***SCRIBE* - A Social Media For Bloggers**

**Created and Developed by Angular,Bootstrap,HTML,CSS & Firebase(Database)**

Thesis Submitted in fulfillment of the Requirement for the degree

Of

**Bachelor of Technology (B.Tech)**

**in**

**Department of Computer Science and Engineering**

**Pralay Ghosh**

**(*Roll No.-*** **500117014064 *Reg. No.-*** **171430110065 of 2017-2018)**

**Sayan Chowdhury**

**(*Roll No.-*** **500117011084 *Reg. No.-*** **171430110085 of 2017-2018)**

**Sunil Maity**

**(*Roll No.-*** **500117011109 *Reg. No.-*** **171430110111 of 2017-2018)**

**Promise Dutta**

**(*Roll No.-*** **500117011066 *Reg. No.-*** **171430110067 of 2017-2018)**

**Sankha Mondal**

**(*Roll No.-*** **500117011082 *Reg. No.-*** **171430110083 of 2017-2018)**

**Under the guidance of**

**Prof. Nirupam Saha**

**Assistant Professor of Deprt. Of CSE**

**Guru Nanak Institute of Technology, Kolkata-700110**

**C:\Users\SONY\Desktop\logo.png**

**ACKNOWLEDGEMENT**

Date: 27.05.2021 **Pralay Ghosh**

Place:Kolkata

Date: 27.05.2021 **Sayan Chowdhury**

Place: Kolkata

Date: 27.05.2021 **Promise Dutta**

Place: Kolkata

Date: 27.05.2021 **Sankha Mondal**

Place: Kolkata

Date: 27.05.2021 **Sunil Maity**

Place: Kolkata

**GURUNANAK INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that the thesis entitled, “**SCRIBE - A Social Media For Bloggers**

**Created and Developed by Angular, Bootstrap, HTML,CSS & Firebase(Database)**” which is going to be submitted by  **Pralay Ghosh,** roll no-14311214001,for the award of the degree of partial fulfilment of Bachelor of Technology in the branch Computer Science and Engineering of Guru Nanak Institute of Technology, is a record of bonafideresearch work carried out under the supervision and guidance of the undersigned. Mr.DevduttaChakrabarti has worked for nearly one year on the aforementioned subject at the Department of Computer Science and Engineering, Guru Nanak Institute of Technology, Kolkata and this has reached the standard fulfilment of the requirements and the regulation relating to the degree.

The contents of this thesis, in full or part, have not been submitted to any other university or institution for the award of any degree.

**Supervisor**

----------------------------

**<Supervisor Name with Designation>**

Department of Computer Science and Engineering

GNIT, Kolkata.

**Head of the Department**

**----------------------------------**

**Mrs.IpsitaSaha**

Head of the Department

Department of Computer Science and Engineering,

GNIT, Kolkata

**Contents**

**Page No.**

* Abstract
* Introduction 01
* Literature survey 02
* Problem Statement 03
* Proposed System 06
* System Model
  1. Software Requirement Specification 08
  2. Design 14
* Implementation
  1. Modules 24
  2. Module description 44
* Experimental Results 45
  1. Test Results
  2. Screen Shots
* Conclusion & Future Study 53
* References

**INTRODUCTION**

**Scribe** is a social media platform for bloggers. It is a thought sharing platform where you can share your craziest ideas with the world. Scribe analyzes your content, and tells you exactly how to gently tweak it for better search engine rankings. Scribe also analyzes your overall site content to help you execute on your go-forward content strategy, which has become vital since the Google Panda update. Scribe helps you crosslink your content to increase usability, time on site, and indexing, identify relevant websites for guest content, strategic alliances, and link building, recruit authoritative industry writers, and locate social media users who’ll want to share your content.

**Scribe Web** is completely turn-key – we provide you a user account & password when you signed up. Once you’re logged in you can do research, read content, relationship and other connection activity all from one place, then write and publish your own content with the publish button also you can comment on other blogger’s content & interact with them.

**LITERATURE SURVEY**

Blogs represent an especially interesting informative site of any topics like-Fashion, Technology, Gadgets, Business-News etc. A blog is a discussion or informational [website](https://en.wikipedia.org/wiki/Website) published on the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web) consisting of discrete, often informal diary-style text entries (posts). Posts are typically displayed in [reverse chronological order](https://en.wikipedia.org/wiki/Reverse_chronology), so that the most recent post appears first, at the top of the [web page](https://en.wikipedia.org/wiki/Web_page). Blogging provides anyone with the opportunity to publish content. According to market research we notice that there is no particular website only for bloggers where they can post their contents and also look up others content and can comment and share that. Mainly there are many sites only for personal blogging where you are only visitor. There are you tube where we post content in video form. There is also facebook where we can post anything video audio written content etc. There is another mobile app that is **“your quotes”** that has no web version and we can write here with limited words.

So, we came up with an idea where we promote only bloggers and it will be informative for us. Here will be different subject like **ABP ANANDA app**, not only that there will be multiple features also.

Problem Statement

Blogging provides anyone with the opportunity to publish content and the potential to express their thoughts. Because of this, there are hundreds or thousands of new blogs launched every day. Unfortunately, there are some significant challenges that most new bloggers face, and in many cases the challenges are significant enough to lead the blogger to give up.

1. **No comments from Visitors/Readers**: -

It can be very frustrating to dedicate time creating a blog post, only to sit and wait for comments that never come. The interactive aspect of blogging is what draws many people to get started with their own blog, but in reality, many new blogs have trouble attracting comments. If you find that your blog posts don’t usually get the attention or feedback that you had hoped for, you’re not alone.

**2)Not enough traffic: -**

Possibly the most frustrating challenge for many bloggers is trying to attract more visitors. In fact, most bloggers don’t give very much thought to how they will attract visitors until after the blog has been launched and they find that not many people are visiting. Regardless of how much traffic you have, you’ll always want more. But for new blogs this can be a discouraging problem to face, and it’s one that leads many bloggers to give up and stop blogging.

**3)No clear purpose: -**

Every blog should have a clear purpose. If you’re maintaining a personal blog your focus is probably just to communicate with friends and family, so you can pretty much post whatever you want. But if you are blogging for business purposes you should have an idea of what you want to get out of blogging. Many people start blogs because other people are doing it and they feel that it is important, but they don’t always know specifically why.

If your blog does not have a clear purpose it will be difficult to retain the interest of your visitors. They may like one post that they stumble across, but the other content may not be of any interest to them, so they’ll stop coming back.

**4)Very few repeat visitors: -**

While getting traffic is certainly a concern for many bloggers, repeat traffic really is what is most important to a blog’s long-term success. In order for a blog to achieve and maintain a high level of success, it must be able to build a community. The community of users are people who come to the blog on a consistent basis, hopefully they leave comments at least occasionally, and they’re also more like to share a link or vote for the content on social media sites.

**5)** **Fear of publishing: -**

There is a part of blogging that is absolutely intimidating for many people. Each and every time you publish a blog post, you are truly putting yourself out there.

Your thoughts, opinions, writing style, and more are out there for comment and criticism each time you make one of your blog posts available to the public.

First, here’s a reality check: Not everybody will like you or your blog. Some will genuinely find something in your writing style, your branding, or your opinions that they truly dislike.

In that case, take their criticism, consider it carefully, improve where you can, then let it go. You’ll also run into trolls. The best way to deal with them is simply not to feed them or respond.

Mostly, keep in mind that the blogging community as a whole is both supportive and forgiving, as is the majority of your audience.

Make a sincere effort to post quality content, continue to educate yourself and grow as a blogger, and you will be just fine. The positive feedback that you receive will have much more of an impact on you than the negative.

Although social media offer easier and more accessible ways to interact with others, how we communicate and interact hinders the quality of our interpersonal relationships. The quality of interpersonal relationships is at risk because "as the Internet becomes more intertwined in daily life, the use of the Internet or social media becomes a higher priority. The usage of social media is deemed more important because users are able to easily interact and communicate with those in their social network by just liking a photo on Instagram or commenting on a post on Facebook. However, this type of online communication and interaction is changing how we think about relationships and how we maintain them. Individuals are no longer using face-to-face interaction to facilitate communication, rather they are depending on social media to do the job. Today, those in romantic relationships are writing on one another’s wall on Facebook to display and express love and affection. College students are frequently checking and updating their social networking sites daily in order to stay connected with friends and family back home. Even in friendships, individuals believe that if you are friends on Facebook or followers on Instagram and Twitter that you are friends in real life. Porter et al. Users are enticed by the idea of easier and convenient ways to keep in touch with family and friends, thus individuals are becoming dependent on social media to facilitate communication efforts in their relationships, As a result, social media can ultimately affect interpersonal relationships when we choose to allow them to control our communication in a relationship. When we become dependent on social media to facilitate communication, there are shifts in relationship maintenance behaviours, misconstrued thoughts about what a true relationship really is, and there are emotional barriers, where there is lack of emotional intimacy. Hence, causing the quality of relationships to deteriorate and “weak ties” in a relationship to develop. Influence marketing has become the social media equivalent of local advertising, and in my view, the effects are similar: incremental improvement of soft metrics such as awareness and engagement, while hard metrics like conversions and sales at scale.

**Proposed system**

Blog is a Web site on which an individual or group of users record opinions, information, etc. on a regular basis. Our software ‘***SCRIBE-*** *a social media for blogger’* is designed to simplify creating and maintaining weblogs. This software is designed for any kind of individuals. The member who is signed up as a user can use our sites totally Free. Our software also works as a **daily diary**. ***SCRIBE*** is a powerful semantic publishing platform, and it comes with a great set of features designed to make your experience as a publisher on the Internet as easy, pleasant and appealing as possible. We are proud to offer you a freely distributed, standards-compliant, fast, light and free content management system, with Sensible default settings and features, and an extremely customizable core. Blogging is often seen as a good way to get in touch with students. This blog is a type of website that is comprised of entries either made by the students or other members. A blog is a great way for a person to read & learn their topic of interests. As specialized content management systems, ***SCRIBE*** applications support the authoring, editing, and publishing of blog posts and comments, with special functions for post management, web syndication, and comment moderation. The main aim of developing this platform is to create a community of bloggers, which allowing multiple bloggers to contribute to a here and create their community and revenue out of this posts. Here anyone who have an valid email id can create an account and post what they want to post. Our Proposed System work like a **Consumer-Producer** Model like **Youtube(Broadcasting media).** Here you can read other’s blog as well as can write your own blog.

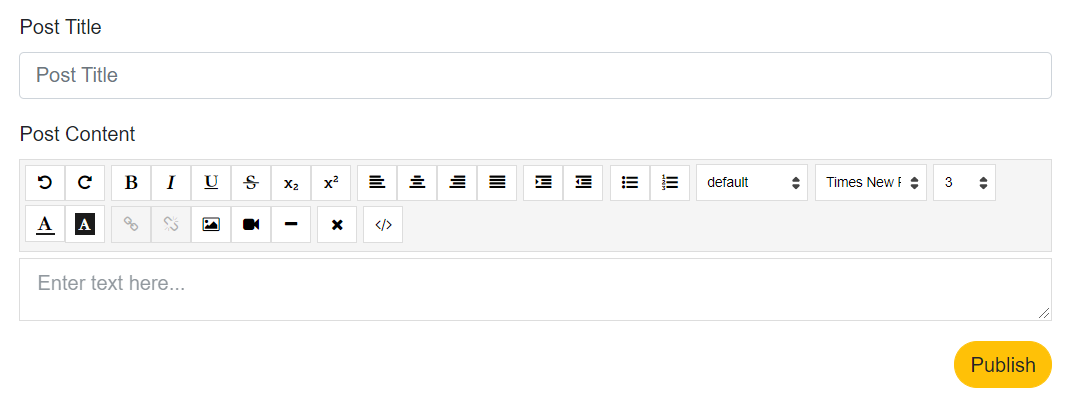
Consumer

Producer

Producer

The main factor in our *Proposed model*, Verified Bloggers can earn a handsome amount of money from our site by writing blogs. Our **SCRIBE** app is very user-friendly and hustle-free. Users/Bloggers don’t need to think about reach, community or anything. They focus on their writing and creative skills *without any technical difficulties* like- Maintaining Sites , SEO or Purchasing Domain etc. They also don’t need to pay one single penny from their pocket for accessing our site. They also get a *huge blogging community* from here where they can read, write and also can exchange each other’s thoughts and perspectives through comments.

Here as our app fully based on Consumer-Producer model that means here Consumer is Producer and Producer is consumer. So we here visitors and writers are the same person. One can consume content from here and also produce his/her own content. So this way we resolve the issue of **Few Repetitive Viewers**. Here You also can share your post link with your friends.

We make our Web-app so user-friendly that one Publish button is enough to publish your content across the internet. Now after that part if your content 

doesn’t have so much quality then other blogger’s can help you to improve and can also publish their view about your blog. You also need not to search here and there in Google for reading blogs. You can read your fellow bloggers post and improve yourself in very less amount of time and without any hustle. Actually our **SCRIBE- A social media for bloggers** is a integrated web app of bloggers for *read, write comment and grow* Together.

