**ST. ANDREWS INSTITUTE OF TECHNOLOGY &**

**MANAGEMENT**

**Gurgaon Delhi (NCR)**

**Approved by AICTE, Govt. of India, New Delhi Affiliated to Maharshi Dayanand University**

**‘A ’Grade State University, accredited by NAAC Session: 2020 – 2024**

**Department Computer Science Engineering FriendSphere**

SUBMITTED IN COMPLETE FULFILLMENT OF THE REQUIREMENT FOR MINOR PROJECT

SUBMITTED BY: -

Name: Rishab Raj Branch: B. Tech (CSE)

College Roll No. :203023 Univ. Reg. No.: 2012191141

**St. Andrews Institute of Technology & Management Gurgaon**

TO WHOM IT MAY CONCERN

I hereby certify that **“Rishab Raj”** Roll No **“203023”** of St. Andrew’s Institute of Technology and management, Gurgaon, has undergone forty- five days month Industrial training from

January 2023 to April 2023 at our organization to fulfil the requirements for the award of degree of B.Tech. (Branch).

He worked on **FriendSphere** project during the training under the supervision of **Mrs. Ranjana Rajput**. During his tenure with us we found him sincere and hard working.

Wishing him a great success in the future. Signature of the Student

Rishab Raj,

Univ. Reg. No.: 2012191141.

Signature of the Director (S) Signature of the HOD (S)

(Seal of Organization)

**ACKNOWLEDGEMENT**

### I am highly grateful to Prof. (Dr.) Rakesh Rajpal, Director, St. Andrew’s Institute of Technology and Management, Gurgaon, for providing this opportunity to carry out the Summer Internship training at ST. ANDREWS INSTITUTE OF TECHNOLOGY

AND MANAGEMENT.

The constant guidance and encouragement received from Dean T&P, SAITM, Gurgaon, has been of great help in carrying out the project work and is acknowledged with reverential thanks.

Without the wise counsel and able guidance, it would have been impossible to complete the report in this manner.

The help rendered by Mrs. Ranjana Rajput, Supervisor Ms. Preetishree

. Patnaik HOD CS for experimentation is greatly acknowledged.

I would like to express gratitude to other faculty members of Computer Science Engineering department of SAITM for their intellectual support throughout the course of this work.

Finally, the authors are indebted to all whosoever have contributed in this report work and friendly stay at

### ST. ANDREWS INSTITUTE OF TECHNOLOGY

AND MANAGEMENT.

[ Rishab Raj]

**ST. ANDREWS INSTITUTE OF TECHNOLOGY & MANAGEMENT, GURGAON**

**JANUARY- MAY, 2023**

**BONAFIDE CERTIFICATE**

This is to certify that the project entitled **FriendSphere** is the bonafide record of project work done by Mr. / Ms. **Rishab Raj** URN**:** 2012191141 of B.TECH / CSE during the year **2 022**.

**Signature of the Signature of**

**DIRECTOR HOD**

Submitted for the Project Viva-Voce examination held on

.

**DECLARATION**

I affirm that the project work titled **FriendSphere** being submitted in partial fulfilment for the award of the degree of B. Tech is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other University.

Signature of the Candidate

Rishab Raj

URN: 2012191141

I certify that the declaration made above by the candidate is true.

Signature of the Guide

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **TOPIC** | **PAGE NO.** |
| **1** | **Introduction to HTML** | 7 |
| **2** | **HTML tags used in the Project** | 8-9 |
| **3** | **Introduction to CSS.** | 10-11 |
| **4** | **CSS properties used in the project and JS** | 12-14 |
| **5** | **React.js** | 15 |
| **6** | **Redux** | 16 |
| **7** | **MongoDB** |  |
| **8** | **ExpressJS** |  |
| **9** | **Node.JS** |  |
| **10** | **API** |  |
| **11** | **E-R Diagram** |  |
| **12** | **MERN Stack** |  |
| **6** | **IDE Visual Studio code** | 16 |
| **7** | **About the project** | 17 |
| **8** | **Code, HTML, CSS, JAVASCRIT** | 18-40 |
| **9** | **OUTPUT** | 41 |
| **10** | **Conclusion** | 42 |
| 11 | **Bibliography** | 43 |

# INTRODUCTION TO HTML

**HTML** (Hypertext Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript)."Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.

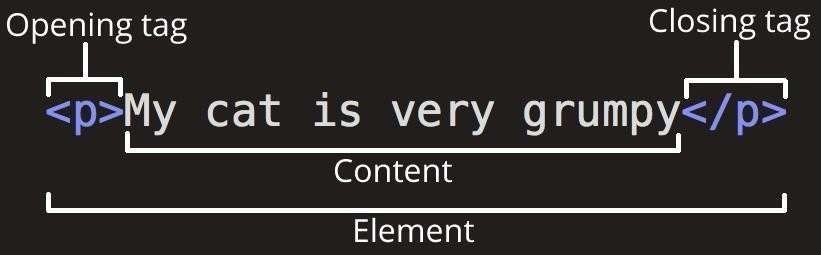
HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as: -

<head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>

, <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>

, <progress>, <video>, <ul>, <ol>, <li>and many others.

An HTML element is set off from other text in a document by "tags", which consist of the element name surrounded by “<” and “>”. The name of an element inside a tag is case insensitive. That is, it can be written in uppercase, lowercase, or a mixture. For example, the <title>tag can be written as<Title><TITLE>, or in any other way.



# HTML Tags used in the Project: -

The <!DOCTYPE html> declaration defines that this document is an HTML5 document. The <html> element is the root element of an HTML page.

The <head> element contains meta information about the HTML page.

The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.<meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.

The <title> tag defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab. The <title> tag is required in HTML documents!

The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

The <h1> element defines a large heading. There are h1 to h6 tags. The <h1> defines the most important heading. <h6> defines the least important heading.

The <a> tag defines a hyperlink, which is used to link from one page to another. The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

The <p> element defines a paragraph Tags.

The <div> tag defines a division or a section in an HTML document. The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript. The <div> tag is easily styled by using the class or id attribute. Any sort of content can be put inside the <div> tag!

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. The <img> tag is used to embed an image in an HTML page.

Images are not technically inserted into a web page; images are linked to web pages. The

<img> tag creates a holding space for the referenced image.

O The <img> tag has two required attributes:

* src - Specifies the path to the image
* alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed.

You can add comments to your HTML source by using the following syntax:

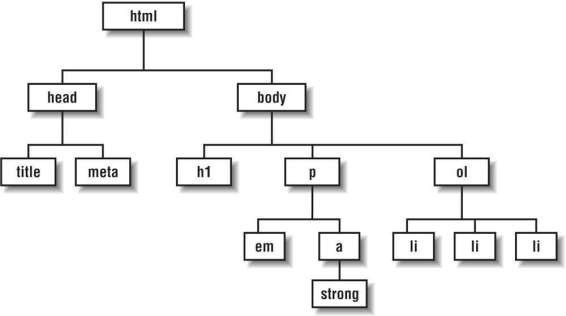
<!-- Write your comments here —>

The <nav> tag defines a set of navigation links. Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

The <link> tag defines the relationship between the current document and an external resource. The <link> tag is most often used to link to external style sheets or to add a favicon to your website. The <link> element is an empty element; it contains attributes only.

The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document. The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

The class attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.



*Representation of html tags*

# INTRODUCTION TO CSS

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.CSS is among the core languages of the open web and is standardized across Web browsers. Previously, development of various parts of CSS specification was done synchronously, which allowed versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, CSS3.

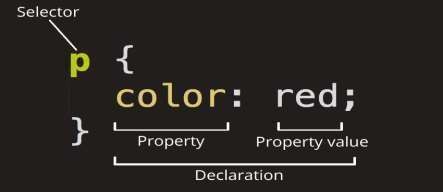
However, CSS4 has never become an official version. Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.CSS Solved a Big Problem HTML was NEVER intended to contain tags for formatting a web page! HTML was created to describe the content of a web page, like:

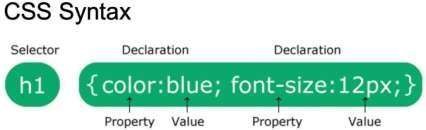
<h1>This is a heading</h1>

<p>This is a paragraph.</p>

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.



CSS removed the style formatting from the HTML page. CSS Saves a lot of work. The style definitions are normally saved in external .css files. With an external stylesheet file,



you can change the look of an entire website by changing just one file!

