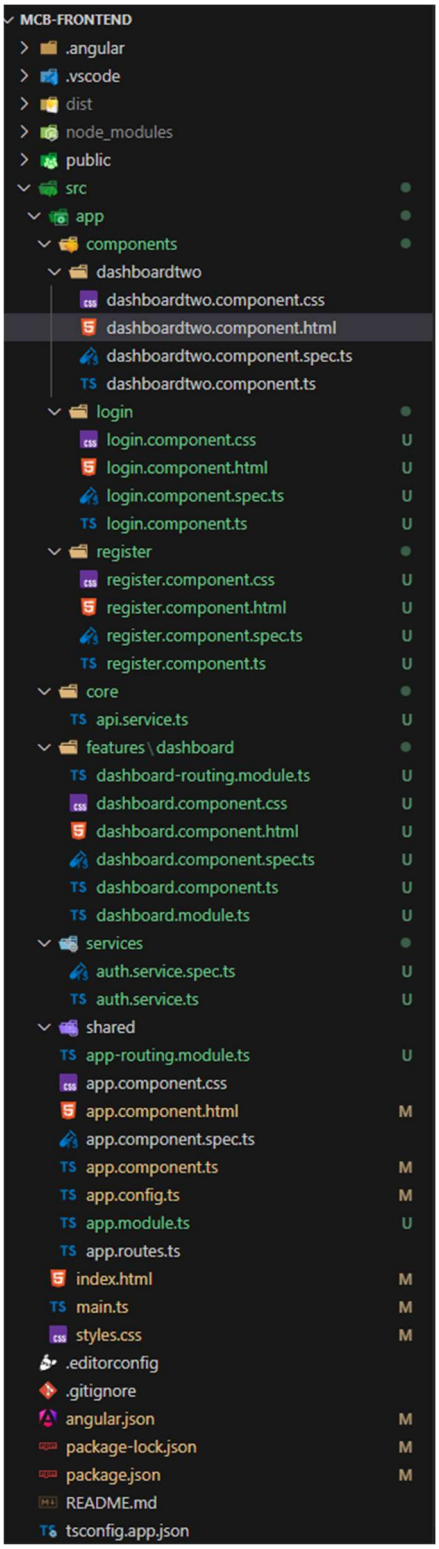
MCB Frontend Report

Overview

The frontend of the MCB project is built using Angular and follows a modular structure with reusable components, services, and a routing system. It provides user authentication, data visualization, and reporting functionalities. Below is an overview of its key components and their functionalities.



Project Structure

Key Directories & Files

- **src/app/components/**: Contains UI components like dashboard, dashboardtwo, login, and register.
- **src/app/core/**: Includes the api.service.ts file for API communication.
- **src/app/features/dashboard/**: Contains dashboard module and related files.
- **src/app/services/**: Holds authentication services (auth.service.ts and auth.service.spec.ts).
- **src/app/shared/**: Contains global configurations, routing, and shared components.
- **Root files**: Includes index.html, main.ts, styles.css, and app.component.* files.

Component Breakdown

1. Dashboard Component (dashboard.component.html & dashboardtwo.component.html)

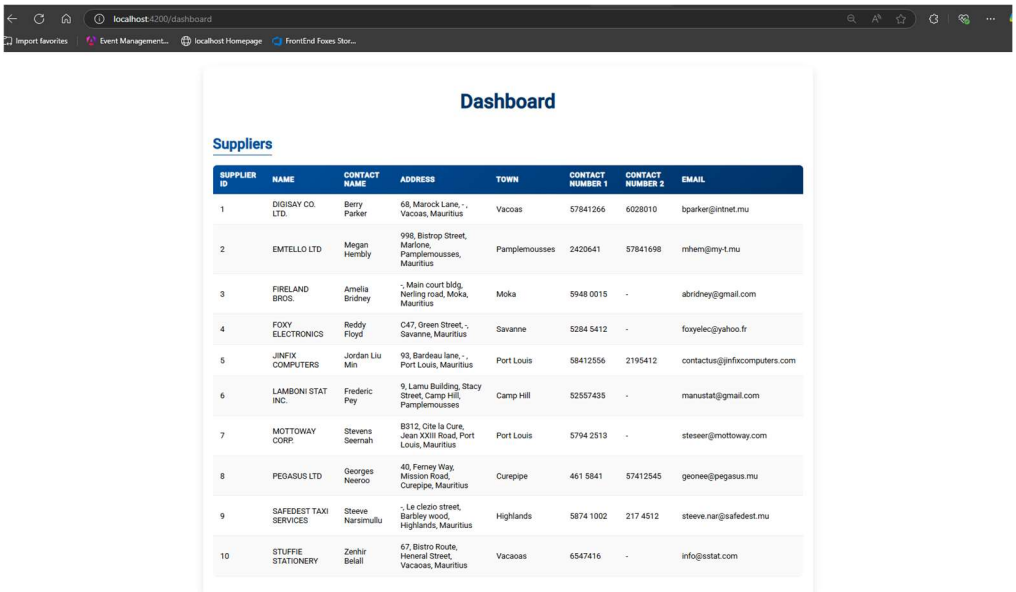
The **Dashboard** modules provide tabular data representation for suppliers, orders, and invoices.

Dashboard Features:

- **Error Handling**: Displays an error message when applicable.
- **Loading State**: Shows a loading message until data is fully fetched.
- **Data Display**: Uses Angular directives (*ngIf, *ngFor) to iterate and render tables.
- **Data Formatting**: Uses Angular Pipes for date and currency formatting.

