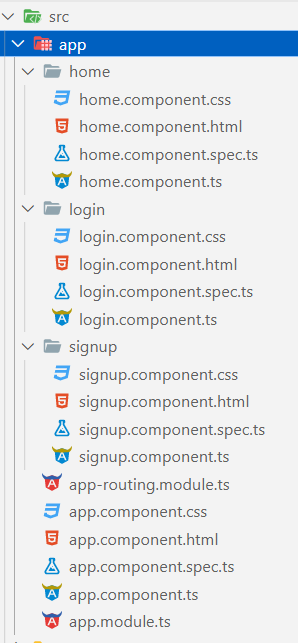
# Directory Structure



# index.html

| <!doctype html>  <html lang="en">  <head>    <meta charset="utf-8">    <title>MyAssignment</title>    <base href="/">    <meta name="viewport" content="width=device-width, initial-scale=1">    <link rel="icon" type="image/x-icon" href="favicon.ico">    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-aFq/bzH65dt+w6FI2ooMVUpc+21e0SRygnTpmBvdBgSdnuTN7QbdgL+OapgHtvPp" crossorigin="anonymous">  </head>  <body>    <app-root></app-root>  </body>  </html> |
| --- |

# app.component.ts

| import { Component } from '@angular/core';  @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {    title = 'my-assignment';  } |
| --- |

# app-routing.module.ts

| import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { HomeComponent } from './home/home.component';  import { LoginComponent } from './login/login.component';  import { SignupComponent } from './signup/signup.component';  const routes: Routes = [    {      path: "",      component: HomeComponent    },    {      path: "login",      component: LoginComponent    },    {      path: "signup",      component: SignupComponent    },  ];  @NgModule({    imports: [RouterModule.forRoot(routes)],    exports: [RouterModule]  })  export class AppRoutingModule { } |
| --- |

# app.component.html

| <router-outlet></router-outlet> |
| --- |

# signup.component.ts

| import { Component } from '@angular/core';  import { Router } from '@angular/router';  @Component({      selector: 'app-signup',      templateUrl: './signup.component.html',      styleUrls: ['./signup.component.css']  })  export class SignupComponent {      usernameError: string = "";      passwordError: string = "";      constructor(private router: Router) { }      async getData(username: any, password: any) {          this.usernameError = username ? "" : "Username is required";          this.passwordError = password ? "" : "Password is required";          if (username && password) {              var myHeaders = new Headers();              myHeaders.append("Content-Type", "application/json");              var raw = JSON.stringify({                  "username": username,                  "password": password              });              var requestOptions: Object = {                  method: 'POST',                  headers: myHeaders,                  body: raw,                  redirect: 'follow'              };              fetch("http://localhost:2324/signup", requestOptions)                  .then(response => {                      if (response.ok) {                          return response.json();                      } else {                          this.usernameError = "Username Already Exists";                          throw Error("Not OK")                      }                  })                  .then(result => {                      if (result.token) {                          document.cookie = "jwt=" + result.token                          localStorage.setItem("user", JSON.stringify(result.user))                          this.router.navigate(['']);                      }                  })                  .catch(error => console.log('error', error));          }      }  } |
| --- |

# signup.component.html

| <div class="container py-5">      <div class="col-12 col-sm-10 col-md-6 col-lg-4 m-auto border shadow rounded p-4">          <h3 class="mb-4 pb-2 border-bottom">User Signup</h3>          <div class="mb-3">              <label for="username" class="form-label">username</label>              <input type="username" class="form-control" name="username" #username aria-describedby="usernameHelpId" placeholder="abc">              <small id="usernameHelpId" class="form-text text-danger">{{usernameError}}</small>          </div>          <div class="mb-3">              <label for="" class="form-label">Password</label>              <input type="password" class="form-control" name="password" #password aria-describedby="passwordHelpId" placeholder="Your Password">              <small id="passwordHelpId" class="form-text text-danger">{{passwordError}}</small>          </div>          <button type="button" class="btn btn-primary" #loginBtn (click)="getData(username.value, password.value)">Login</button>          <br>          <hr />          <a href="/login">Already Have An Account?</a>      </div>  </div> |
| --- |

# login.component.ts

| import { Component } from '@angular/core';  import { Router } from '@angular/router';  @Component({      selector: 'app-login',      templateUrl: './login.component.html',      styleUrls: ['./login.component.css']  })  export class LoginComponent {      usernameError: string = "";      passwordError: string = "";      constructor(private router: Router) {      async getData(username: any, password: any) {          this.usernameError = username ? "" : "Username is required";          this.passwordError = password ? "" : "Password is required";          if (username && password) {              var myHeaders = new Headers();              myHeaders.append("Content-Type", "application/json");              var raw = JSON.stringify({                  "username": username,                  "password": password              });              var requestOptions: Object = {                  method: 'POST',                  headers: myHeaders,                  body: raw,                  redirect: 'follow'              };              fetch("http://localhost:2324/login", requestOptions)                  .then(response => response.json())                  .then(result => {                      if (result.token) {                          document.cookie = "jwt=" + result.token                          localStorage.setItem("user", JSON.stringify(result.user))                          this.router.navigate(['']);                      } else {                          this.usernameError = result.message == "Username Not Found" ? result.message : "";                          this.passwordError = result.message == "Incorrect Password" ? result.message : "";                      }                  })                  .catch(error => console.log('error', error));          }      }  } |
| --- |

