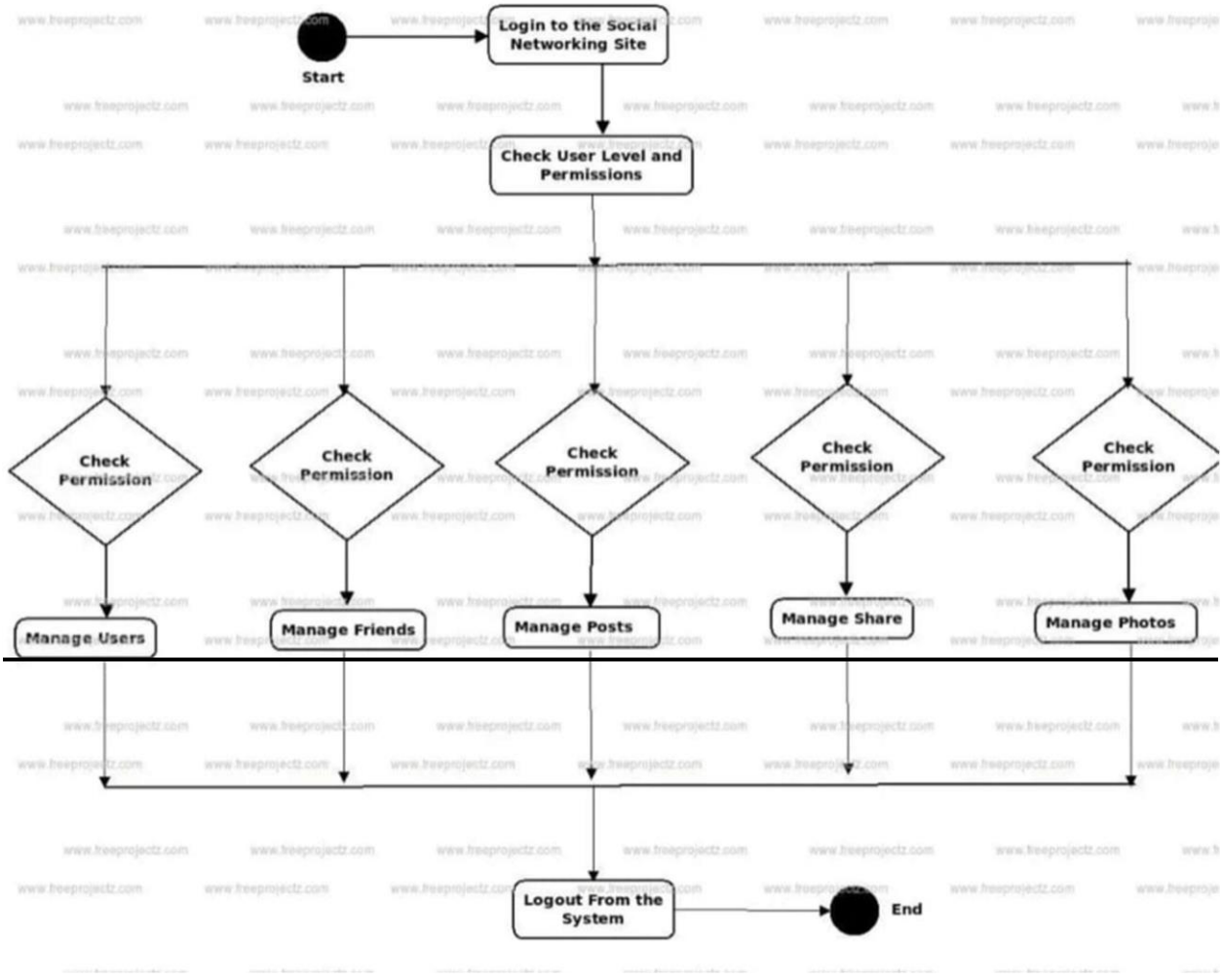
Aim :- Draw the sequence diagram for any two scenarios.




Login Success
Users Management
Friends Management
Post Management
Share Management
Photos Management
Add / Edit Friends
Add / Edit Friends
Add / Edit Friends
Add / Edit Friends
Add / Edit Friends
Login to page
Manage Users
save / Update Users
Save / Update Friends
Save / Update Friends
Save / Update Friends
Save / Update Friends
List / Delete Friends
List/Delete Post
List/Delete Post
Manage Friends Details
List / Delete Friends
Manage Post Details
Manage Share Details
Manage Photos Details
List / Delete Users

