Here’s a breakdown of how to complete Assignment 3, including setting up a GitHub repository, Docker container, and an Angular application with user authentication functionality.

**Assignment 3a: Creating a GitHub Repository and Pushing Code Using Git Commands**

**1. Create a GitHub Account**

* Go to [GitHub](https://github.com) and sign up or log in.

**2. Create a New Repository**

* Click on the **+** icon at the top right corner and select **New repository**.
* Name your repository and initialize it with a README.md file if desired.

**3. Set Up Git Locally**

* Open your terminal/command prompt and configure Git (if not already set up):

bash

Copy code

git config --global user.name "Your Name"

git config --global user.email "youremail@example.com"

**4. Clone the Repository Locally**

* In the GitHub repository page, copy the repository URL and run the following command:

bash

Copy code

git clone <repository-url>

* This will create a local copy of the repository.

**5. Add Your Code**

* Navigate to the repository folder:

bash

Copy code

cd <repository-name>

* Create or add your code files to this folder.

**6. Add, Commit, and Push Changes to GitHub**

* Stage all changes:

bash

Copy code

git add .

* Commit the changes with a message:

bash

Copy code

git commit -m "Initial commit with code"

* Push the changes to GitHub:

bash

Copy code

git push origin main

* If your branch is named differently, replace main with your branch name.

**Assignment 3b: Creating a Docker Container Environment**

**1. Install Docker**

* Download and install Docker Desktop from [Docker’s official website](https://www.docker.com/).

**2. Create a Dockerfile**

* Inside your project folder, create a Dockerfile:

dockerfile

Copy code

# Use an official Node.js image as the base

FROM node:14

# Set the working directory

WORKDIR /app

# Copy package.json and package-lock.json files

COPY package\*.json ./

# Install dependencies

RUN npm install

# Copy the rest of the application

COPY . .

# Expose port 4200 (default Angular port)

EXPOSE 4200

# Command to run the Angular application

CMD ["npm", "start"]

**3. Create a docker-compose.yml (Optional)**

* Create a docker-compose.yml file for container orchestration:

yaml

Copy code

version: '3'

services:

angular-app:

build: .

ports:

- "4200:4200"

volumes:

- .:/app

stdin\_open: true

tty: true

**4. Build and Run the Docker Container**

* Open your terminal and navigate to the project directory:

bash

Copy code

docker build -t angular-app .

* Run the container:

bash

Copy code

docker run -p 4200:4200 angular-app

* Alternatively, use Docker Compose:

bash

Copy code

docker-compose up

**Assignment 3c: Create an Angular Application with User Authentication**

**1. Set Up a New Angular Project**

* Install the Angular CLI if you haven’t:

bash

Copy code

npm install -g @angular/cli

* Create a new Angular application:

bash

Copy code

ng new user-auth-app

* Navigate to the project directory:

bash

Copy code

cd user-auth-app

* Generate necessary components:

bash

Copy code

ng generate component register

ng generate component login

ng generate component profile

ng generate service user

**2. Create User Service for Authentication**

* In src/app/user.service.ts:

typescript

Copy code

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

interface User {

username: string;

password: string;

}

@Injectable({

providedIn: 'root'

})

export class UserService {

private users: User[] = [];

register(user: User) {

this.users.push(user);

localStorage.setItem('users', JSON.stringify(this.users));

}

login(username: string, password: string): boolean {

const storedUsers = JSON.parse(localStorage.getItem('users') || '[]');

return storedUsers.some((user: User) => user.username === username && user.password === password);

}

getUser(username: string): User | undefined {

const storedUsers = JSON.parse(localStorage.getItem('users') || '[]');

return storedUsers.find((user: User) => user.username === username);

}

}

**3. Create Register, Login, and Profile Components**

* In src/app/register/register.component.ts:

typescript

Copy code

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

import { UserService } from '../user.service';

@Component({

selector: 'app-register',

templateUrl: './register.component.html'

})

export class RegisterComponent {

username = '';

password = '';

constructor(private userService: UserService) {}

register() {

this.userService.register({ username: this.username, password: this.password });

alert('User registered successfully!');

}

}

* Similarly, implement login.component.ts to authenticate the user, and profile.component.ts to display user data upon successful login.

**4. Create Basic Routing**

* In src/app/app-routing.module.ts:

typescript

Copy code

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

import { RouterModule, Routes } from '@angular/router';

import { RegisterComponent } from './register/register.component';

import { LoginComponent } from './login/login.component';

import { ProfileComponent } from './profile/profile.component';

const routes: Routes = [

{ path: 'register', component: RegisterComponent },

{ path: 'login', component: LoginComponent },

{ path: 'profile', component: ProfileComponent },

{ path: '', redirectTo: '/login', pathMatch: 'full' }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }