

ANGULAR FINAL ASSIGNMENT – FREELANCER HUB

Step 1: Setting Up the Angular Project

Install Angular CLI (If Not Installed)

```
npm install -g @angular/17cli
```

```
D:\>npm install -g @angular/cli@17
npm WARN deprecated read-package-json@7.0.1: This package is no longer supported. Please use @npmcli/package-json instead.

changed 244 packages in 14s

43 packages are looking for funding
  run `npm fund` for details

D:\>|
```

Create a New Angular Project

```
ng new freelancer-hub --standalone
```

```
D:\>ng new freelancer-hub --standalone
? Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS ]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE freelancer-hub/angular.json (2727 bytes)
CREATE freelancer-hub/package.json (1085 bytes)
CREATE freelancer-hub/README.md (1095 bytes)
CREATE freelancer-hub/tsconfig.json (889 bytes)
CREATE freelancer-hub/.editorconfig (290 bytes)
CREATE freelancer-hub/.gitignore (629 bytes)
CREATE freelancer-hub/tsconfig.app.json (277 bytes)
CREATE freelancer-hub/tsconfig.spec.json (287 bytes)
CREATE freelancer-hub/.vscode/extensions.json (134 bytes)
CREATE freelancer-hub/.vscode/launch.json (490 bytes)
CREATE freelancer-hub/.vscode/tasks.json (980 bytes)
CREATE freelancer-hub/src/main.ts (256 bytes)
CREATE freelancer-hub/src/favicon.ico (15086 bytes)
CREATE freelancer-hub/src/index.html (312 bytes)
CREATE freelancer-hub/src/styles.css (81 bytes)
CREATE freelancer-hub/src/app/app.component.html (20239 bytes)
CREATE freelancer-hub/src/app/app.component.spec.ts (969 bytes)
CREATE freelancer-hub/src/app/app.component.ts (323 bytes)
CREATE freelancer-hub/src/app/app.component.css (0 bytes)
CREATE freelancer-hub/src/app/app.config.ts (235 bytes)
CREATE freelancer-hub/src/app/app.routes.ts (80 bytes)
CREATE freelancer-hub/src/assets/.gitkeep (0 bytes)
✓ Packages installed successfully.
  Successfully initialized git.
```

Navigate to the project folder:

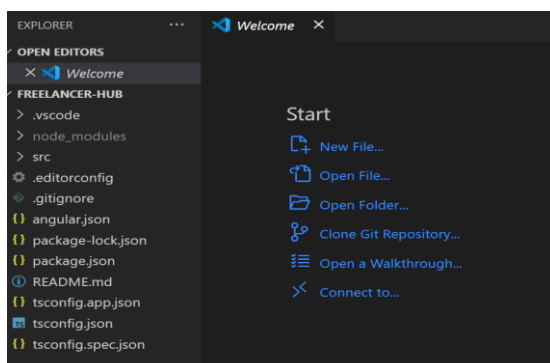
```
cd freelancer-hub
```

```
D:\>cd freelancer-hub

D:\freelancer-hub>
```

Open Project in VSCODE:

Code .



Run the project:

ng serve



Hello, freelancer-hub

Congratulations! Your app is running. 🎉

[Explore the Docs](#)[Learn with Tutorials](#)[CLI Docs](#)[Angular Language Service](#)[Angular DevTools](#)

This will start the project at <http://localhost:4200>.

Step 2: Generate Components

Commands:

