VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**BELAGAVI, KARNATAKA – 590018**



# MINI PROJECT REPORT ON

# “SOCIAL NETWORK”

Submitted in partial fulfillment of the requirements as a part of the **DBMS Laboratory with Mini Project(18CSL58)** for the V semester of degree of **Bachelor of Engineering in Computer Science and Engineering** of Visvesvaraya Technological University, Belagavi

# Submitted by:

**PRAJWAL V- 1JS19CS119 PRANAVA B- 1JS19CS121**

**Under the Guidance of**

## Mr. Rohitaksha K.,

Assistant Professor, Dept of CSE, JSSATE, Bengaluru

## JSS ACADEMY OF TECHNICAL EDUCATION

**Dr.Vishnuvardhan Rd, Srinivasapura Post, Bengaluru – 560060 (2021-2022)**

# JSS MAHA VIDYAPEETHA, MYSORE

JSS ACADEMY OF TECHNICAL EDUCATION

JSS Campus, Uttarahalli - Kengeri main road, Bengaluru – 560060



# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING CERTIFICATE

This is to certify that the project work entitled “**SOCIAL NETWORK”** is a bona fide work carried out by **Mr. Prajwal V (1JS19CS119)** and **Mr. Pranava B (1JS19CS121)** in partial fulfillments of the requirements for DBMS Laboratory with Mini Project (18CSL58) of V semester **Bachelor of Engineering in Computer Science and Engineering** of the academic year 2021-2022. It is certified that all the corrections and suggestions indicated for Internal Assessments have been incorporated in the report deposited in the department Library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

## Mr. Rohitaksha K,

Assistant Professor, Dept of CSE JSSATE, Bengaluru

## Dr. Naveen N.C,

Professor and Head, Dept of CSE JSSATE, Bengaluru

## Name of the Examiners: Signature with Date:

**1. ………………………… ……………………………**

**2. ………………………… ……………………………**

# ABSTRACT

A Database Management System (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between the end user and a database, allowing the users to create, read, update, and delete data in the database.

This project, **SOCIAL NETWORK**

We express our humble pranams to His Holiness **Jagadguru Sri Sri Sri Shivaratri Deshikendra mahaswamiji** for showering his blessings on us to receive good education and have a successful career.

The completion of any project involves the effort of many people. We have been very lucky to have received support and guidance from all kinds of sources to complete our project. So, we take this opportunity to express our gratitude to all those who gave us guidance and encouragement to successfully complete this project.

We express a sincere thanks to our beloved Principal **Dr. Mrityunjaya V Latte** for having supported us in all academic endeavours.

We are also forever grateful to **Dr. Naveen N.C** Head of Department, Computer science and Engineering for his unending support, guidance, and encouragement in all are ventures.

We are very thankful for the resourceful guidance and timely assistance and graceful gesture of our guide **Mr. Rohitaksha K** Assistant Professor Department of Computer Science and Engineering who has helped us in every aspect of our project work.

Last but not the least, we would be immensely pleased to express our heartfelt thanks to all the teaching and non-teaching staff of the Department of CSE and our friends for the timely help support and guidance.

# PRAJWAL V

# PRANAVA B

**Chapter Title Page No.**

Abstract i

Acknowledgement ii

Table of Contents iii

List of Figures iv

[Chapter 1: Preamble](#_TOC_250019)

* 1. [Introduction 1](#_TOC_250018)
     1. [VSCode Code editor 1](#_TOC_250017)
     2. Server – Nodejs/Express , EJS……………………………………….2
     3. Javascript, HTML/CSS 2
     4. [Database – MongoDB 2](#_TOC_250016)
     5. MongoDB Compass 3
  2. [Objectives 4](#_TOC_250015)
  3. Organization of Report 4
  4. [Summary… 5](#_TOC_250014)

Chapter 2: Requirements Specification

* 1. [Software Specification 6](#_TOC_250013)
  2. [Hardware Specification 6](#_TOC_250012)
  3. [User Characteristics 6](#_TOC_250011)

[Chapter 3: Database Design and Implementation](#_TOC_250010)

* 1. [Introduction 7](#_TOC_250009)
  2. [ER Diagram 8](#_TOC_250007)
  3. [Schema Diagram 9](#_TOC_250006)
  4. [Implementation 10](#_TOC_250005)
     1. Collections 10
     2. [Pseudo Code 15](#_TOC_250004)
     3. [Queries… 17](#_TOC_250003)

[Chapter 4: Results and Discussions 18](#_TOC_250002)

[Chapter 5: Conclusion and Future Enhancements 26](#_TOC_250001)

[References… 26](#_TOC_250000)

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **Figure No** | **Name of Figure** | **Page No.** |
| Fig: 3.2 | ER Diagram | 8 |
| Fig: 3.3 | Schema Diagram | 9 |
| Fig: 3.4 | Description of Users Collection | 11 |
| Fig: 3.5 | Description of Posts collection | 12 |
| Fig: 3.6 | Description of Comment collection | 13 |
| Fig: 3.7 | Description of FriendsList collection | 13 |
| Fig: 4.1 | First/Welcome Page | 18 |
| Fig: 4.2 | About us Page | 18 |
| Fig: 4.3 | Admin Login Page | 19 |
| Fig: 4.4 | Admin Operations | 19 |
| Fig: 4.5 | Add new Student | 20 |
| Fig: 4.6 | View all Registered Student | 20 |
| Fig: 4.7 | Insert New Result | 21 |
| Fig: 4.8 | View all Student result | 21 |
| Fig: 4.9 | Delete Student details/Result | 22 |
| Fig: 4.10 | Failed Students | 22 |
| Fig: 4.11 | Backlog Student entry | 23 |
| Fig: 4.12 | View Backlog Students | 23 |
| Fig: 4.13 | Student Login | 24 |
| Fig: 4.14 | Result Page of Student 1 | 24 |
| Fig: 4.15 | Result Page of Student 2 | 25 |

# Chapter 1: Preamble

# Introduction

A database is an organised collection of data. A relational database more restrictively is a collection of schemas, tables, queries, reports, views, and other elements. A Database Management System (DBMS) is a computer software application that interacts with end users, other applications, and the database itself to capture and analyse the data. A general-purpose DBMS allows the definition, creation, querying, update, and administration of databases

# VSCode Code Editor

# Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++. Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

# Server -Node Js:

**Node Js:**

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

**Express Js:**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework −

• Allows to set up middlewares to respond to HTTP Requests.

• Defines a routing table which is used to perform different actions based on HTTP Method and URL.

Allows to dynamically render HTML Pages based on passing arguments to templates

**EJS (Embedded Javascript Templating):**

is one of the most popular template engines for JavaScript. As the name suggests, it lets us embed JavaScript code in a template language that is then used to generate HTML.

Initialize a new Node project in the folder by running **npm init -y**

then to install Express and EJS, run: **npm install express ejs**

to run server run : **npm run** **index.js**

# Javascript , HTML/CSS

**Javascript :**

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

**H****TML:**

HTML stands for **Hyper Text Markup Language** HTML is the standard markup language for creating Web pages HTML describes the structure of a Web page HTML consists of a series of elements HTML elements tell the browser how to display the content HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

# CSS:

CSS stands for **Cascading Style Sheets**. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files.

# Database MongoDB

# MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

# Collection

# Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

# Document

# A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

# MongoDB Compass

MongoDB Compass is a powerful GUI for querying, aggregating, and analyzing your MongoDB data in a visual environment , Compass is an interactive tool for querying, optimizing, and analyzing your MongoDB data. Get key insights, drag and drop to build pipelines, and more.



# Objectives

The main objective of Social Network is:

//few points here

# Organization of the Report

Chapter 1 provides the information about the basics of NodesJs Javascript html MongoDB.

Chapter 2 we discuss the software and hardware requirements to run the above applications. Chapter 3 gives the idea of the project and its actual implementation.