The whole structure is called a ruleset. (The term ruleset is often referred to as just rule.) Note the names of the individual parts:

###### Selector:-

This is the HTML element name at the start of the ruleset. It defines the element(s) to be

styled (in this example, <p> elements). To style a different element, change the selector.

###### Declaration:-

This is a single rule like color: red; . It specifies which of the element's properties you want

to style.

###### Properties:-

These are ways in which you can style an HTML element. (In this example, color is a

property of the <p> elements.) In CSS, you choose which properties you want to affect in

the rule.

###### Property value:-

To the right of the property—after the colon—there is the property value. This chooses one out of many possible appearances for a given property. (For example, there are many color values in addition to red.)

# CSS properties used in the Project: -

Background color: — The background-color property specifies the background color of an element. With CSS, a color is most often specified by:

* a valid color name - like "red"
* an RGB value - like "RGB(255,0,0)"

The background image can also be set for specific elements, like the <p> element Font Weight. The font-weight property specifies the weight of a font.

Font Size. The font-size property sets the size of the text. The font-size value can be an absolute, or relative size.

CSS Height and Width. The CSS height and width properties are used to set the height and width of an element. The CSS max-width property is used to set the maximum width of an element.

The CSS margin properties are used to create space around elements, outside of any defined borders. With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

CSS padding the CSS padding properties are used to generate space around an element's content, inside of any defined borders. With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

The text-align property specifies the horizontal alignment of text in an element.

The z-index property of CSS which specifies the stack order of an element. An element with greater stack order is always in front of an element with a lower stack order. So, if the value of z-index of any element is greater than the element will stay above the other element.

Text-shadow property provides shadow to the text. It accepts a comma-separated list of shadows to be applied to the text and any of its decorations. Each shadow is described by some combination of X and Y offsets from the element, blur radius, and color.

The display CSS property sets whether an element is treated as a block or inline element and the layout used for its children, such as flow layout, grid or flex. Formally, the display property sets an element's inner and outer display types.

Position absolute takes the document out of the document flow. This means that it no longer takes up any space like what static and relative does. When position absolute is used on an element, it is positioned absolutely with reference to the closest parent that has a position relative value.

The background color is the very basic property that is used in CSS to give any element a background color.

The color property in CSS is used to give colors to the text again it is very common property used in CSS.

JAVASCRIPT

JavaScript is *an object-based scripting language* which is lightweight and cross-platform. JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

JavaScript (JS) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser.

**Elements used in JS:**

const keyword:

Variables defined with const cannot be Redeclared. Variables defined with const cannot be Reassigned. Variables defined with const have Block Scope.

JavaScript Variable: Variables are containers for storing data (storing data values). 4 Ways to Declare a JavaScript Variable:

Using var Using let Using const Using nothing

JavaScript Const:

The const keyword was introduced in ES6 (2015). Variables defined with const cannot be Redeclared. Variables defined with const cannot be Reassigned. Variables defined with const have Block Scope.

JavaScript Loops: Loops are handy, if you want to run the same code over and over again, each time with a different value.

Different Kinds of Loops: JavaScript supports different kinds of loops:

* + for - loops through a block of code a number of times
  + for/in - loops through the properties of an object
  + for/of - loops through the values of an iterable object

. while - loops through a block of code while a specified condition is true

* + do/while - also loops through a block of code while a specified condition is true

JavaScript Loop:

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array. There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop
5. JavaScript For loop:

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iterations is known.

The While Loop: The while loop loops through a block of code as long as a specified condition is true. The Do While Loop:

The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

JavaScript Let: The let keyword was introduced in ES6 (2015). Variables defined with let cannot be Redeclared. Variables defined with let must be Declared before use. Variables defined with let have Block Scope.

JavaScript Event Listener: The addEventListener() method attaches an event handler to the specified element. The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element.

Event Handlers in JavaScript:

* 1. onClick
  2. onDblClick
  3. onDragDrop
  4. onError

# REACT.JS

ReactJS is JavaScript library used for building reusable UI components. According to React official documentation, following is the definition −

React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on

the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

React Features

* **JSX** − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
* **Components** − React is all about components. You need to think of everything as a component.
* This will help you maintain the code when working on larger scale projects.
* **Unidirectional data flow and Flux** − React implements one-way data flow which makes it easy to
* reason about your app. Flux is a pattern that helps keeping your data unidirectional.
* **License** − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

## React Advantages

* Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript
* virtual DOM is faster than the regular DOM.
* Can be used on client and server side as well as with other frameworks.
* Component and data patterns improve readability, which helps to maintain larger apps.

## React Limitations

* Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
* Uses inline templating and JSX, which might seem awkward to some developers.
* React is an open source, JavaScript library for developing user interface (UI) in web application.
* React is developed and released by Facebook. Facebook is continuously working on the React library and enhancing it by fixing bugs and introducing new features. This tutorial starts with the architecture of React, how-to guide to setup projects, creating components, JSX and then walks through advanced concepts like state management, form programming, routing and finally conclude with step by step
* working example.

# Redux

Redux is a predictable state container for JavaScript apps.

It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as [live code editing combined with a time traveling debugger](https://github.com/reduxjs/redux-devtools).

You can use Redux together with [React](https://reactjs.org/), or with any other view library. It is tiny (2kB, including dependencies), but has a large ecosystem of addons available.

### Redux Toolkit[​](https://redux.js.org/introduction/getting-started#redux-toolkit)

[**Redux Toolkit**](https://redux-toolkit.js.org/) is our official recommended approach for writing Redux logic. It wraps around the Redux core, and contains packages and functions that we think are essential for building a Redux app. Redux Toolkit builds in our suggested best practices, simplifies most Redux tasks, prevents common mistakes, and makes it easier to write Redux applications.

RTK includes utilities that help simplify many common use cases, including [store setup](https://redux-toolkit.js.org/api/configureStore), [creating reducers and writing immutable update logic](https://redux-toolkit.js.org/api/createreducer), and even [creating entire "slices" of state at once](https://redux-toolkit.js.org/api/createslice).

Whether you're a brand new Redux user setting up your first project, or an experienced user who wants to simplify an existing application, [Redux Toolkit](https://redux-toolkit.js.org/) can help you make your Redux code better.

Redux Toolkit is available as a package on NPM for use with a module bundler or in a Node application:

# NPM  
npm install @reduxjs/toolkit  
  
# Yarn  
yarn add @reduxjs/toolkit

### Create a React Redux App[​](https://redux.js.org/introduction/getting-started#create-a-react-redux-app)

The recommended way to start new apps with React and Redux is by using the [official Redux+JS template](https://github.com/reduxjs/cra-template-redux) or [Redux+TS template](https://github.com/reduxjs/cra-template-redux-typescript) for [Create React App](https://github.com/facebook/create-react-app), which takes advantage of [Redux Toolkit](https://redux-toolkit.js.org/) and React Redux's integration with React components.

# Redux + Plain JS template  
npx create-react-app my-app --template redux  
  
# Redux + TypeScript template  
npx create-react-app my-app --template redux-typescript

# MongoDB

MongoDB is a [source-available](https://en.wikipedia.org/wiki/Source-available) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [document-oriented database](https://en.wikipedia.org/wiki/Document-oriented_database) program. Classified as a [NoSQL](https://en.wikipedia.org/wiki/NoSQL) database program, MongoDB uses [JSON](https://en.wikipedia.org/wiki/JSON)-like documents with optional [schemas](https://en.wikipedia.org/wiki/Database_schema). MongoDB is developed by [MongoDB Inc.](https://en.wikipedia.org/wiki/MongoDB_Inc.) and licensed under the [Server Side Public License](https://en.wikipedia.org/wiki/Server_Side_Public_License) (SSPL) which is deemed non-free by several distributions. MongoDB is a member of the [MACH Alliance](https://en.wikipedia.org/wiki/MACH_Alliance).

Document Database

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

{

field:value

name: ”Rishab”,

age: 27,

roll\_no: 203023,

}

The advantages of using documents are:

* Documents correspond to native data types in many programming languages.
* Embedded documents and arrays reduce need for expensive joins.
* Dynamic schema supports fluent polymorphism.

**Collections/Views/On-Demand Materialized Views**

MongoDB stores documents in [collections](https://www.mongodb.com/docs/manual/core/databases-and-collections/#std-label-collections). Collections are analogous to tables in relational databases.