When Anyone starting their own blogging sites they have to market it and for marketing through Google Ad-sense they have to pay a hefty amount but in our case Our app is totally free for all users and they also need not to think about traffics.

All bloggers earn their majority amount of money from **Brand Endorsement**. In our sites they also can endorse brands with their post like this.



And we also run customized ads based on user profile and their interests. From these revenue we also pay the verified bloggers as they can put their heart and soul for making one content and post it to our sites and make our site much informative and premium.

**System Model**

*I* . *Software Requirement Specification:-*

For making this Project we use some recent Technologies like- **Angular, Bootstrap** , **Node JS**, **CSS**, **HTML**. For storing data and authentication, Database that We use **Firebase**.

Angular is a Typescript based, open-source, front-end web application development framework. Angular uses Typescript as the programming language. It's not JavaScript, it's typescript. Typescript is a superset language of JavaScript. Typescript code is "transpiled" to regular JavaScript Code.

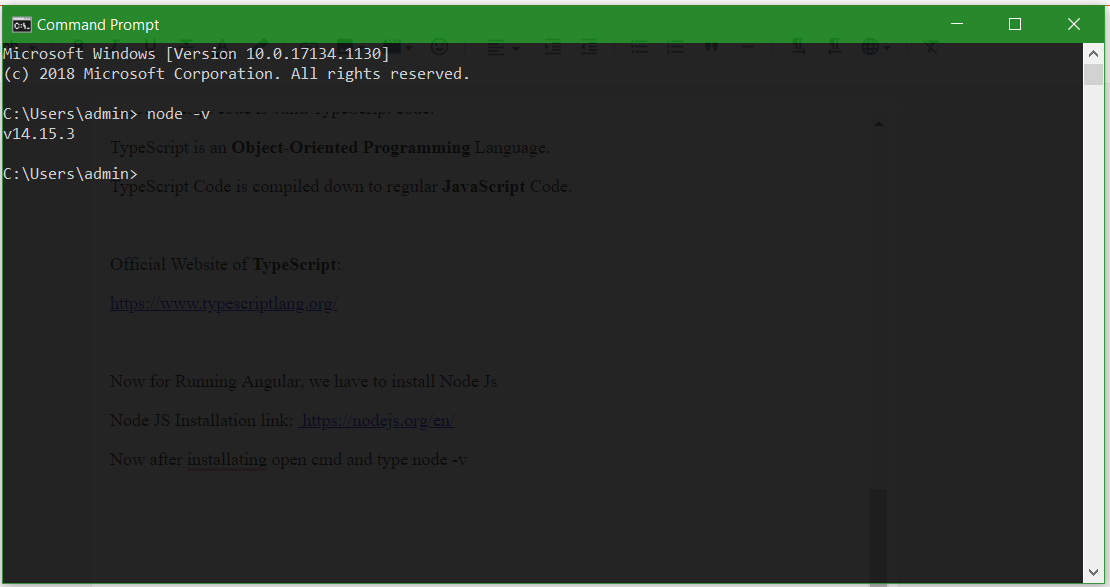
\*\* Typescript is an **Object-Oriented Programming** Language.

Here we use Angular Version 2+ and for using Angular we use Angular CLI (Command Line Interpreter). For using Angular we have to install **Node JS**.

Installation process are described here step by step:

1. We install Node JS from the Installation link: [https://nodejs.org/en/](https://www.blogger.com/blog/post/edit/7036150795473817630/5754529785545654599) for using **Angular CLI** , **Node JS** is mandatory.
2. Now after installation open cmd and type **node –v**. Now Type **npm –v**.

* npm is  **node package manager.**

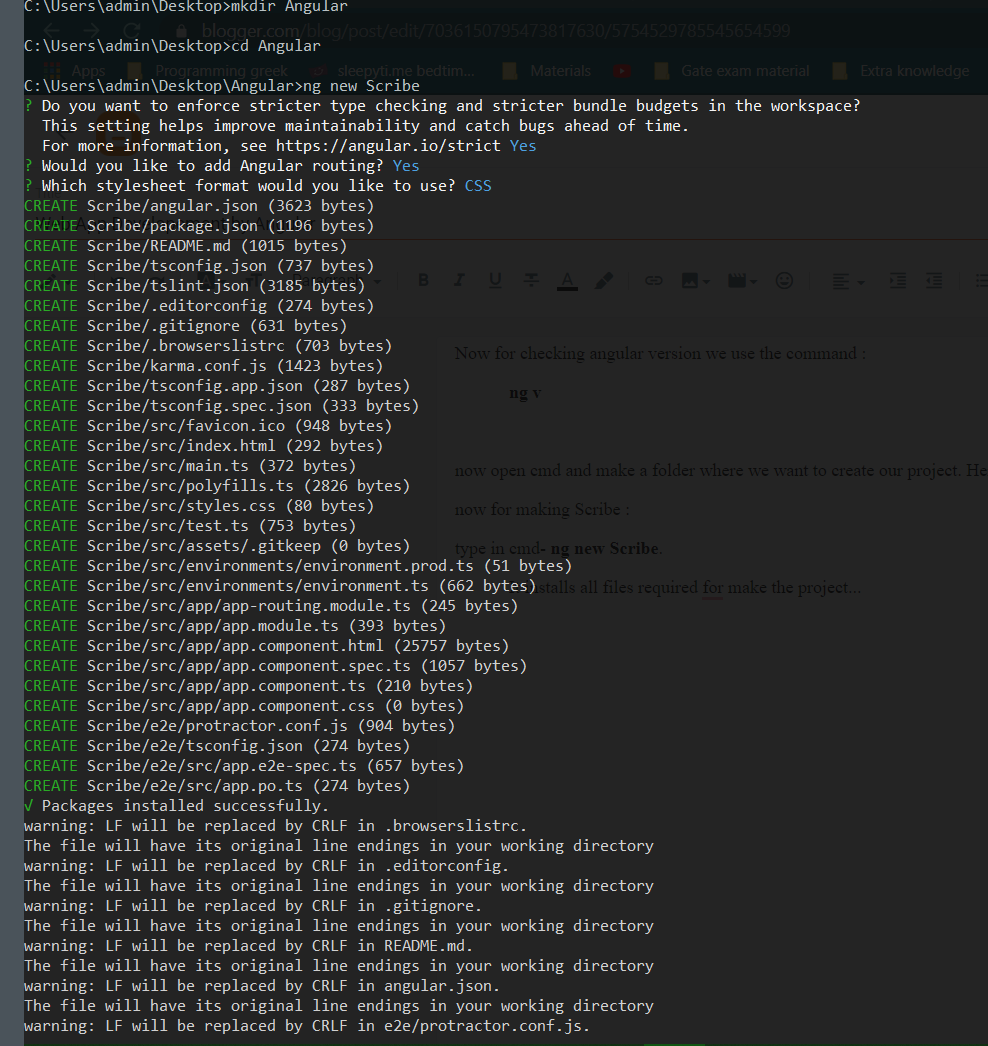


1. Now we install VS-Code from their official website.
2. After that, we install **Angular CLI**- Command Line Interface For Angular.
3. The Angular CLI created the Angular Project with all required files and folders.
4. Angular CLI also allows you to build and test your projects using a local development server.
5. Now To install angular CLI we have to open CMD and type the command:

**npm install -g @angular/cli .**

**-g** flag installs the **@angular/cli** package globally. globally means we can use the angular cli entire our machine not just in a folder.

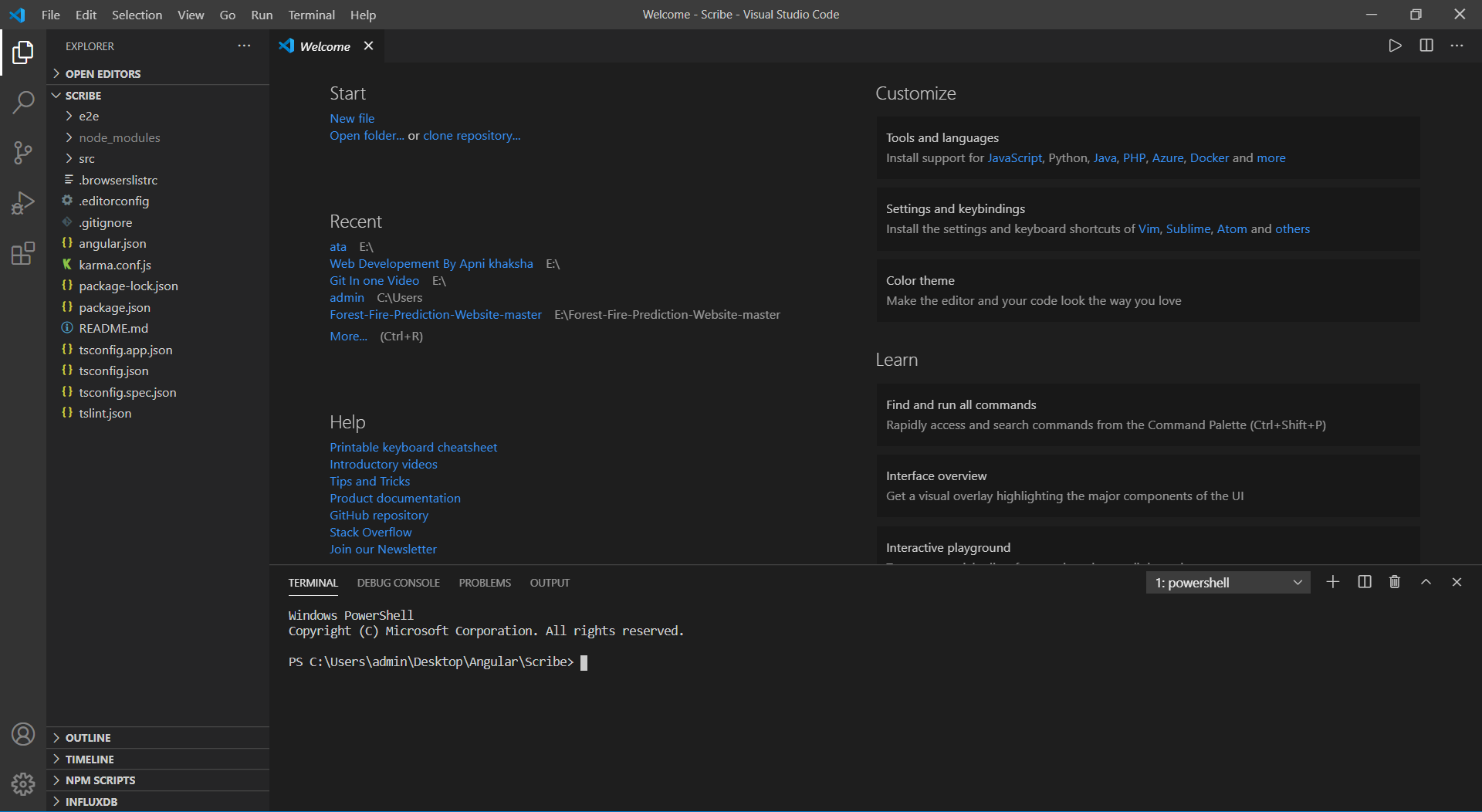
1. Now for checking angular version we use the command: **ng v**
2. Now open cmd and make a folder where we want to create our project. Here we make a website Scribe.
3. Now for making Scribe : type in cmd- **ng new Scribe**.
4. It installs all files required to make the project...