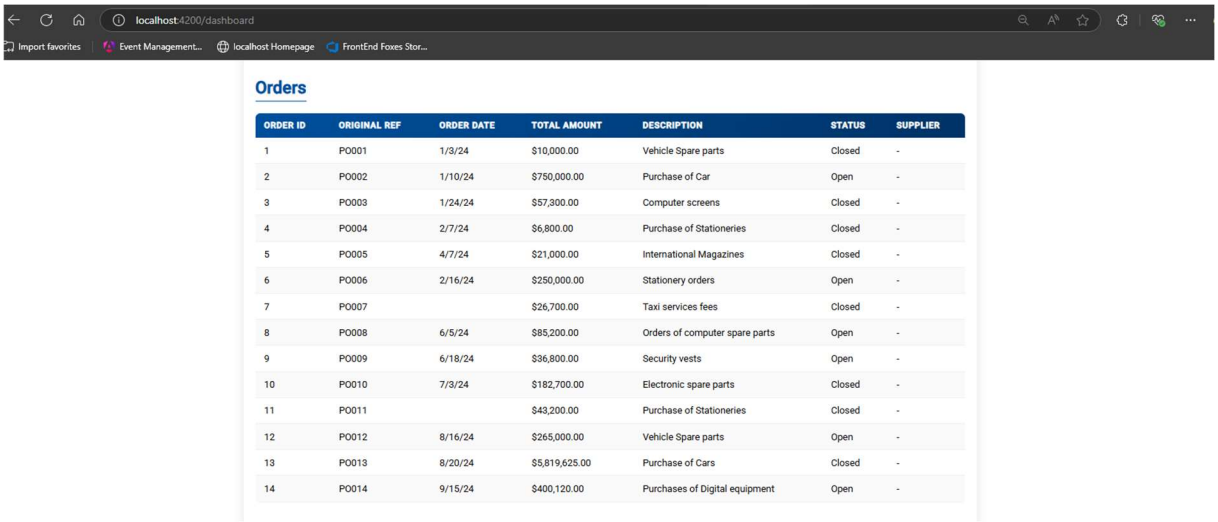
Tables Included:

- **Suppliers Table**: Displays supplier information (ID, name, contact details, address, etc.).



SUPPLIER ID	NAME	CONTACT NAME	ADDRESS	TOWN	CONTACT NUMBER 1	CONTACT NUMBER 2	EMAIL
1	DIGISAY CO. LTD.	Berry Parker	68, Marock Lane, - , Vacoas, Mauritius	Vacoas	57841266	6028010	bparker@intnet.mu
2	EMTELLO LTD	Megan Hembly	998, Bistrop Street, Marlone, Pamplemousses, Mauritius	Pamplemousses	2420641	57841698	mhem@my-t.mu
3	FIRELAND BROS.	Amelia Bridney	-, Main court bldg, Nerling road, Moka, Mauritius	Moka	5948 0015	-	abridney@gmail.com
4	FOXY ELECTRONICS	Reddy Floyd	C47, Green Street, -, Savanne, Mauritius	Savanne	5284 5412	-	foxyelec@yahoo.fr
5	JINFIX COMPUTERS	Jordan Liu Min	93, Bardeau lane, - , Port Louis, Mauritius	Port Louis	58412556	2195412	contactus@jinfixcomputers.com
6	LAMBONI STAT INC.	Frederic Pey	9, Lamu Building, Stacy Street, Camp Hill, Pamplemousses	Camp Hill	52557435	-	manustat@gmail.com
7	MOTTOWAY CORP.	Stevens Seernah	B312, Cite la Cure, Jean XXIII Road, Port Louis, Mauritius	Port Louis	5794 2513	-	steseer@mottoway.com
8	PEGASUS LTD	Georges Neeroo	40, Ferney Way, Mission Road, Curepipe, Mauritius	Curepipe	461 5841	57412545	geonee@pegasus.mu
9	SAFEDEST TAXI SERVICES	Steeve Narsimullu	-, Le clezio street, Barbley wood, Highlands, Mauritius	Highlands	5874 1002	217 4512	steeve.nar@safedest.mu
10	STUFFIE STATIONERY	Zenhir Belali	67, Bistro Route, Heneral Street, Vacaoas, Mauritius	Vacaoas	6547416	-	info@sstat.com

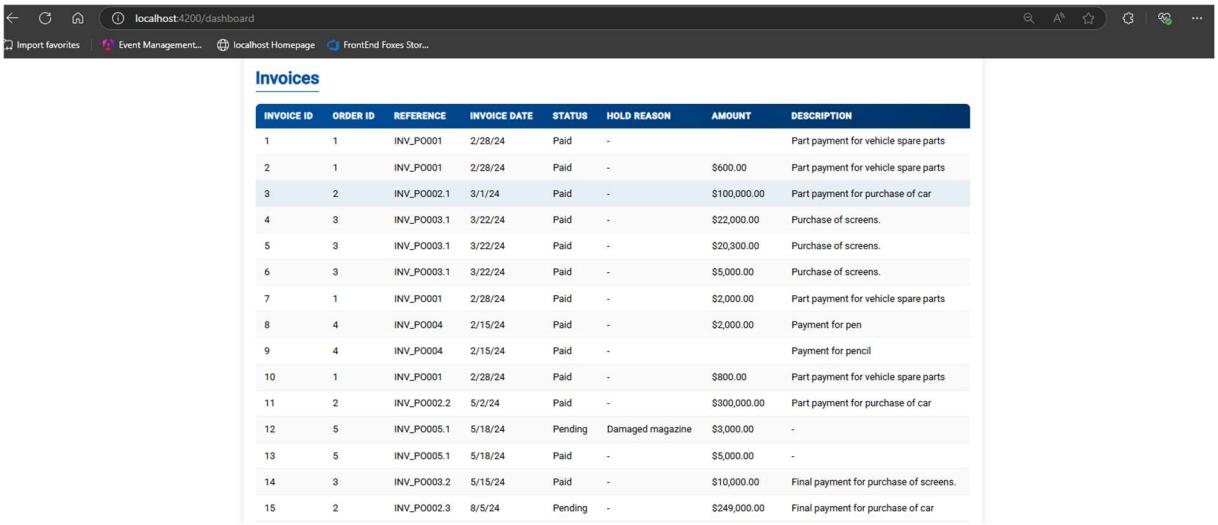
- **Orders Table**: Displays order details (ID, reference, date, amount, status, supplier info).