In addition to collections, MongoDB supports:

* Read-only [Views](https://www.mongodb.com/docs/manual/core/views/) (Starting in MongoDB 3.4)
* [On-Demand Materialized Views](https://www.mongodb.com/docs/manual/core/materialized-views/) (Starting in MongoDB 4.2).

Key Features

**High Performance**

MongoDB provides high performance data persistence. In particular,

* Support for embedded data models reduces I/O activity on database system.
* Indexes support faster queries and can include keys from embedded documents and arrays.

**Query API**

The MongoDB Query API supports [read and write operations (CRUD)](https://www.mongodb.com/docs/manual/crud/) as well as:

* [Data Aggregation](https://www.mongodb.com/docs/manual/core/aggregation-pipeline/)
* [Text Search](https://www.mongodb.com/docs/manual/text-search/) and [Geospatial Queries.](https://www.mongodb.com/docs/manual/tutorial/geospatial-tutorial/)

**High Availability**

MongoDB's replication facility, called [replica set](https://www.mongodb.com/docs/manual/replication/), provides:

* *automatic* failover
* data redundancy.

A [replica set](https://www.mongodb.com/docs/manual/replication/) is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

**Horizontal Scalability**

MongoDB provides horizontal scalability as part of its *core* functionality:

* [Sharding](https://www.mongodb.com/docs/manual/sharding/#std-label-sharding-introduction) distributes data across a cluster of machines.
* Starting in 3.4, MongoDB supports creating [zones](https://www.mongodb.com/docs/manual/core/zone-sharding/#std-label-zone-sharding) of data based on the [shard key](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-shard-key). In a balanced cluster, MongoDB directs reads and writes covered by a zone only to those shards inside the zone. See the [Zones](https://www.mongodb.com/docs/manual/core/zone-sharding/#std-label-zone-sharding) manual page for more information.

**Support for Multiple Storage Engines**

MongoDB supports [multiple storage engines:](https://www.mongodb.com/docs/manual/core/storage-engines/)

* [WiredTiger Storage Engine](https://www.mongodb.com/docs/manual/core/wiredtiger/) (including support for [Encryption at Rest)](https://www.mongodb.com/docs/manual/core/security-encryption-at-rest/)
* [In-Memory Storage Engine.](https://www.mongodb.com/docs/manual/core/inmemory/)

In addition, MongoDB provides pluggable storage engine API that allows third parties to develop storage engines for MongoDB.

# ExpressJS

Express.js, or simply Express, is a [back end](https://en.wikipedia.org/wiki/Front_end_and_back_end) [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) for building [REST](https://en.wikipedia.org/wiki/Representational_state_transfer)ful APIs with [Node.js](https://en.wikipedia.org/wiki/Node.js), released as [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software) under the [MIT License](https://en.wikipedia.org/wiki/MIT_License). It is designed for building [web applications](https://en.wikipedia.org/wiki/Web_application) and [APIs](https://en.wikipedia.org/wiki/API). It has been called the [de facto standard](https://en.wikipedia.org/wiki/De_facto_standard) server framework for [Node.js](https://en.wikipedia.org/wiki/Node.js).

The original author, TJ Holowaychuk, described it as a [Sinatra](https://en.wikipedia.org/wiki/Sinatra_(software))-inspired server, meaning that it is relatively minimal with many features available as plugins. Express is the back-end component of popular development stacks like the [MEAN](https://en.wikipedia.org/wiki/MEAN_(software_bundle)), MERN or MEVN stack, together with the [MongoDB](https://en.wikipedia.org/wiki/MongoDB) database software and a [JavaScript](https://en.wikipedia.org/wiki/JavaScript) front-end framework or library.

# I D E: - VISUAL STUDIO CODE

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas.

Extensions used in project: -

\****Prettier***: It is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

* ***Bracket Pair Colorizer:*** This extension allows matching brackets to be identified with colors. The user can define which characters to match, and which colors to use.

\****Live Share:*** It is an extension for Visual Studio Code that enables real-time collaboration between developers. It gives users the ability to share a session with someone else, allowing them to edit code as well as share a server and debugging session.

***\*Code Runner***: It helps to run code for JavaScript, PHP, Python, Perl, Ruby, Go, Lua, Groovy, Scala, VBScript, PowerShell easily.

***\*Live server:*** It launch’s a development local Server with live reload feature for static & dynamic pages

# About the project

**Friends list Website**

#### Friends list website is a social media networking site that allows users to connect with friends, family, co-workers and others, including groups of people who share similar interests. Users can share their profile picture with their friends. This website is simply takes care of your Friends all over globe and handle their status profile and there details which is necessary. This website helps you to connect you over the globe privately. While many argue that social media networks only distract employees, the opposite may be true. When employees take breaks to do something that interests them, they tend to be happier and more productive. If employees enjoy a quick break to check the Facebook News Feed and feel more connected to friends and family, they may be more relaxed and focused when they get back to work.

“Short and unobtrusive breaks, such as a quick surf on the internet, enables the mind to rest itself, leading to a higher net total concentration for a day’s work and, as a result, increased productivity,” said Brent Coker, researcher and lecturer in the management and marketing department at the University of Melbourne.

H T M L

Layout.handlebars

<!DOCTYPE html>

<html>

<head>

  <title>LoginApp</title>

  <link rel="stylesheet" href="/css/bootstrap.css" />

  <link rel="stylesheet" href="/css/style.css" />

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

</head>

<body>

  <div class="container">

      <div class="header clearfix">

        <nav id="reload">

          <ul class="nav nav-pills pull-right">

            {{#if user}}

              <input type="hidden"  id="currentuser" value="{{user.username}}">

              <li role="presentation"><a href="/search">Search <i class="fa fa-search"></i></a></li>

              <li>

                <div class="dropdown" style="">

                <button class="btn btn-primary dropdown-toggle" type="button" data-toggle="dropdown"><i class="fa fa-user-plus"></i>

                <span class="caret"></span></button>

                  <ul class="dropdown-menu" style="width: 300px; height: 400px; overflow-y: scroll;">

                    {{#if user.request}}

                    {{#each newfriend}}

                    <li style="border-bottom: 1px solid lightgray">

                      <a href="#">

                          <div>

                            <h4>{{this.username}}</h4>

                          </div>

                          <div>

                              <div style="width: 50%; display: inline;">

                                <input type="hidden" name="senderId" id="senderId" value="{{this.userId}}">

                                <input type="hidden" name="senderName" id="senderName" value="{{this.username}}">

                                <button type="submit" id="accept\_friend" class="btn btn-primary" style="width: 8em">Accept</button>

                              </div>

                              <div style="width: 50%;  display: inline;">

                                 <input type="hidden" name="user\_Id" id="user\_Id" value="{{this.userId}}">

                                <button type="submit" id="cancel\_friend" class="btn btn-primary" style="width: 8em">Cancel</button>

                              </div>

                          </div>

                      </a>

                    </li>

                    {{/each}}

                    {{else}}

                    <div>

                      <h4 style="text-align:center">No requests</h4>

                    </div>

                    {{/if}}

                  </ul>

                </div>

              </li>

              <li role="presentation"><a href="/users/logout">Logout</a></li>

            {{else}}

              <li role="presentation"><a href="/users/login">Login</a></li>

              <li role="presentation"><a href="/users/register">Register</a></li>

            {{/if}}

          </ul>

        </nav>

        <h3 class="text-muted"><img src="../upload/panda.png" style="width: 100px; position: absolute;left: 200px"> Panda DOTS</h3>

      </div>

      <div class="row">

        <div class="col-lg-12">

          {{#if success\_msg}}

            <div class="alert alert-success">{{success\_msg}}</div>

          {{/if}}

          {{#if error\_msg}}

            <div class="alert alert-danger">{{error\_msg}}</div>

          {{/if}}

          {{#if error}}

            <div class="alert alert-danger">{{error}}</div>

          {{/if}}

          {{{body}}}

        </div>

      </div>

      <footer class="footer">

        <p>&copy; 2015 LoginApp, Inc.</p>

      </footer>

    </div> <!-- /container -->

</body>

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

  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

  <script type="text/javascript" src="/socket.io/socket.io.js"></script>

  <script type="text/javascript" src="/js/sendrequest.js"></script>

  <script type="text/javascript">

  </script>

</html>

**Search.handlebars:-**

<h2>Search Friends</h2>

<form action="/search" method="post">

    <input type="text" name="searchfriend" placeholder="username">

    <input type="submit" value="search">

</form>

<hr>

    {{#each result}}

    <div class="usercard col-lg-3">

        <img class="usercard-image" src="upload/{{this.userImage}}">

        <h4 class="usercard-name">{{this.name}}</h4>

        <p class="usercard-username">@({{this.username}})</p>

        <form action="" method="get" class="add\_friend">

            <input type="hidden" name="receiverName" class="receiverName" value="{{this.username}}">

