Step 5: Integrate Okta Sign-In Widget

File: angular.json

```
...
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "node_modules/@fortawesome/fontawesome-free/css/all.min.css",
  "node_modules/@okta/okta-signin-widget/dist/css/okta-sign-in.min.css"
],
...
```

add reference for okta sign in css

Step 5: Integrate Okta Sign-In Widget

```
$ ng generate component components/login
```

File: login.component.html

```
<!-- Container to inject the Okta Sign-In Widget -->
<div class="pt-5">
    <div id="okta-sign-in-widget" class="pt-5"></div>
</div>
```

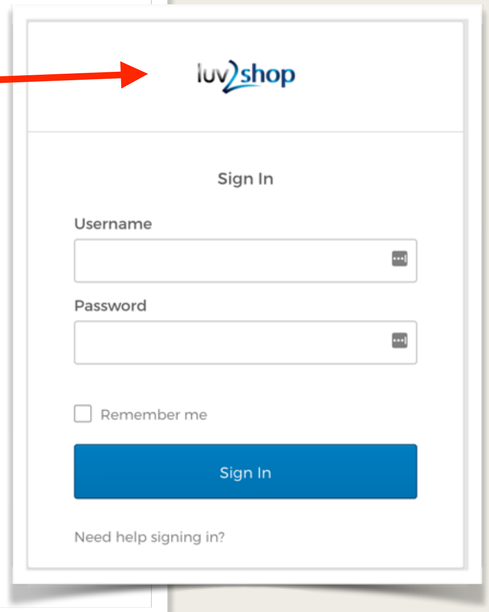
File: login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { OktaAuthService } from '@okta/okta-angular';
import * as OktaSignIn from '@okta/okta-signin-widget';

import myAppConfig from '../../config/my-app-config';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  oktaSignIn: any;

  constructor(private oktaAuthService: OktaAuthService) {

    this.oktaSignIn = new OktaSignIn({
      logo:'assets/images/logo.png',
      baseUrl: myAppConfig.oidc.issuer.split('/oauth2')[0],
      clientId: myAppConfig.oidc.clientId,
      redirectUri: myAppConfig.oidc.redirectUri,
      authParams: {
        pkce: true,
        issuer: myAppConfig.oidc.issuer,
        scopes: myAppConfig.oidc.scopes
      }
}
```

Proof Key for Code Exchange

File: my-app-config.ts

```
export default {

    oidc: {
        clientId: '0oa2agnulNvI7JykI5d6',
        issuer: 'https://dev-5389590.okta.com/oauth2/default',
        redirectUri: 'http://localhost:4200/login/callback',
        scopes: ['openid', 'profile', 'email'],
      }

}
```



Authorization Code Flow with PKCE

- PKCE: Proof Key for Code Exchange
- Recommended approach for controlling access between app and auth server
- Protects against Authorization Code Interception attacks
- Introduces concept of dynamic secrets
- Implemented with a code verifier, code challenge and method

http://www.luv2code.com/okta-pkce

File: login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { OktaAuthService } from '@okta/okta-angular';
import * as OktaSignIn from '@okta/okta-signin-widget';

import myAppConfig from '../../config/my-app-config';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  oktaSignIn: any;

  constructor(private oktaAuthService: OktaAuthService) {

    this.oktaSignIn = new OktaSignIn({
      logo:'assets/images/logo.png',
      baseUrl: myAppConfig.oidc.issuer.split('/oauth2')[0],
      clientId: myAppConfig.oidc.clientId,
      redirectUri: myAppConfig.oidc.redirectUri,
      authParams: {
        pkce: true,
        issuer: myAppConfig.oidc.issuer,
        scopes: myAppConfig.oidc.scopes
      }
    });

  }
  ...
}
```

File: my-app-config.ts

```
export default {

    oidc: {
        clientId: '0oa2agnulNvI7JykI5d6',
        issuer: 'https://dev-5389590.okta.com/oauth2/default',
        redirectUri: 'http://localhost:4200/login/callback',
        scopes: ['openid', 'profile', 'email'],
      }

}
```