Chapter 4 discusses about the results and discussions of the program

Chapter 5 concludes by giving the direction for future enhancement and the reference

# Summary

The chapter discussed before is an overview about the Nodejs/Express, javascript, HTML, CSS, and MongoDB Database. The scope of study and objectives of the project are mentioned clearly. The organization of the report has been pictured to increase the readability. Further, coming up chapters depicts the use of various queries to implement various changes like insert, update, delete and triggers to perform various functions

# Chapter 2: Requirement Specifications

# Software Specification

* Operating System : Windows 10/windows 11
* Front End : Javascript, HTML, CSS
* Back End : Nodejs/Express .
* Database : MongoDB
* Tools : VSCode , mongoDB compass

# Hardware Specification

* Processor: x86 compatible processor with 1.7 GHz Clock Speed
* RAM: 512 MB or greater
* Hard Disk: 20 GB or greater
* Monitor: VGA/SVGA
* Keyboard: 104 keys standard
* Mouse: 2/3 button. Optical/Mechanical.

# User Characteristics

Every user:

* + - Should be comfortable with basic working of the computer
    - Must have basic knowledge of English
    - should have Email ID and a unique username inorder to login into social network .

# Chapter 3: Database Design and Implementation

# Introduction

Database design is the process by which an agent creates a specification of a database

artifact intended to accomplish goals, using a set of primitive components and subject

to constraints. Database design usually involves problem – solving and planning a

software solution.

A wide range of tools are used to design the Social Network database are

• VSCode

Used to create and edit the source code

• Nodejs

Used to create server and to install npm dependencies

• mongoDB compass

Used to store data and handle transactions

Project is implemented in NodeJs, on a Windows 10 based computer. NodeJs is used

within VSCode., which provides integrated terminal to handle server, installing npm dependencies graphical UI for debugging and other features .

# Schema diagram

//schema here

# Implementation

In this section, we will discuss about the creation of collection insertion of values and the Queries that we have implemented in our application program.

## Collections :

In MongoDB collection are created when we insert first document inside it we don’t need

to manually create collection prior of insertion operation

Below are the collection schema that we have created/ in our project .

#### Collection: Users

Fields :

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | | Data type |
|  |  | \_id | Id |
|  |  | name | String |
|  |  | username | String |
|  |  | email | String |
|  |  | password | String |
|  |  | gender | String |
|  |  | reset\_token | String |
|  |  | profileImage | String |
|  |  | coverPhoto | String |
|  |  | dob | String |
|  |  | city | String |
|  |  | country | String |
|  |  | aboutMe | String |
|  |  | accessToken | String |

Linked from

|  |  |  |
| --- | --- | --- |
| Collection | | Join |
|  | [comment](#_bookmark3) | **users**\_id = comment.comments.user.\_id |
|  | [friendsList](#_bookmark4) | **users**\_id = friendsListuser\_id |
|  | [friendsList](#_bookmark4) | **users**\_id = friendsList.friends.f\_id |
|  | [notification](#_bookmark5) | **users**\_id = notification.notifications.\_id |
|  | [notification](#_bookmark5) | **users**\_id = notificationuser\_id |
|  | [posts](#_bookmark6) | **users**\_id = posts.user.\_id |
|  | [posts](#_bookmark6) | **users**\_id = posts.likers.\_id |

Unique keys

|  |  |
| --- | --- |
| fields | |
|  | \_id |
|  | email |
|  | username |

#### Collection: Posts

fields

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | | Data type |
|  |  | \_id | Id |
|  |  | caption | String |
|  |  | image | String |
|  |  | type | String |
|  |  | createdAt | Double |
|  |  | likers | Document[] |
|  |  | likers.\_id | Id |
|  |  | likers.name | String |
|  |  | likers.profileImage | String |
|  |  | user | Document |
|  |  | user.\_id | Id |
|  |  | user.name | String |
|  |  | user.username | String |
|  |  | user.profileImage | String |

Links to

|  |  |  |
| --- | --- | --- |
| Collection | | Join |
|  | [users](#_bookmark7) | **posts**.user.\_id = users\_id |
|  | [users](#_bookmark7) | **posts**.likers.\_id = users\_id |

Linked from

|  |  |  |
| --- | --- | --- |
| Collection | | Join |
|  | [comment](#_bookmark3) | **posts**\_id = commentpost\_id |
|  | [notification](#_bookmark5) | **posts**\_id = notification.notifications.post.\_id |

#### Collection: friendsList

fields

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | | Data type |
|  |  | \_id | Id |
|  |  | user\_id | Id |
|  |  | friends | Document[] |
|  |  | friends.f\_id | Id |
|  |  | friends.name | String |
|  |  | friends.profileImage | String |
|  |  | friends.status | String |
|  |  | friends.sentByMe | Boolean |

Links to

|  |  |  |
| --- | --- | --- |
| Collection | | Join |
|  | [users](#_bookmark7) | **friendsList**user\_id = users\_id |
|  | [users](#_bookmark7) | **friendsList**.friends.f\_id = users\_id |

#### Collection: comment

fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | | Data type |  |
|  |  | \_id | Id |
|  |  | post\_id | String |
|  |  | comments | Document [] |
|  |  | comments.user | Document |
|  |  | comments.user.\_id | Id |
|  |  | comments.user.name | String |
|  |  | comments.user.profileImage | String |
|  |  | comments.comment | String |
|  |  | comments.createdAt | Double |

Links to

|  |  |  |  |
| --- | --- | --- | --- |
| Collection | | Join |  |
|  | [posts](#_bookmark6) | **comment.**post\_id = posts\_id | |
|  | [users](#_bookmark7) | **comment**.comments.user.\_id = users\_id | |

#### Collection: notification

fields

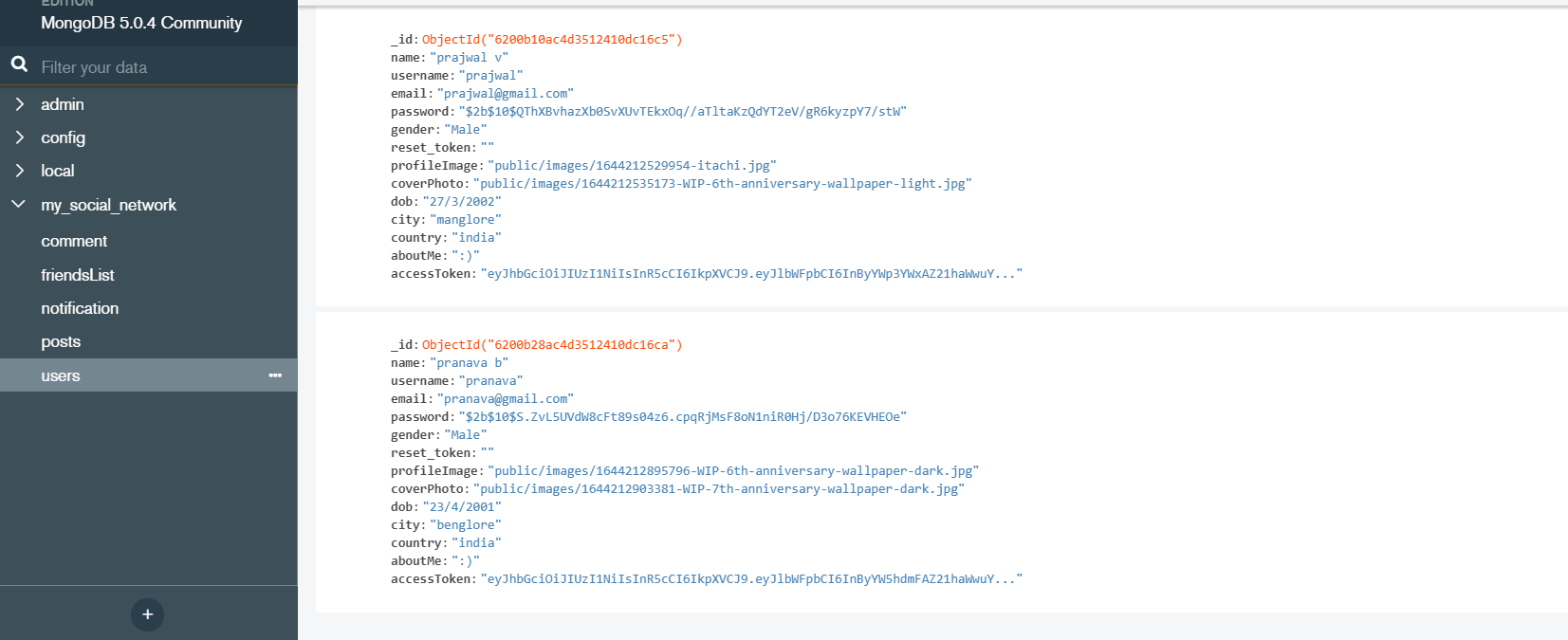
|  |  |  |  |
| --- | --- | --- | --- |
| Name | | | Data type |
|  |  | \_id | Id |
|  |  | user\_id | Id |
|  |  | notifications | Document[] |
|  |  | notifications.\_id | Id |
|  |  | notifications.type | String |
|  |  | notifications.username | String |
|  |  | notifications.content | String |
|  |  | notifications.profileImage | String |
|  |  | notifications.isRead | Boolean |
|  |  | notifications.createdAt | Double |
|  |  | notifications.post | Document |
|  |  | notifications.post.\_id | Id |