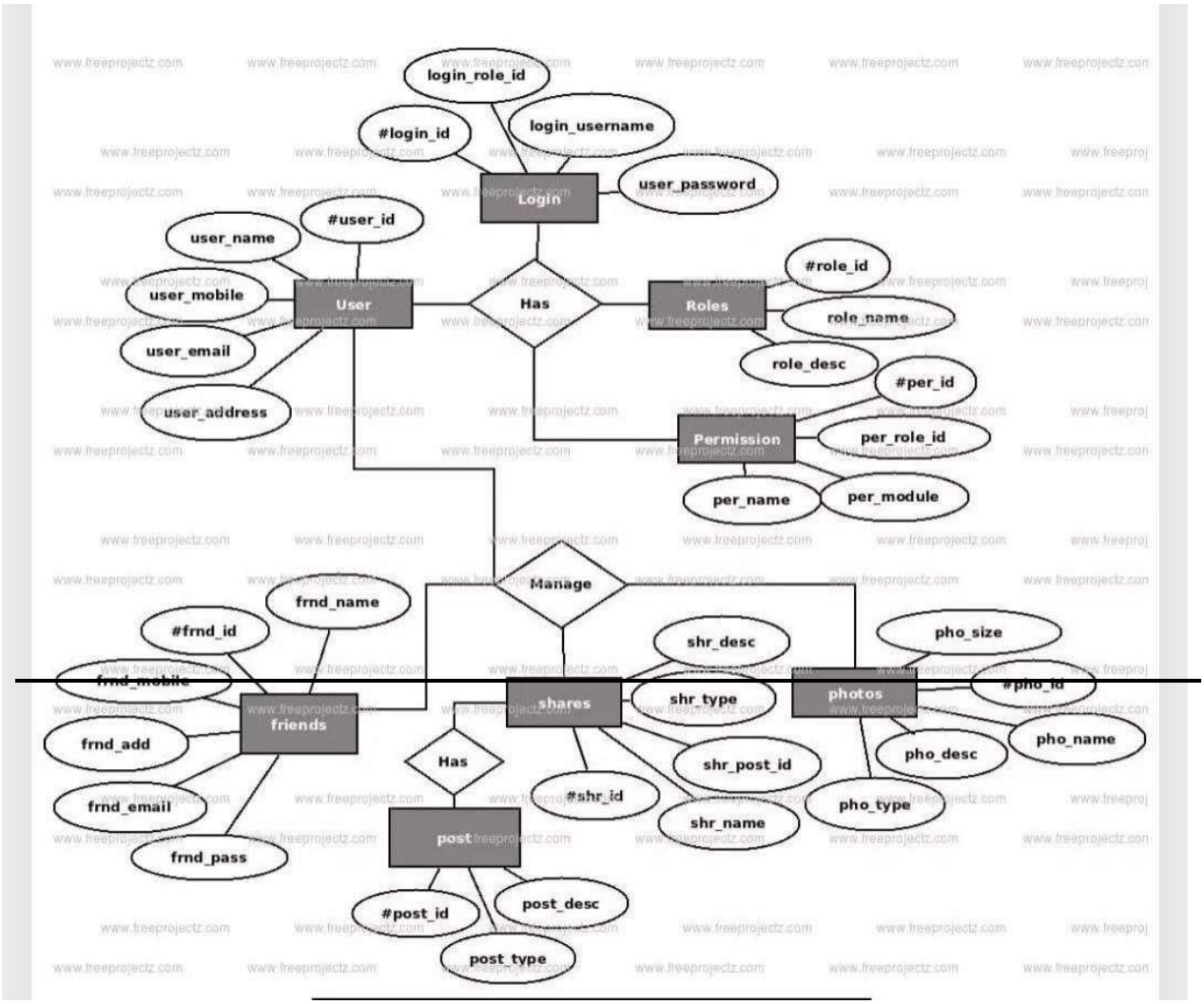
Practical- 8

A i m : - Draw the collaboration diagram for the specified Case Study.



Practicle :- 9

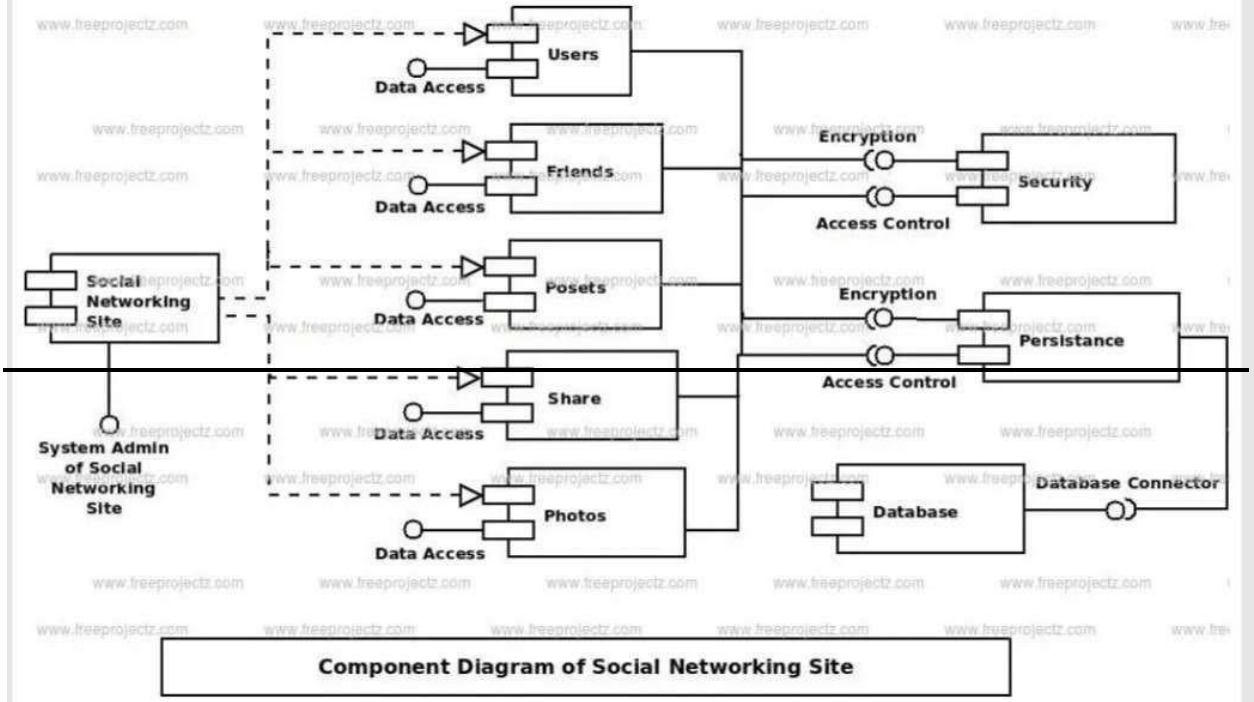
Aim :- Draw the state chart diagram for the specified Case Study.


login_role_id
#login_id
login_username
user_password
Login
#user_id
user_name
user_mobile
user_email
user_address
User
Has
Roles
#role_id
role_name
role_desc
#per_id
per_role_id
per_name
per_module
Permission
Manage
frnd_name
#frnd_id
frnd_mobile
frnd_add
frnd_email
frnd_pass
friends
shr_desc
shr_type
shares
Has
#shr_id
shr_post_id
shr_name
post
#post_id
post_desc
post_type
pho_size
#pho_id
photos
pho_name
pho_desc
pho_type