ng generate component components/login

ng generate component components/signin

ng generate component components/home

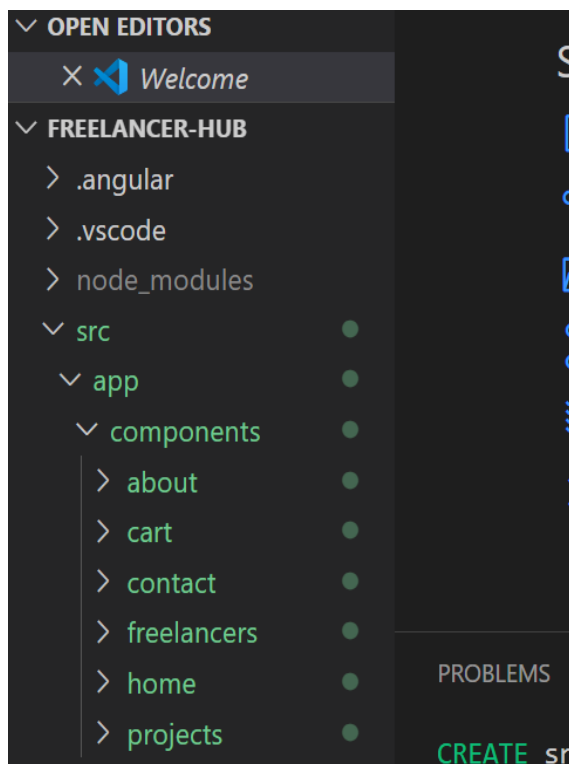
ng generate component components/projects

ng generate component components/freelancers

ng generate component components/cart

ng generate component components/contact

ng generate component components/about



Components Created.

Step 3: Setup Routing for Components

```

src > app > TS app.routes.ts > ...
1  import { Routes } from '@angular/router';
2  import { HomeComponent } from '../components/home/home.component';
3  import { ProjectsComponent } from '../components/projects/projects.component';
4  import { FreelancersComponent } from '../components/freelancers/freelancers.component';
5  import { CartComponent } from '../components/cart/cart.component';
6  import { ContactComponent } from '../components/contact/contact.component';
7  import { AboutComponent } from '../components/about/about.component';
8
9  export const routes: Routes = [
10   { path: '', component: HomeComponent }, // Default route
11   { path: 'projects', component: ProjectsComponent },
12   { path: 'freelancers', component: FreelancersComponent },
13   { path: 'cart', component: CartComponent },
14   { path: 'contact', component: ContactComponent },
15   { path: 'about', component: AboutComponent }
16 ];
17

```

Step 4: Implement Navigation in App Component

```

TS app.routes.ts M    <> app.component.html M X
src > app > <> app.component.html > ...
1  <nav class="navbar">
2    <a routerLink="/" class="nav-item">Home</a>
3    <a routerLink="/projects" class="nav-item">Projects</a>
4    <a routerLink="/freelancers" class="nav-item">Freelancers</a>
5    <a routerLink="/cart" class="nav-item">Cart</a>
6    <a routerLink="/contact" class="nav-item">Contact</a>
7    <a routerLink="/about" class="nav-item">About</a>
8  </nav>
9
10 <!-- Display the selected route component -->
11 <router-outlet></router-outlet>
12

```

Step 5: Implement Content for Each Component

1,Signin:

1. FormGroup Creation define the form's structure in your component.
2. Input Binding connect HTML fields to the defined form.
3. Data Validation ensure entered data adheres to requirements.
4. Error Feedback display messages for invalid input.
5. Backend Submission send the collected data to the local storage.