Links to

|  |  |  |
| --- | --- | --- |
| collection | | Join |
|  | [posts](#_bookmark6) | **notification**.notifications.post.\_id = posts\_id |
|  | [users](#_bookmark7) | **notification**.notifications.\_id = users\_id |
|  | [users](#_bookmark7) | **notification**user\_id = users\_id |

**Inserted values :**

Users collection :



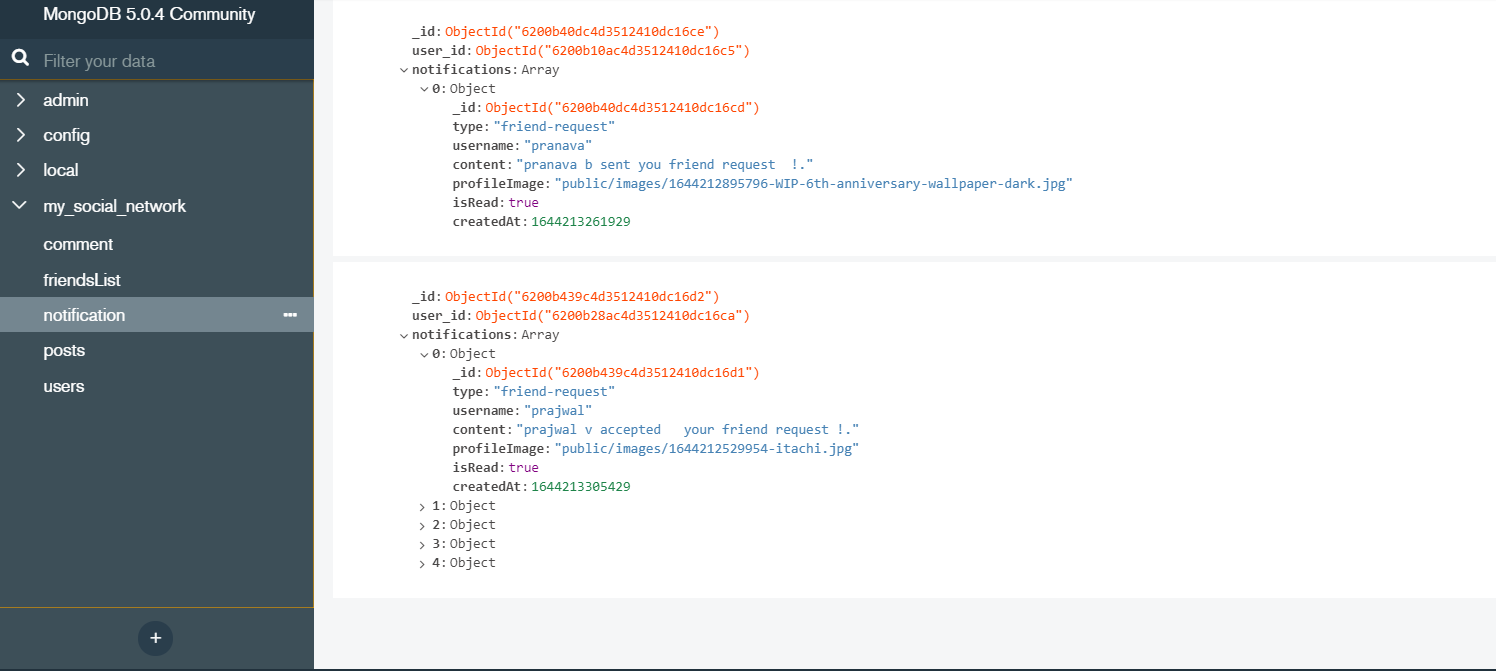
Posts collection :



Comment collection :



Notification collection :



friendsList collection :



## Pseudo Code :

Pseudocode is an informal high-level description of the operating principle of a computer program or

other algorithm. It uses the structural conventions of a normal programming language, but is intended

for human reading rather than machine reading.

Here are some of the algorithms used to implement social network application :

➢ Algorithm for User Login :

1. Begin
2. Fetch credentials from text boxes
3. Verifying credentials with values in database
4. If verified then redirect user to home page
5. Else display message to create new account
6. End

➢ Algorithm for creating User account :

1. Begin
2. Fetch credentials from input text boxes
3. Check weather username and email already exists in database
4. If exists then display message ‘username or email already exists’
5. Else insert credentials into database and redirect user to home page
6. End

➢ Algorithm for updating profile information :

1. Begin
2. Fetch user details from input text box
3. Find the user document of currently logged in user
4. Then update that user document with fetched value
5. End

➢ Algorithm for adding post :

1. Begin
2. Fetch input text and image fields value
3. If fetched values are not null/empty
4. Then add into database
5. End

➢ Algorithm for deleting post :

1. Begin
2. Fetch the id of post when user clicks delete
3. Find the corresponding post and delete it
4. End

➢ Algorithm for sending friend request and accepting :

1. Begin
2. Get the id of the friend when request is sent by user
3. Add a new friend in friendsList of both the users
4. If friend request is accepted by friend
5. Then update friendsList
6. End

➢ Algorithm for notification :

1. Begin
2. When friend like,comment,or send/accepts friend request
3. Add new notification to user in database
4. End

# Basic Queries :

Query is a request for data or information from a collection or combination of collections

Below are the some of the basic queries that we perform in mongodb

➢ inserting document in collection :

Database.collection(<collection name>).insertOne(<document>);

Database.collection(<collection name>).insertMany([document1,document2]);

➢ updating document in collection :

Database.collection(<collection name>).updateOne({query} ,{updateValues},{options});

Database.collection(<collection name>).updateMany({query},{updateValues},{options});

Different operators are can be used to update values few of them are :

$set : used to set fields in document

$inc : increments value in field

$setOnInsert : used to give default value on inserting

$unSet : removes fields

$avg : calculates the average

➢ Deleting documents in collection :

Database.collection(<collection name>).delete({query});

Database.collection(<collection name>).deleteMany({query});

Different query operators available in mongodb are :

$gt : matches the values strictly greater then given value

$eq : matches the values exactly equal to given value

$lt : matches the values strictly less than to given value

$in :matches the values in given array of values

$nin : matches the values that are not in array

# 

# Chapter 4:Observation And Results

The project is compiled and executed in VSCode and mongodb compass below are the few

Snapshots of working application ,server started in localhost at port 3000 .