Orders

ORDER ID	ORIGINAL REF	ORDER DATE	TOTAL AMOUNT	DESCRIPTION	STATUS	SUPPLIER
1	PO001	1/3/24	$10,000.00	Vehicle Spare parts	Closed	-
2	PO002	1/10/24	$750,000.00	Purchase of Car	Open	-
3	PO003	1/24/24	$57,300.00	Computer screens	Closed	-
4	PO004	2/7/24	$6,800.00	Purchase of Stationeries	Closed	-
5	PO005	4/7/24	$21,000.00	International Magazines	Closed	-
6	PO006	2/16/24	$250,000.00	Stationery orders	Open	-
7	PO007		$26,700.00	Taxi services fees	Closed	-
8	PO008	6/5/24	$85,200.00	Orders of computer spare parts	Open	-
9	PO009	6/18/24	$36,800.00	Security vests	Open	-
10	PO010	7/3/24	$182,700.00	Electronic spare parts	Closed	-
11	PO011		$43,200.00	Purchase of Stationeries	Closed	-
12	PO012	8/16/24	$265,000.00	Vehicle Spare parts	Open	-
13	PO013	8/20/24	$5,819,625.00	Purchase of Cars	Closed	-
14	PO014	9/15/24	$400,120.00	Purchases of Digital equipment	Open	-

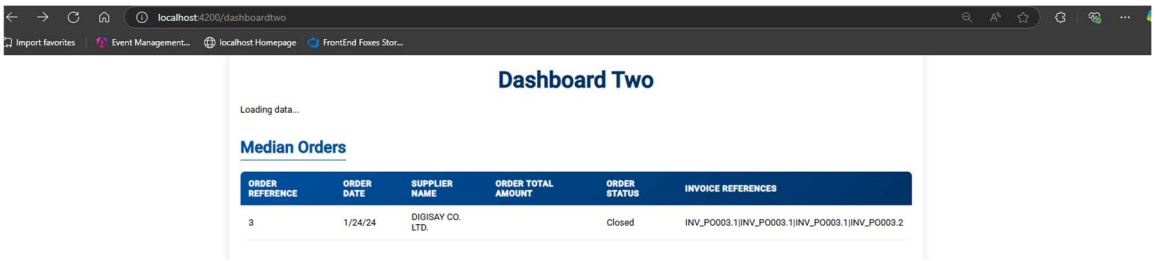
- **Invoices Table**: Shows invoice details (ID, order reference, date, amount, hold reason, status, etc.).



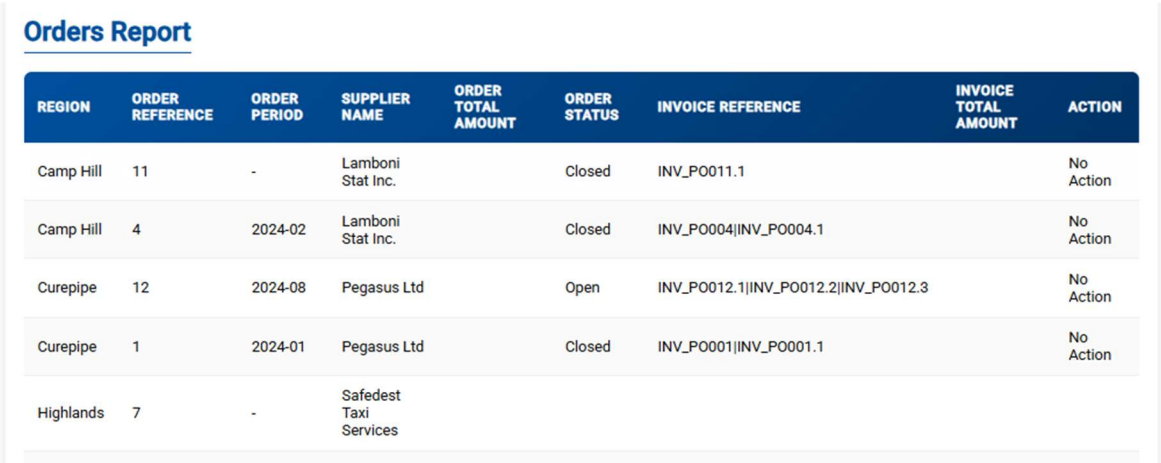
Invoices

INVOICE ID	ORDER ID	REFERENCE	INVOICE DATE	STATUS	HOLD REASON	AMOUNT	DESCRIPTION
1	1	INV_PO001	2/28/24	Paid	-		Part payment for vehicle spare parts
2	1	INV_PO001	2/28/24	Paid	-	$600.00	Part payment for vehicle spare parts
3	2	INV_PO002.1	3/1/24	Paid	-	$100,000.00	Part payment for purchase of car
4	3	INV_PO003.1	3/22/24	Paid	-	$22,000.00	Purchase of screens.
5	3	INV_PO003.1	3/22/24	Paid	-	$20,300.00	Purchase of screens.
6	3	INV_PO003.1	3/22/24	Paid	-	$5,000.00	Purchase of screens.
7	1	INV_PO001	2/28/24	Paid	-	$2,000.00	Part payment for vehicle spare parts
8	4	INV_PO004	2/15/24	Paid	-	$2,000.00	Payment for pen
9	4	INV_PO004	2/15/24	Paid	-		Payment for pencil
10	1	INV_PO001	2/28/24	Paid	-	$800.00	Part payment for vehicle spare parts
11	2	INV_PO002.2	5/2/24	Paid	-	$300,000.00	Part payment for purchase of car
12	5	INV_PO005.1	5/18/24	Pending	Damaged magazine	$3,000.00	-
13	5	INV_PO005.1	5/18/24	Paid	-	$5,000.00	-
14	3	INV_PO003.2	5/15/24	Paid	-	$10,000.00	Final payment for purchase of screens.
15	2	INV_PO002.3	8/5/24	Pending	-	$249,000.00	Final payment for purchase of car

- **Dashboard Two Additional Reports**:
  - **Median Orders Table**: Summarizes order references, suppliers, and amounts.