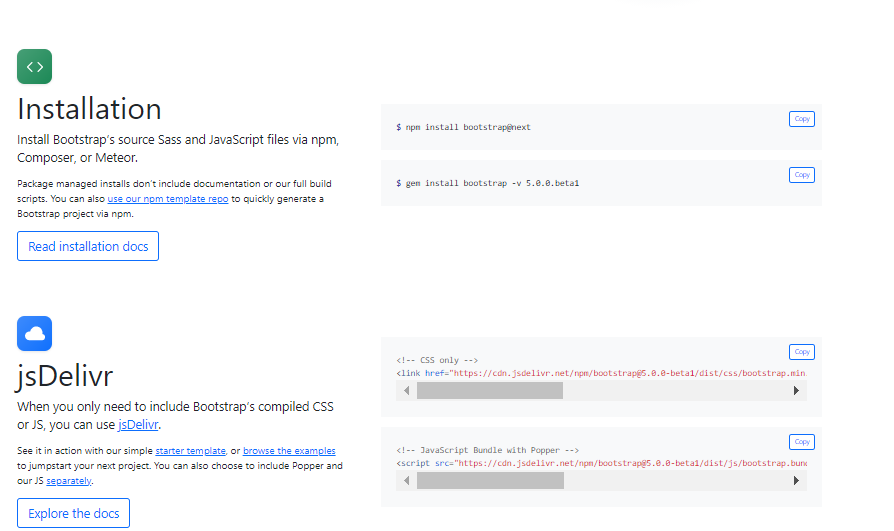
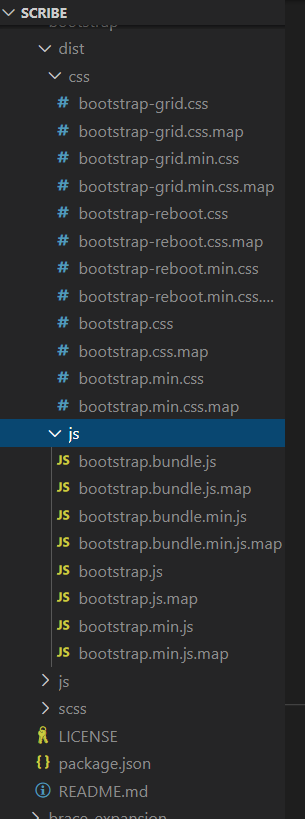


Now for write code, we have to use Visual Studio Code :

  so in **cmd**, we have to write- **code <project name>**

For our case we write: **code Scribe.**

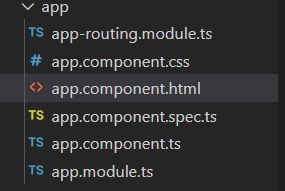
****

1. Now here we can see different folders and files. But for now, we will just work on the src folder.
2. We start with **main.ts** file -- This is a TypeScript File.
3. An angular project is divided into modules. Each module can further be divided into more modules then modules can contain components.
4. Components are the basic building block of any application.
5. We also use Bootstrap here. It allows us to help a user responsive user Interface.
6. Download Bootstrap from their official Sites :
7. Open VS-Code then **View🡪 Terminal**. There we type: npm install **bootstrap jquery --save**  to install bootstrap.
8.  **--save** is used to update the **package.json.file.**
9. Now This is our Bootstrap folder.
10. Now we have to use **bootstrap.min.css** and **bootstrap. min.js**
11. Now we go to **angular.json** file and add



**Now we ready to use Bootstrap using VS-code.**

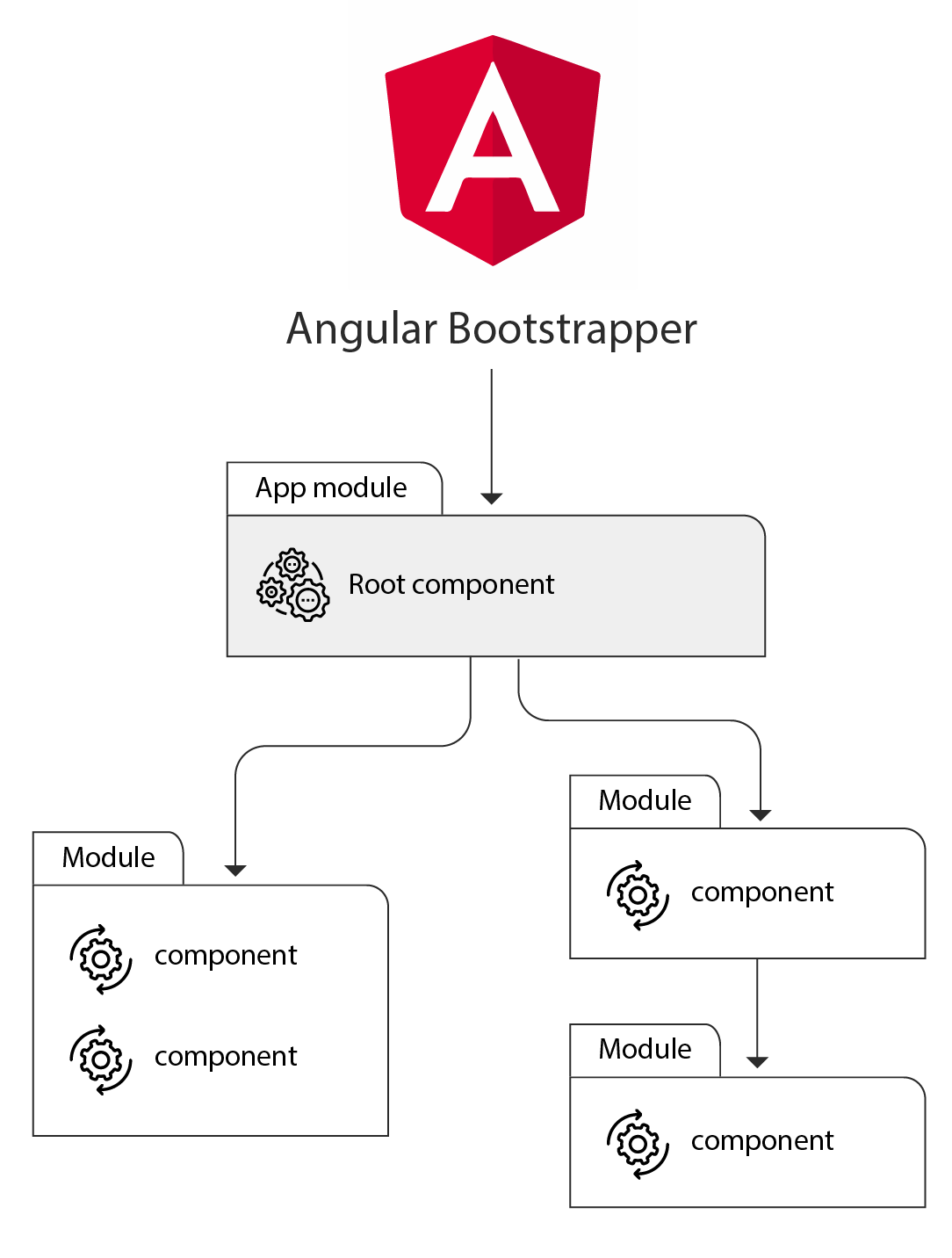
* *Our Main App modules in Angular are:*



|  |  |  |  |
| --- | --- | --- | --- |
| **Disk Footprint** | **200 MB** | **Database** | **Firebase(No-SQL Cloud Database)** |
| **Processor** | **Pentium 4 or newer processor**  **(>1.6GHz fast)**  **SSE3 Capable** | **Platform** | **Windows7, 8,8.1,10 or later**  **Android:Android Lollipop 5.0**  **Mac: OS X**  **El Capitan 10.11 or later** |
| **RAM** | **1GB** |  |  |

**Minimum Hardware Requirements**

**Design**

****

**Angular Components:** Components are the main building blocks for any Angular applications. Each component consists of:

* An **HTML template** that declares what renders on the page.
* A Typescript class that defines behavior.
* A **CSS** selector that defines how the component is used in a template.
* Optionally, CSS styles applied to the template.



**Creating Components:**

To create a component using the Angular CLI:

1. From a terminal window, navigate to the directory containing your application.
2. Run the **ng generate component <component-name>** command, where **<component-name>** is the name of your new component.

By default, this command creates the following:

* A folder named after the component.
* A component file**, <component-name>.component.ts**
* A template file, **<component-name>.component.html**
* A CSS file**, <component-name>.component.css**
* A testing specification file, **<component-name>.component.spec.ts**

Where **<component-name>** is the name of your component.

Now We Design the model of Our components which we will use further as routing purpose & Make our App fully ready.

If you open up the **app.module.ts**  file, it has some libraries which are imported and also a declarative which is assigned the appcomponent as follows –

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule

],

providers: [],

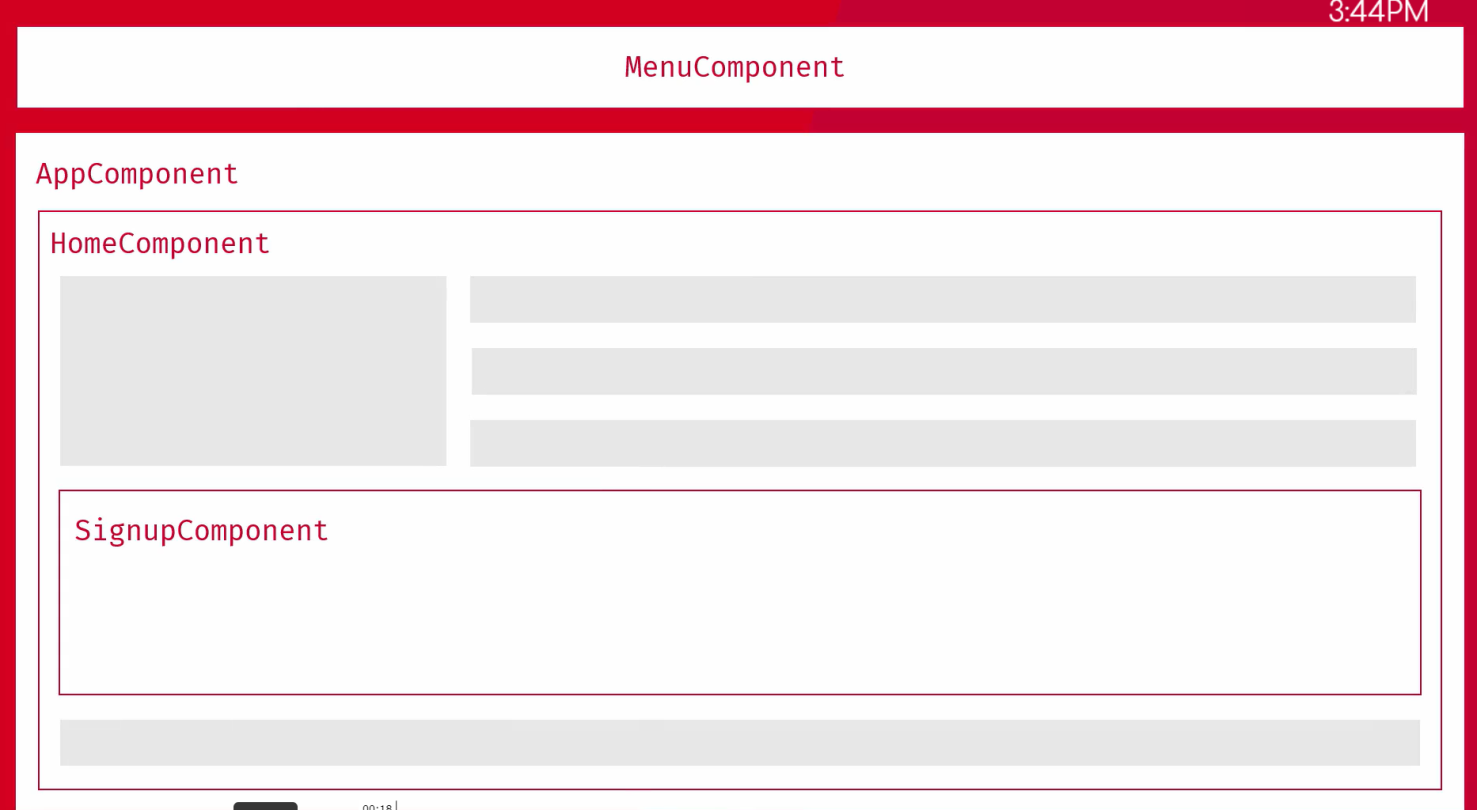
bootstrap: [AppComponent]