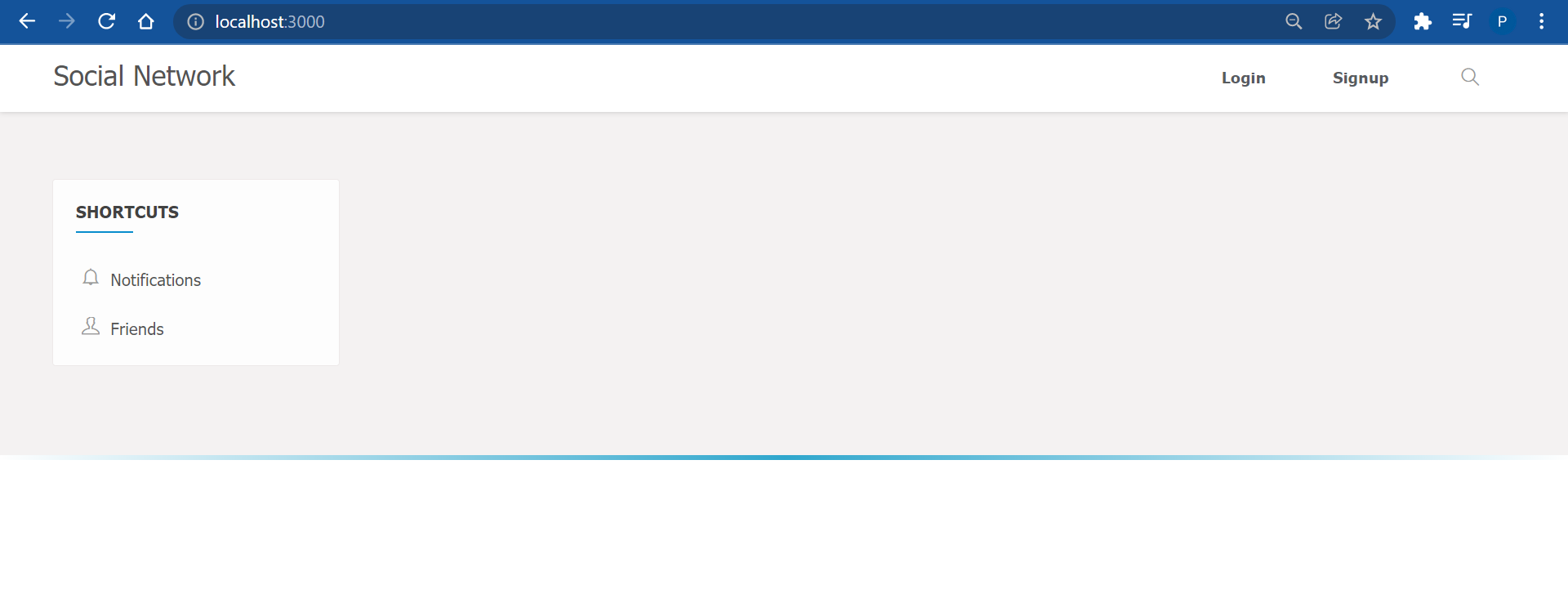


Fig 4.1(first page)

Fig 4.1 is the first user will see when they visit application for the first time they can login through

Login button or they can create their account if they don’t have an by singup button on right top

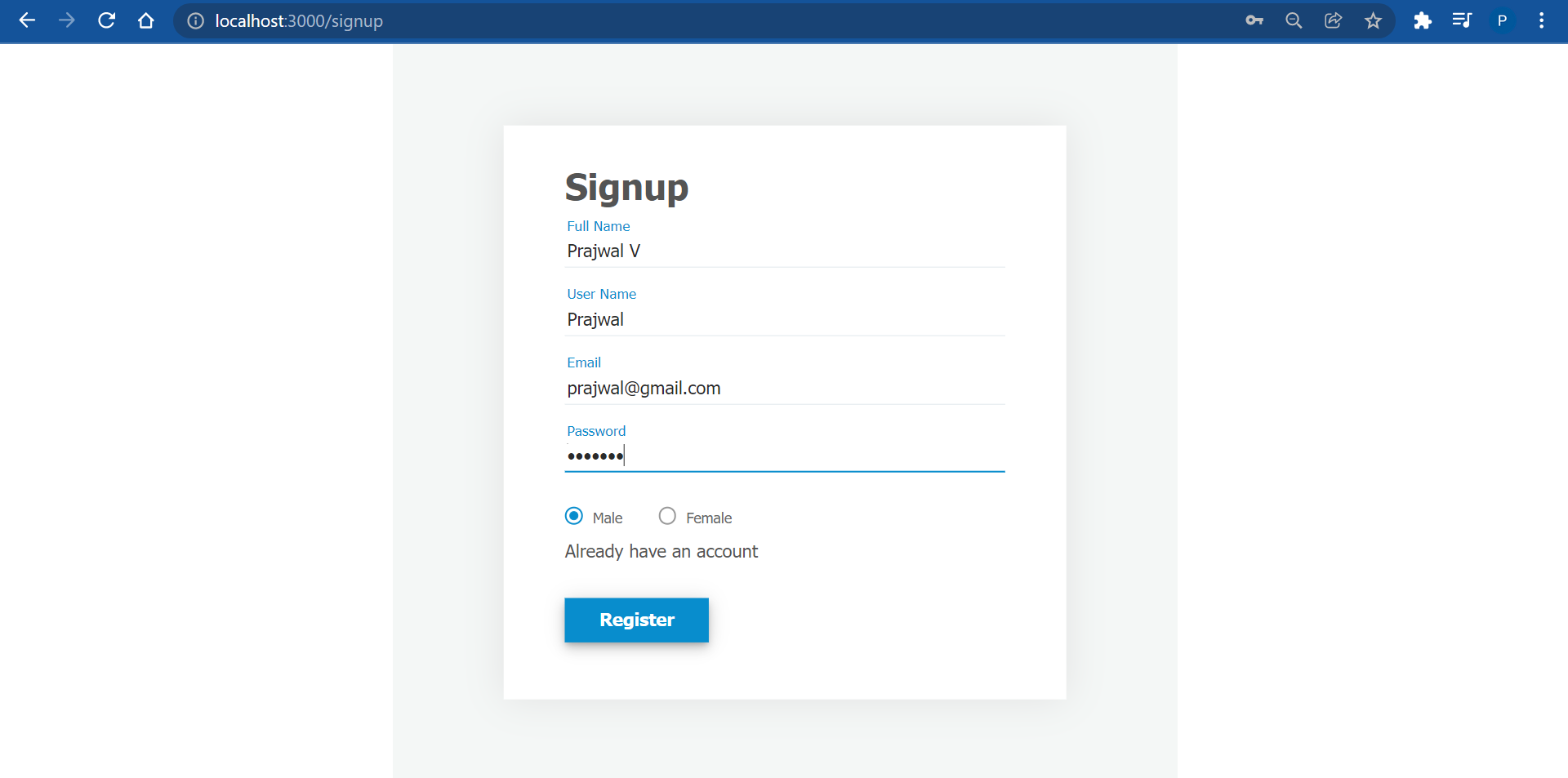


Fig 4.2 (sign up page)

Fig 4.2 is the singup page where user can create their account by having unique email and usename