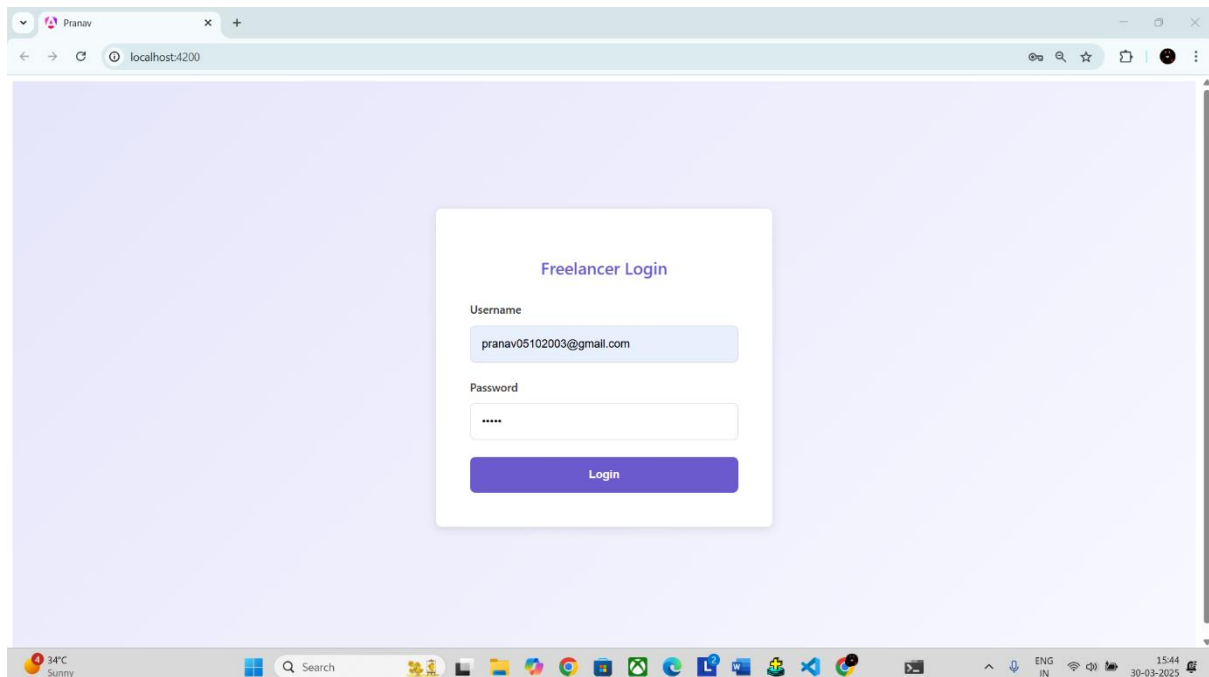
The screenshot displays a web browser window with the address bar showing 'localhost:4200/signup'. The page features a 'Freelancer Signup' form with the following fields and values:

- First Name: THASARATHA
- Last Name: PRANAV K
- Email: pranav05102003@gmail.com
- Username: pranav0510
- Password: (masked with dots)
- Confirm Password: (masked with dots)
- Role: Client (selected from a dropdown menu)
- Skills (Comma-separated): coding

A blue 'Signup' button is located at the bottom of the form. The browser's taskbar at the bottom shows various application icons, a search bar, and system status indicators including 'Hot weather Now', 'ENG IN', and the time '15:55' on '30-03-2025'.

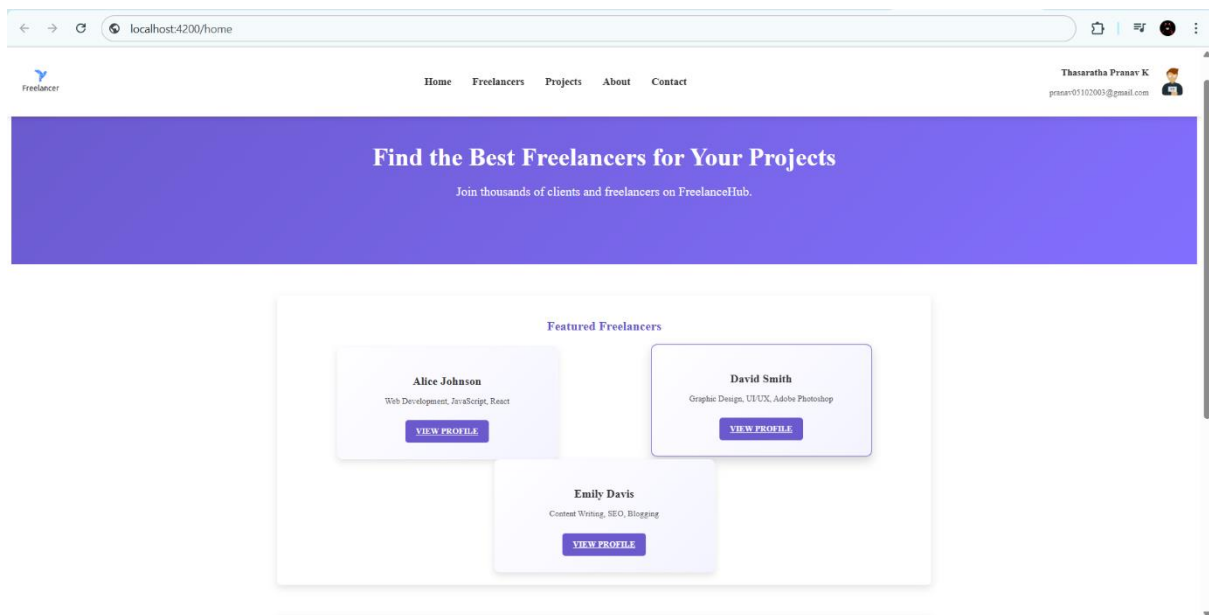
2, Login:

- Component Data stores username, password, and error messages.
- Form Binding links HTML inputs to component data using ngModel.
- Service Call sends login data to the UserService for authentication.
- Response Handling navigates on success, displays errors on failure.
- Visual Styling CSS provides a clean login form appearance.



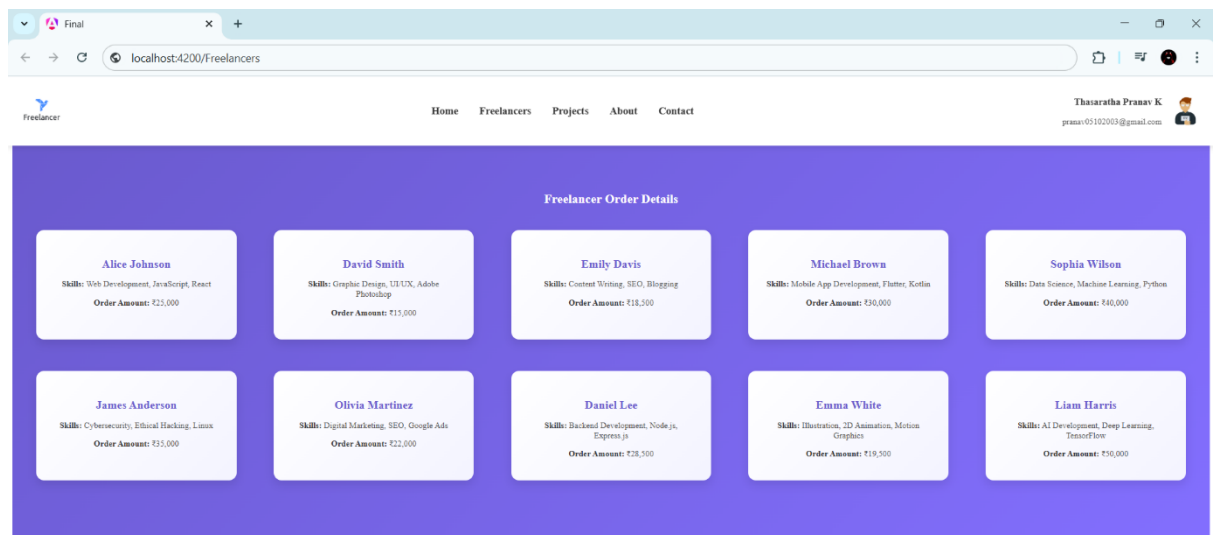
3, Home:

- The HomeComponent holds the data for the featured freelancers.
- The FreelancerCardComponent is responsible for displaying each freelancer's information.
- Using `@Input()` and `@Output()`, the components communicate effectively, allowing the **parent component** to handle the navigation when a freelancer profile is clicked.
- The freelancer card component is reusable, and the home component is clean.