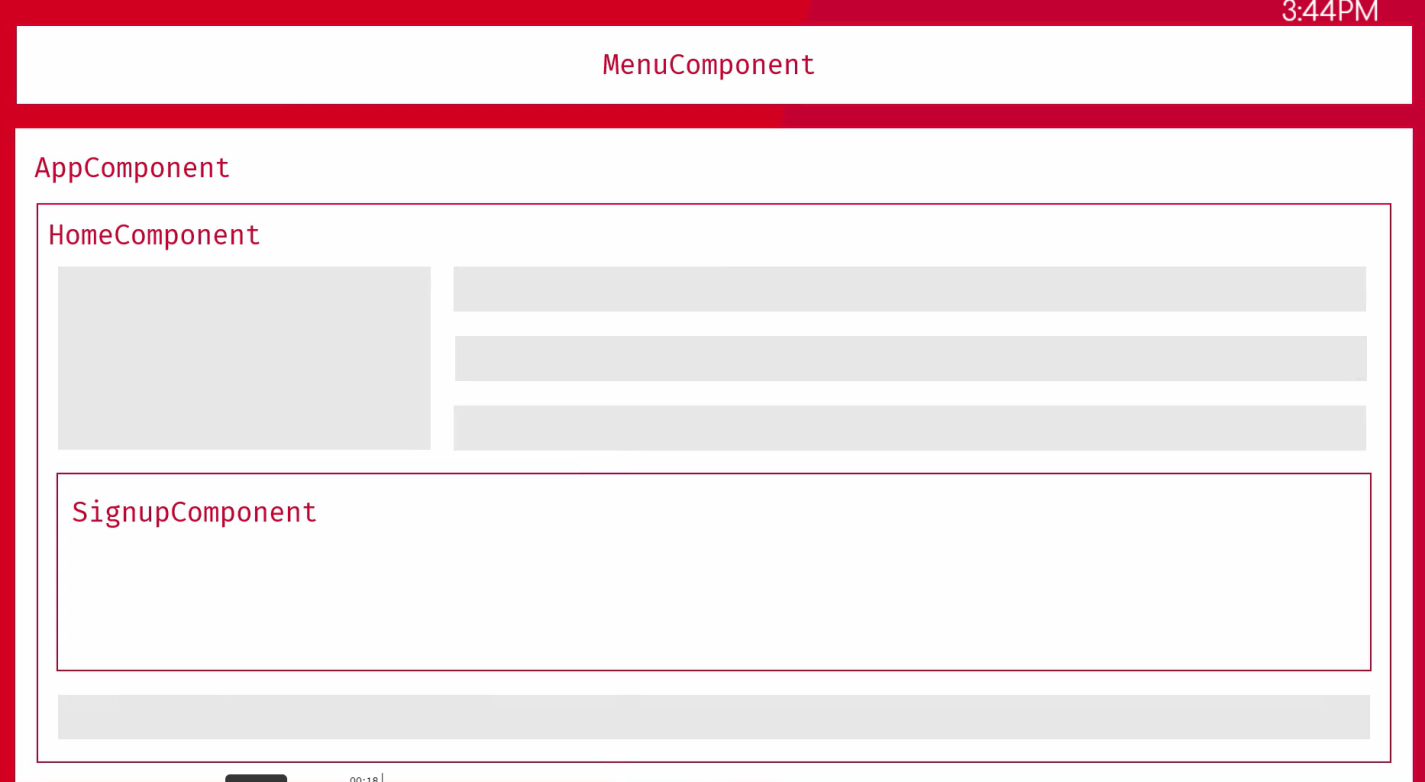
})

export class AppModule { }

**Design Of Menu Component**

The Components of App-Component will keep on changing using Routing. It depends where the user want to be in the Application. We can see a Menu-bar on the top which is essentially a Menu component. The Menu component will change the menu items and it will change different menu-items based on whether the User is logged in or not.

**Design Of App Component**

Below the menu-component , there is App-Component and this is where the rest of the application were left. The Contents inside the App-Component will keep on changing using Routing depending upon where the user is in our Web-App.

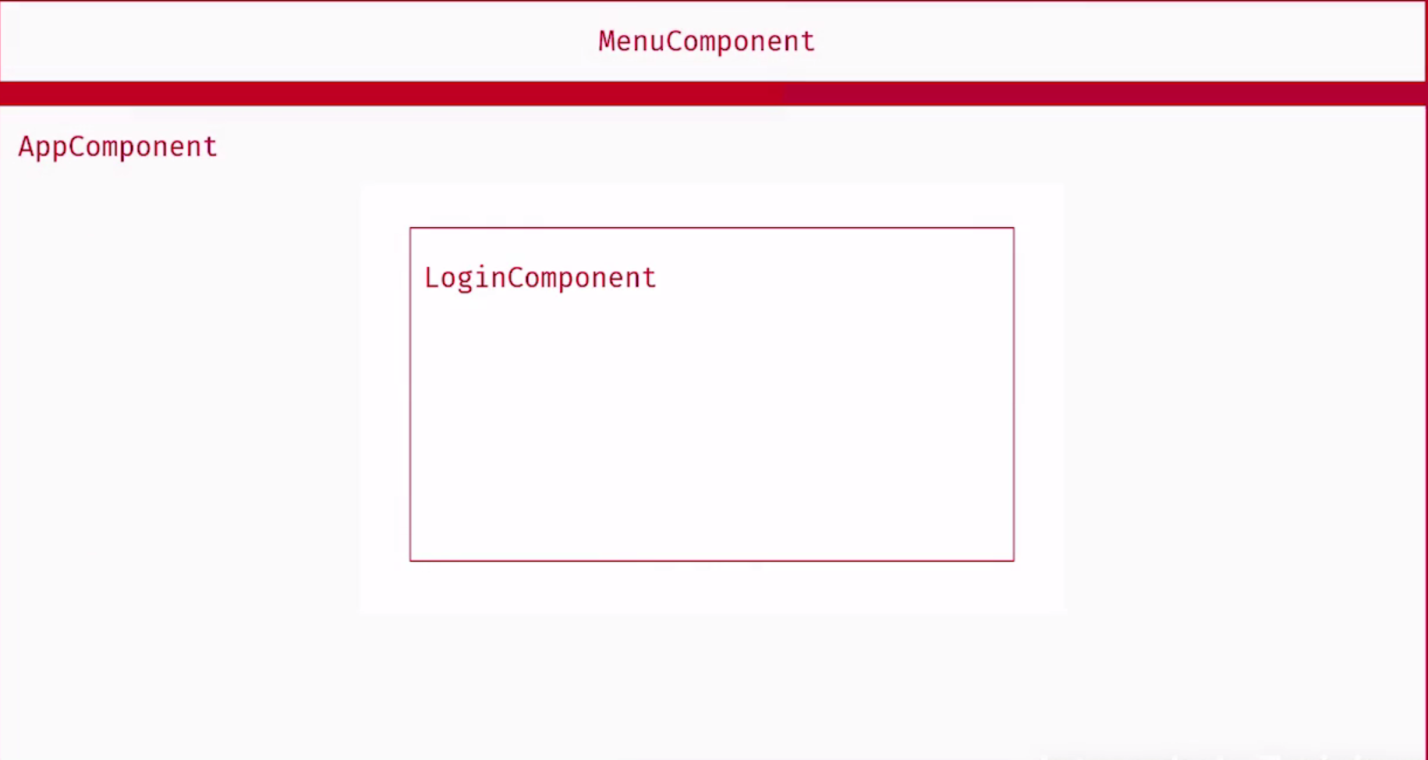
**Design Of Home & Sign-UP Component**

When Our Application Loads Up user will present on the Home Route. This will Present the Home component to the user which will obviously loaded inside the app-component. Home Component will contain some information about our Project, text and pictures that will make the application look good. Here will also display the signup component which will let user create their personal Account.

If the user already have an account they will click the Login button and user will be taken to Login Routes.

**Design Of Login Component**

Login Component will allow the users to Login in their personal account by using a valid email id and password.



Login Component

Valid Password

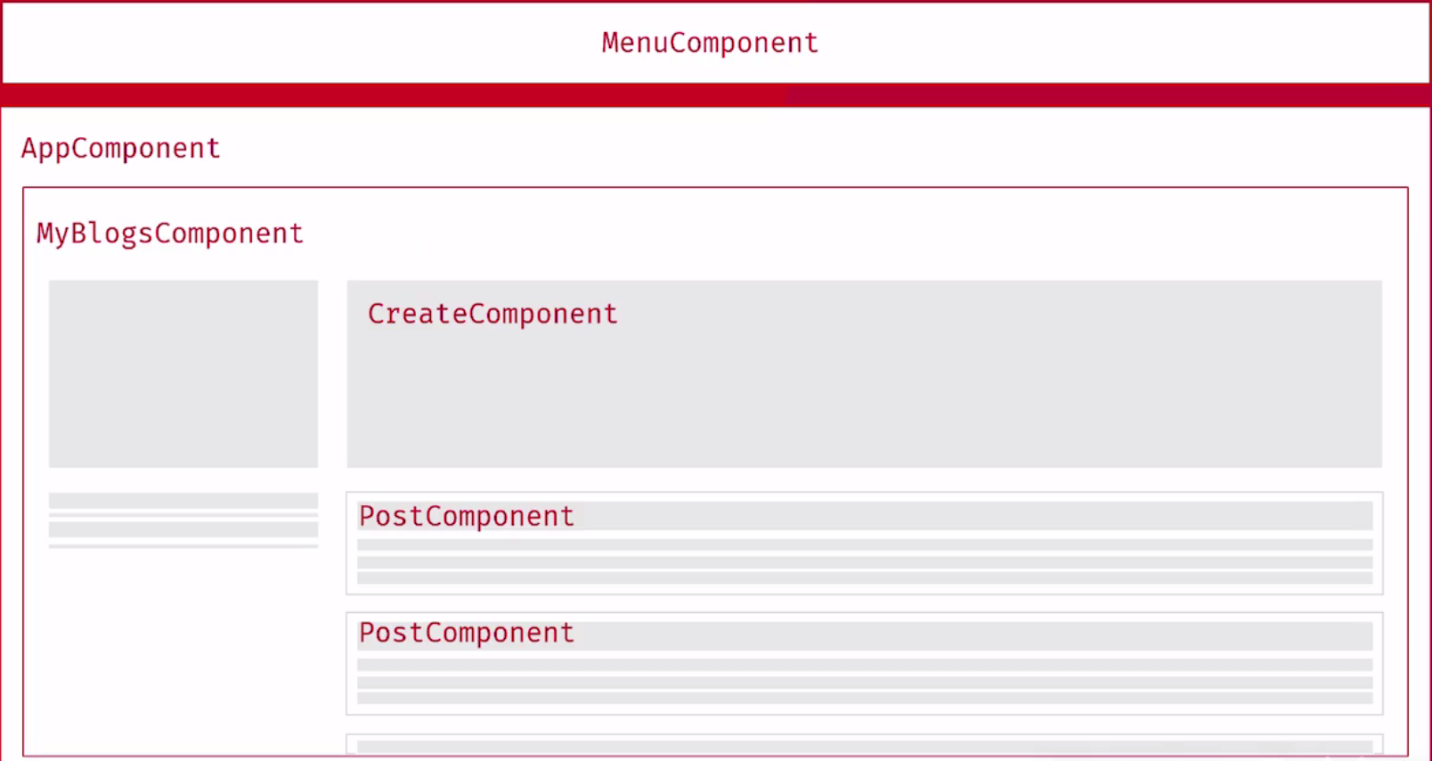
Valid Email Id

**Design Of MyBlogs Component**

Now after Login or Signup will route to MyBlogs Component. Now MyBlogs component have details about the user on the left and on the right it will contain **Create component**. About Create Component will discuss on further pages.

Actually Create component used to create a blogs or post by the User. After that Create Component there is Post Component where we can see all posts created by other bloggers.

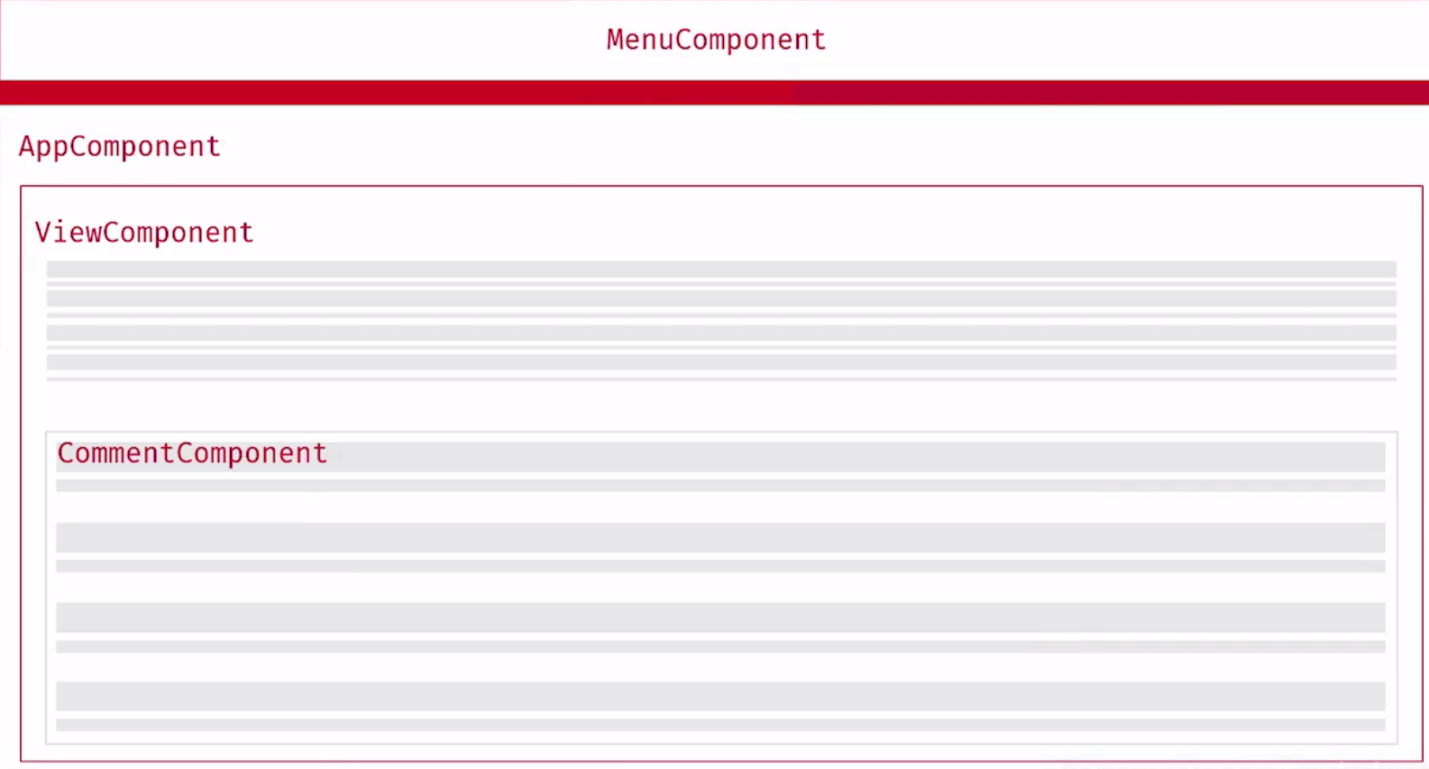
Here we use ngFor directives so that we have one post component for each post.