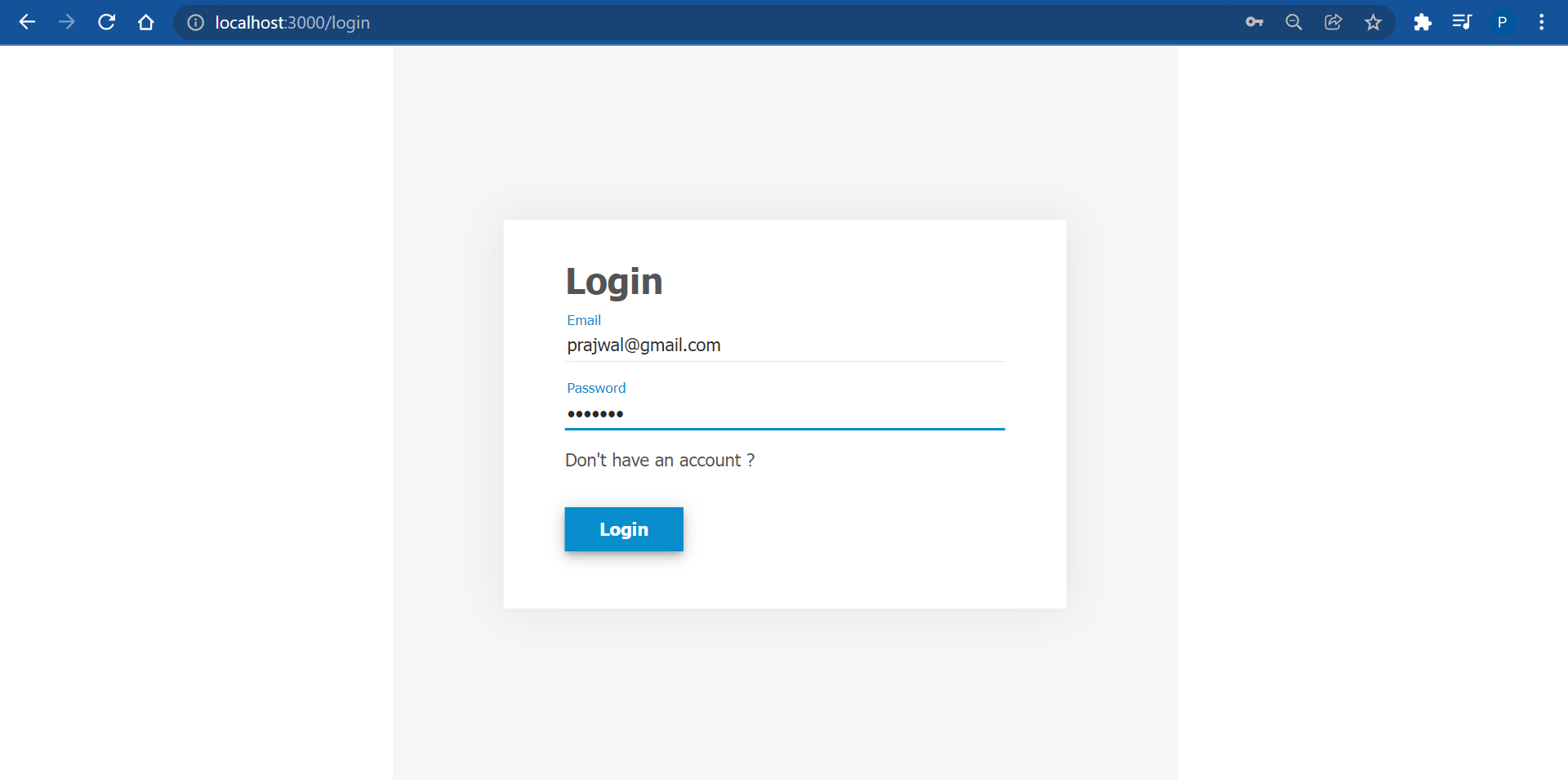


Fig 4.3 (user log in page)

In login page user enter their email and password on successful authentication they will be Redirected

to home page otherwise they can create new account and login.

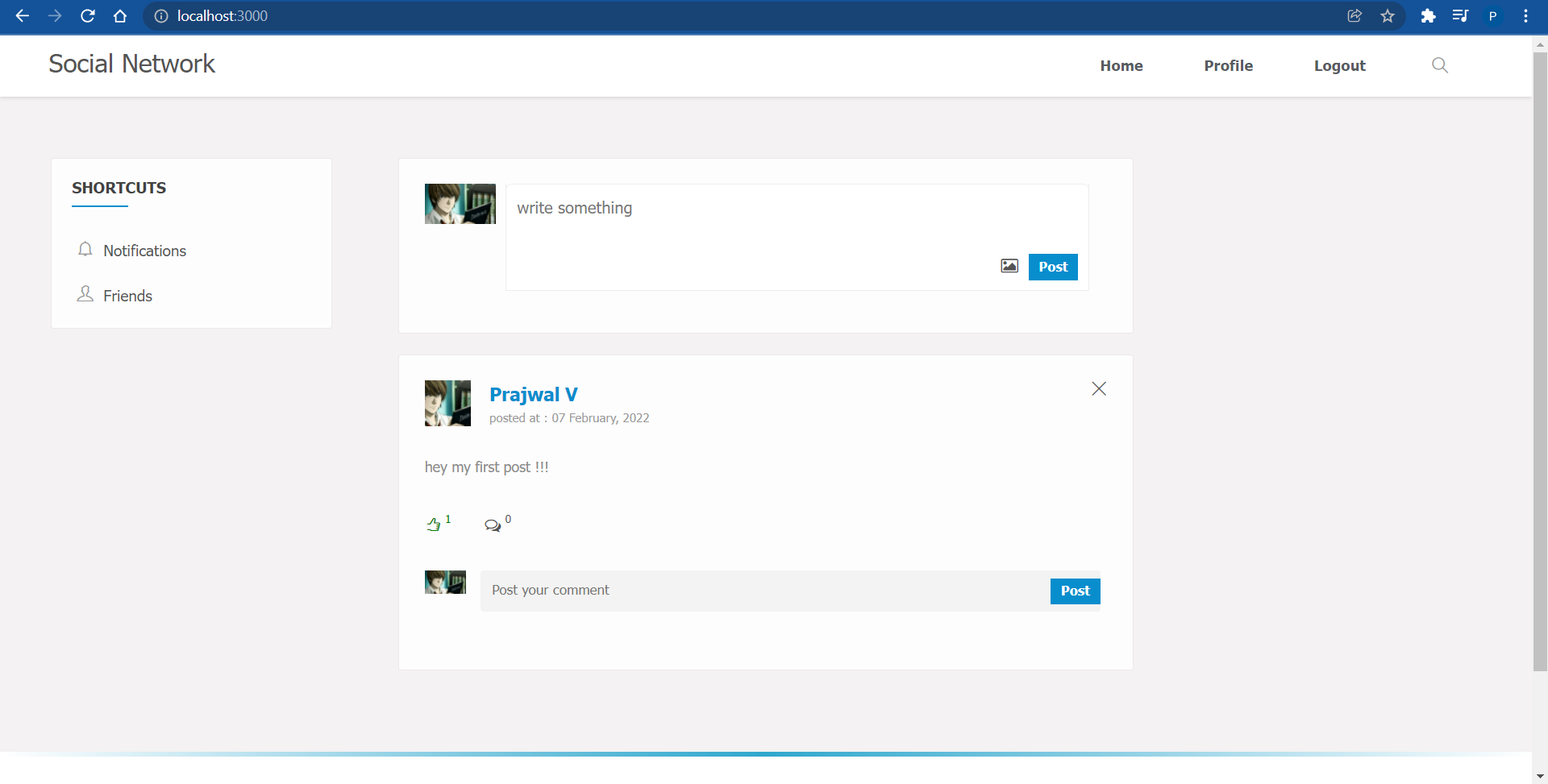


Fig 4.4 (home page)

On successful log in user will be redirected to home page where they can post , comment ,

like , view profile and Seach other users and also can see their notifications and friends list