  - **Orders Report Table**: Provides regional order breakdown with invoice details.



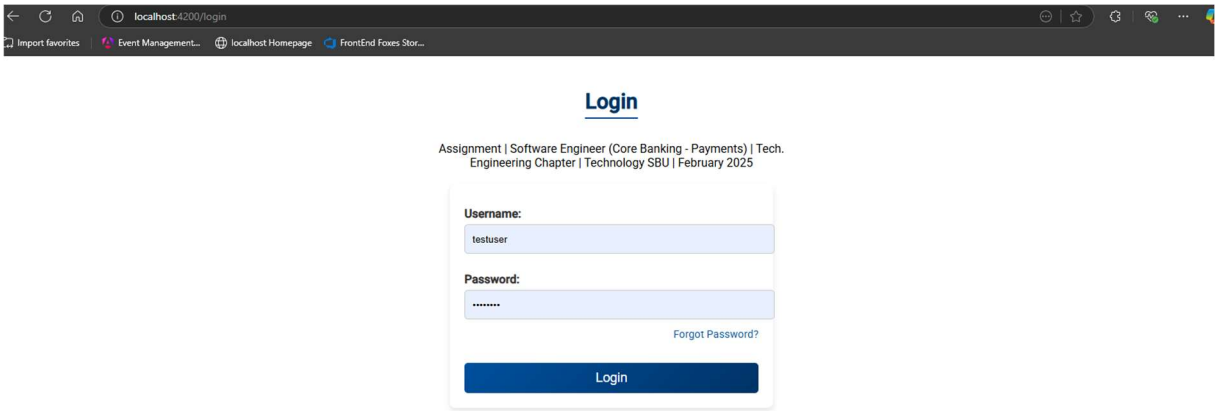
Orders Report

REGION	ORDER REFERENCE	ORDER PERIOD	SUPPLIER NAME	ORDER TOTAL AMOUNT	ORDER STATUS	INVOICE REFERENCE	INVOICE TOTAL AMOUNT	ACTION
Camp Hill	11	-	Lamboni Stat Inc.		Closed	INV_PO011.1		No Action
Camp Hill	4	2024-02	Lamboni Stat Inc.		Closed	INV_PO004|INV_PO004.1		No Action
Curepipe	12	2024-08	Pegasus Ltd		Open	INV_PO012.1|INV_PO012.2|INV_PO012.3		No Action
Curepipe	1	2024-01	Pegasus Ltd		Closed	INV_PO001|INV_PO001.1		No Action
Highlands	7	-	Safedest Taxi Services					

  - **Supplier Orders Summary**: Monthly breakdown of suppliers and total orders.

2. Login Component (login.component.html)

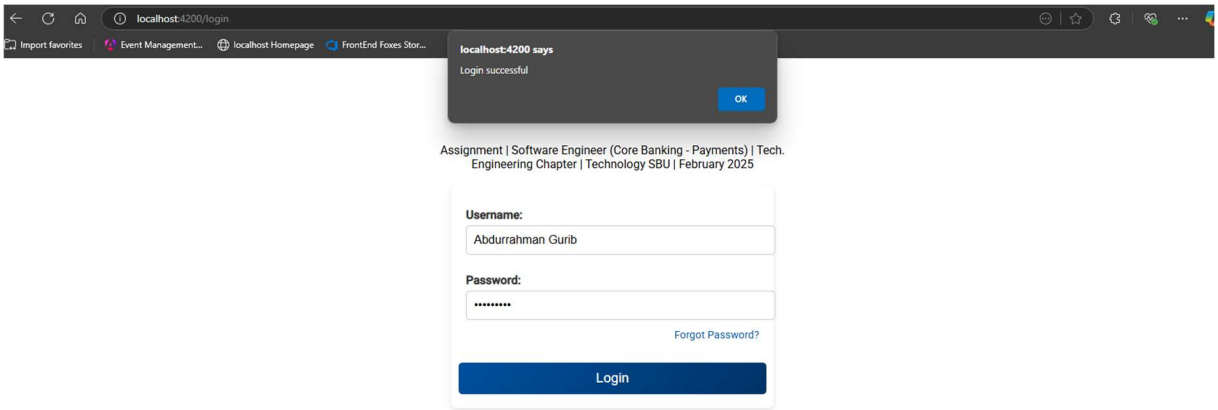
Handles user authentication via login credentials.



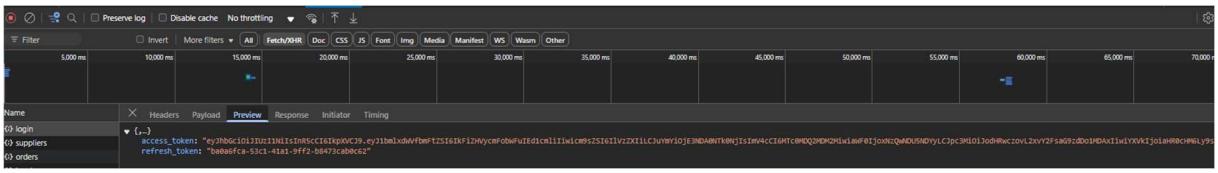
Login Features:

- **User Input Fields**: Email and password fields.
- **Form Validation**: Checks for required fields and valid credentials.
- **Error Handling**: Displays error messages if login fails.
- **API Integration**: Calls authentication service to verify user login.