****

**Design Of View & Comment Component**

If the post component belongs more than 10-words in post data then there is an option **<view more>**. After clicking view more we can see the full post data.

Now view component also contains another component called the Comment Component. This Comment Component will not only allow other users to post comments on this post but will also display all the comments made so far. That’s pretty much for **view and comment** component.

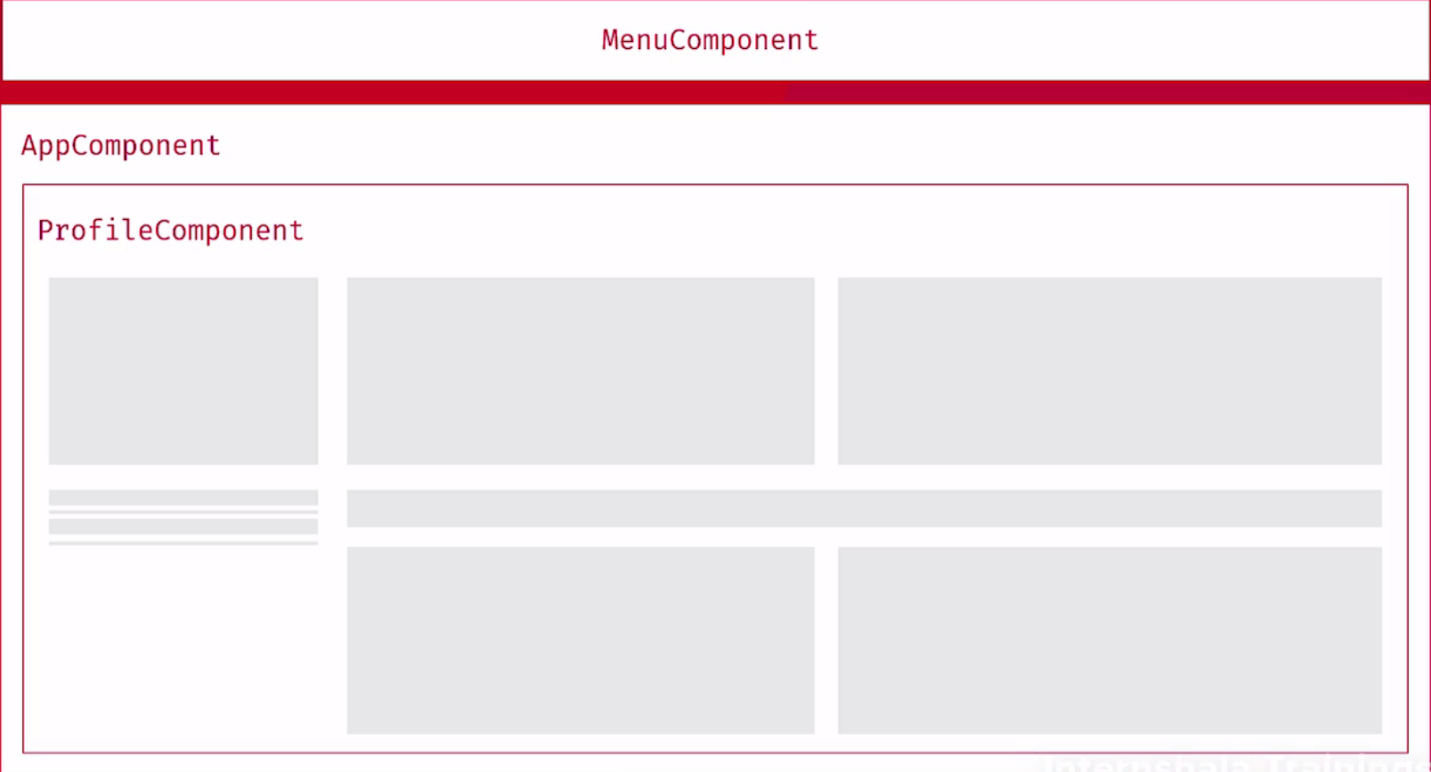


****



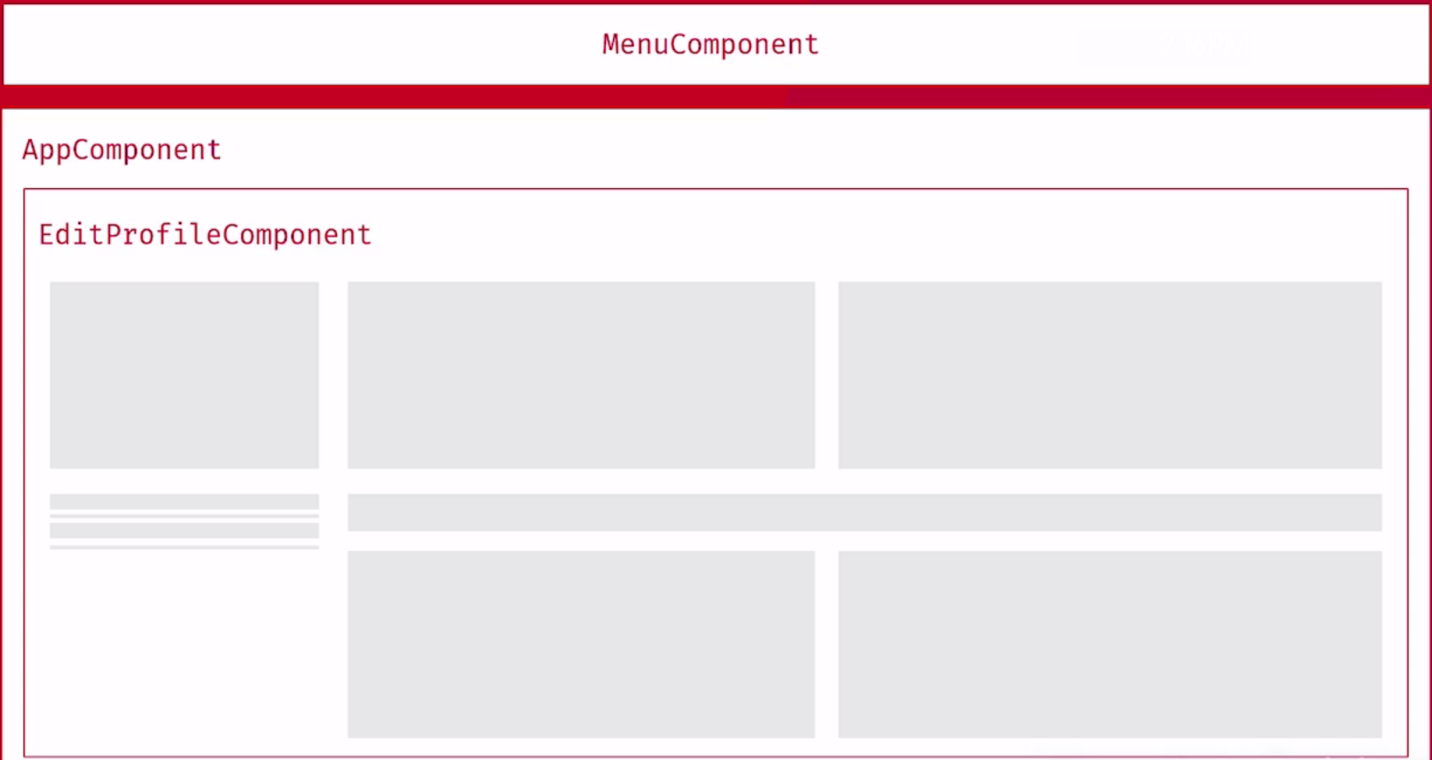


**Design Of User Profile**

Other than previous core functionalities, we are also going to have a profile system for each users. A user can visit the Profile Component using the Profile Routes and their they can see all the information about any user in the application. We will display the link of visit writer’s profile below the every post. From here you can check any bloggers profile if you like their post & also every user have their personal feed that contains their personal data, info & their all posts.



**Design Of Edit-User Profile**

When you login to your account then only you can access that edit-user Profile component. Here we can add extra information to our profile and edit previous given data like as-Name, Bio etc.

* Now In angular this modules routes as per given condition given in Typescript and Service components.

**Our Database(Google Firebase)**

**Firebase** is a Backend-as-a-Service(Baas). It is a Platform as a Service. It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure. The Best Part of this Google Firebase- It is a No SQL database.



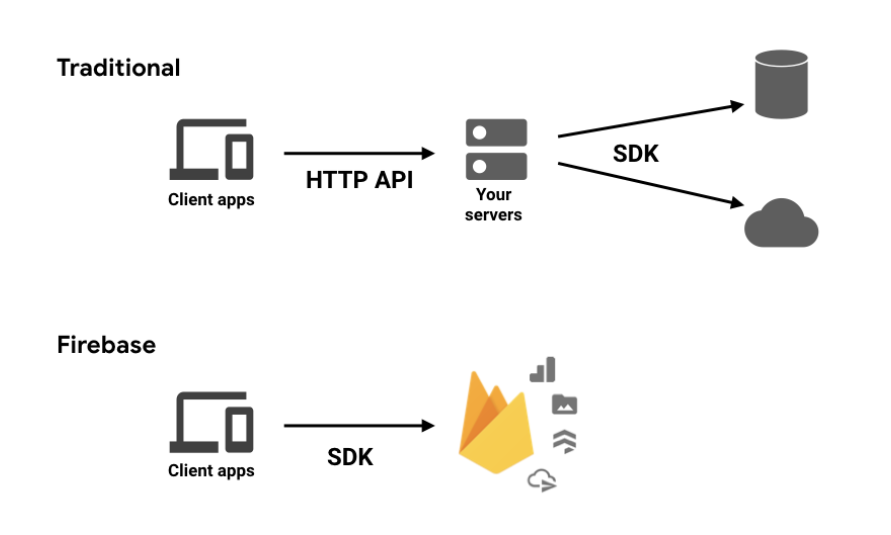
In Firebase, a document is a set of key-value pairs defined by a schema. A group of documents makes up a collection.



***Main Features Of Firebase :***

1. **Authentication** — user login and identity.
2. **Realtime Database** — real-time, cloud hosted, No-SQL database.
3. **Cloud Fire-store** — real-time, cloud hosted, No-SQL database.
4. **Cloud Storage** — massively scalable file storage.
5. **Cloud Functions** — “serverless”, event driven backend.
6. **Firebase Hosting** — global web hosting.
7. **ML Kit** —SDK for common ML tasks.

***Why we prefer Firebase over Traditional SQL Database :***



***Implementation***

*Modules & Modules Description :*

Module in Angular refers to a place where you can group the components, directives, pipes, and services, which are related to the application. In case we are developing a website, the header, footer, left, center and the right section become part of a module. To define module, we can use the **NgModule**. When you create a new project using the **Angular –cli command**, the ***ngmodule*** is created in the **app.module.ts** file by default and it looks as follows –

The **NgModule** needs to be imported as follows –

import { NgModule } from '@angular/core';

The structure for the **ngmodule** is as shown below −

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

It starts with @NgModule and contains an object which has declarations, import s, providers and bootstrap.

**Declaration**

It is an array of components created. If any new component gets created, it will be imported first and the reference will be included in declarations as shown below −

declarations: [

AppComponent,

NewCmpComponent

]

**Import**

It is an array of modules required to be used in the application. It can also be used by the components in the Declaration array. For example, right now in the @NgModule we see the Browser Module imported. In case your application needs forms, you can include the module as follows –

import { FormsModule } from '@angular/forms';

The import in the **@NgModule** will be like the following –

imports: [

BrowserModule,

FormsModule

]

**Providers**

This will include the services created.

**Bootstrap**

This includes the main app component for starting the execution.

\*\* Angular is mainly based on **components**, so here We will further discuss about our **App building Components.**

***Menu Component:***

Our menu changes itself as the user logs in and logs out from our app. Now every component has three basic files where we work:

* menu.component.css
* menu.component.ts
* menu.component.html

and in every component we have to import Firebase because that is our primary database.so we import it in **menu.component.ts** file:

import \* as firebase from 'firebase/app';

import 'firebase/auth';

Now we write our **menu.component.html** file for making the basic building blocks with bootstraps classes.