Practicle :- 10

Aim :- Draw the component diagram for the specified Case Study

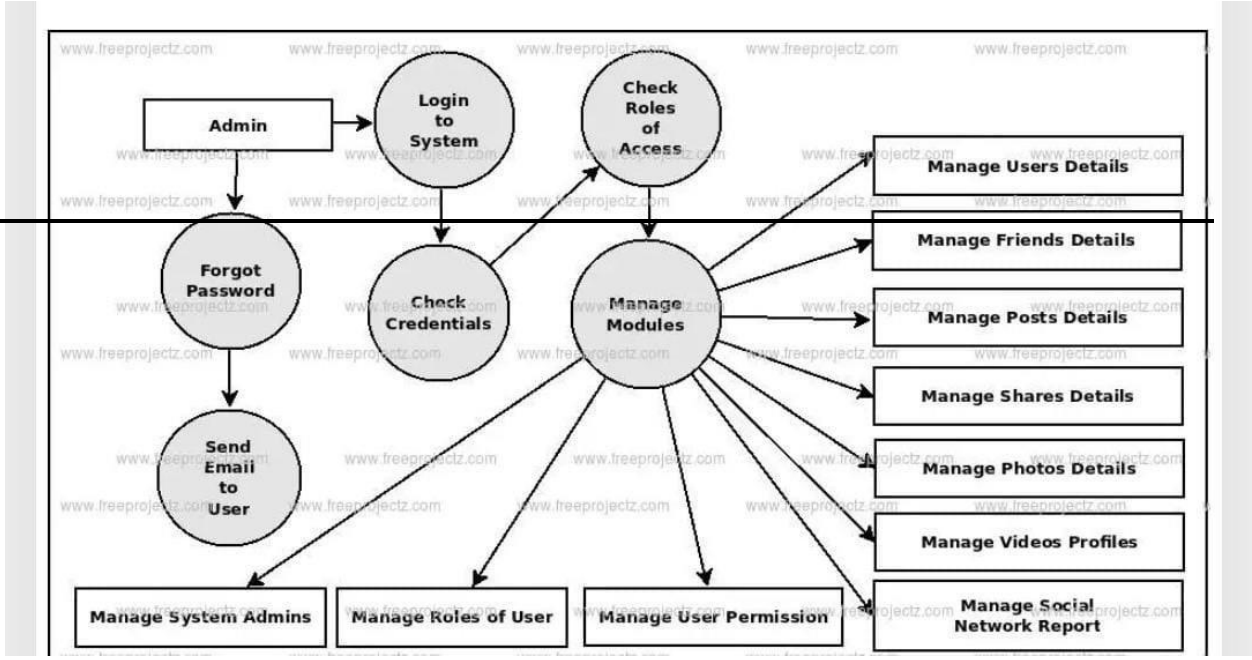
Component Diagram:



Component Diagram of Social Networking Site

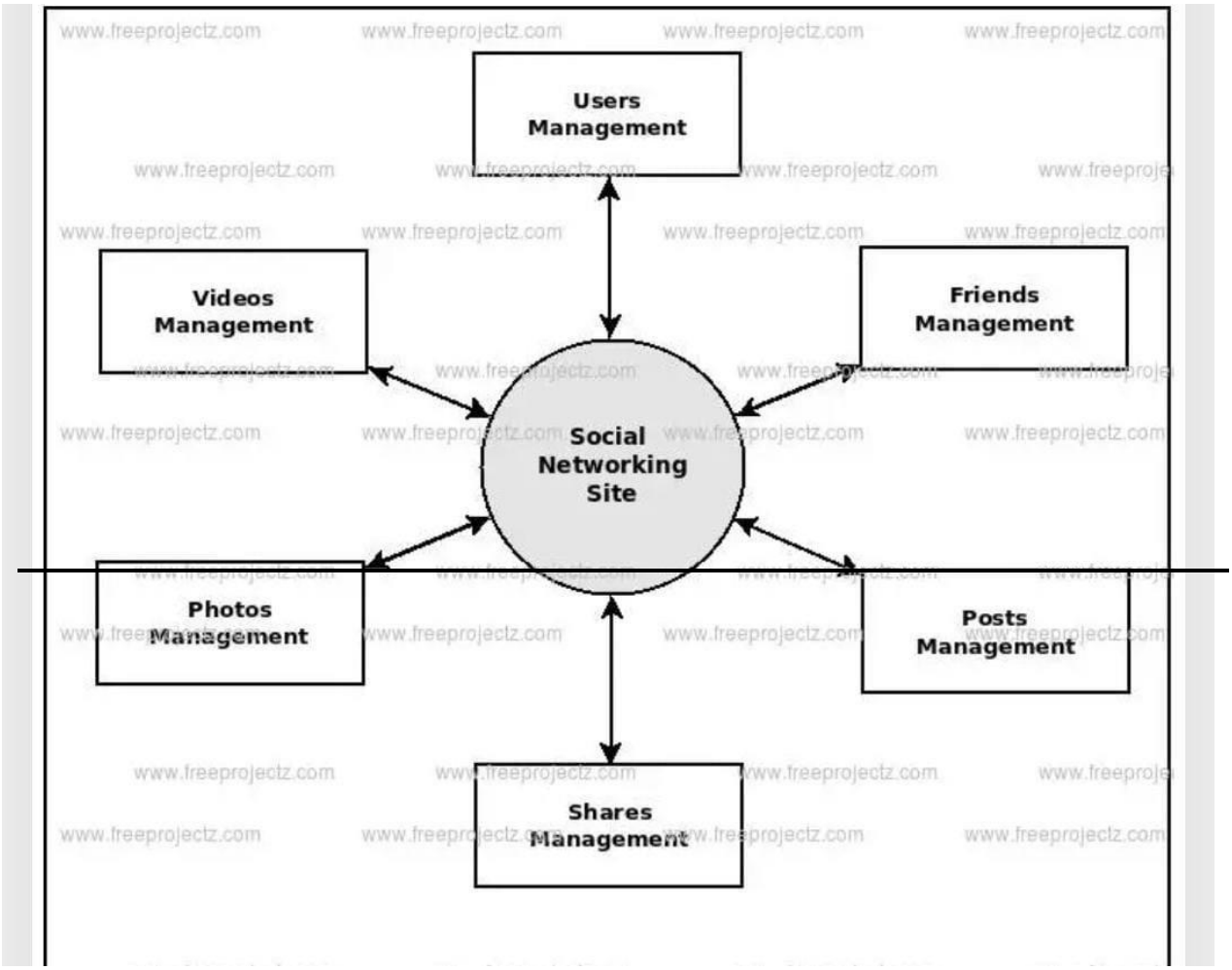
Practicle :- 11

Aim :- Draw the deployment diagram for the specified Case Study.



Practicle :- 12

Aim :- Design a test suite for the specified Case Study.