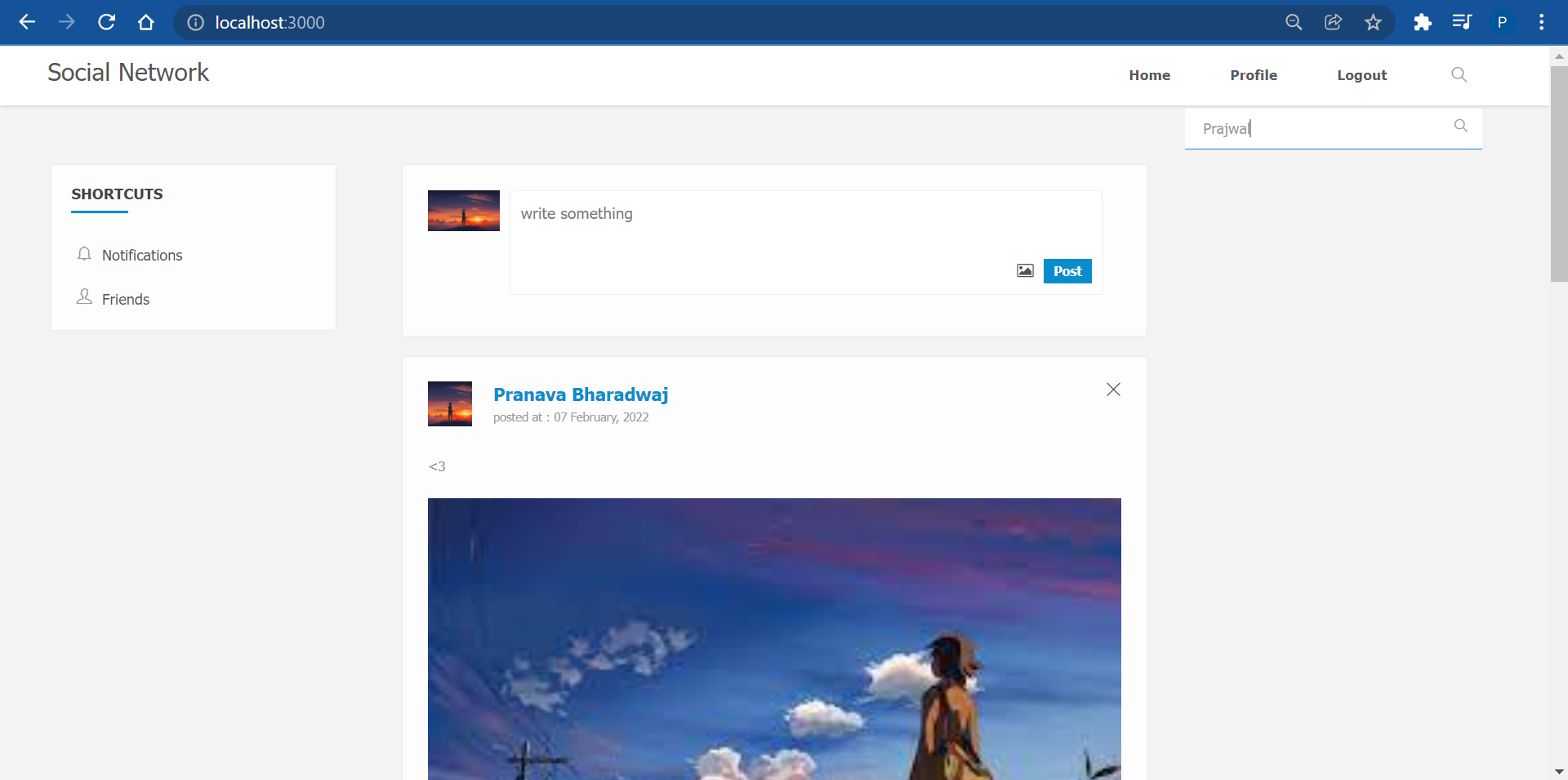


Fig 4.5 ( search friend)

In fig 4.5 currently logged in user can search any user by typing their username in search box

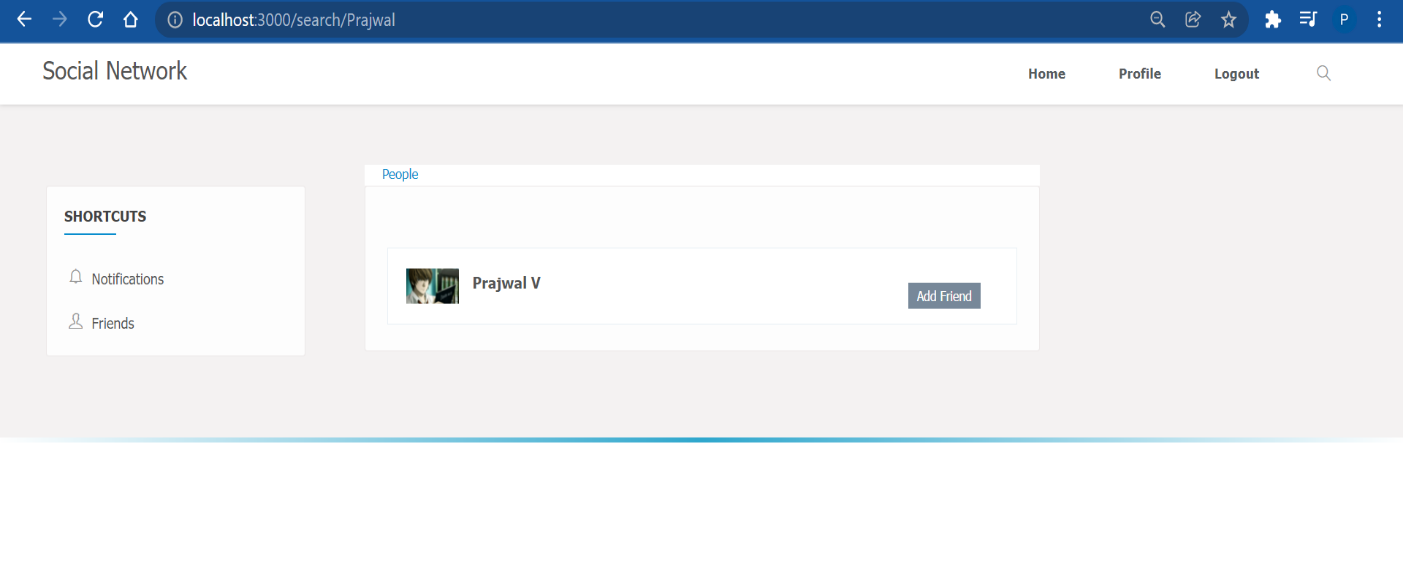


Fig 4.6 ( search results)

User will see list of others peoples profile if the profile is already friend with user they can

Unfriend them or else they add send friend request

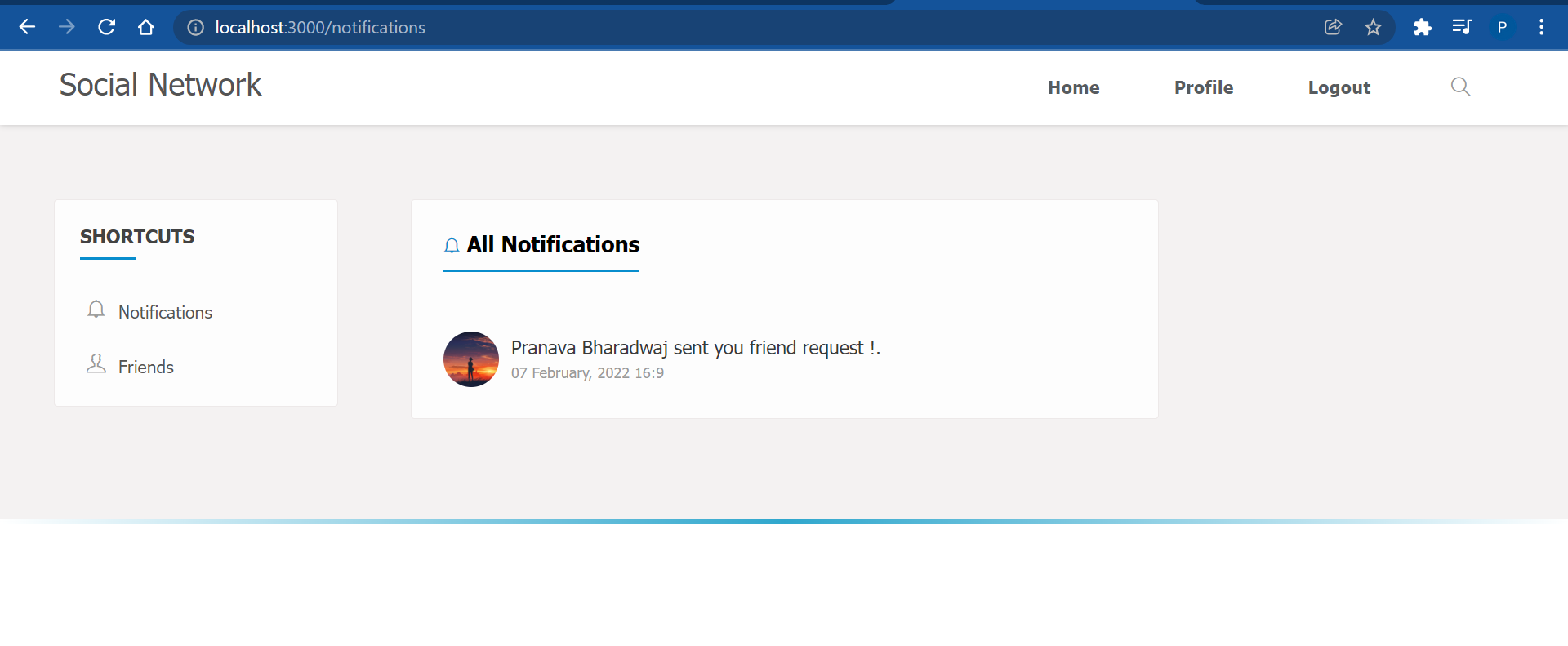


Fig 4.7 (user receiving notification)

In fig 4.7 we can user has received friend request sent by other user , so user can then move

to friends tab and accept or decline their friend request .