            <input type="hidden" name="sender-name" class="sender-name" value="{{user.username}}">

            <button type="submit" id="" onclick="addFriend('{{this.username}}')" class="btn add accept friend-add"><i class="fa fa-user"></i> Add Friend</button>

        </form>

    </div>

    {{/each}}

**Registes.handlebars:-**

h2 class="page-header">Register</h2>

{{#if user}}

<        <div class="alert alert-danger">Username has taken already try new</div>

{{/if}}

{{#if mail}}

        <div class="alert alert-danger">Email has taken already try new</div>

{{/if}}

{{#if errors}}

      {{#each errors}}

        <div class="alert alert-danger">{{msg}}</div>

      {{/each}}

{{/if}}

<form method="post" action="/users/register">

   <div class="form-group">

    <label>Name</label>

    <input type="text" class="form-control" placeholder="Name" name="name">

  </div>

  <div class="form-group">

    <label>Username</label>

    <input type="text" class="form-control" placeholder="Username" name="username">

  </div>

   <div class="form-group">

    <label>Email</label>

    <input type="email" class="form-control" placeholder="Email" name="email">

  </div>

  <div class="form-group">

    <label>Password</label>

    <input type="password" class="form-control" placeholder="Password" name="password">

  </div>

  <div class="form-group">

    <label>Confirm Password</label>

    <input type="password" class="form-control" placeholder="Password" name="password2">

  </div>

  <button type="submit" class="btn btn-default">Submit</button>

<form>

**Login.handlebars:-**

<h2 class="page-header">Register</h2>

{{#if user}}

        <div class="alert alert-danger">Username has taken already try new</div>

{{/if}}

{{#if mail}}

        <div class="alert alert-danger">Email has taken already try new</div>

{{/if}}

{{#if errors}}

      {{#each errors}}

        <div class="alert alert-danger">{{msg}}</div>

      {{/each}}

{{/if}}

<form method="post" action="/users/register">

   <div class="form-group">

    <label>Name</label>

    <input type="text" class="form-control" placeholder="Name" name="name">

  </div>

  <div class="form-group">

    <label>Username</label>

    <input type="text" class="form-control" placeholder="Username" name="username">

  </div>

   <div class="form-group">

    <label>Email</label>

    <input type="email" class="form-control" placeholder="Email" name="email">

  </div>

  <div class="form-group">

    <label>Password</label>

    <input type="password" class="form-control" placeholder="Password" name="password">

  </div>

  <div class="form-group">

    <label>Confirm Password</label>

    <input type="password" class="form-control" placeholder="Password" name="password2">

  </div>

  <button type="submit" class="btn btn-default">Submit</button>

</form>

# CSS

/\* Space out content a bit \*/

body {

  padding-top: 20px;

  padding-bottom: 20px;

}

/\* Everything but the jumbotron gets side spacing for mobile first views \*/

.header,

.marketing,

.footer {

  padding-right: 15px;

  padding-left: 15px;

}

/\* Custom page header \*/

.header {

  padding-bottom: 20px;

  border-bottom: 1px solid #e5e5e5;

}

/\* Make the masthead heading the same height as the navigation \*/

.header h3 {

  margin-top: 0;

  margin-bottom: 0;

  line-height: 40px;

}

/\* Custom page footer \*/

.footer {

  padding-top: 19px;

  color: #777;

  border-top: 1px solid #e5e5e5;

}

/\* Customize container \*/

@media (min-width: 768px) {

  .container {

    max-width: 730px;

  }

}

.container-narrow > hr {

  margin: 30px 0;

}

/\* Main marketing message and sign up button \*/

.jumbotron {

  text-align: center;

  border-bottom: 1px solid #e5e5e5;

}

.jumbotron .btn {

  padding: 14px 24px;

  font-size: 21px;

}

/\* Supporting marketing content \*/

.marketing {

  margin: 40px 0;

}

.marketing p + h4 {

  margin-top: 28px;

}

/\* Responsive: Portrait tablets and up \*/

@media screen and (min-width: 768px) {

  /\* Remove the padding we set earlier \*/

  .header,

  .marketing,

  .footer {

    padding-right: 0;

    padding-left: 0;

  }

  /\* Space out the masthead \*/

  .header {

    margin-bottom: 30px;

  }

  /\* Remove the bottom border on the jumbotron for visual effect \*/

  .jumbotron {

    border-bottom: 0;

  }

}

.page-header{

  margin-top:0;

}

footer{

  margin-top:40px;

}

.profilestyle {

  width: 250px;

  height: 250px;

  border-radius: 50%;

  border:1px solid #ccc;

  object-fit: contain;

}

.usercard{

  width: 190px;

  height: 300px;

  /\*display: inline;\*/

  background-color: white;

  border-radius: 5px;

  box-shadow: 3px 3px 10px #A9A9A9;

  padding: 20px;

  margin: 20px;

  overflow: visible;

}

.usercard-image {

  width: 100px;

  height: 100px;

  border-radius: 50%;

  margin-left: 15%;

  object-fit: contain;

}

.usercard-name {

  margin-left: 10%;

  margin-right: 10%

}

.usercard-username {

  margin-left: 10%;

  margin-right: 10%;

  color: blue

}

.usercard-button {

  width: 100px;

  padding: 5px;

  margin:5px;

  margin-left: 15%;

  border-radius: 5px;

  color: white;

  border: none;

}

.accept {

  background-color: green;

  color: white;

  margin-left: 15%

}

.reject {

  background-color: red;}

}

JAVASCRIPT&REACT.JS

**Index.js**

var express = require('express');

var router = express.Router();

var formidable = require('formidable');

var User = require('../models/user');

var path = require('path');

var async = require('async');

// Get Homepage

router.get('/', ensureAuthenticated, function(req, res){

    res.render('index', {

        newfriend: req.user.request

    });

});

router.get('/search', ensureAuthenticated, function(req, res){

    var sent =[];

    var friends= [];

    var received= [];

    received= req.user.request;

    sent= req.user.sentRequest;

    friends= req.user.friendsList;

    User.find({username: {$ne: req.user.username}}, function(err, result){

        if (err) throw err;

        res.render('search',{

            result: result,

            sent: sent,

            friends: friends,

            received: received

        });

    });

});

router.post('/search', ensureAuthenticated, function(req, res) {

      var searchfriend = req.body.searchfriend;

    if(searchfriend) {

        var mssg= '';

        if (searchfriend == req.user.username) {

            searchfriend= null;

        }

         User.find({username: searchfriend}, function(err, result) {

             if (err) throw err;

                 res.render('search', {

                 result: result,

                 mssg : mssg

             });

        });

    }

    async.parallel([

        function(callback) {

            if(req.body.receiverName) {

                    User.update({

                        'username': req.body.receiverName,

                        'request.userId': {$ne: req.user.\_id},

                        'friendsList.friendId': {$ne: req.user.\_id}

                    },

                    {

                        $push: {request: {

                        userId: req.user.\_id,

                        username: req.user.username

                        }},

                        $inc: {totalRequest: 1}

                        },(err, count) =>  {

                            console.log(err);

                            callback(err, count);

                        })

            }

        },

        function(callback) {

            if(req.body.receiverName){

                    User.update({

                        'username': req.user.username,

                        'sentRequest.username': {$ne: req.body.receiverName}

                    },

                    {

                        $push: {sentRequest: {

                        username: req.body.receiverName

                        }}

                        },(err, count) => {

                        callback(err, count);

                        })

            }

        }],

    (err, results)=>{

        res.redirect('/search');

    });

            async.parallel([

                // this function is updated for the receiver of the friend request when it is accepted

                function(callback) {

                    if (req.body.senderId) {

                        User.update({

                            '\_id': req.user.\_id,

                            'friendsList.friendId': {$ne:req.body.senderId}

                        },{

                            $push: {friendsList: {

                                friendId: req.body.senderId,

                                friendName: req.body.senderName

                            }},

                            $pull: {request: {

                                userId: req.body.senderId,

                                username: req.body.senderName

                            }},

                            $inc: {totalRequest: -1}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                },