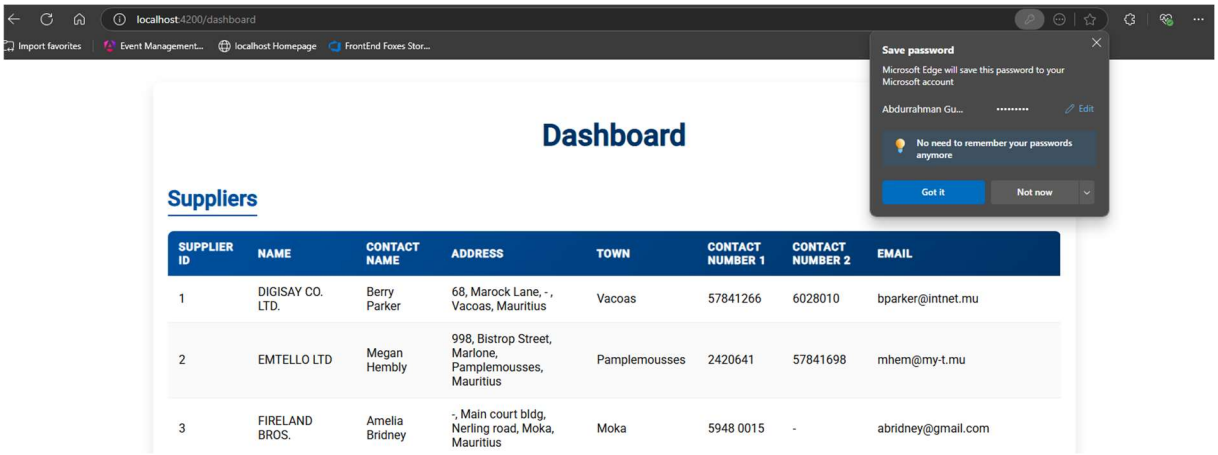
Unit Test: Logged with correct User name and Password



Secure Access token and refresh token created in Developer Tools In the Network Tab

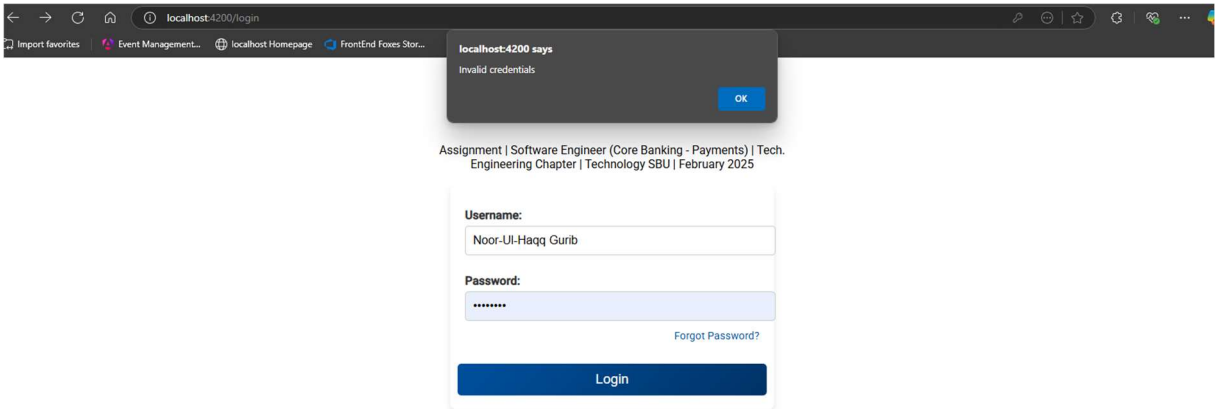


Then directed into dashboard screen

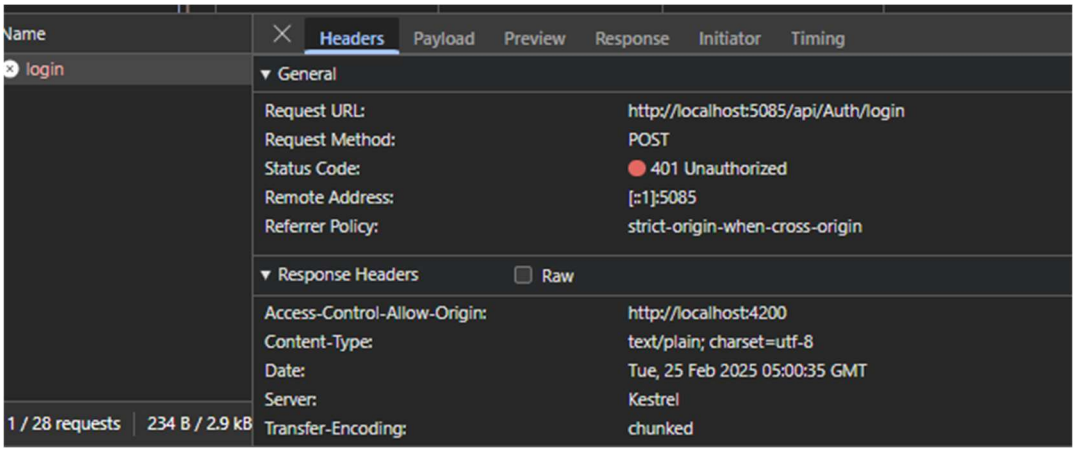


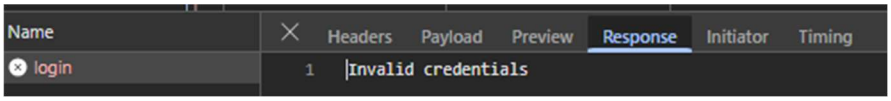
SUPPLIER ID	NAME	CONTACT NAME	ADDRESS	TOWN	CONTACT NUMBER 1	CONTACT NUMBER 2	EMAIL
1	DIGISAY CO. LTD.	Berry Parker	68, Marock Lane, - , Vacoas, Mauritius	Vacoas	57841266	6028010	bparker@intnet.mu
2	EMTELLO LTD	Megan Hembly	998, Bistrop Street, Marlone, Pamplemousses, Mauritius	Pamplemousses	2420641	57841698	mhem@my-t.mu
3	FIRELAND BROS.	Amelia Bridney	-, Main court bldg, Nerling road, Moka, Mauritius	Moka	5948 0015	-	abridney@gmail.com