4, Freelancers:

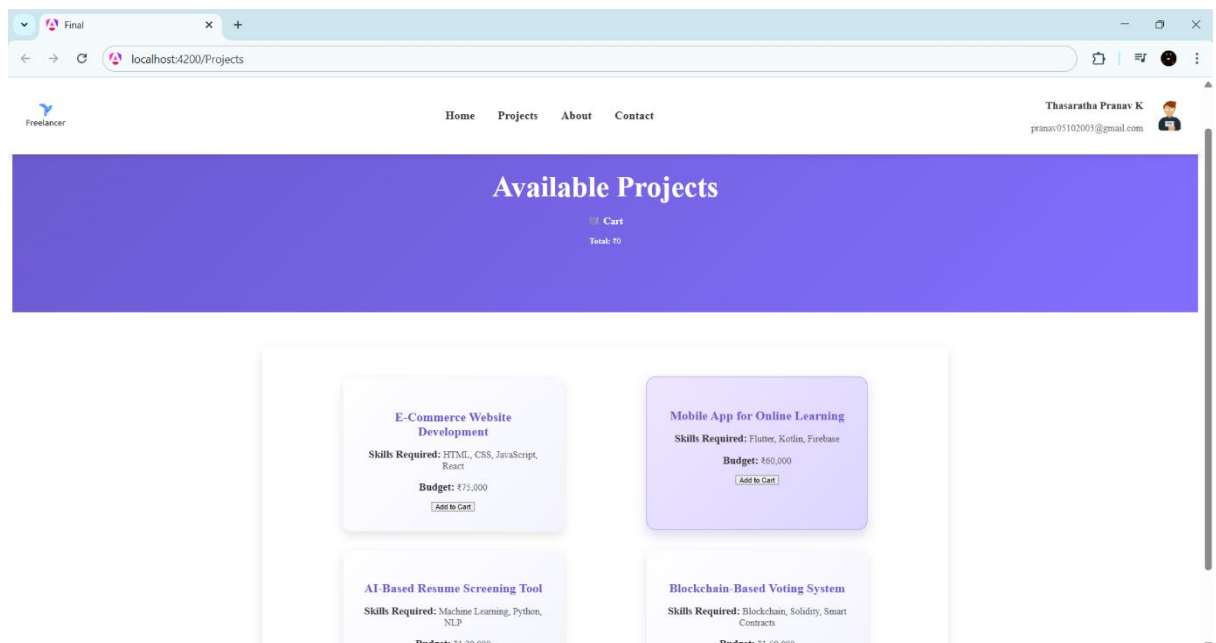
- The FreelancerListComponent holds an array of freelancers containing data about each freelancer.
- The HTML template uses *ngFor to dynamically generate the freelancer cards.
- The CSS styles the layout, creating a responsive grid for the cards and applying styles to each card.
- The *ngFor directive simplifies the process of displaying a list of data without manually creating each card, making the code more efficient and maintainable.
- It is the child component



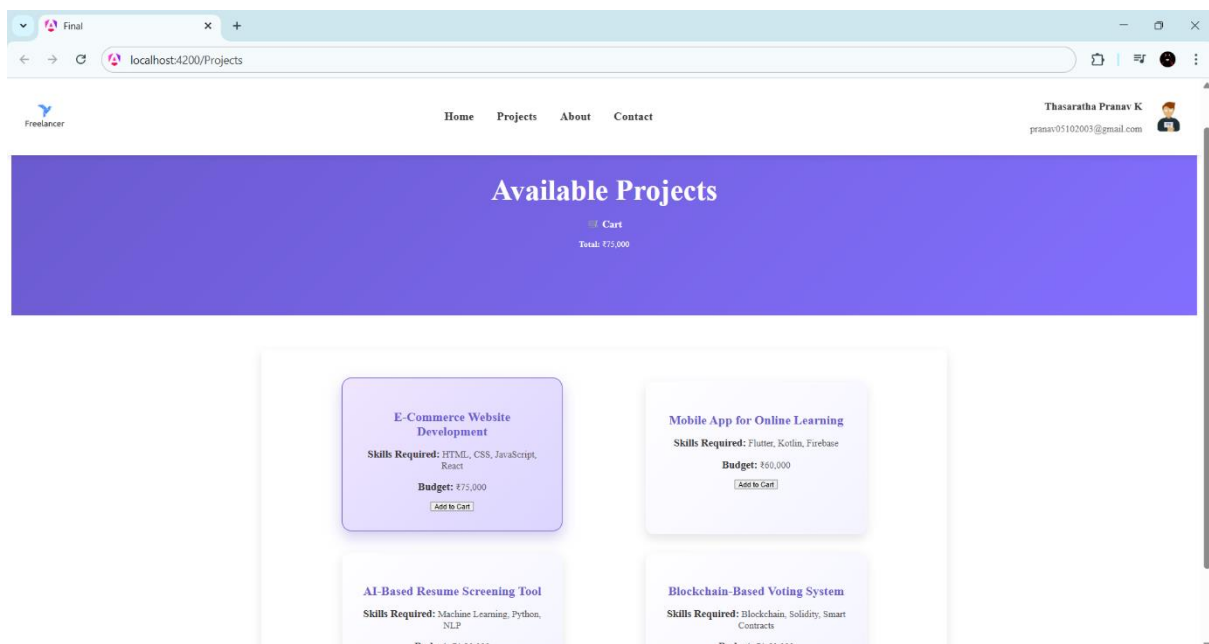
5,Projects:

The ProjectsComponent holds an array of projects with data about each project.

- The HTML template uses *ngFor to dynamically generate the project cards.
- The CurrencyPipe formats the budget property for each project, making it more user-friendly.
- The addToCart() method increases the cartCount for a project when the "Add to Cart" button is clicked.
- The CSS styles the layout, creating a responsive grid for the cards and applying styles to each card.

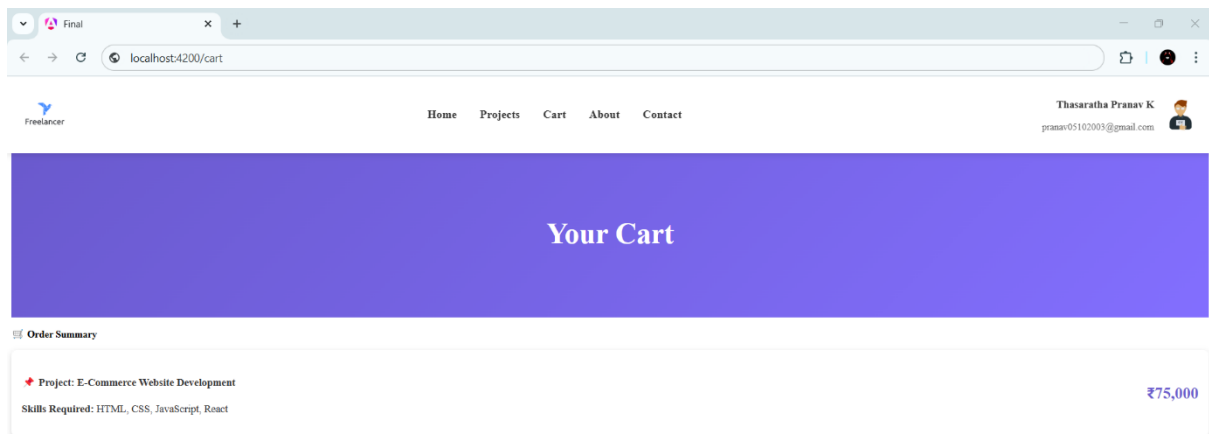


Selecting the required project the amount will be updated in cart.