Users Management
Videos Management
Friends Management
Social Networking Site
Photos Management
Posts Management
Shares Management

PROJ

PROJECT CODE

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="./icon-web-01.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></scr
ipt>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"></sc
ript>

    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
```

```
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.2/css/all.min.css" rel="stylesheet">

    <title>FunCloud</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

INDEX.JS

```
import React from 'react';
import ReactDOM from 'react-dom';
import './styles/global.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import DataProvider from './redux/store'

ReactDOM.render(
  <React.StrictMode>
    <DataProvider>
      <App />
    </DataProvider>
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

APP.JS

```
import { useEffect } from 'react'
import {BrowserRouter as Router, Route} from 'react-router-dom'

import PageRender from './customRouter/PageRender'
import PrivateRouter from './customRouter/PrivateRouter'

import Home from './pages/home'
import Login from './pages/login'
import Register from './pages/register'

import Alert from './components/alert/Alert'
import Header from './components/header/Header'
import StatusModal from './components/StatusModal'

import { useSelector, useDispatch } from 'react-redux'
import { refreshToken } from './redux/actions/authAction'
import { getPosts } from './redux/actions/postAction'
import { getSuggestions } from './redux/actions/suggestionsAction'

import io from 'socket.io-client'
import { GLOBALTYPES } from './redux/actions/globalTypes'
import SocketClient from './SocketClient'

import { getNotifies } from './redux/actions/notifyAction'
import CallModal from './components/message/CallModal'
import Peer from 'peerjs'

function App() {
  const { auth, status, modal, call } = useSelector(state => state)
  const dispatch = useDispatch()

  useEffect(() => {
    dispatch(refreshToken())

    const socket = io()
    dispatch({type: GLOBALTYPES.SOCKET, payload: socket})
    return () => socket.close()
  },[dispatch])
```

```
useEffect(() => {
  if(auth.token) {
    dispatch(getPosts(auth.token))
    dispatch(getSuggestions(auth.token))
    dispatch(getNotifies(auth.token))
  }
}, [dispatch, auth.token])


useEffect(() => {
  if (!("Notification" in window)) {
    alert("This browser does not support desktop notification");
  }
  else if (Notification.permission === "granted") {}
  else if (Notification.permission !== "denied") {
    Notification.requestPermission().then(function (permission) {
      if (permission === "granted") {}
    });
  }
},[])


useEffect(() => {
  const newPeer = new Peer(undefined, {
    path: '/', secure: true
  })

  dispatch({ type: GLOBALTYPES.PEER, payload: newPeer })
},[dispatch])


return (
  <Router>
    <Alert />

    <input type="checkbox" id="theme" />
    <div className={`App ${(status || modal) && 'mode'}`}>
      <div className="main">
        {auth.token && <Header />}
        {status && <StatusModal />}
        {auth.token && <SocketClient />}
        {call && <CallModal />}

        <Route exact path="/" component={auth.token ? Home : Login} />
```

```
          <Route exact path="/register" component={Register} />

          <PrivateRouter exact path="/:page" component={PageRender} />
          <PrivateRouter exact path="/:page/:id" component={PageRender} />

        </div>
      </div>
    </Router>
  );
}

export default App;
```

SOCKET CLIENT.JS

```
import React, { useEffect, useRef } from 'react'
import { useSelector, useDispatch } from 'react-redux'
import { POST_TYPES } from './redux/actions/postAction'
import { GLOBALTYPES } from './redux/actions/globalTypes'
import { NOTIFY_TYPES } from './redux/actions/notifyAction'
import { MESS_TYPES } from './redux/actions/messageAction'

import audiobell from './audio/got-it-done-613.mp3'


const spawnNotification = (body, icon, url, title) => {
    let options = {
        body, icon
    }
    let n = new Notification(title, options)

    n.onclick = e => {
        e.preventDefault()
        window.open(url, '_blank')
    }
}

const SocketClient = () => {
    const { auth, socket, notify, online, call } = useSelector(state => state)
    const dispatch = useDispatch()

    const audioRef = useRef()

    // joinUser
    useEffect(() => {
        socket.emit('joinUser', auth.user)
    },[socket, auth.user])

    // Likes
    useEffect(() => {
        socket.on('likeToClient', newPost =>{
            dispatch({type: POST_TYPES.UPDATE_POST, payload: newPost})
        })

        return () => socket.off('likeToClient')
    },[socket, dispatch])
```

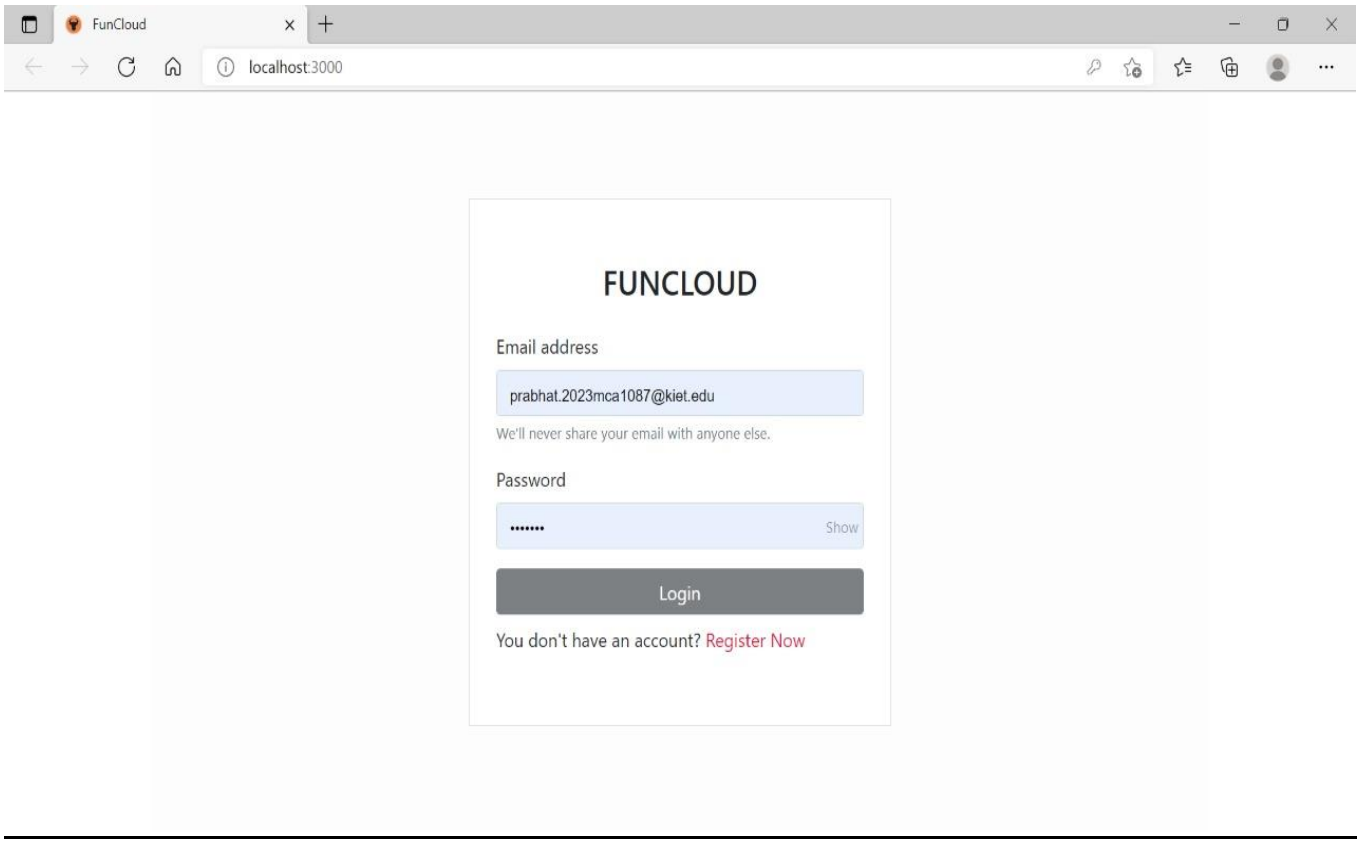
```
    useEffect(() => {
        socket.on('unLikeToClient', newPost =>{
            dispatch({type: POST_TYPES.UPDATE_POST, payload: newPost})
        })