File: login.component.ts

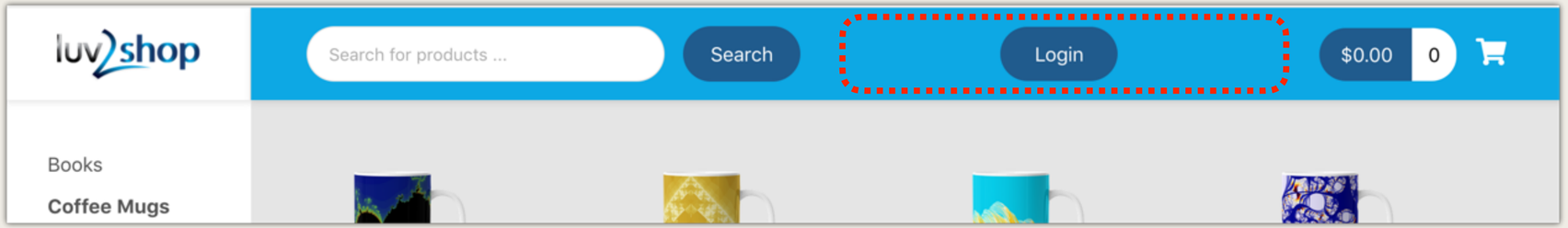
```
import { Component, OnInit } from '@angular/core';
import { OktaAuthService } from '@okta/okta-angular';
import * as OktaSignIn from '@okta/okta-signin-widget';

import myAppConfig from '../../config/my-app-config';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  oktaSignIn: any;

 ...

  ngOnInit() {
    this.oktaSignIn.remove();

    this.oktaSignIn.renderEl({
      el: '#okta-sign-in-widget'}, //this name should be same as div tag id in login.component.html
      (response) => {
        if (response.status === 'SUCCESS') {
          this.oktaAuthService.signInWithRedirect();
        }
      },
      (error) => {
        throw error;
      }
    );
  }
}
```

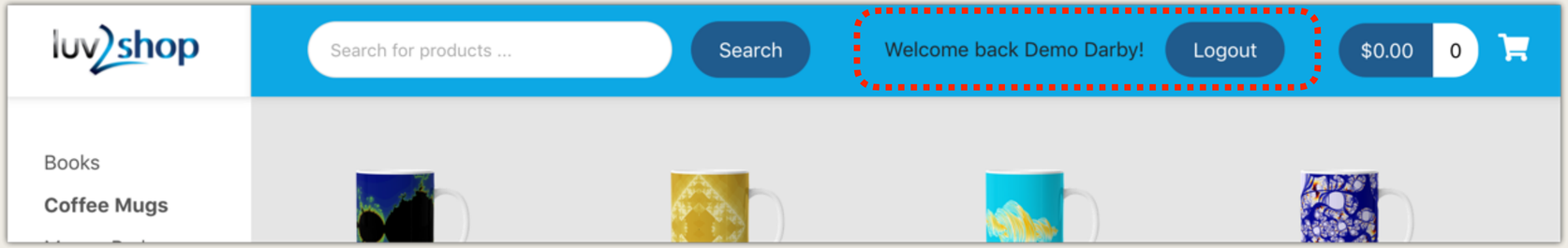
Render element with given id

File: login.component.html

```
<!-- Container to inject the Okta Sign-In Widget -->
<div class="pt-5">
    <div id="okta-sign-in-widget" class="pt-5"></div>
</div>
```

Step 6: Develop login status component for login/logout buttons





Step 6: Develop login status component for login/logout buttons

```
$ ng generate component components/LoginStatus
```

Step 6: Develop login status component for login/logout buttons

File: login-status.component.ts

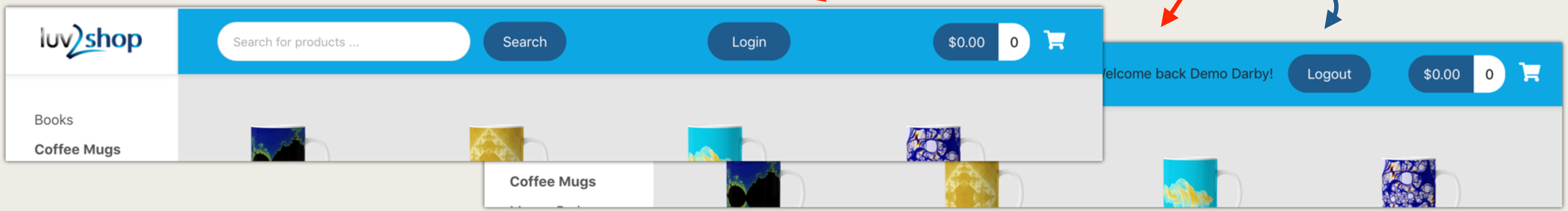
```
import { Component, OnInit } from '@angular/core';
import { OktaAuthService } from '@okta/okta-angular';

@Component({
  selector: 'app-login-status',
  templateUrl: './login-status.component.html',
  styleUrls: ['./login-status.component.css']
})
export class LoginStatusComponent implements OnInit {

  isAuthenticated: boolean = false;
  userFullName: string;

  constructor(private oktaAuthService: OktaAuthService) {
  }

  ngOnInit() {

    // Subscribe to authentication state changes
    this.oktaAuthService.$authenticationState.subscribe(
      (result) => {
        this.isAuthenticated = result;
        this.getUserDetails();
      }
    );
  }
  ...
```

```
  getUserDetails() {
    if (this.isAuthenticated) {

      // Fetch the logged in user details (user's claims)
      //
      // user full name is exposed as a property name
      this.oktaAuthService.getUser().then(
        res => {
          this.userFullName = res.name;
        }
      )
    }
  }

 logout() {
    // Terminates the session with Okta and removes current tokens.
    this.oktaAuthService.signOut();
  }
}
```