                // this function is updated for the sender of the friend request when it is accepted by the receiver

                function(callback) {

                    if (req.body.senderId) {

                        User.update({

                            '\_id': req.body.senderId,

                            'friendsList.friendId': {$ne:req.user.\_id}

                        },{

                            $push: {friendsList: {

                                friendId: req.user.\_id,

                                friendName: req.user.username

                            }},

                            $pull: {sentRequest: {

                                username: req.user.username

                            }}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                },

                function(callback) {

                    if (req.body.user\_Id) {

                        User.update({

                            '\_id': req.user.\_id,

                            'request.userId': {$eq: req.body.user\_Id}

                        },{

                            $pull: {request: {

                                userId: req.body.user\_Id

                            }},

                            $inc: {totalRequest: -1}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                },

                function(callback) {

                    if (req.body.user\_Id) {

                        User.update({

                            '\_id': req.body.user\_Id,

                            'sentRequest.username': {$eq: req.user.username}

                        },{

                            $pull: {sentRequest: {

                                username: req.user.username

                            }}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                }

            ],(err, results)=> {

                res.redirect('/search');

            });

});

router.post('/', function(req, res) {

    var form =new formidable.IncomingForm();

    form.parse(req);

    let reqPath= path.join(\_\_dirname, '../');

    let newfilename;

    form.on('fileBegin', function(name, file){

        file.path = reqPath+ 'public/upload/'+ req.user.username + file.name;

        newfilename= req.user.username+ file.name;

    });

    form.on('file', function(name, file) {

        User.findOneAndUpdate({

            username: req.user.username

        },

        {

            'userImage': newfilename

        },

        function(err, result){

            if(err) {

                console.log(err);

            }

        });

    });

    req.flash('success\_msg', 'Your profile picture has been uploaded');

    res.redirect('/');

});

function ensureAuthenticated(req, res, next){

    if(req.isAuthenticated()){

        return next();

    } else {

        //req.flash('error\_msg','You are not logged in');

        res.redirect('/users/login');

    }

}

module.exports = router;

**App.js:-**

var express = require('express');

var path = require('path');

var http = require('http');

var cookieParser = require('cookie-parser');

var bodyParser = require('body-parser');

var exphbs = require('express-handlebars');

var expressValidator = require('express-validator');

var flash = require('connect-flash');

var session = require('express-session');

var passport = require('passport');

var LocalStrategy = require('passport-local').Strategy;

var mongo = require('mongodb');

var mongoose = require('mongoose');

var socketIO = require('socket.io');

mongoose.connect('mongodb://localhost/blogtest');

var db = mongoose.connection;

var routes = require('./routes/index');

var users = require('./routes/users');

// Init App

var app = express();

const server = http.createServer(app);

const io= socketIO(server);

require('./socket/friend')(io);

// View Engine

app.set('views', path.join(\_\_dirname, 'views'));

app.engine('handlebars', exphbs({

  helpers: {

    ifIn: function(elem, list, options) {

      if(list.indexOf(elem) > -1) {

        return options.fn(this);

      }

      return options.inverse(this);

    }

  },

  defaultLayout:'layout'

}));

app.set('view engine', 'handlebars');

// BodyParser Middleware

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use(cookieParser());

// Set Static Folder

app.use(express.static(path.join(\_\_dirname, 'public')));

// Express Session

app.use(session({

    secret: 'secret',

    saveUninitialized: true,

    resave: true

}));

// Passport init

app.use(passport.initialize());

app.use(passport.session());

// Express Validator

app.use(expressValidator({

  errorFormatter: function(param, msg, value) {

      var namespace = param.split('.')

      , root    = namespace.shift()

      , formParam = root;

    while(namespace.length) {

      formParam += '[' + namespace.shift() + ']';

    }

    return {

      param : formParam,

      msg   : msg,

      value : value

    };

  }

}));

// Connect Flash

app.use(flash());

// Global Vars

app.use(function (req, res, next) {

  res.locals.success\_msg = req.flash('success\_msg');

  res.locals.error\_msg = req.flash('error\_msg');

  res.locals.error = req.flash('error');

  res.locals.user = req.user || null;

  next();

});

app.use('/', routes);

app.use('/users', users);

// Set Port

 app.set('port', (process.env.PORT || 5000));

server.listen(app.get('port'),function(){

      console.log('listening on port 5000');

    });

**Friend.js: -**

module.exports= function(io) {

    io.on('connection', (socket) =>{

        // console.log('A user is connected');

        socket.on('friendRequest', (friend, callback)=> {

            // console.log(friend.sender+ " "+ friend.receiver);

            io.to(friend.receiver).emit('newFriendRequest', {

                from: friend.sender,

                to: friend.receiver

            });

            callback();

        });

    });

}

**Users.js: -**

var express = require('express');

var router = express.Router();

var formidable = require('formidable');

var User = require('../models/user');

var path = require('path');

var async = require('async');

// Get Homepage

router.get('/', ensureAuthenticated, function(req, res){

    res.render('index', {

        newfriend: req.user.request

    });

});

router.get('/search', ensureAuthenticated, function(req, res){

    var sent =[];

    var friends= [];

    var received= [];

    received= req.user.request;

    sent= req.user.sentRequest;

    friends= req.user.friendsList;

    User.find({username: {$ne: req.user.username}}, function(err, result){

        if (err) throw err;

        res.render('search',{

            result: result,

            sent: sent,

            friends: friends,

            received: received

        });

    });

});

router.post('/search', ensureAuthenticated, function(req, res) {

      var searchfriend = req.body.searchfriend;

    if(searchfriend) {

        var mssg= '';

        if (searchfriend == req.user.username) {

            searchfriend= null;

        }

         User.find({username: searchfriend}, function(err, result) {

             if (err) throw err;

                 res.render('search', {

                 result: result,

                 mssg : mssg

             });

        });

    }

    async.parallel([

        function(callback) {

            if(req.body.receiverName) {

                    User.update({

                        'username': req.body.receiverName,

                        'request.userId': {$ne: req.user.\_id},

                        'friendsList.friendId': {$ne: req.user.\_id}

                    },

                    {

                        $push: {request: {

                        userId: req.user.\_id,

                        username: req.user.username

                        }},

                        $inc: {totalRequest: 1}

                        },(err, count) =>  {

                            console.log(err);

                            callback(err, count);

                        })

            }

        },

        function(callback) {

            if(req.body.receiverName){

                    User.update({

                        'username': req.user.username,

                        'sentRequest.username': {$ne: req.body.receiverName}

                    },

                    {

                        $push: {sentRequest: {

                        username: req.body.receiverName

                        }}

                        },(err, count) => {

                        callback(err, count);

                        })

            }

        }],

    (err, results)=>{

        res.redirect('/search');

    });

            async.parallel([

                // this function is updated for the receiver of the friend request when it is accepted

                function(callback) {

                    if (req.body.senderId) {

                        User.update({

                            '\_id': req.user.\_id,

                            'friendsList.friendId': {$ne:req.body.senderId}

                        },{

                            $push: {friendsList: {

                                friendId: req.body.senderId,

                                friendName: req.body.senderName

                            }},

                            $pull: {request: {

                                userId: req.body.senderId,

                                username: req.body.senderName

                            }},

                            $inc: {totalRequest: -1}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                },

                // this function is updated for the sender of the friend request when it is accepted by the receiver

                function(callback) {

                    if (req.body.senderId) {

                        User.update({

                            '\_id': req.body.senderId,

                            'friendsList.friendId': {$ne:req.user.\_id}

                        },{

                            $push: {friendsList: {

                                friendId: req.user.\_id,

                                friendName: req.user.username

                            }},

                            $pull: {sentRequest: {

                                username: req.user.username

                            }}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                },

                function(callback) {

                    if (req.body.user\_Id) {

                        User.update({

                            '\_id': req.user.\_id,

                            'request.userId': {$eq: req.body.user\_Id}

                        },{

                            $pull: {request: {

                                userId: req.body.user\_Id

                            }},

                            $inc: {totalRequest: -1}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                },

                function(callback) {

                    if (req.body.user\_Id) {

                        User.update({

                            '\_id': req.body.user\_Id,

                            'sentRequest.username': {$eq: req.user.username}

                        },{

                            $pull: {sentRequest: {

                                username: req.user.username

                            }}

                        }, (err, count)=> {

                            callback(err, count);