<nav class="navbar navbar-expand-lg navbar-dark bg-primary">

  <a class="navbar-brand" href="#">

    <img src="assets/Scribe-Logo.png">

  </a>

  <button class="navbar-toggler" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

    aria-expanded="false" aria-label="Toggle navigation">

    <span class="navbar-toggler-icon"></span>

  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">

    <ul class="navbar-nav mr-auto">

      <li class="nav-item" routerLinkActive="active" data-toggle="collapse"

data-target=".navbar-collapse" \*ngIf= "!loggedIn">

        <a class="nav-link" routerLink="/home">Home</a>

      </li>

      <li class="nav-item" routerLinkActive="active" data-toggle="collapse"

 data-target=".navbar-collapse" \*ngIf= "loggedIn">

        <a class="nav-link" routerLink="/myblogs">Home</a>

      </li>

    </ul>

    <ul class="navbar-nav ml-auto">

      <li class="nav-item right" \*ngIf= "!loggedIn" routerLinkActive="active" data-toggle="collapse" data-target=".navbar-collapse">

        <button class="btn btn-outline-light px-5" routerLink="/login">Login</button>

      </li>

      <li class="nav-item right" \*ngIf="loggedIn">

        <a class="nav-link" [routerLink]= "'/profile/' + user.uid">My Profile

</a>

      </li>

      <li class="nav-item right" \*ngIf= "loggedIn">

        <a class="nav-link" routerLink="/" (click)= "logout()">Log out</a>

      </li>

    </ul>

  </div>

</nav>

Now write the logical section in typescript files . In typescript file we actually import Firebase and routing everything.

import { Component, OnInit } from '@angular/core';

import \* as firebase from 'firebase/app';

import 'firebase/auth';

@Component({

  selector: 'app-menu',

  templateUrl: './menu.component.html',

  styleUrls: ['./menu.component.css']

})

export class MenuComponent implements OnInit {

  loggedIn: boolean = false;

  user: any;

  constructor() {

    this.user = firebase.auth().currentUser;

    if(this.user) {

      this.loggedIn = true;

    } else {

      this.loggedIn = false;

    }

    firebase.auth().onAuthStateChanged((user) => {

      this.user = user;

      if(user){

        this.loggedIn = true;

      } else {

        this.loggedIn = false;

      }

    })

  }

  ngOnInit() {

  }

  logout(){

    firebase.auth().signOut();

  }

}

Now we add some css class in **menu.component.css** file for add some extra styling:

.btn-outline-light {

    margin-left: 10px;

    border-radius: 50px;

}

nav {

    border-bottom: 1px solid #6ca3f888;

}

***Login Component:***

When we press the Login button then will route us to Login Component. Now Login component have same 3 modules/files:

* login.component.ts
* login.component.css
* login.component.html

Now in same manner firstly, we have to build up the Login page using **html:**

<div class="row background">

  <div class="offset-sm-4 col-sm-4">

    <div class="card text-blank">

      <div class="card-body">

        <div class="card-title text-center"><h5>Login using your

email</h5></div>

        <div class="card-text">

          <form [formGroup]= "myForm" (ngSubmit)= "onSubmit(myForm)">

            <div class="form-group">

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

[formControl]= "myForm.controls['email']">

            </div>

            <div class="alert alert-danger" \*ngIf= "!myForm.controls['email'].valid &&

myForm.controls['email'].touched">

              A valid Email is required.

            </div>

            <div class="form-group">

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

 [formControl]= "myForm.controls['password']">

            </div>

            <div class= "alert alert-danger" \*ngIf= "!myForm.controls['password'].valid &&

myForm.controls['password'].touched">

              Password is required.

            </div>

            <div class="text-center">

              <button type="submit" class="btn btn-block btn-warning round" [disabled]= "!myForm.valid">Submit</button>

            </div>

          </form>

          <div class="text-center mt-3">

            <button class="btn btn-link" routerLink="/home">Don't have an account? Create One</button>

          </div>

        </div>

      </div>

    </div>

    <div class="alert alert-success" \*ngIf="message.length > 0">

      You have been logged in successfully.

    </div>

    <div class="alert alert-danger" \*ngIf="userError">

      {{ userError.message }}

    </div>

  </div>

</div>

Then develop the typescript file **login.component.ts** :

import { Component, OnInit } from '@angular/core';

import { FormGroup, FormBuilder, Validators } from '@angular/forms';

import { AuthService } from '../auth.service';

import { Router } from '@angular/router';

@Component({

  selector: 'app-login',

  templateUrl: './login.component.html',

  styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

  myForm: FormGroup;

  message: string = "";

  userError: any;

  constructor(public fb: FormBuilder, public authService: AuthService,

public router: Router) {

    this.myForm = this.fb.group({

      email: ['', [Validators.email, Validators.required]],

      password: ['', [Validators.required]]

    })

  }

  ngOnInit() {

  }

  onSubmit(form){

    this.authService.login(form.value.email,

form.value.password).then((data) => {

      console.log(data);

      this.message = "You have been logged in successfully."

      this.userError = null;

      this.router.navigate(['/myblogs'])

    }).catch((error) => {

      console.log(error);

      this.message = null;

      this.userError = error;

    })

  }

}

Atlast, for adding some extra background colors and styles we us css file **login.component.css:**

.background {

    background-color: #48D1CC;

    height: 100%;

}

.card {

    background-color: transparent !important;

    color: #fff;

    border: none;

    margin-top: 35%;

}

.btn[type=submit]{

    border-radius: 50px;

}

.btn-link {

    color: #fff;

}

h1 {

    color: #fff;

}

.vertical-divider{

    border-right:1px solid #ddd;

}

First I go to my home component and open **home.component.html** and their write the Signup Selector:

<app-signup></app-signup>

Then come to our signup component and write the basic html building blocks:

<div class="row background">

  <div class="col-sm-6 background-left p-5 pb-5">

    <h1>Don't just think! Blog It!</h1>

  </div>

  <div class="col-sm-6 background-right p-5 pb-5">

    <div class="row">

      <div class="offset-sm-2 col-sm-8">

        <div class="card text-blank border-0">

          <div class="card-body">

            <div class="card-title text-center">

              <h5>Sign up for free</h5>

            </div>

            <div class="card-text">

              <form [formGroup]= "myForm" (ngSubmit)= "onSubmit(myForm)">

                <div class="form-row">

                  <div class="form-group col-sm-6">

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

[formControl]= "myForm.controls['firstName']">

                  </div>

                  <div class="form-group col-sm-6">

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

[formControl]= "myForm.controls['lastName']">

                  </div>

                </div>

                <div class="alert alert-danger" \*ngIf="(!myForm.controls['firstName'].valid ||

!myForm.controls['lastName'].valid) &&

(myForm.controls['firstName'].touched ||

 myForm.controls['lastName'].touched)">

                  First and Last name are required

                </div>

                <div class="form-group">

                  <input type="email" placeholder="Enter your email"

class="form-control" [formControl]="myForm.controls['email']">

                  <small class="form-text text-muted">

                    We'll never share your email with anyone else

                  </small>

                </div>

                <div class="alert alert-danger" \*ngIf= "!myForm.controls['email'].valid &&

 myForm.controls['email'].touched">

                  A valid email is required

                </div>

                <div class="form-group">

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

[formControl]= "myForm.controls['password']">

                </div>

                <div class="form-group">

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

[formControl]= "myForm.controls['confirmPassword']">

                </div>

                <div class="alert alert-danger" \*ngIf= "!myForm.controls['password'].valid &&

myForm.controls['password'].touched">

                  Password should be atleast 8 characters long.

                </div>

                <div class="text-center">

                  <button type="submit" class="btn btn-block btn-warning" [disabled]= "!myForm.valid">Sign Up</button>

                </div>

              </form>

            </div>

          </div>

        </div>

        <div class="alert alert-success" \*ngIf= "message.length > 0">

          {{ message }}

        </div>

        <div class="alert alert-danger" \*ngIf= "userError">

          {{ userError.message }}

        </div>

      </div>

    </div>

  </div>

</div>

Then Open **signup.component.ts** file and import router,firebase & firebase authentication. As we are making Signup form we also have to import the formbuilder, formgroup,f ormcontrol and validators.

import { FormBuilder, FormGroup, FormControl, Validators } from '@angular/forms';

import { AuthService } from '../auth.service';

import \* as firebase from 'firebase/app';

import 'firebase/firestore';

import { Router } from '@angular/router';

now we have to build the form and collect every data and have to store that in Firebase:

import { Component, OnInit } from '@angular/core';

import { FormBuilder, FormGroup, FormControl, Validators } from '@angular/forms';

import { AuthService } from '../auth.service';

import \* as firebase from 'firebase/app';

import 'firebase/firestore';

import { Router } from '@angular/router';

@Component({

  selector: 'app-signup',

  templateUrl: './signup.component.html',

  styleUrls: ['./signup.component.css']

})

export class SignupComponent implements OnInit {

  myForm: FormGroup;

  message: string = "";

  userError: any;

  constructor(public fb: FormBuilder, public authService: AuthService, public router: Router) {

    this.myForm = this.fb.group({

      firstName: ['', [Validators.required]],

      lastName: ['', [Validators.required]],

      email: ['', [Validators.required]],

      password: ['', [Validators.required, Validators.minLength(8)]],

      confirmPassword: ['', [Validators.required]]

    }, {

      validator: this.checkIfMatchingPasswords("password", "confirmPassword")

    })

  }

  checkIfMatchingPasswords(passwordKey: string, confirmPasswordKey: string){

    return (group: FormGroup) => {

      let password = group.controls[passwordKey];

      let confirmPassword = group.controls[confirmPasswordKey];

      if(password.value == confirmPassword.value){

        return;

      } else {

        confirmPassword.setErrors({

          notEqualToPassword: true

        })

      }

    }

  }

  onSubmit(signupform){

    let email: string = signupform.value.email;

    let password: string = signupform.value.password;

    let firstName: string = signupform.value.firstName;

    let lastName: string = signupform.value.lastName;

    this.authService.signup(email, password, firstName, lastName).then((user: any) => {

      firebase.firestore().collection("users").doc(user.uid).set({

        firstName: signupform.value.firstName,

        lastName: signupform.value.lastName,

        email: signupform.value.email,

        photoURL: user.photoURL,

        interests: "",

        bio: "",

        hobbies: ""

      }).then(() => {

        this.message = "You have been signed up successfully.";

        this.userError = null;

        this.router.navigate(['/myblogs'])

      })

    }).catch((error) => {

      console.log(error);

      this.userError = error;

    })

  }

  ngOnInit() {

  }

}

Now for some Extra styling and background puposes we have to use **CSS** **:**

.background-left {

    background-color: #FAFAD2;

    background-image: url('/assets/Sign-Up-BG-Image.png');

    background-repeat: no-repeat;

    background-size: contain;

    background-position: left bottom;

}

.background-right {

    background-color: #FAFAD2;

}

.card {

    background-color: transparent;

    color: #000;

}

.text-muted{

    color: #191970 !important;

}

.btn[type=submit]{

    border-radius: 50px;

}

h1 {

    color: #F08080;

}

***Home Component:***

Now We have added some Pics, texts to our home component so at first we write the Html File.

<app-signup></app-signup>

<div class="row">

  <div class="col-sm-6 text-right pt-5">

    <img class="align-middle m-5" height="300px" src="/assets/What-Is-Scribe-Image.png">

  </div>

  <div class="col-sm-4">

    <h2 class="mx-5 my-5">What is Scribe?</h2>

    <h6 class="mx-5 my-1">

      Scribe is a thought sharing revolutionary platform where you can

share your thoughts and craziest ideas with the world. You can also

interact with other users by commenting on their posts.

      <br/><br/>

      Create a free account and get started.

      <br/><br/>

      Let's get started!

    </h6>

  </div>

</div>

<hr/>

<div class="row">

  <div class="offset-sm-2 col-sm-8 text-center pt-5 mt-5">

    <h5>Contact Us</h5>

    <div class="row">

      <div class="offset-sm-2 col-sm-4 text-left pt-5 vertical-divider mt-5 ">

        <i class="fa fa-phone mt-3"></i> 9123354166

        <br/>

        <i class="fa fa-envelope mt-3"></i>  pralayghoshg9@gmail.com

        <br/>

        <i class="fa fa-copyright mt-3"></i> All Rights reserved by @Yeapralay

        <br/>

      </div>

      <div class="col-sm-6 text-left py-5 my-5 px-5">

        <i class="fa fa-map-marker mt-3"></i> Scribe Pvt. Ltd.<br/>

        <i class="fa fa-map-marker mt-1 invisible"></i> C-809, Business Zone,<br/>

        <i class="fa fa-map-marker mt-1 invisible"></i> Nirvana Apartment, South City 2,<br/>

        <i class="fa fa-map-marker mt-1 invisible"></i> Bangalore, India - 122018<br/>

      </div>

    </div>

  </div>

</div>

**\*\***Now for using **<fa fa-map-marker>** weneed to install a library called- ‘font-awesome’ . so, open terminal and type:

**npm install font-awesome –save**

Now go to **angular.json** & add the following path to:

"styles": [

              "node\_modules/font-awesome/css/font-awesome.min.css",

              "node\_modules/bootstrap/dist/css/bootstrap.min.css",

              "src/styles.css"

            ],

Then we can access the font-style and formats.