        return () => socket.off('unLikeToClient')
    },[socket, dispatch])


    // Comments
    useEffect(() => {
        socket.on('createCommentToClient', newPost =>{
            dispatch({type: POST_TYPES.UPDATE_POST, payload: newPost})
        })

        return () => socket.off('createCommentToClient')
    },[socket, dispatch])

    useEffect(() => {
        socket.on('deleteCommentToClient', newPost =>{
            dispatch({type: POST_TYPES.UPDATE_POST, payload: newPost})
        })

        return () => socket.off('deleteCommentToClient')
    },[socket, dispatch])


    // Follow
    useEffect(() => {
        socket.on('followToClient', newUser =>{
            dispatch({type: GLOBALTYPES.AUTH, payload: {...auth, user:
newUser}})
        })

        return () => socket.off('followToClient')
    },[socket, dispatch, auth])

    useEffect(() => {
        socket.on('unFollowToClient', newUser =>{
            dispatch({type: GLOBALTYPES.AUTH, payload: {...auth, user:
newUser}})
        })

        return () => socket.off('unFollowToClient')
```

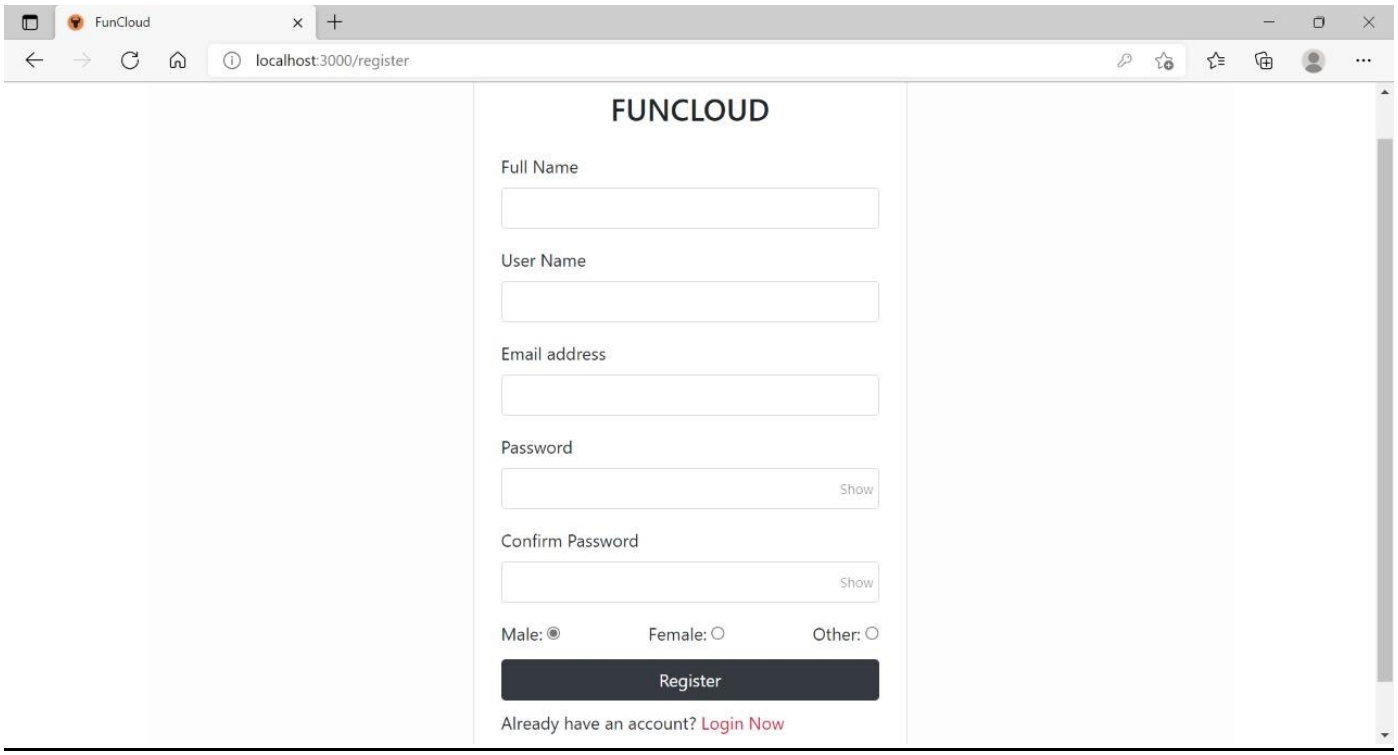
```
    },[socket, dispatch, auth])


    // Notification
    useEffect(() => {
        socket.on('createNotifyToClient', msg =>{
            dispatch({type: NOTIFY_TYPES.CREATE_NOTIFY, payload: msg})

            if(notify.sound) audioRef.current.play()
            spawnNotification(
                msg.user.username + ' ' + msg.text,
                msg.user.avatar,
                msg.url,
                'V-NETWORK'
            )
        })

        return () => socket.off('createNotifyToClient')
    },[socket, dispatch, notify.sound])

    useEffect(() => {
        socket.on('removeNotifyToClient', msg =>{
            dispatch({type: NOTIFY_TYPES.REMOVE_NOTIFY, payload: msg})
        })

        return () => socket.off('removeNotifyToClient')
    },[socket, dispatch])