                        });

                    }

                }

            ],(err, results)=> {

                res.redirect('/search');

            });

});

router.post('/', function(req, res) {

    var form =new formidable.IncomingForm();

    form.parse(req);

    let reqPath= path.join(\_\_dirname, '../');

    let newfilename;

    form.on('fileBegin', function(name, file){

        file.path = reqPath+ 'public/upload/'+ req.user.username + file.name;

    });

    form.on('file', function(name, file) {

        User.findOneAndUpdate({

            username: req.user.username

        },

        {

            'userImage': newfilename

        },

        function(err, result){

            if(err) {

                console.log(err);

            }

        });

    });

    req.flash('success\_msg', 'Your profile picture has been uploaded');

    res.redirect('/');

});

function ensureAuthenticated(req, res, next){

    if(req.isAuthenticated()){

        return next();

    } else {

        //req.flash('error\_msg','You are not logged in');

        res.redirect('/users/login');

    }

}

module.exports = router;

user.js:-

   var mongoose = require('mongoose');

var bcrypt = require('bcryptjs');

// User Schema

var UserSchema = mongoose.Schema({

    username: {

        type: String,

        index:true

    },

    password: {

        type: String

    },

    email: {

        type: String

    },

    name: {

        type: String

    },

    userImage : {

        type:String,

        default:'default.png'

    },

    sentRequest:[{

        username: {type: String, default: ''}

    }],

    request: [{

        userId: {type: mongoose.Schema.Types.ObjectId, ref: 'User'},

        username: {type: String, default: ''}

    }],

    friendsList: [{

        friendId: {type: mongoose.Schema.Types.ObjectId, ref: 'User'},

        friendName: {type: String, default: ''}

    }],

    totalRequest: {type: Number, default:0}

});

var User = module.exports = mongoose.model('User', UserSchema);

module.exports.createUser = function(newUser, callback){

    bcrypt.genSalt(10, function(err, salt) {

        bcrypt.hash(newUser.password, salt, function(err, hash) {

            newUser.password = hash;

            newUser.save(callback);

        });

    });

}

module.exports.getUserByUsername = function(username, callback){

    var query = {username: username};

    User.findOne(query, callback);

}

module.exports.getUserById = function(id, callback){

    User.findById(id, callback);

}

module.exports.comparePassword = function(candidatePassword, hash, callback){

    bcrypt.compare(candidatePassword, hash, function(err, isMatch) {

        if(err) throw err;

        callback(null, isMatch);

    });

}

Sendrequest.js:-

var socket = io();

var sender = $('#currentuser').val();

var receiverName;

function addFriend(name) {

      $.ajax({

        url: '/search',

        type: 'POST',

        data: {

          receiverName: name

        },

        success: function() {

        }

      })

}

$(document).ready(function(){

        $('.friend-add').on('click', function(e){

            e.preventDefault();

        });

        $('#accept\_friend').on('click', function(){

            var senderId= $('#senderId').val();

            var senderName= $('#senderName').val();

            $.ajax({

                url: '/search/',

                type: 'POST',

                data: {

                    senderId:senderId,

                    senderName: senderName

                },

                success: function() {

                    $(this).parent().eq(1).remove();

                }

            });

        $('#reload').load(location.href + ' #reload');

        });

        $('#cancel\_friend').on('click', function(){

            var user\_Id= $('#user\_Id').val();

            // console.log(user\_Id);

            $.ajax({

                url: '/search',

                type: 'POST',

                data: {

                    user\_Id: user\_Id

                },

                success: function() {

                    $(this).parent().eq(1).remove();

                }

            });

        $('#reload').load(location.href + ' #reload');

        });

});

Hendlbars.js:-

var socket = io();

var sender = $('#currentuser').val();

var receiverName;

function addFriend(name) {

      $.ajax({

        url: '/search',

        type: 'POST',

        data: {

          receiverName: name

        },

        success: function() {

        }

      })

}

$(document).ready(function(){

        $('.friend-add').on('click', function(e){

            e.preventDefault();

        });

        $('#accept\_friend').on('click', function(){

            var senderId= $('#senderId').val();

            var senderName= $('#senderName').val();

            $.ajax({

                url: '/search/',

                type: 'POST',

                data: {

                    senderId:senderId,

                    senderName: senderName

                },

                success: function() {

                    $(this).parent().eq(1).remove();

                }

            });

        $('#reload').load(location.href + ' #reload');

        });

        $('#cancel\_friend').on('click', function(){

            var user\_Id= $('#user\_Id').val();

            // console.log(user\_Id);

            $.ajax({

                url: '/search',

                type: 'POST',

                data: {

                    user\_Id: user\_Id

                },

                success: function() {

                    $(this).parent().eq(1).remove();

                }

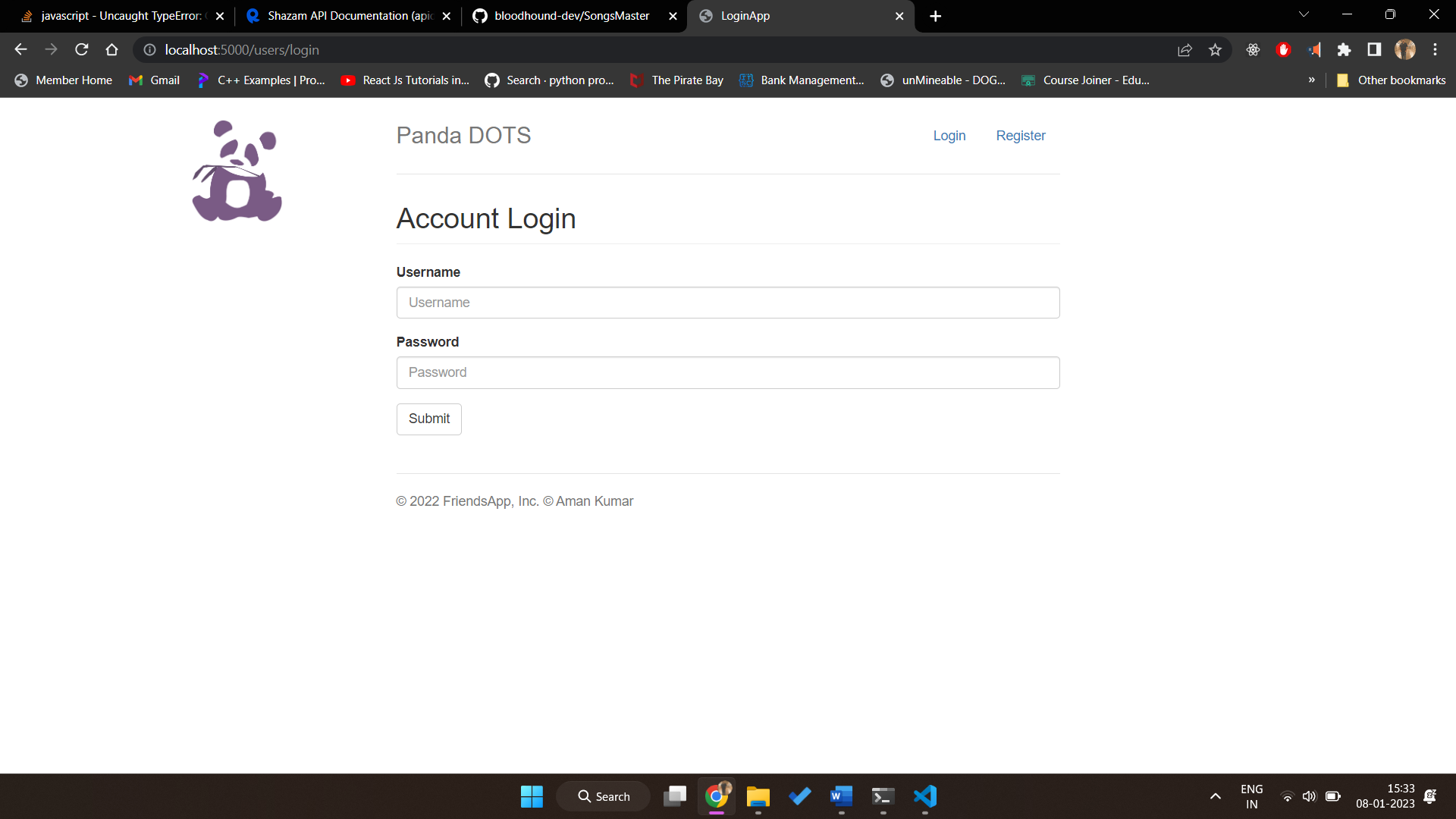
            });