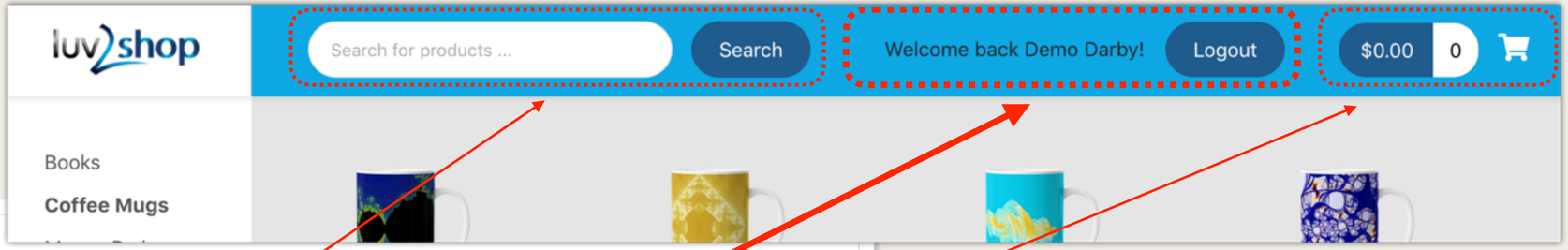
Step 6: Develop login status component for login/logout buttons

File: login-status.component.html

```
<div class="login-status-header">

    <div *ngIf="isAuthenticated && userFullName" class="login-status-user-info">
        Welcome back {{ userFullName }}!
    </div>

    <button *ngIf="!isAuthenticated" routerLink="/login" class="security-btn">Login</button>

    <button *ngIf="isAuthenticated" (click)="logout()" class="security-btn">Logout</button>

</div>
```



Step 6: Develop login status component for login/logout buttons



File: app.component.html

```
<!-- HEADER DESKTOP-->
<header class="header-desktop">
  <div class="section-content section-content-p30">
    <div class="container-fluid">
      <div class="header-wrap">

        <app-search></app-search>

        <app-login-status></app-login-status>

        <app-cart-status></app-cart-status>

      </div>
      <div class="account-wrap"></div>
    </div>
  </div>
</header>
<!-- END HEADER DESKTOP-->
```

Add our new login-status-component to header section

Step 7: Update App Module configs to connect routes

File: app.module.ts

```
...
import {
  OKTA_CONFIG,
  OktaAuthModule,
  OktaCallbackComponent
} from '@okta/okta-angular';

import myAppConfig from './config/my-app-config';

const oktaConfig = Object.assign({
  onAuthRequired: (injector) => {
    const router = injector.get(Router);

    // Redirect the user to your custom login page
    router.navigate(['/login']);
  }
}, myAppConfig.oidc);
...
```

```
const routes: Routes = [
  {path: 'login/callback', component: OktaCallbackComponent },
  {path: 'login', component: LoginComponent },
  ...
]
```

Once the user is authenticated, they are redirected to your app

Normally you would need to parse the response and store the OAuth+OIDC tokens

The OktaCallbackComponent does this for you :-)

Step 7: Update App Module configs to connect routes

File: app.module.ts

```
...
imports: [
  RouterModule.forRoot(routes),
  BrowserModule,
  HttpClientModule,
  NgbModule,
  ReactiveFormsModule,
  OktaAuthModule
],
providers: [ProductService, { provide: OKTA_CONFIG, useValue: oktaConfig }],
...
```

Documentation for Okta Components

- Okta Sign-In Widget
  - https://github.com/okta/okta-signin-widget

- Okta Angular SDK
  - https://github.com/okta/okta-angular