# login.component.html

| <div class="container py-5">      <div class="col-12 col-sm-10 col-md-6 col-lg-4 m-auto border shadow rounded p-4">          <h3 class="mb-4 pb-2 border-bottom">User Login</h3>          <div class="mb-3">              <label for="username" class="form-label">username</label>              <input type="username" class="form-control" name="username" #username aria-describedby="usernameHelpId" placeholder="abc">              <small id="usernameHelpId" class="form-text text-danger">{{usernameError}}</small>          </div>          <div class="mb-3">              <label for="" class="form-label">Password</label>              <input type="password" class="form-control" name="password" #password aria-describedby="passwordHelpId" placeholder="Your Password">              <small id="passwordHelpId" class="form-text text-danger">{{passwordError}}</small>          </div>          <button type="button" class="btn btn-primary" #loginBtn (click)="getData(username.value, password.value)">Login</button>          <br>          <hr />          <a href="/signup">Create An Account?</a>      </div>  </div> |
| --- |

# [home.component.ts](http://home.component.ts)

| import { Component } from '@angular/core';  import { CookieService } from 'ngx-cookie-service';  import { Router } from '@angular/router';  @Component({      selector: 'app-home',      templateUrl: './home.component.html',      styleUrls: ['./home.component.css']  })  export class HomeComponent {      message: string      constructor(private cookieService: CookieService, private router: Router) {          let user: string = localStorage.getItem("user") ?? "{}";          const userDetails = JSON.parse(user)          this.message = userDetails.username ? "Hello " + userDetails.username : "LOGIN";          this.validate()      }      validate() {          // verify          const jwt = this.cookieService.get('jwt');          console.log(jwt)          const myHeaders = new Headers({              'Content-Type': 'application/json',              'Cookie': `jwt=${jwt}`          });          var requestOptions: Object = {              method: 'GET',              headers: myHeaders,              redirect: 'follow',              credentials: 'include'          };          fetch("http://localhost:2324", requestOptions)              .then(response => {                  if (!response.ok) {                      throw new Error(String(response.status));                  }                  return response.json();              })              .then(result => {                  console.log(result)                  if (result.message == "Token is not valid") {                      // redirect to login                      localStorage.removeItem("user")                      this.router.navigate(['/login']);                  }                  else if (result.message == "Token is not provided") {                      // redirect to login                      localStorage.removeItem("user")                      this.router.navigate(['/login']);                  }                  if (this.message == "LOGIN") {                      this.router.navigate(['/login']);                  }              })              .catch(error => console.log('error', error));      }      logout() {          localStorage.removeItem("user")          this.router.navigate(['/login']);      }  } |
| --- |

# [home.component.html](http://home.component.html)

| <p class="text-center h1 m-5">{{message}}</p>  <div class="text-center">      <button class="btn btn-danger my-5" (click)="logout()">LogOUT</button>  </div> |
| --- |

# BACKEND: index.js

| import express from "express"  import User from "./User.js"  import cookieParser from "cookie-parser";  import jwt from "jsonwebtoken";  import connection from "./mongodb.js";  import cors from 'cors';  import dotenv from "dotenv"  dotenv.config();  const app = express();  //middleware  app.use(express.json());  app.use(cookieParser());  app.use(function (req, res, next) {      res.setHeader('Access-Control-Allow-Origin', '\*');      res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');      res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');      res.setHeader('Access-Control-Allow-Credentials', true);      next();  });  app.use(cors({      origin: 'http://localhost:4200',      credentials: true  }));  const authMiddleware = (req, res, next) => {      const jwToken = req.cookies.jwt      if (jwToken) {          jwt.verify(jwToken, "This is My Secret", (error, decode) => {              if (error) {                  res.json({ message: "Token is not valid" })              } else {                  req.\_id = decode.\_id;                  next()              }          })      } else {          res.json({ message: "Token is not provided" })      }  }  // functions  async function createToken(\_id, res) {      const token = await jwt.sign({ \_id }, "This is My Secret", {          expiresIn: "2d"      })      res.cookie("jwt", token, {          maxAge: 1000 \* 60 \* 60 \* 24 \* 2,          httpOnly: true      });      return token;  }  app.post("/login", async (req, res) => {      try {          const user = await User.login(req.body);          if (user == 1) {              res.clearCookie("jwt");              return res.send({ message: "Username Not Found" })          } else if (user == 2) {              res.clearCookie("jwt");              return res.send({ message: "Incorrect Password" })          }          const token = await createToken(user.\_id, res)          return res.send({ user, token });      } catch (error) {          res.status(404).send({ error })      }  })  app.post("/signup", async (req, res) => {      const { username, password } = req.body;      if (username && password) {          try {              const user = await User.create({ username, password });              const token = await createToken(user.\_id, res)              return res.send({ user, token });          } catch (error) {              return res.status(404).send({ error })          }      } else {          return res.status(404).json({ message: "`username` and `password` are required" })      }  })  app.get("/", authMiddleware, async (req, res) => {      try {          const user = await User.findOne({ \_id: req.\_id })          res.status(200).json({ username: user.username })      } catch (error) {          res.status(494).json({ error })      }  })  connection.then(() => {      app.listen(2324, () => {          console.log("server started: http://localhost:2324")      })  }).catch(err => {      console.log(err)  }) |
| --- |