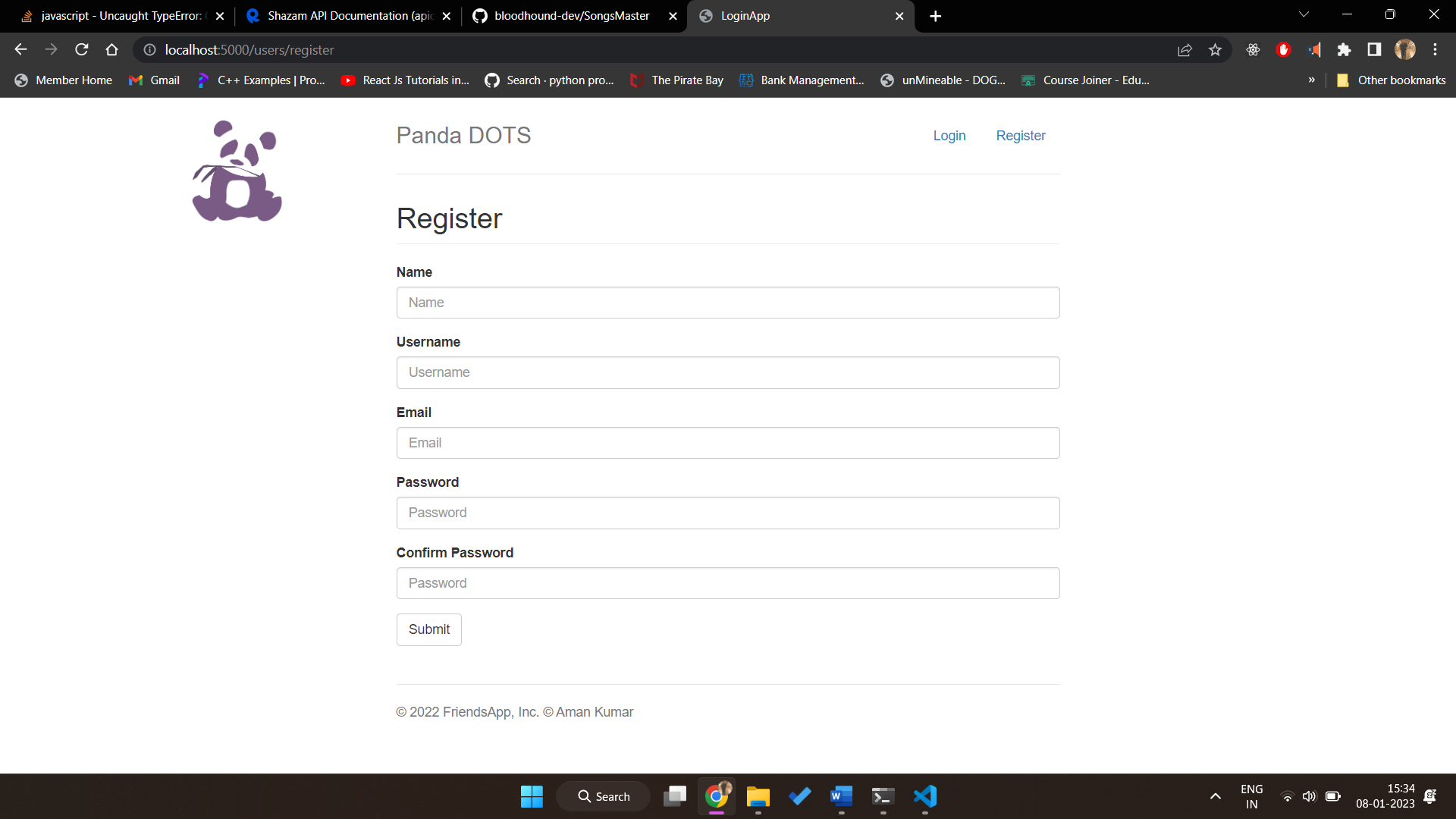
        $('#reload').load(location.href + ' #reload');

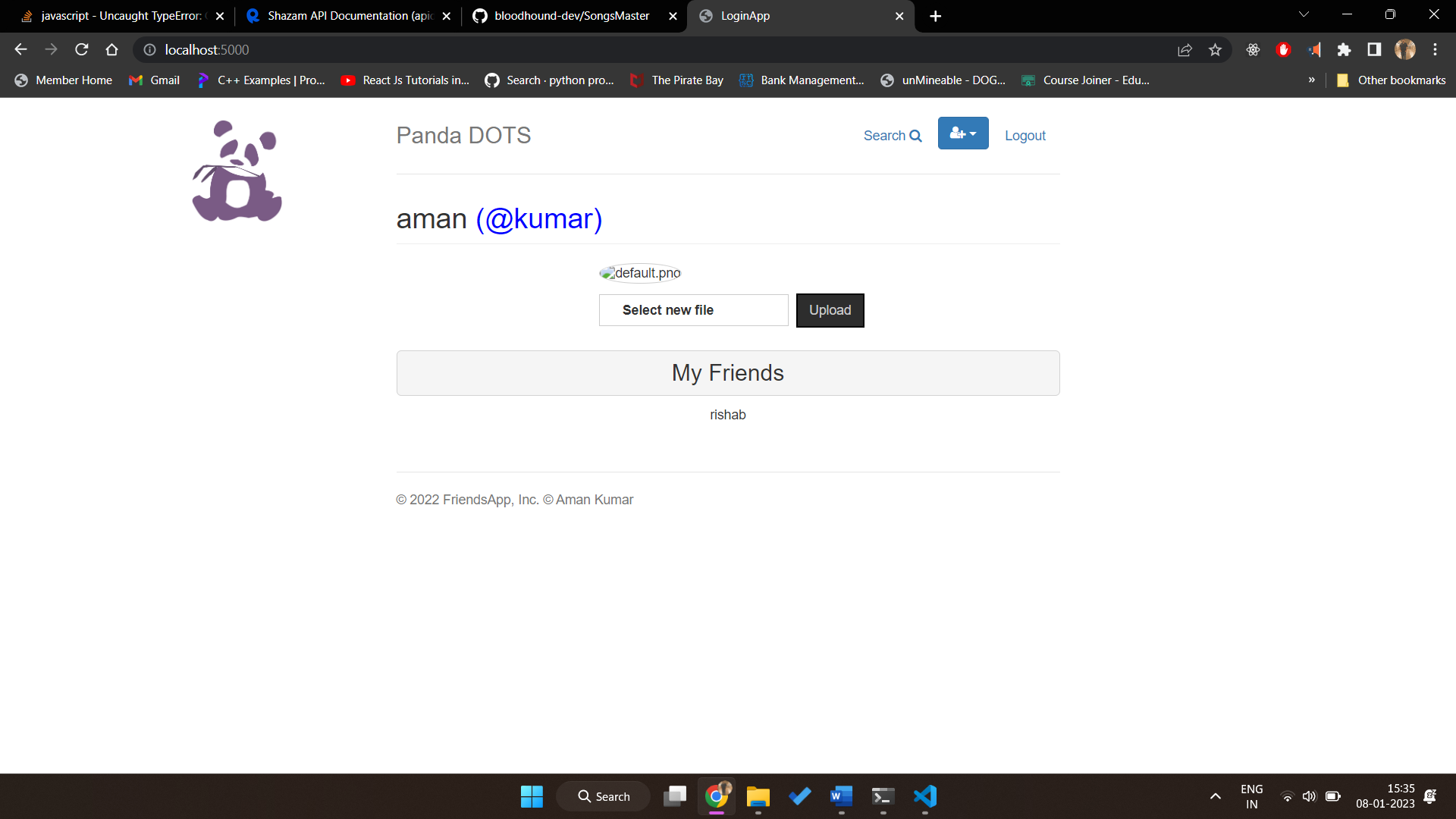
        });