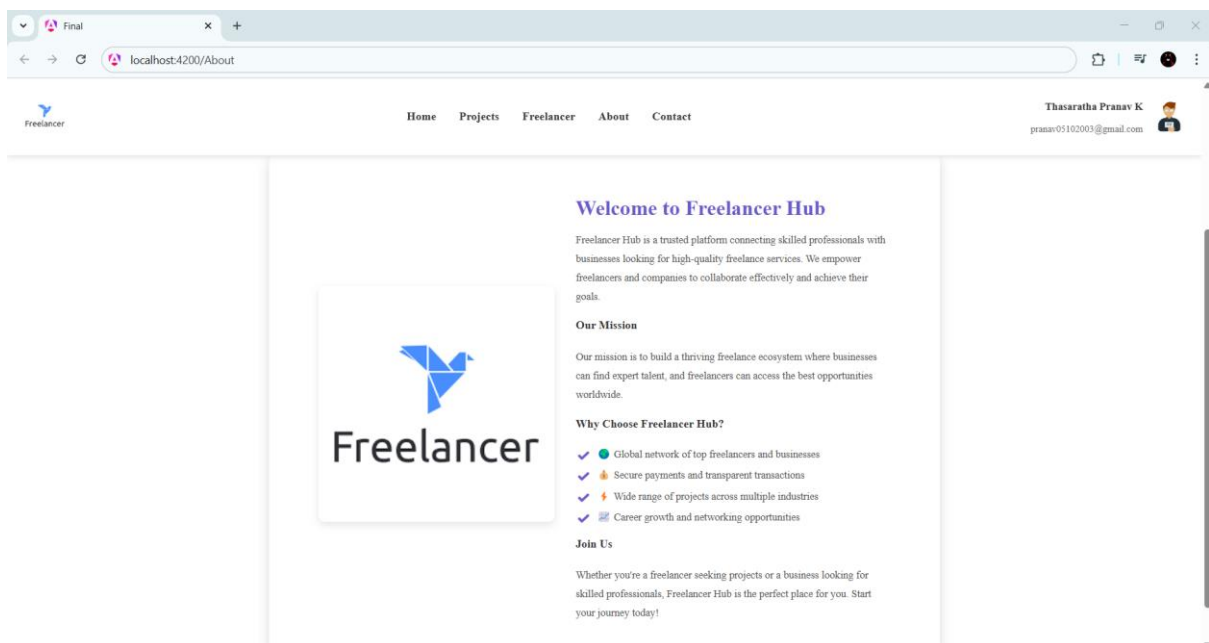
7, Cart

- The CartComponent retrieves and displays the cart items from local storage.
- The ProjectsComponent (or wherever the "Add to Cart" functionality is) stores the cart items in local storage.
- Local storage allows us to persist the cart data even if the user refreshes the page or closes the browser.
- Local storage is a simple key-value storage mechanism that is available in the browser.
- It is useful for storing small amounts of data that don't need to be sent to a local storage.



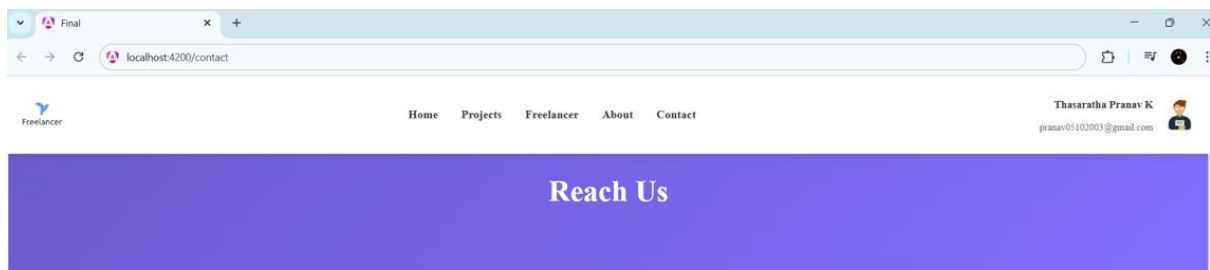
8, About

- The AboutComponent holds the data for the about page.
- The HTML template uses *ngFor to dynamically generate the list of reasons to choose Freelancer Hub.
- We created a custom CapitalizePipe that capitalizes the first letter of each word in a string.
- By using {{ title | capitalize }}, {{ mission | capitalize }}, {{ item | capitalize }}, and {{ joinUs | capitalize }}, we apply the custom pipe to the text, making the first letter of each word uppercase.
- The custom pipe is declared in the app.module.ts to make it available for use in the template.



