Architecture & Design Patterns

Component-Based Architecture

The application follows Angular's component-based architecture where each feature is encapsulated in its own component with clear responsibilities.

Modular Design

- **Feature Modules**: Components are organized by feature
- **Shared Module**: Common components, pipes, and directives
- **Core Module**: Singleton services and app-wide functionality

Design Patterns Used

1. **Dependency Injection**

Services are injected into components via Angular's DI system:

```
constructor(
  private apiService: ApiService,
  private router: Router
) {}
```

2. **Observer Pattern (RxJS)**

Used extensively for handling asynchronous data:

```
this.apiService.getProducts().subscribe(
  data => this.products = data,
  error => this.handleError(error)
);
```

3. **Guard Pattern**

Route guards protect authenticated routes:

```
{ path: 'cart', canActivate: [AuthGuard], component: CartComponent }
```

4. **Interceptor Pattern**

HTTP interceptors handle cross-cutting concerns:

- Authentication token injection
- Error handling
- Loading states

5. **Service Locator Pattern**

Services are registered at the root level and accessed via DI:

```
@Injectable({ providedIn: 'root' })
export class ApiService { }
```

Revision #1
Created 24 February 2026 07:22:19 by ondeliveloper
Updated 24 February 2026 07:22:59 by ondeliveloper