# BACKEND: User.js

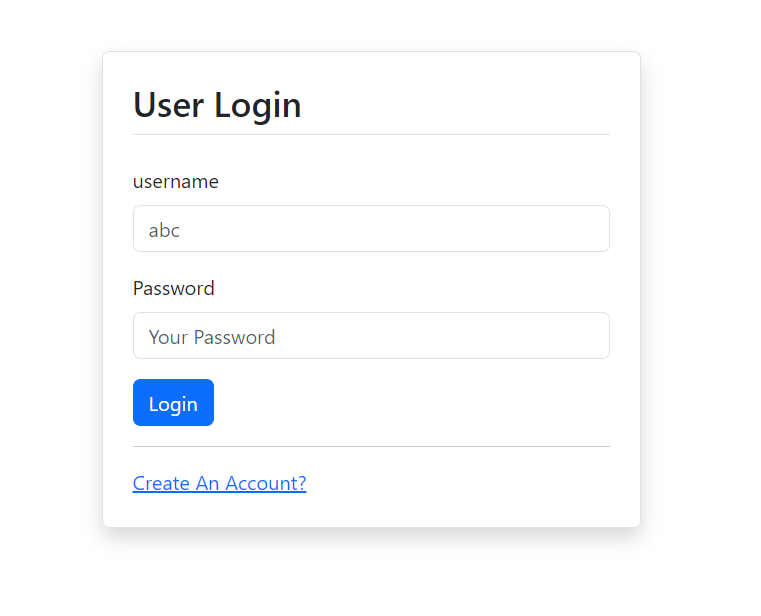
| import mongoose from "mongoose"  import bcrypt from "bcrypt"  const User = new mongoose.Schema({      username: {          type: String,          unique: true,          trim: true,          required: true      },      password: {          type: String,          trim: true,          required: true      }  });  User.pre("save", async function (next) {      this.password = await bcrypt.hash(this.password, 10);      next();  });  User.statics.login = async function ({ username, password }) {      try {          const user = await this.findOne({ username })          if (user) {              return await bcrypt.compare(password, user.password) ? user : 2          } else {              return 1;          }      } catch (error) {          console.log(error)      }  }  export default mongoose.model("user", User); |
| --- |

# BACKEND:mongodb.js

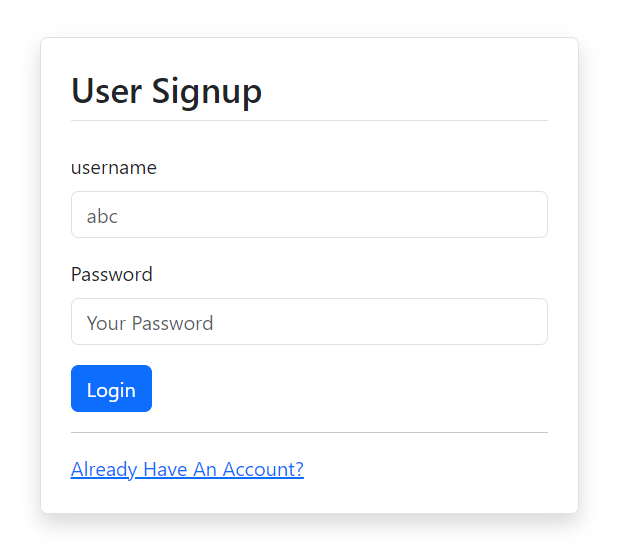
| import mongoose from "mongoose";  import dotenv from "dotenv"  dotenv.config();  const connection = mongoose.connect(process.env.DB\_URI)  export default connection; |
| --- |

# OUTPUT:

## Login Page:



## Signup Page:



## Home Page (After Login/Signup):