Then We write the just one css property on **css** file:

.vertical-divider{

    border-right:1px solid #ddd;

}

***myblogs Component:***

Actually myblogs component contains every blogs that have been posted on our site. For myblogs component we have to write in **myblogs.component.html** :

<div class="row mb-5">

  <div class="col-sm-3 offset-sm-1">

    <div class="card mt-5">

      <img class="card-img-top" [src]="user.photoURL">

      <div class="card-body">

        <h4 class="card-title">{{ user.displayName }}</h4>

        <a routerLink="/edit-profile/{{user.uid}}" class="btn btn-sm btn-primary">Edit Profile</a>

      </div>

    </div>

  </div>

  <div class="col-sm-7">

    <app-create (postCreated)="onPostCreated()"></app-create>

    <app-post \*ngFor="let post of posts" [post]="post" (onDelete)="onDelete()"></app-post>

  </div>

</div>

In **typescript** file we code:

import { Component, OnInit } from '@angular/core';

import \* as firebase from 'firebase/app';

import 'firebase/auth';

import 'firebase/firestore';

@Component({

  selector: 'app-myblogs',

  templateUrl: './myblogs.component.html',

  styleUrls: ['./myblogs.component.css']

})

export class MyblogsComponent implements OnInit {

  user: any = {};

  posts: any[] = [];

  constructor() {

    // firebase.firestore().settings({

    //   timestampsInSnapshots: true

    // });

    this.user = firebase.auth().currentUser;

    console.log(this.user)

    this.getPosts();

  }

  ngOnInit() {

  }

  getPosts(){

    // get the list of posts

    firebase.firestore().collection("posts")

    .orderBy("created", "desc")

    .get().then((querySnapshot) => {

      console.log(querySnapshot.docs);

      this.posts = querySnapshot.docs;

    }).catch((err) => {

      console.log(err);

    })

  }

  onPostCreated(){

    // refresh the list of posts

    this.posts = [];

    this.getPosts();

  }

  onDelete(){

    // refresh the list of posts

    this.posts = [];

    this.getPosts();

  }

}

***create Component:***

First We write basic html code for building the basic of create component.

<div class="row mt-5">

  <div class="col">

    <div class="form-group">

      <label>Post Title</label>

      <input type="text" class="form-control" id="title" placeholder="Post Title" name="title" [(ngModel)]="title">

    </div>

    <div class="form-group">

      <label>Post Content</label>

      <angular-editor [placeholder]="'Enter text here...'" [(ngModel)]= "content"></angular-editor>

    </div>

    <div class="form-group float-right">

      <button (click)="createPost()" class="btn btn-warning round">Publish</button>

    </div>

  </div>

</div>

Then we need Angular **ngx-editor**. **Ngx-editor** is a complete text editor that we can use in our Angular Projects.

Now go to terminal and type : **ngx –editor ngx –bootstrap --save**

then import files in **app.module.ts :**

import { NgxEditorModule } from 'ngx-editor';

import { AngularEditorModule } from '@kolkov/angular-editor';

import { HttpClientModule } from '@angular/common/http';

now also have to import this into Arrays:

  imports: [

    BrowserModule,

    FormsModule,

    ReactiveFormsModule,

    AppRoutingModule,

    NgxEditorModule,

    AngularEditorModule,

    HttpClientModule

  ],

After this, write the **create.component.typescript** file:

import { Component, OnInit, Output, EventEmitter } from '@angular/core';

import \* as firebase from 'firebase/app';

import 'firebase/firestore';

import 'firebase/auth';

@Component({

  selector: 'app-create',

  templateUrl: './create.component.html',

  styleUrls: ['./create.component.css'],

})

export class CreateComponent implements OnInit {

  // editorConfig: any;

  title: string;

  // content: string; changed for ngx-editor 5

  content: any;

  @Output('postCreated') postCreated = new EventEmitter();

  constructor() {}

  ngOnInit() {}

  createPost() {

    // let contentNode = schema.nodeFromJSON(this.content);

    // let html: DocumentFragment = DOMSerializer.fromSchema(

    //   schema

    // ).serializeFragment(contentNode.content);

    firebase

      .firestore()

      .collection('posts')

      .add({

        title: this.title,

        content: this.content,

        owner: firebase.auth().currentUser.uid,

        created: firebase.firestore.FieldValue.serverTimestamp(),

      })

      .then((data) => {

        console.log(data);

        this.postCreated.emit();

      })

      .catch((error) => {

        console.log(error);

      });

  }

}

Atlast, adding one small property of css for better looking:

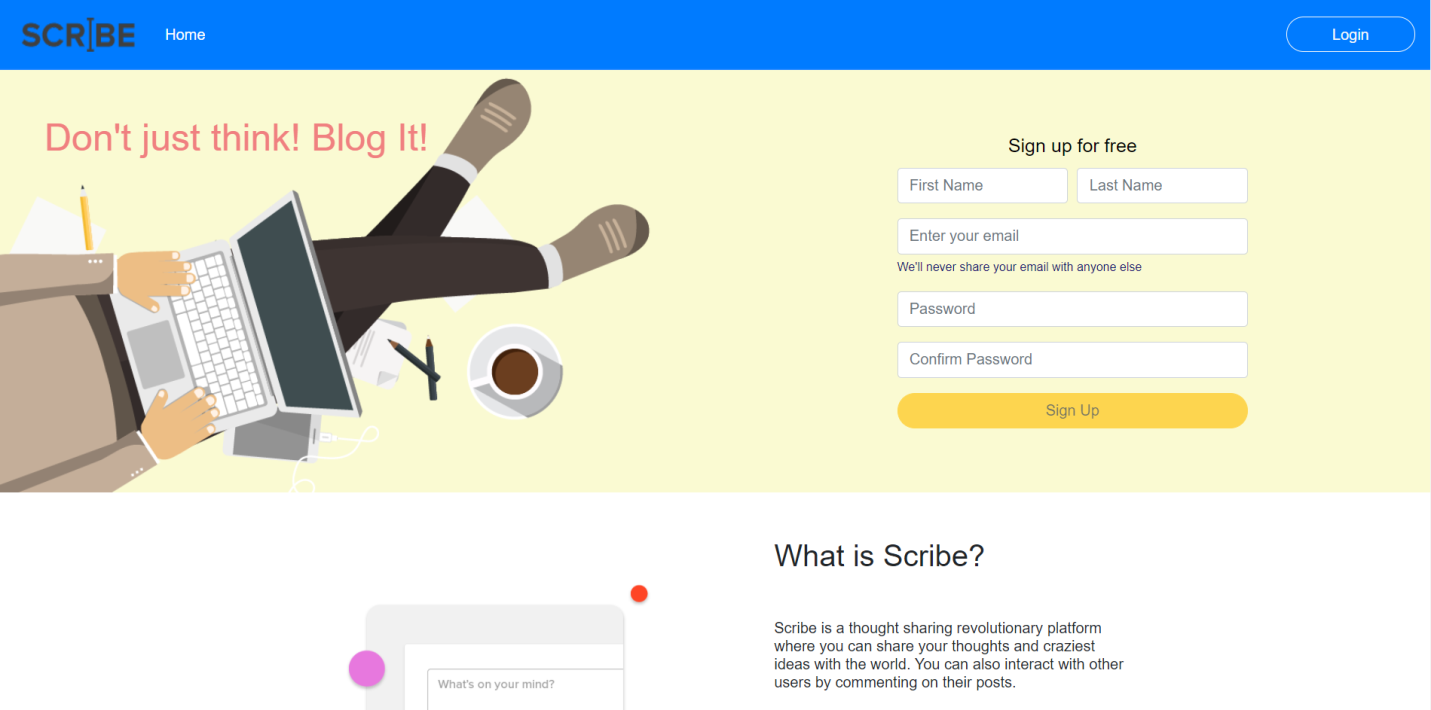
.round {

    border-radius: 50px;