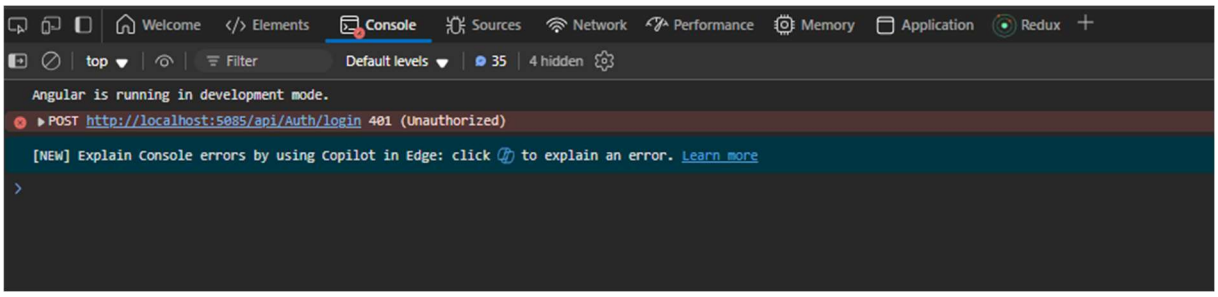
Unit Test Logged with incorrect Username and Password, trigger validations



Unauthorised access Api Status Code to login

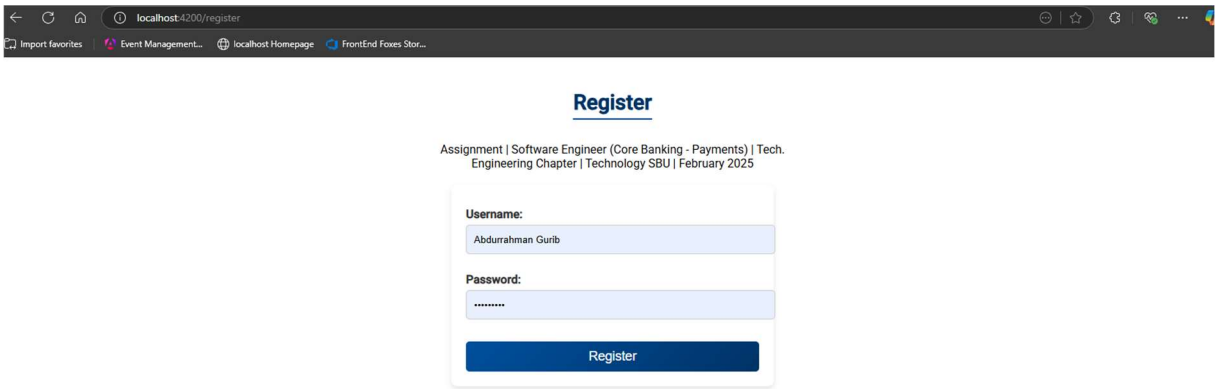






3. Register Component (register.component.html)

Handles new user registration.

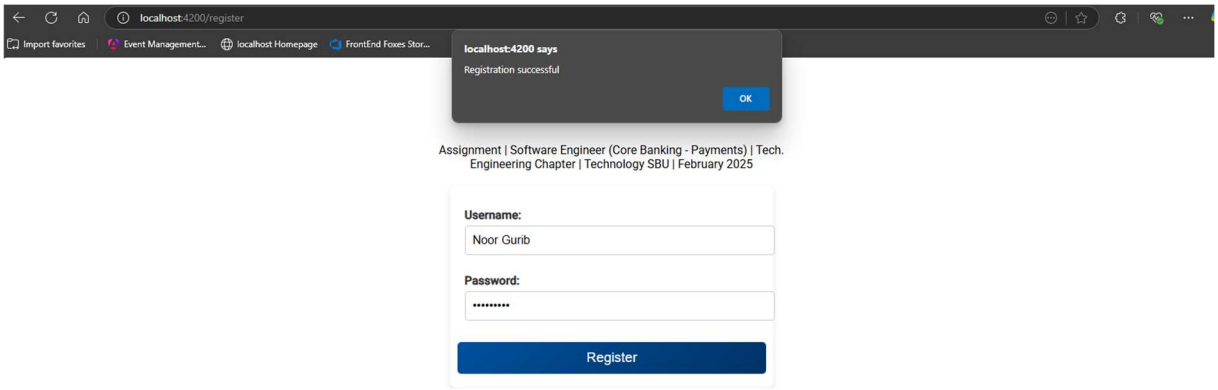


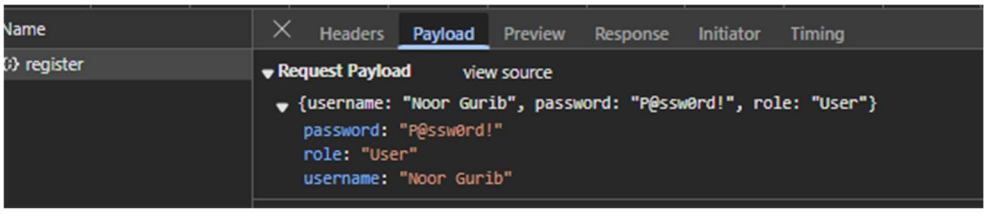
Register Features:

- **User Input Fields**: Name, email, password, and confirm password fields.
- **Form Validation**: Ensures valid input, matching passwords, and required fields.