    // Message
    useEffect(() => {
        socket.on('addMessageToClient', msg =>{
            dispatch({type: MESS_TYPES.ADD_MESSAGE, payload: msg})

            dispatch({
                type: MESS_TYPES.ADD_USER,
                payload: {
                    ...msg.user,
                    text: msg.text,
                    media: msg.media
                }
            })
        })
```

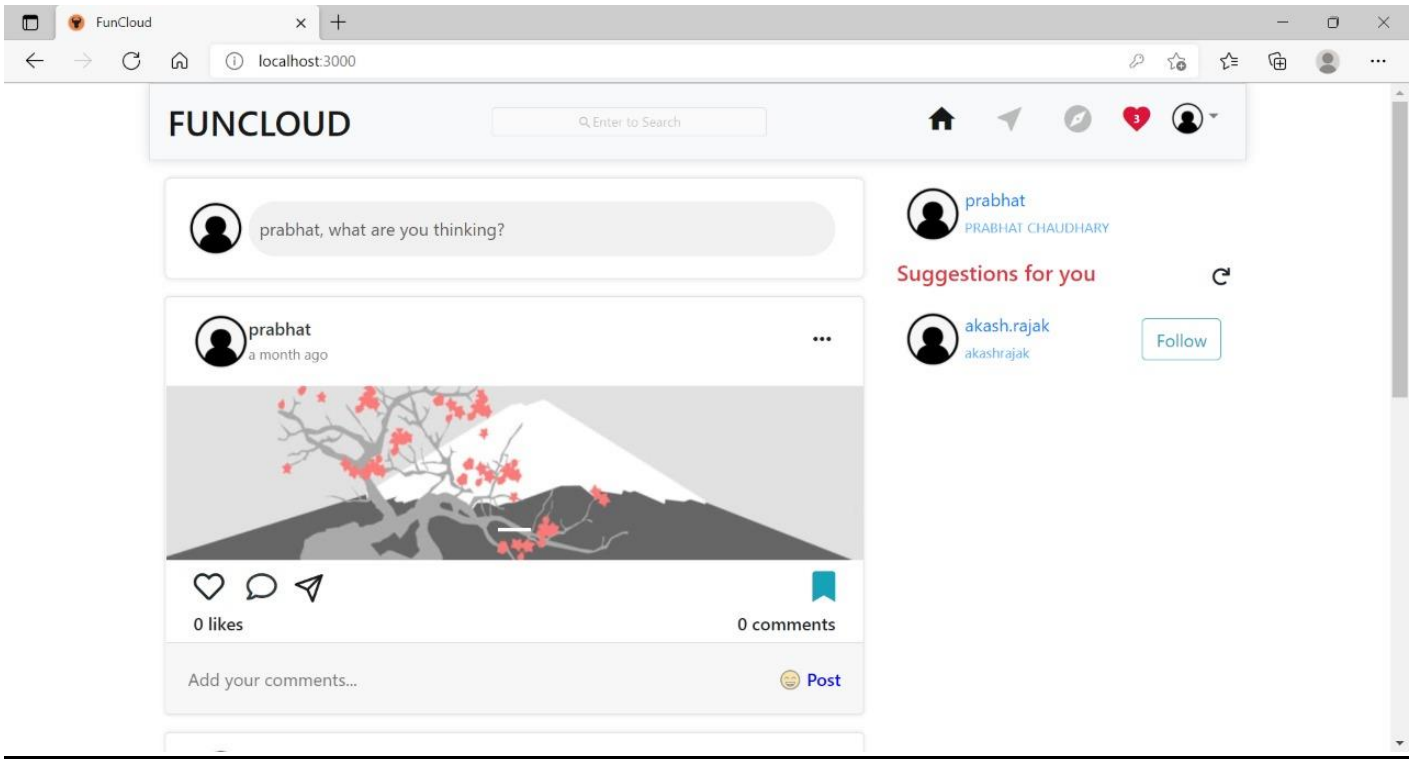
```
        return () => socket.off('addMessageToClient')
    },[socket, dispatch])

    // Check User Online / Offline
    useEffect(() => {
        socket.emit('checkUserOnline', auth.user)
    },[socket, auth.user])

    useEffect(() => {
        socket.on('checkUserOnlineToMe', data =>{
            data.forEach(item => {
                if(!online.includes(item.id)){
                    dispatch({type: GLOBALTYPES.ONLINE, payload: item.id})
                }
            })
        })

        return () => socket.off('checkUserOnlineToMe')
    },[socket, dispatch, online])

    useEffect(() => {
        socket.on('checkUserOnlineToClient', id =>{
            if(!online.includes(id)){
                dispatch({type: GLOBALTYPES.ONLINE, payload: id})
            }
        })

        return () => socket.off('checkUserOnlineToClient')
    },[socket, dispatch, online])

    // Check User Offline
    useEffect(() => {
        socket.on('CheckUserOffline', id =>{
            dispatch({type: GLOBALTYPES.OFFLINE, payload: id})
        })

        return () => socket.off('CheckUserOffline')
    },[socket, dispatch])