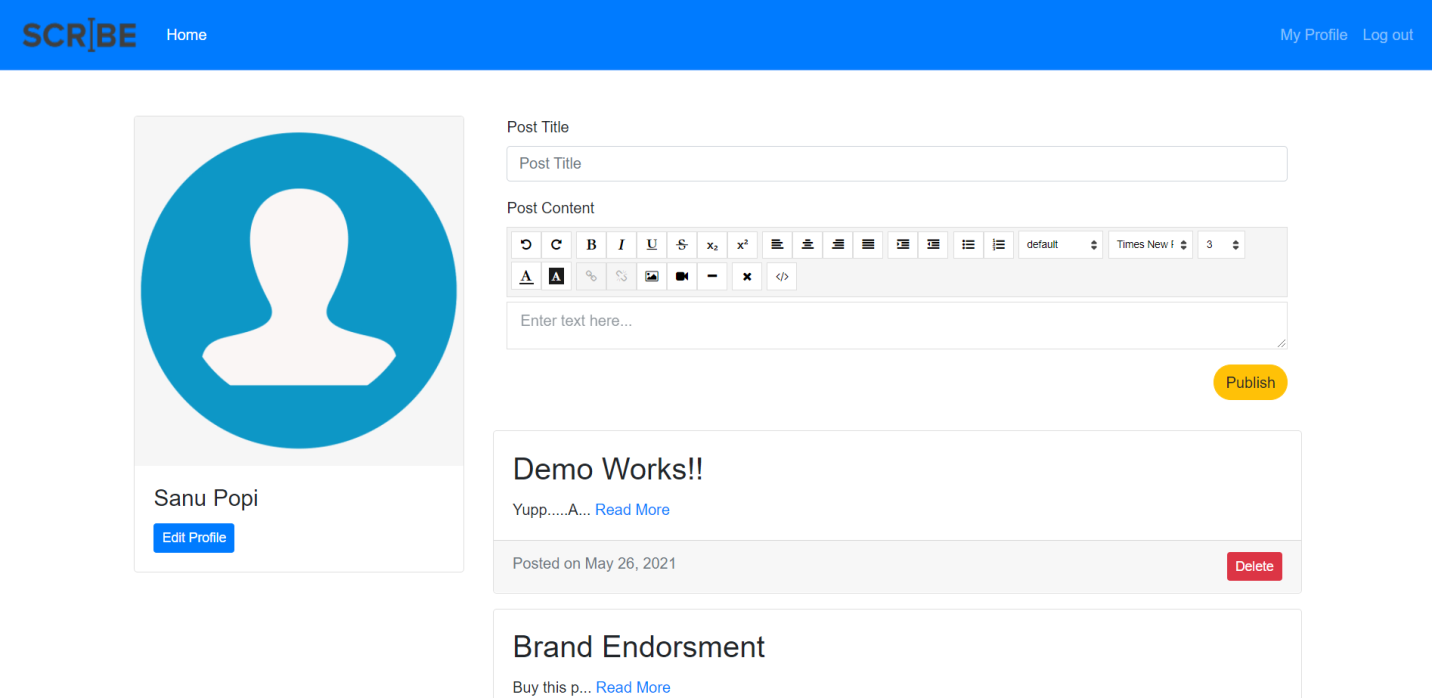
}

***Experimental Results***

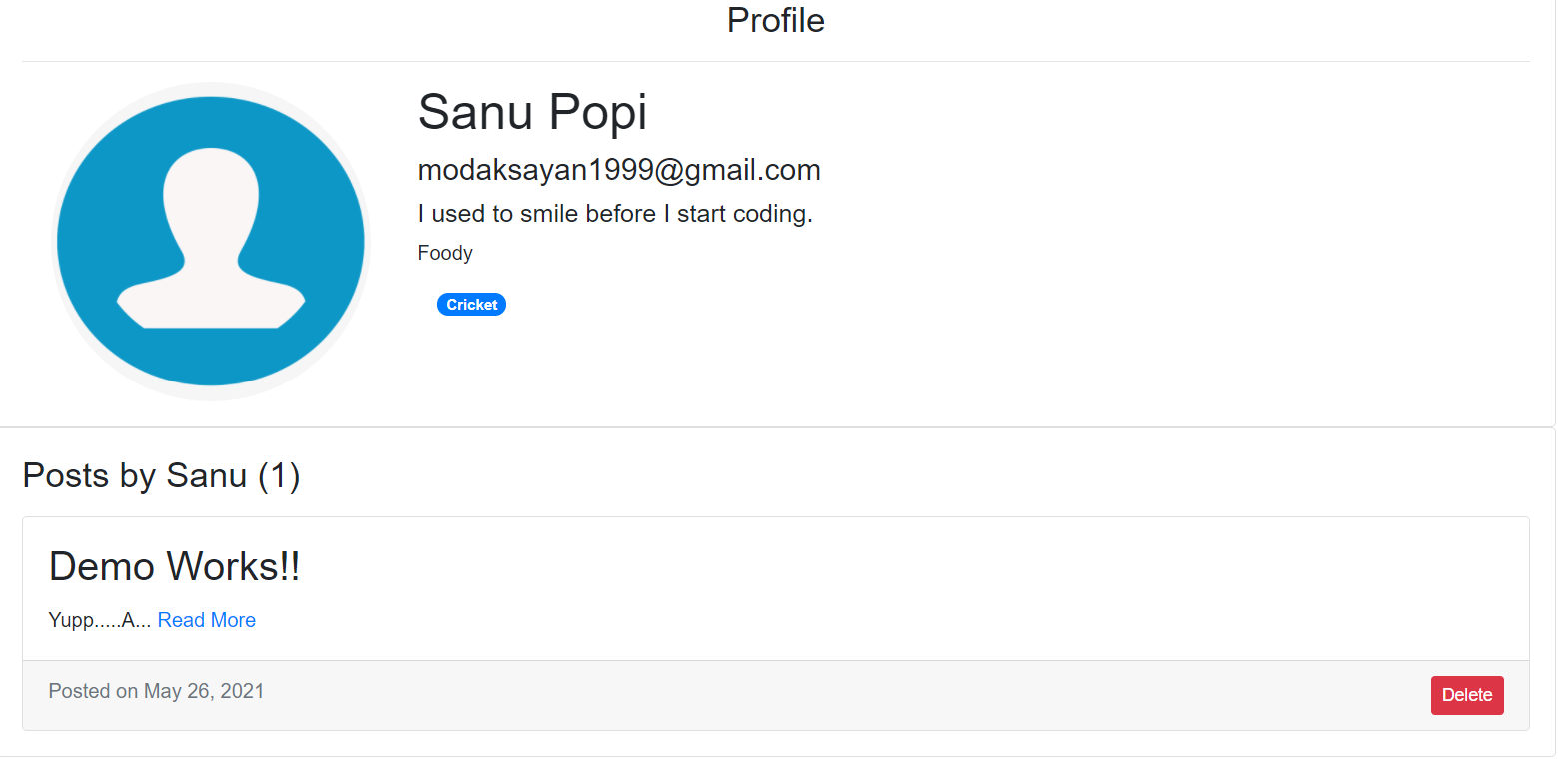
Test Results & Scrennshots:

Home Page For Login & Sign-Up

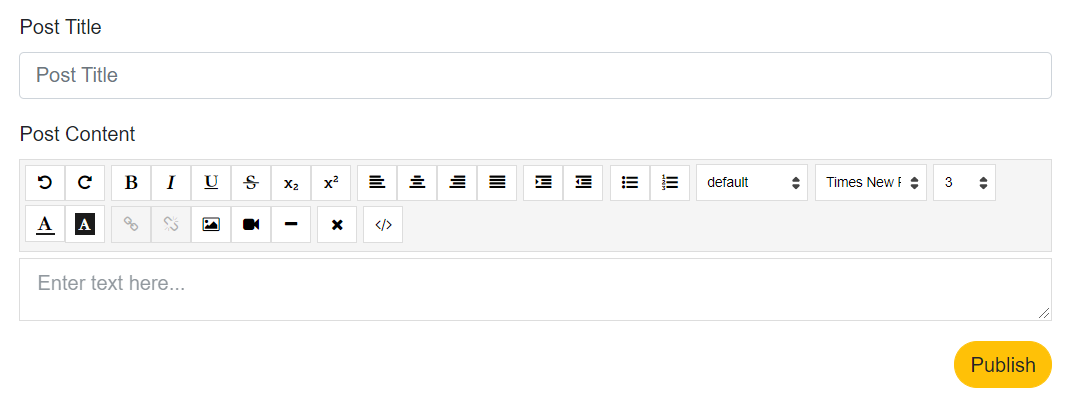
Blog-Feed Page



My Profile Page



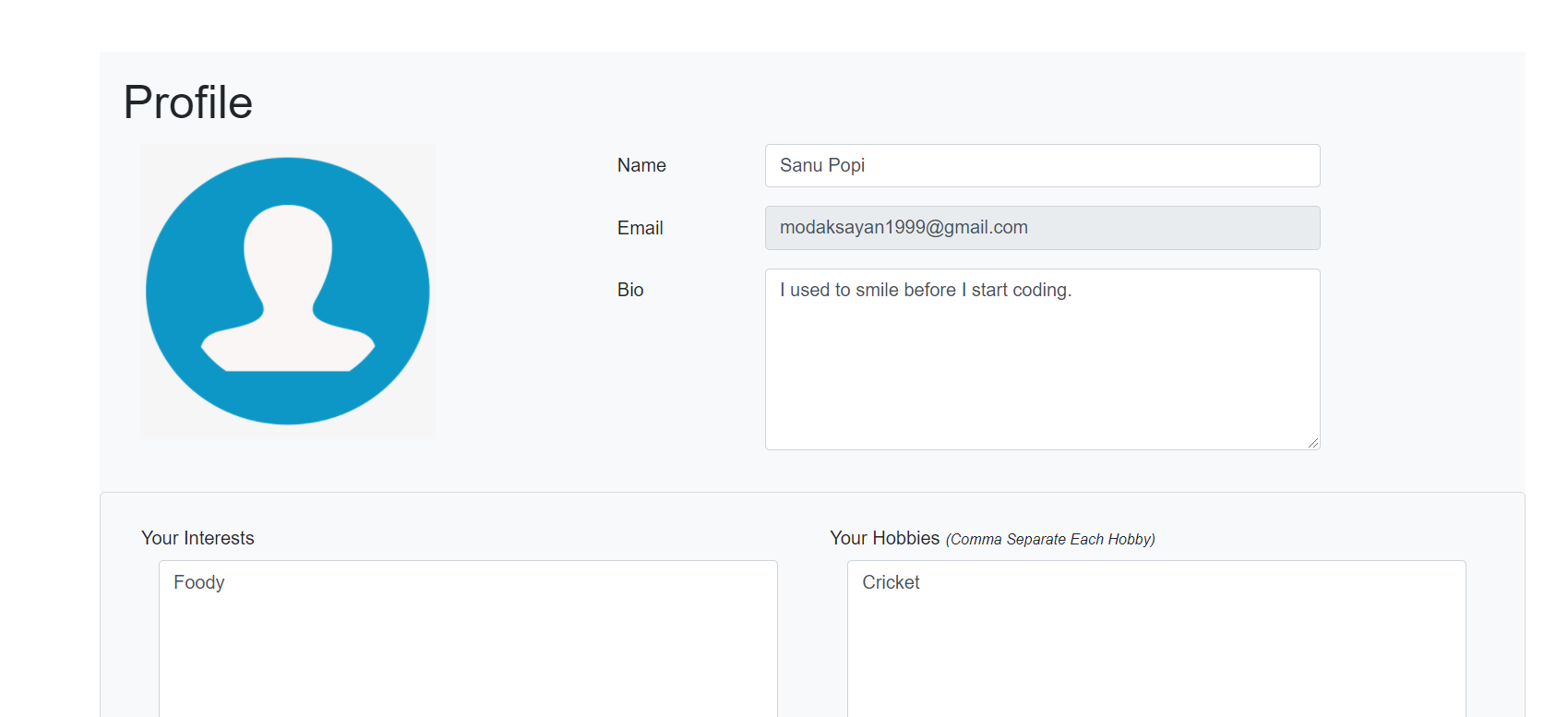
Blog-Publish



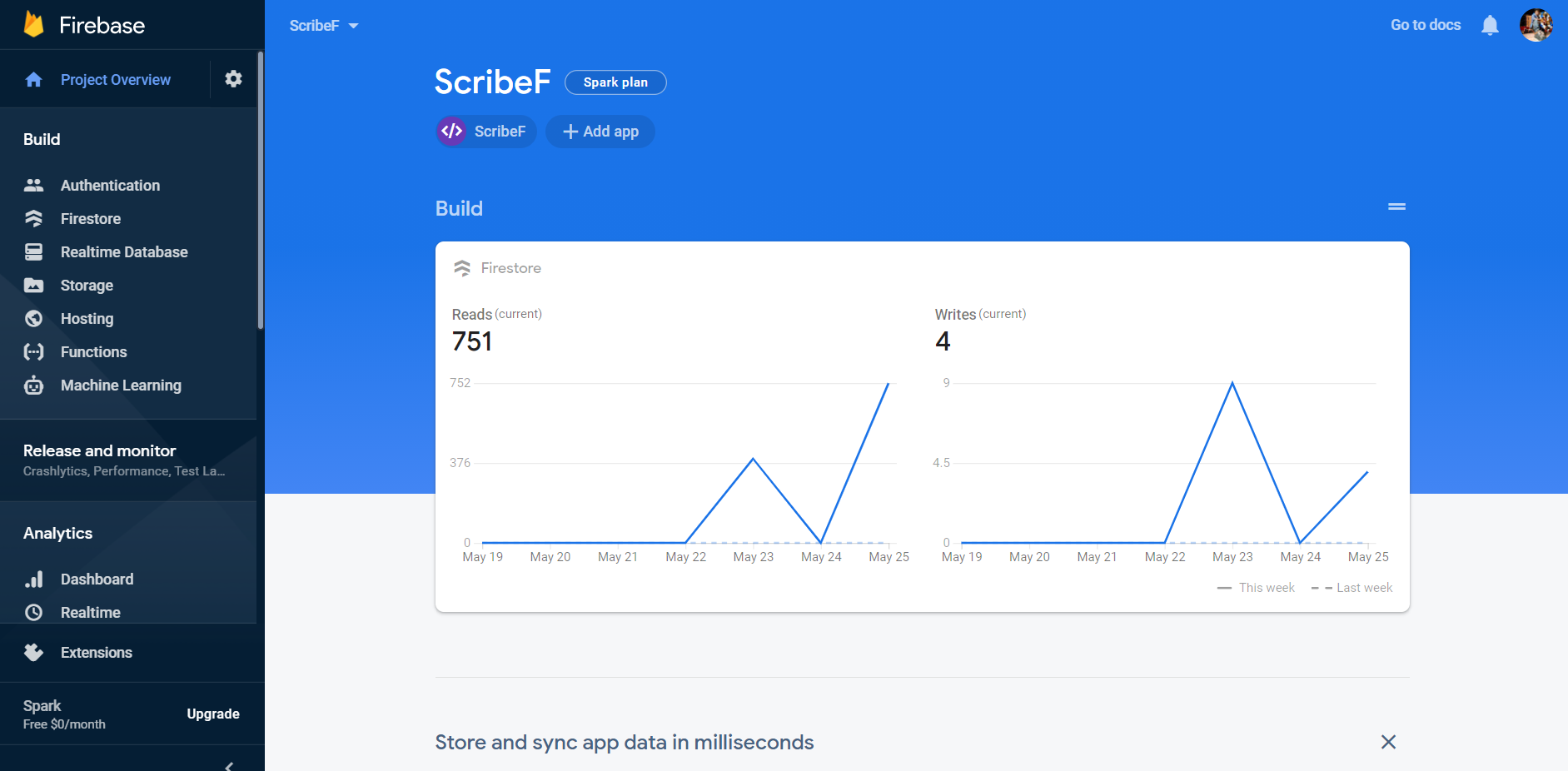
Brand-endorsement & Link any site



Edit Profile



Connected Databases & Traffic Details



Conclusion & Future Study

**Add Google Ad-Sense :**

In Future we can add Google ads to this Web-App. From there we can generate revenue and also we can give some part of our revenue to the Bloggers.

**Brand Promoting :**

We have added that link promotion feature to our web-app. From that every creator can promote brands and also we can feature personalize brands to every consumer as per their interest. We can that interest data from their blog reading habits.

**Employment Opportunity :**

If we can add and personalize ads to every consumer then from brands and ads we can generate our Revenue and then we can give employment to every blogger or creator who is constantly active & contribute in our Web-App.

**V-Log Posts :**

In Future we can also add V-logs feature in Web-App. Here every creator can post their daily life and experience with the consumer. V-log is actually Video-log. So we are thinking to take it from written Blog to Visual Vlogs.

References

1. Angular Docs
   * 1. <https://angular.io/docs>
2. Npm Docs
   * 1. <https://docs.npmjs.com/>
3. Bootstrap Docs
   * 1. <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
4. Html & Css Sites
   * 1. <https://www.w3schools.com/html/>
     2. <https://developer.mozilla.org/en-US/docs/Web/HTML>
     3. <https://www.tutorialspoint.com/html/index.htm>
     4. <https://www.w3schools.com/css/>
     5. <https://developer.mozilla.org/en-US/docs/Web/CSS>
     6. <https://www.tutorialspoint.com/css/index.htm>
5. Firebase Docs-
   * 1. <https://firebase.google.com/docs>