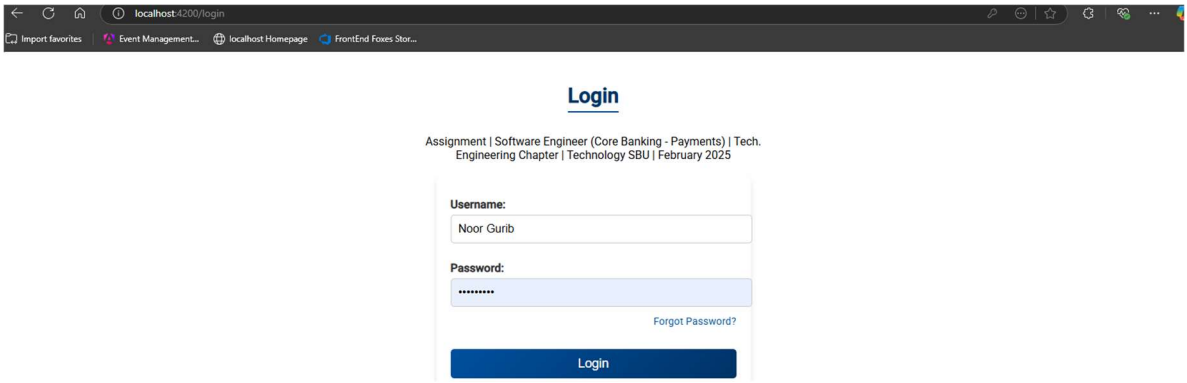
- **Error Handling**: Displays relevant error messages.
- **API Integration**: Sends registration data to the backend.

Unit Test: Created a new account to register a user

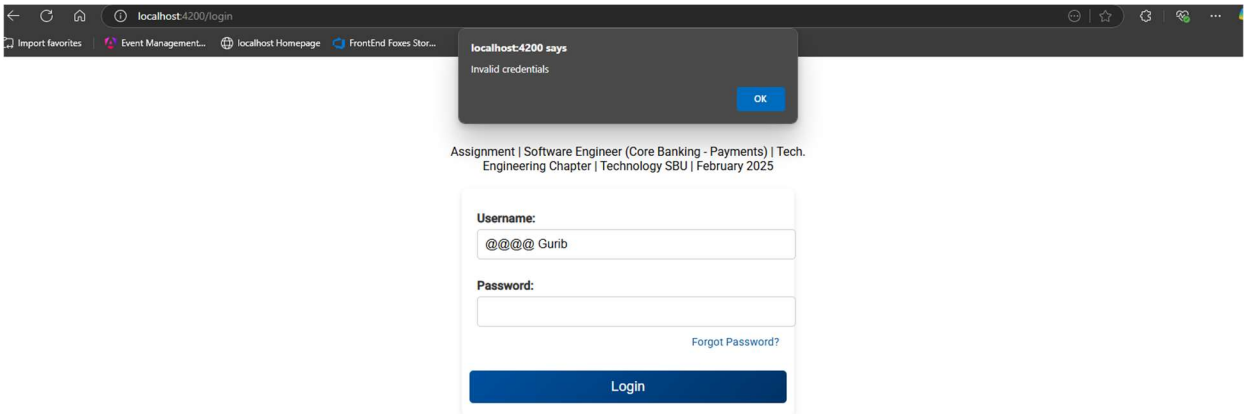




Redirected to login page



Unit Test: Register with invalid characters



Services & API Integration

1. api.service.ts

Handles API communication for retrieving suppliers, orders, invoices, and authentication data.

src > app > core > TS api.service.ts > ApiService > getSupplierOrdersSummary

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ApiService {
  private baseUrl = 'http://localhost:5085/api/Migration'; // adjust if needed

  constructor(private http: HttpClient) { }

  getSuppliers(): Observable<any[]> {
    return this.http.get<any[]>(`${this.baseUrl}/suppliers`);
  }

  getOrders(): Observable<any[]> {
    return this.http.get<any[]>(`${this.baseUrl}/orders`);
  }

  getInvoices(): Observable<any[]> {
    return this.http.get<any[]>(`${this.baseUrl}/invoices`);
  }

  // Fetch Median Order Data
  getMedianOrder(): Observable<any[]> {
    return this.http.get<any[]>(`http://localhost:5085/api/MedianOrder`);
  }

  // Fetch Order Management Data
  getOrderManagement(): Observable<any[]> {
    return this.http.get<any[]>(`http://localhost:5085/api/OrderManagement`);
  }

  // Fetch Orders Report Data
  getOrdersReport(): Observable<any[]> {
    return this.http.get<any[]>(`http://localhost:5085/api/Report/orders`);
  }

  // Fetch Supplier Orders Summary
  getSupplierOrdersSummary(): Observable<any[]> {
    return this.http.get<any[]>(`http://localhost:5085/api/SupplierOrders/summary`);
  }
}
```

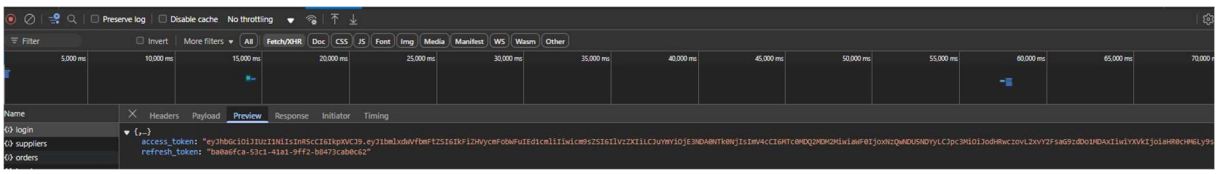
2. Authentication Service (auth.service.ts)

src > app > services > TS auth.service.ts > AuthService > logout

```
1   import { Injectable } from '@angular/core';
2   import { HttpClient } from '@angular/common/http';
3   import { Observable } from 'rxjs';
4
5   @Injectable({
6     providedIn: 'root'
7   })
8   export class AuthService {
9     private apiUrl = 'http://localhost:5085/api/Auth';
10
11    constructor(private http: HttpClient) {}
12
13    register(user: any): Observable<any> {
14      return this.http.post(`${this.apiUrl}/register`, user);
15    }
16
17    login(credentials: any): Observable<any> {
18      return this.http.post(`${this.apiUrl}/login`, credentials);
19    }
20
21    saveToken(token: string) {
22      localStorage.setItem('jwt_token', token);
23    }
24
25    getToken(): string | null {
26      return localStorage.getItem('jwt_token');
27    }
28
29    logout() {
30      localStorage.removeItem('jwt_token');
31    }
32  }
33
```

- **Login & Logout**: Manages user authentication.

- **Token Handling**: Stores and retrieves authentication tokens.



- **User Authorization**: Ensures secure access control.

Routing & Navigation

1. app-routing.module.ts

- Defines routes for the application, including authentication and dashboard pages.
- Uses Angular's RouterModule for navigation control.

2. app.routes.ts

- Centralizes application-wide route definitions.