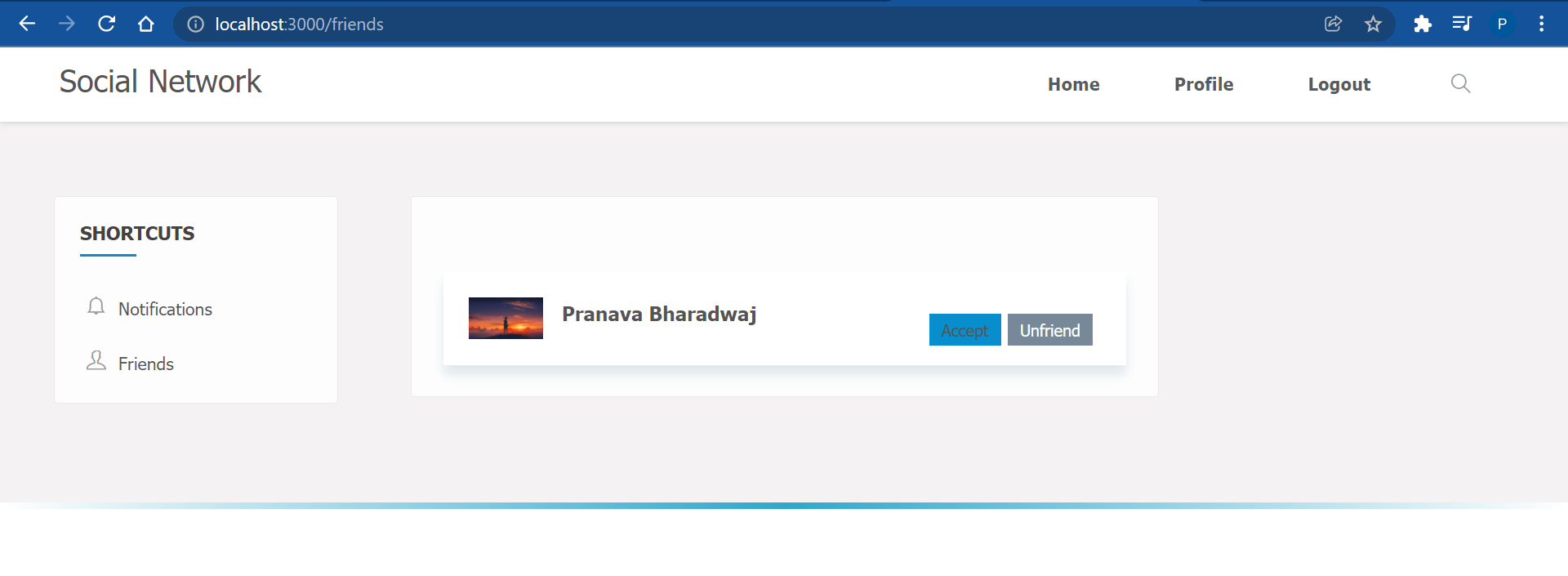


Fig 4.8 (accept or decline friend request)

In fig 4.8 we can see now user can accept or decline friend request he has receive if he accepts

new friend will be added his friendsList collection and sender will also receive notification

that user has accepted his friend request .

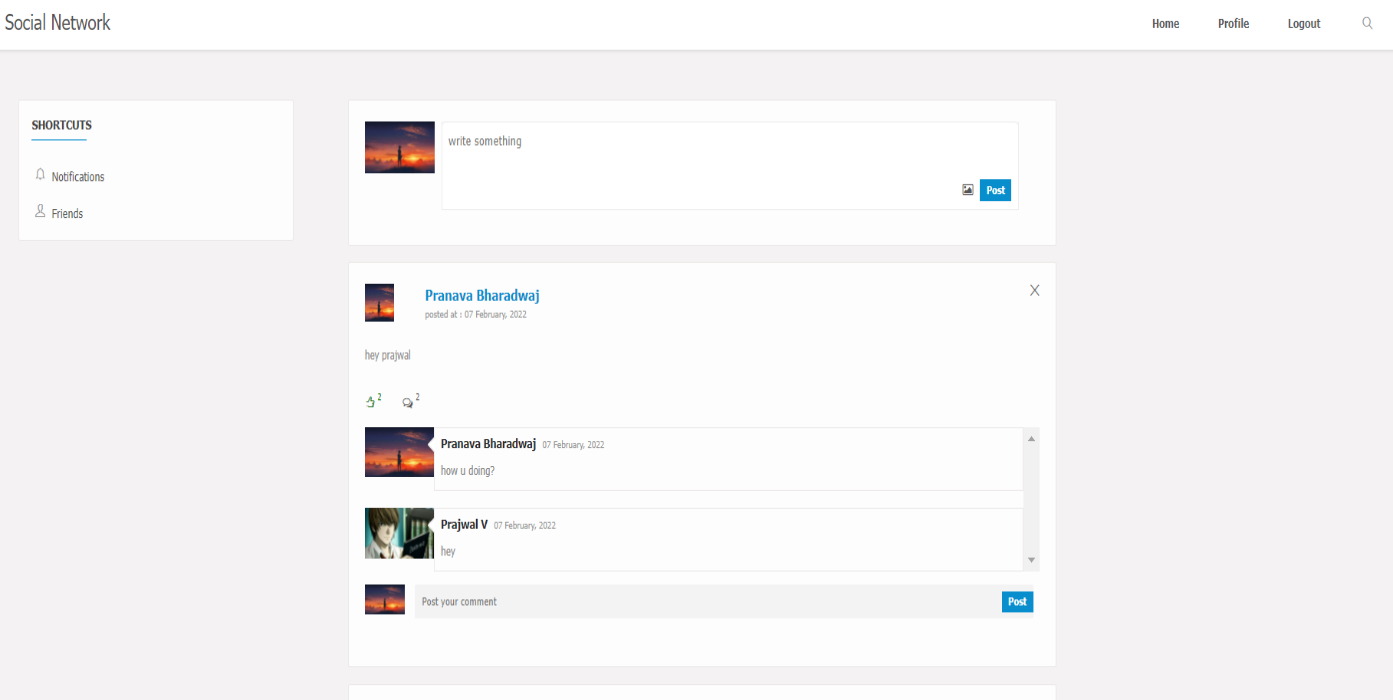


Fig 4.9 (comment and like post)

Once the 2 user are friends with each other they can see each others post and also like or

comment on each others posts .