9, Reach us

- The ContactComponent holds the contact information and message data.
- The HTML template uses template-driven forms to create the message form.
- The NgForm directive and ngModel enable data binding and validation.
- The sendMessage() method logs the message data to the console (you would typically send this data to a backend API).
- The CSS styles the contact page layout and form (Form Handling).



Contact Us

Have any questions or need assistance? Feel free to reach out to us. Our team is always ready to help you.

Address

123 Freelancer Hub Street, Tech City, India

Email

support@freelancerhub.com

Phone

+91 98765 43210

Send Us a Message

Send Message

10, Guard:

Implementing guard:

```

1  import { Injectable } from '@angular/core';
2  import { CanActivate, Router } from '@angular/router';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class AuthGuard implements CanActivate {
8    constructor(private router: Router) {}
9
10   canActivate(): boolean {
11     const isAuthenticated = localStorage.getItem('user');
12     if (!isAuthenticated) {
13       this.router.navigate(['/']);
14       return false;
15     }
16     return true;
17   }

```

11, pipes

implementing pipes:

```

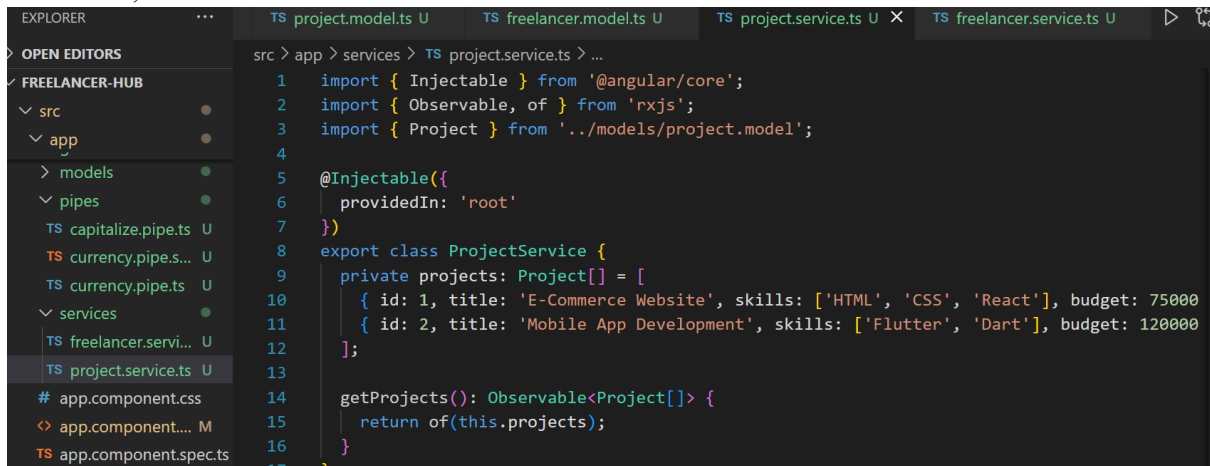
1  // capitalize.pipe.ts
2
3  import { Pipe, PipeTransform } from '@angular/core';
4
5  @Pipe({
6    name: 'capitalize'
7  })
8  export class CapitalizePipe implements PipeTransform {
9    transform(value: string): string {
10     if (!value) return '';
11     return value.replace(/\b\w/g, (char) => char.toUpperCase());
12   }
13 }

```

12, Services:

Two services implemented:

- 1,project.model.ts
- 2,freelancer.model.ts



```
src > app > services > TS project.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { Observable, of } from 'rxjs';
3  import { Project } from '../models/project.model';
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class ProjectService {
9    private projects: Project[] = [
10     { id: 1, title: 'E-Commerce Website', skills: ['HTML', 'CSS', 'React'], budget: 75000
11     { id: 2, title: 'Mobile App Development', skills: ['Flutter', 'Dart'], budget: 120000
12   ];
13
14   getProjects(): Observable<Project[]> {
15     return of(this.projects);
16   }
17 }
```