```
src > app > TS app-routing.module.ts > ...
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './components/login/login.component';
import { RegisterComponent } from './components/register/register.component';
import { DashboardtwoComponent } from './components/dashboardtwo/dashboardtwo.component';


export const routes: Routes = [

  { path: 'dashboardtwo', component: DashboardtwoComponent },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
  {
    path: 'dashboard',
    loadChildren: () => import('./features/dashboard/dashboard.module').then(m => m.DashboardModule)
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Styling & Configuration

1. styles.css

- Provides global styles for the application.

2. app.config.ts

- Stores configuration settings for API endpoints and environment variables.

3. app.module.ts & dashboard.module.ts

- Declares components and imports necessary Angular modules.

Unit Test for Login using **JEST npx Framework**

```
9   describe('LoginComponent', () => {
10    let component: LoginComponent;
11    let fixture: ComponentFixture<LoginComponent>;
12    let authService: AuthService;
13
14    beforeEach(async () => {
15      await TestBed.configureTestingModule({
16        imports: [FormsModule, HttpClientTestingModule, RouterTestingModule],
17        declarations: [LoginComponent],
18        providers: [AuthService]
19      })
20      .compileComponents();
21
22      fixture = TestBed.createComponent(LoginComponent);
23      component = fixture.componentInstance;
24      authService = TestBed.inject(AuthService);
25      fixture.detectChanges();
26    });
27
      Run | Debug
28    it('should create', () => {
29      expect(component).toBeTruthy();
30    });
31
      Run | Debug
32    it('should call login method on form submit', () => {
33      spyOn(component, 'login');
34      const form = fixture.nativeElement.querySelector('form');
35      form.dispatchEvent(new Event('submit'));
36      expect(component.login).toHaveBeenCalled();
37    });
38
      Run | Debug
39    it('should login successfully', () => {
40      const credentials = { username: 'test', password: 'test' };
41      component.credentials = credentials;
42      spyOn(authService, 'login').and.returnValue(of({ token: '12345' }));
43      spyOn(authService, 'saveToken');
44      spyOn(window, 'alert');
45      spyOn(component['router'], 'navigate');
46
47      component.login();
48
49      expect(authService.login).toHaveBeenCalledWith(credentials);
50      expect(authService.saveToken).toHaveBeenCalledWith('12345');
51      expect(window.alert).toHaveBeenCalledWith('Login successful');
52      expect(component['router'].navigate).toHaveBeenCalledWith(['/dashboard']);
53    });
54
      Run | Debug
55    it('should show error on login failure', () => {
56      const credentials = { username: 'test', password: 'wrong' };
57      component.credentials = credentials;
58      spyOn(authService, 'login').and.returnValue(throwError({ status: 401 }));
59      spyOn(window, 'alert');
60
61      component.login();
62
63      expect(authService.login).toHaveBeenCalledWith(credentials);
64      expect(window.alert).toHaveBeenCalledWith('Invalid credentials');
65    });
66
      Run | Debug
67    it('should call forgotPassword method when forgot password link is clicked', () => {
68      spyOn(component, 'forgotPassword');
69      const forgotPasswordLink = fixture.nativeElement.querySelector('a');
70      forgotPasswordLink.click();
```

Unit Test for Register using **Jasmine Framework**

```
Run | Debug
describe('RegisterComponent', () => {
  let component: RegisterComponent;
  let fixture: ComponentFixture<RegisterComponent>;
  let authService: AuthService;
  let router: Router;

  beforeEach(async () => {
    const authServiceMock = {
      register: jasmine.createSpy('register').and.returnValue(of({}))
    };

    const routerMock = {
      navigate: jasmine.createSpy('navigate')
    };

    await TestBed.configureTestingModule({
      imports: [FormsModule, RegisterComponent],
      providers: [
        { provide: AuthService, useValue: authServiceMock },
        { provide: Router, useValue: routerMock }
      ]
    })
    .compileComponents();

    fixture = TestBed.createComponent(RegisterComponent);
    component = fixture.componentInstance;
    authService = TestBed.inject(AuthService);
    router = TestBed.inject(Router);
    fixture.detectChanges();
  });

  Run | Debug
  it('should create', () => {
    expect(component).toBeTruthy();
  });

  Run | Debug
  it('should call register method of AuthService on form submit', () => {
    component.user.username = 'testuser';
    component.user.password = 'password';
    component.register();
    expect(authService.register).toHaveBeenCalledWith(component.user);
  });

  Run | Debug
  it('should navigate to login on successful registration', () => {
    component.register();
    expect(router.navigate).toHaveBeenCalledWith(['/login']);
  });

  Run | Debug
  it('should show alert on successful registration', () => {
    spyOn(window, 'alert');
    component.register();
    expect(window.alert).toHaveBeenCalledWith('Registration successful');
  });

  Run | Debug
  it('should show alert on registration failure', () => {
    (authService.register as jasmine.Spy).and.returnValue(throwError('Registration failed'));
    spyOn(window, 'alert');
    component.register();
    expect(window.alert).toHaveBeenCalledWith('Registration failed');
  });
});
```