    // Call User
    useEffect(() => {
        socket.on('callUserToClient', data =>{
            dispatch({type: GLOBALTYPES.CALL, payload: data})
```

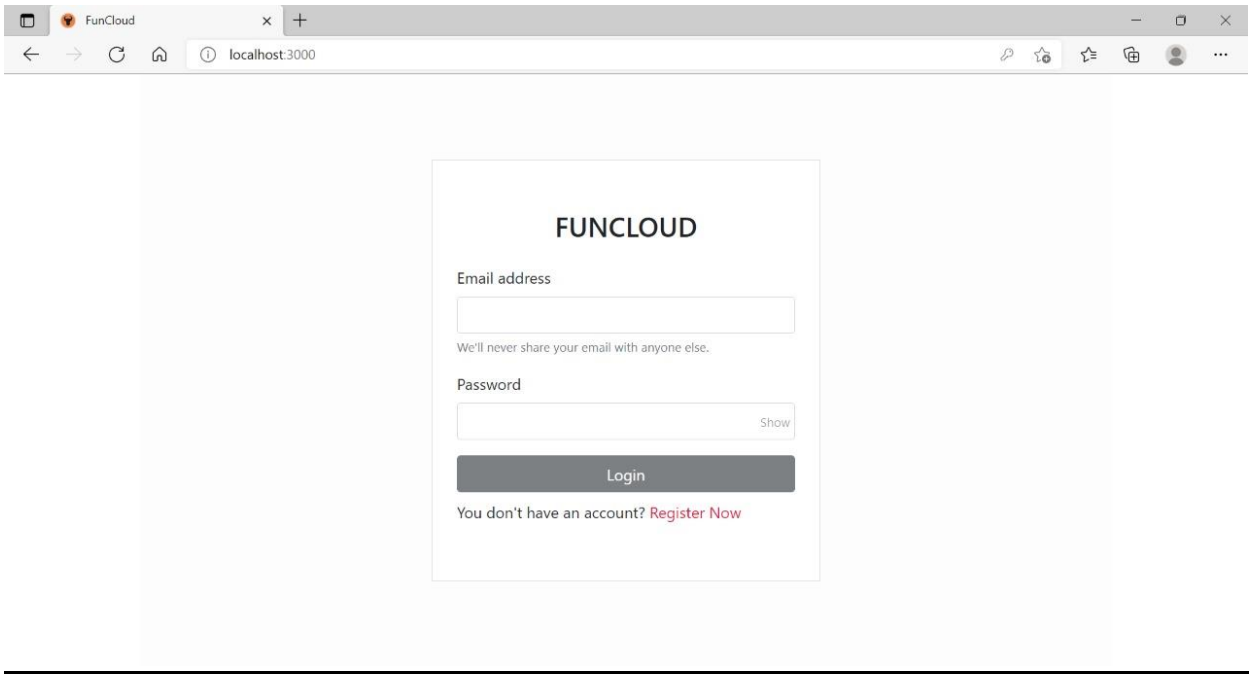
```
        })

        return () => socket.off('callUserToClient')
    },[socket, dispatch])

    useEffect(() => {
        socket.on('userBusy', data =>{
            dispatch({type: GLOBALTYPES.ALERT, payload: {error:
`${call.username} is busy!`}})
        })

        return () => socket.off('userBusy')
    },[socket, dispatch, call])



    return (
        <>
            <audio controls ref={audioRef} style={{display: 'none'}} >
                <source src={audiobell} type="audio/mp3" />
            </audio>
        </>
    )
}