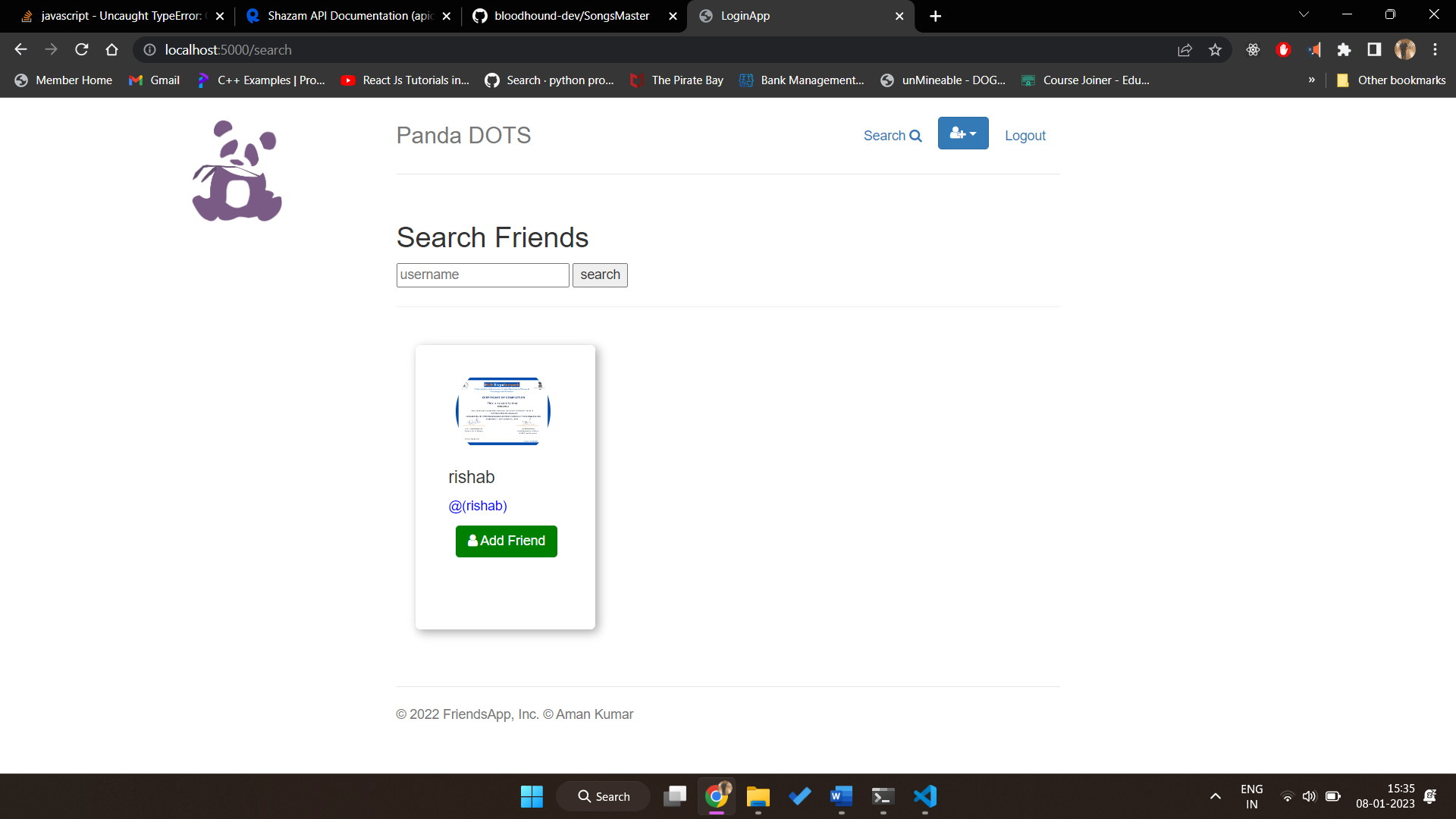
});

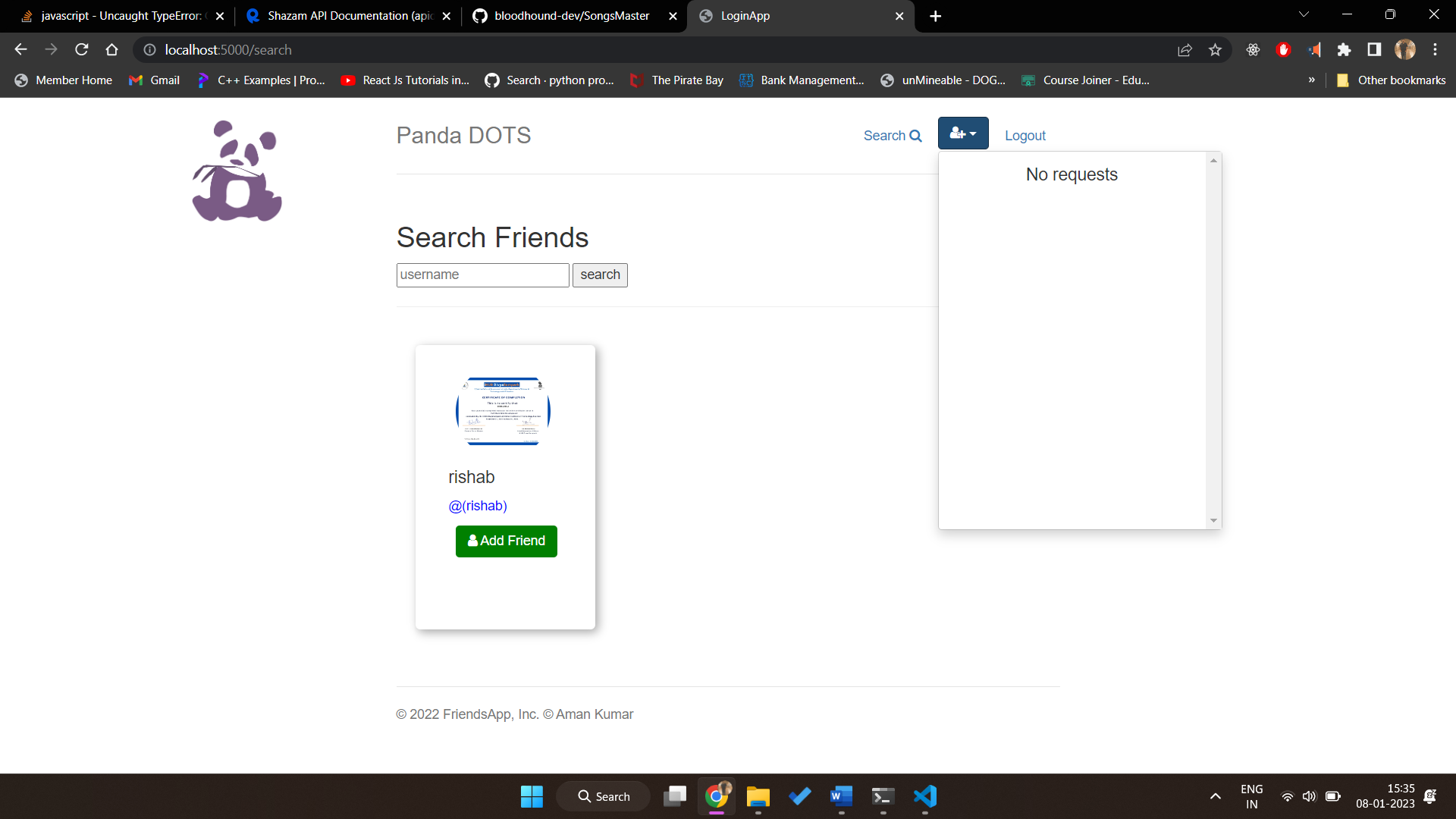
OUTPUT











C o n c l u s i o n

The Above project is Friends list search Website. I had tried to replicate the friends list search website using HTML (Hypertext Markup Language), CSS (Cascading Style Sheet) and JS (JavaScript) and React.js.

From this we learn the follow: - Learn a lot about HTML.

* + Learn a lot about CSS and overall website designing.
  + Learn a lot about JS.
  + Learn a lot about react.js.
  + How VS Code and its extension can make our as software developer’s life easy.
  + We learnt about what difficulties a software developer can facewhile working on a particular project.

Websites: -

B i b l i o g r a p h y

##### https://[www.google.com](http://www.google.com/)

* + - https://[www.youtube.com](http://www.youtube.com/)
    - https://[www.w3school.com](http://www.w3school.com/)
    - https://[www.Geeksforgeeks.com](http://www.Geeksforgeeks.com/)
    - https://[www.tutorialspoint.com](http://www.tutorialspoint.com/)