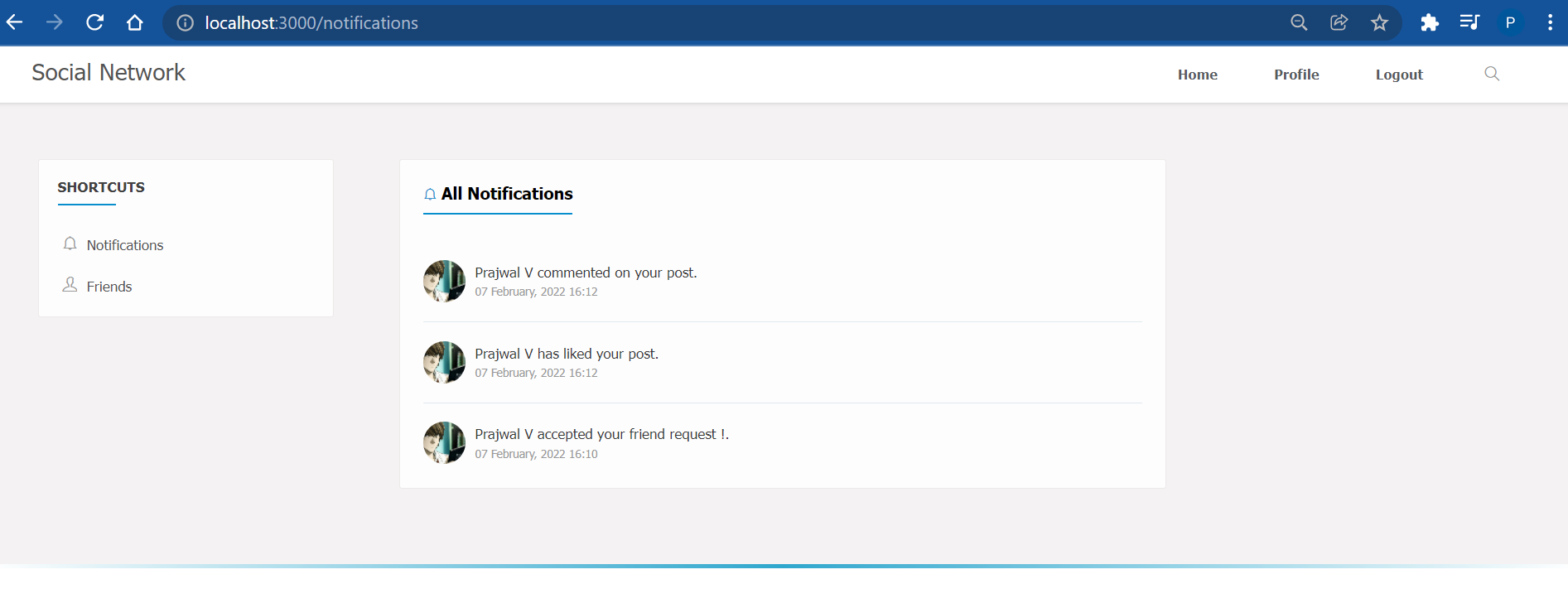
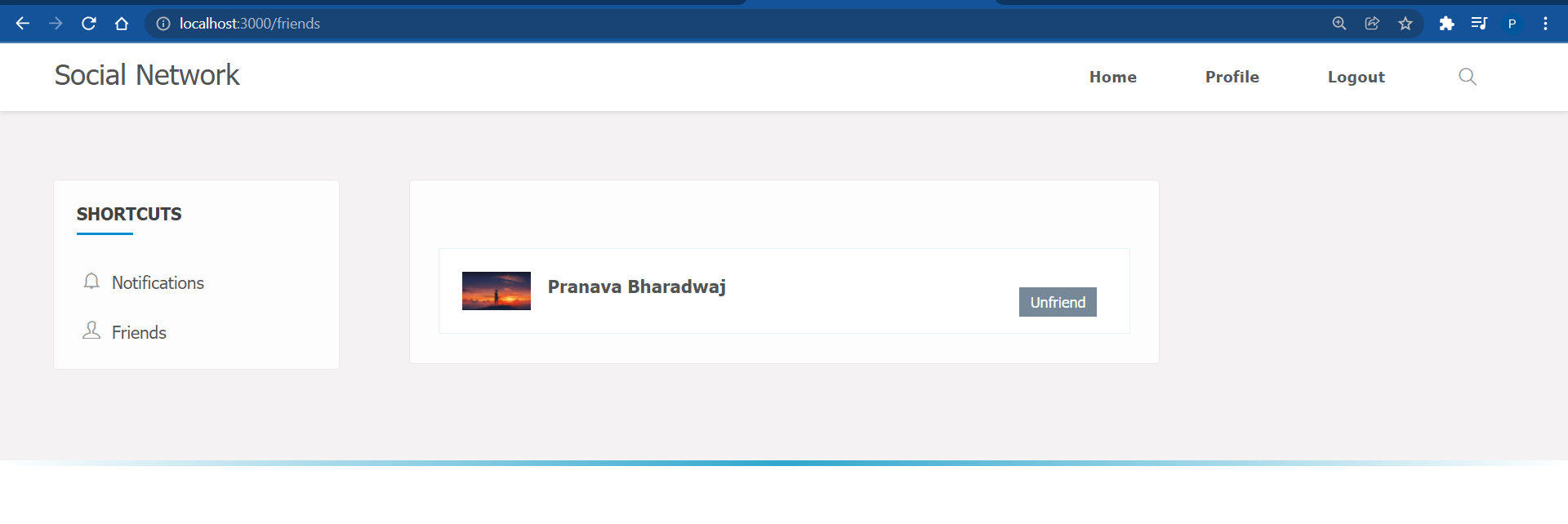


Fig 4.10 (like and comment notification)

In fig 4.10 we can see user will receive notifications when his friend likes and

comments on his post .



4.11(remove friends)

User can remove any of his friend by clicking unfriend button on friends profile in friends

tab .

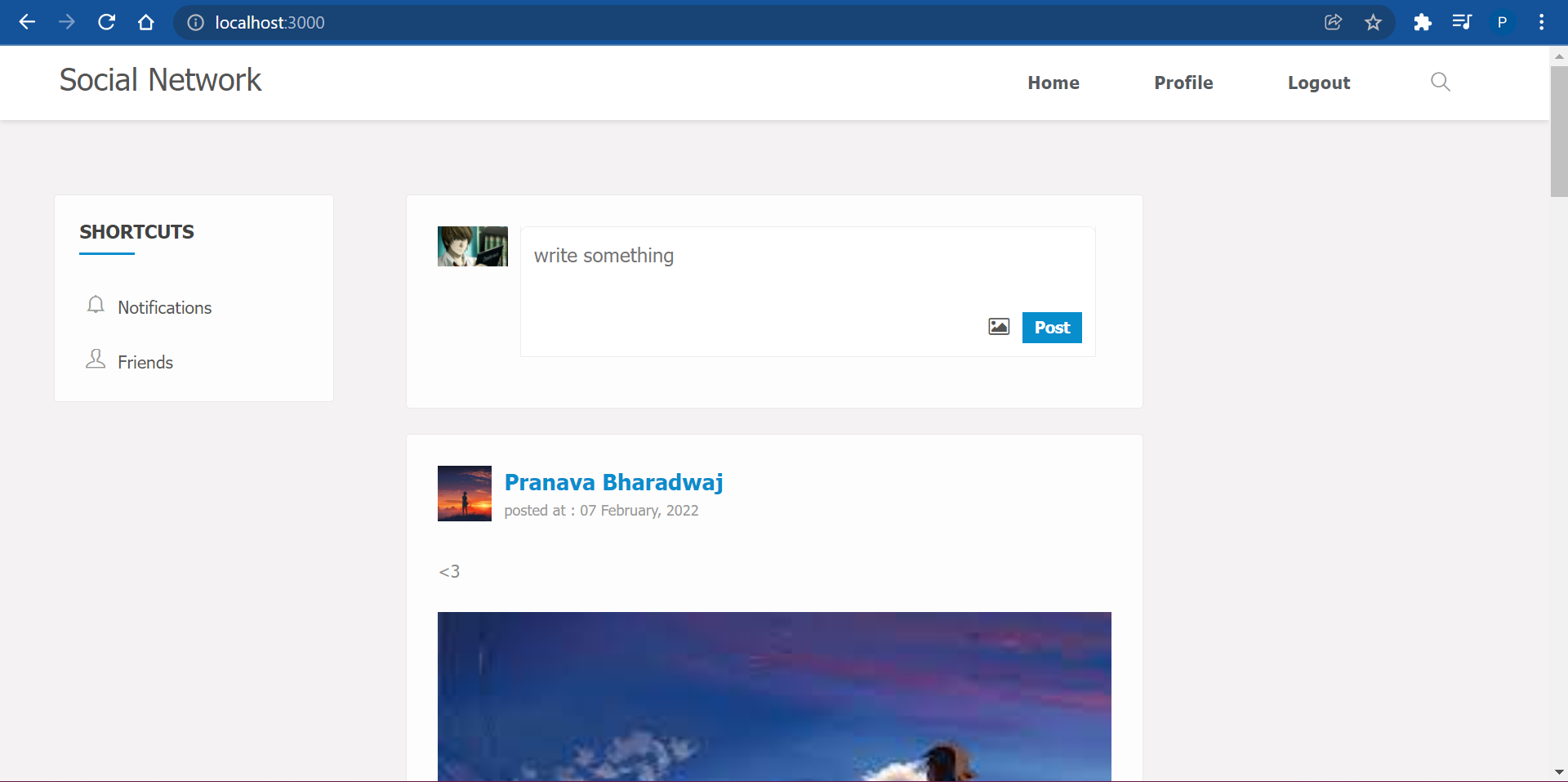


Fig 4.12(all posts and logout)

In the home page user will see all posts of his friend and his he can like ,or comment on each

Posts . Latest posts will fetched and displayed on users home page , on the right top corner

user can logout of application .

# Chapter 5: CONCLUSION AND FUTURE ENHANCEMENTS

**Conclusion:**

# Future Enhancements:

.

# REFERENCES

* Murach's Java Servlets and JSP, 3rd Edition.
* Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
* Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill
* <https://www.w3schools.com/html/html_css.asp>
* <https://www.upgrad.com/blog/how-to-create-dynamic-web-project-using-eclipse/>