export default SocketClient
```

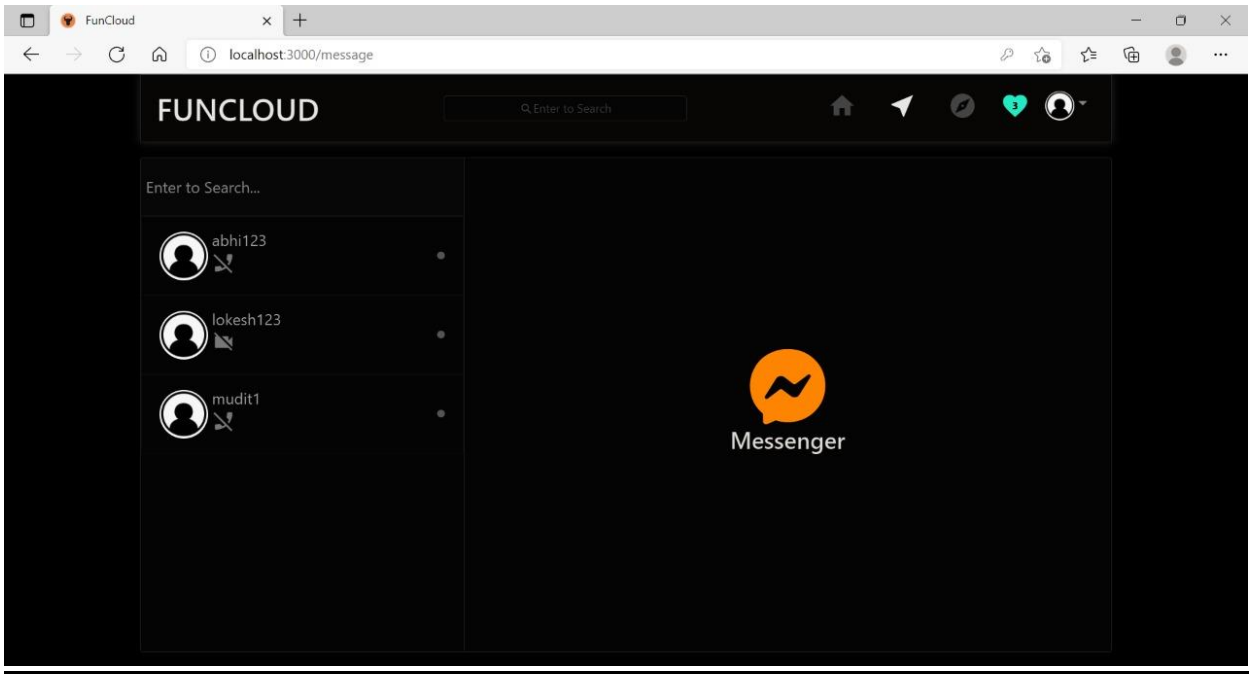
PROJECT SCREENSHOT

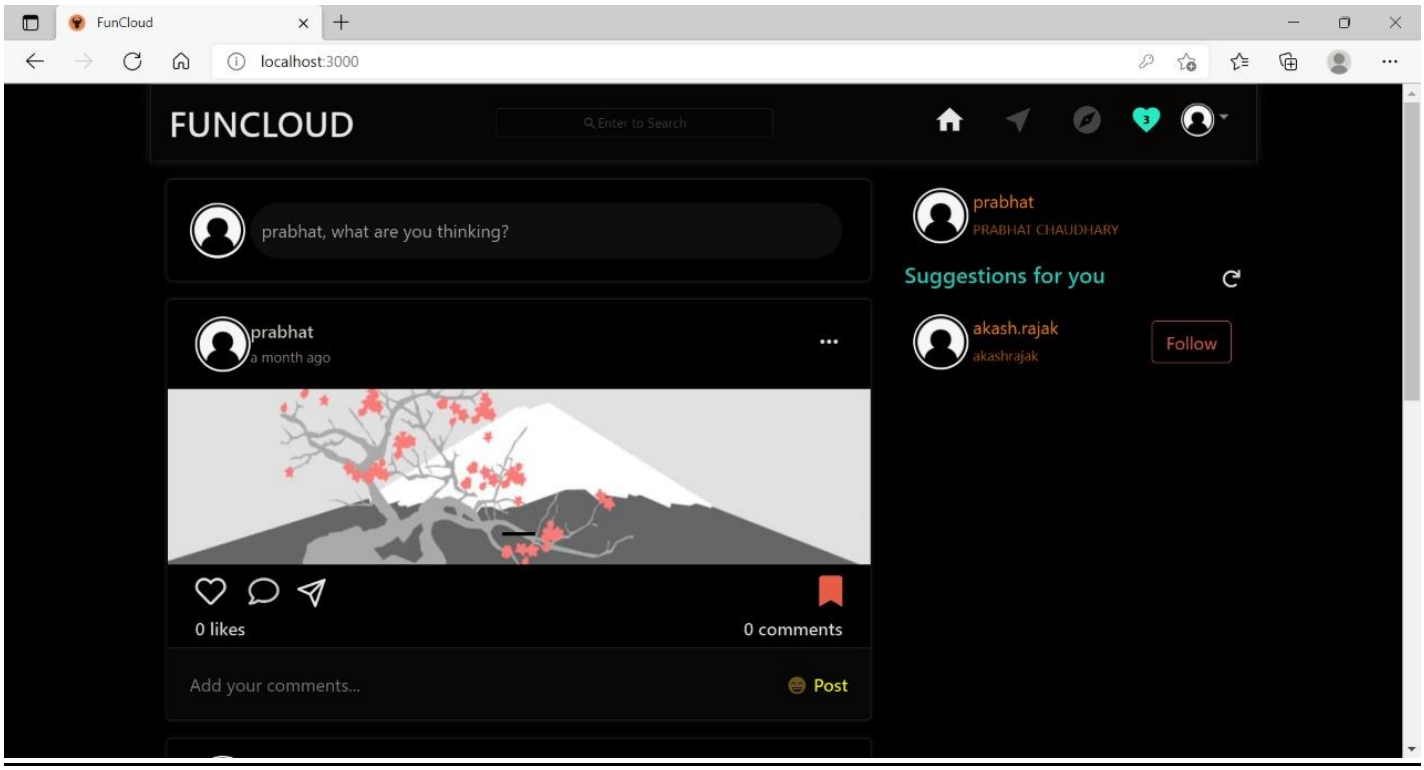


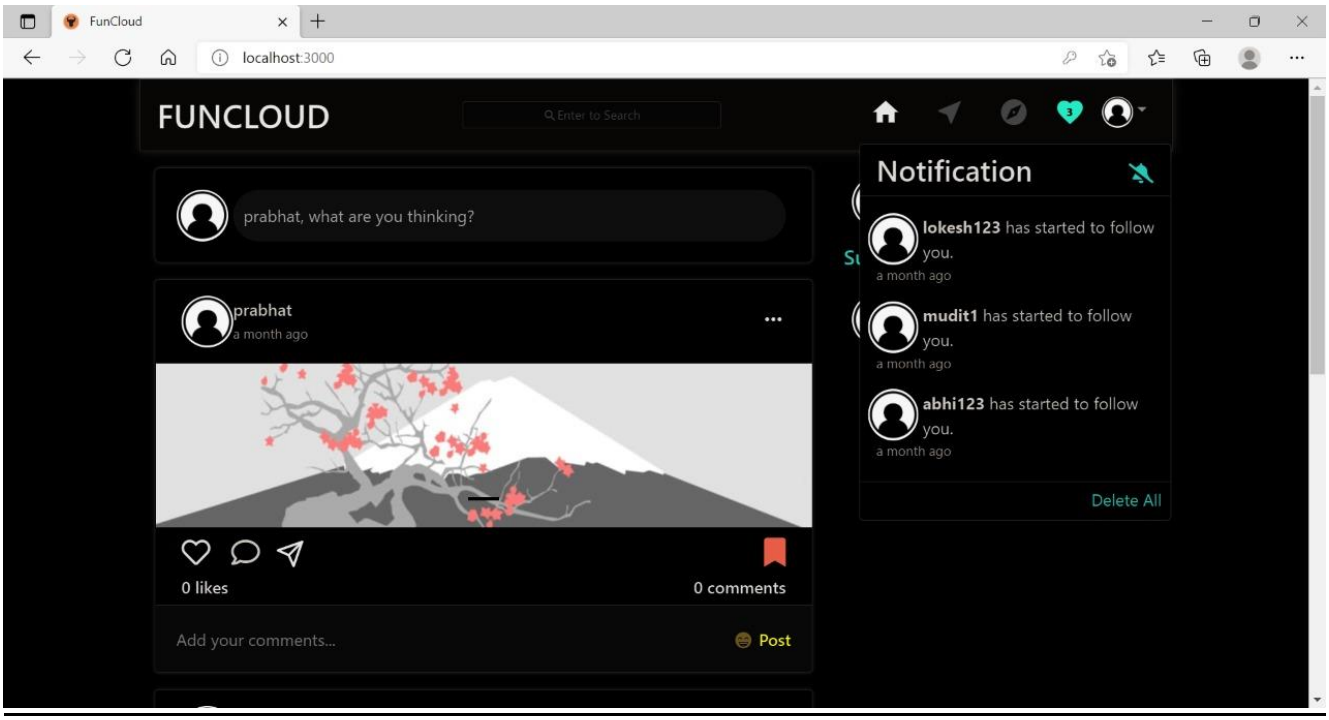

FunCloud
localhost:3000/register
FUNCLOUD
Full Name
User Name
Email address
Password
Show
Confirm Password
Show
Male:
Female:
Other:
Register
Already have an account? Login Now


FunCloud
localhost:3000
FUNCLOUD
Enter to Search
prabhat, what are you thinking?
prabhat
a month ago
0 likes
0 comments
Add your comments...
Post
prabhat
PRABHAT CHAUDHARY
Suggestions for you
akash.rajak
akashrajak
Follow


FunCloud
localhost:3000
FUNCLOUD
Email address
We'll never share your email with anyone else.
Password
Show
Login
You don't have an account? Register Now


FunCloud
localhost:3000/message
FUNCLOUD
Enter to Search
Enter to Search...
abhi123
lokesh123
mudit1
Messenger


FunCloud
localhost:3000
FUNCLOUD
Enter to Search
prabhat, what are you thinking?
prabhat
a month ago
0 likes
0 comments
Add your comments...
Post
prabhat
PRABHAT CHAUDHARY
Suggestions for you
akash.rajak
akashrajak
Follow


FunCloud
localhost:3000
FUNCLOUD
Enter to Search
prabhat, what are you thinking?
prabhat
a month ago
0 likes
0 comments
Add your comments...
Post
Notification
lokesh123 has started to follow you.
a month ago
mudit1 has started to follow you.
a month ago
abhi123 has started to follow you.
a month ago
Delete All