Getting started

Introduction

Project Overview

OnDelivery Dashboard is a comprehensive web-based logistics and delivery management system built with Angular. The application provides a complete suite of tools for managing deliveries, tracking packages, monitoring couriers, handling returns, and generating reports.

Key Features

- □□ Package tracking and scanning (incoming/outgoing)
- □□ Trucking and courier management
- □□ Comprehensive reporting and analytics
- □□ Finance and tariff management
- □□ Marketplace integration
- □□ Real-time notifications
- □□ Map-based tracking with Leaflet
- □□ JWT-based authentication with role management

Technology Stack

Core Framework

- **Angular 14.3.0** - Main frontend framework
- **TypeScript 4.8.4** - Programming language
- **RxJS 7.8.1** - Reactive programming library

UI & Styling

- **Bootstrap 5.3.2** - CSS framework for responsive design
- **Angular Material 14.2.7** - Material Design components
- **Angular Flex Layout 14.0.0** - Responsive layout system
- **ng-bootstrap 13.1.1** - Bootstrap components for Angular

- **Perfect Scrollbar** - Custom scrollbar styling

Additional Libraries

- **Leaflet 1.9.4** - Interactive maps for tracking
- **AngularX QRCode 14.0.0** - QR code generation
- **PDFMake 0.2.9** - PDF generation
- **ExcelJS 4.4.0** - Excel file manipulation
- **DayJS 1.11.10** - Date/time manipulation
- **ng2-dragula 3.2.0** - Drag and drop functionality
- **ngx-image-viewer 1.0.13** - Image viewing component

Development Tools

- **Angular CLI 14.2.13** - Build and development tooling
- **Karma & Jasmine** - Testing framework
- **TSLint** - TypeScript linting
- **Protractor** - End-to-end testing

Starting the project

Prerequisites

- Node.js (v14+ recommended)
- npm (v6+ recommended)
- Angular CLI (`npm install -g @angular/cli`)

Installation

1. **Clone the repository**

```
git clone <repository-url>
cd ondelivery-dash
```

2. **Install dependencies**

```
npm install
```

3. **Configure API endpoints** Edit `src/app/addrezz.ts` to set the appropriate API endpoints:

```
export const OSAS_API = "https://testapi.ondelivery.id";
export const BOOKING_API = "https://demoapisat.ondelivery.id";
export const USAPI_API = "http://localhost:3621";
```

Running the Application

Development Server

```
npm start
# or
ng serve
```

Navigate to `http://localhost:4200/`. The app will automatically reload on file changes.

Production Build

```
npm run build
```

Build artifacts are stored in the `dist/` directory with output hashing for cache busting.

Development Build

```
npm run build-dev
```

Builds without production optimizations.

Run Tests

```
npm test
```

Run Linter

```
npm run lint
```

Project Architecture

Directory Structure

```
src/
├── app/
│   ├── account-management/       # Account operations
│   ├── admin-menu/               # Admin navigation
│   ├── finance-management/       # Financial operations
│   ├── home/                     # Dashboard/home page
│   ├── incoming-scan/            # Incoming package scanning
│   ├── indopaket/                # Regional package handling
│   ├── layouts/                  # Layout components (header, sidebar, breadcrumb)
│   ├── login/                    # Authentication
│   ├── manifest/                 # Manifest management
│   ├── manual-handling/          # Manual handling operations
│   ├── marketplace/              # Marketplace integration
│   ├── material-component/       # Material Design wrappers
│   ├── mitra-courier/            # Courier partner management
│   ├── modal/                    # Modal dialogs
│   ├── models/                   # Data models
│   ├── monitoring-booking/       # Booking monitoring
│   ├── monitoring-courier/       # Courier monitoring
│   ├── notification/             # Notification system
│   ├── outgoing-scan/            # Outgoing package scanning
│   ├── package-return/           # Package returns
│   ├── package-return-handling/  # Return processing
│   ├── pod-approval/             # Proof of Delivery approval
│   ├── profile/                  # User profile
│   ├── reports/                  # Standard reports
│   ├── reports-dc/               # Distribution center reports
│   ├── saldo/                    # Account balance
│   ├── services/                 # Business logic services
│   ├── shared/                   # Shared utilities and components
│   ├── tariff/                   # Pricing and tariff
│   ├── tracking/                 # Package tracking
```

```
│   ├── trucking/              # Trucking operations
│   ├── user-manage/           # User management
│   ├── waybill/               # Waybill management
│   ├── api.service.ts         # Main API service
│   ├── auth.guard.ts          # Authentication guard
│   ├── role.guard.ts          # Role-based authorization guard
│   ├── app.module.ts          # Root module
│   └── app.routing.ts         # Routing configuration
├── assets/                    # Static assets (images, icons)
├── environments/              # Environment configurations
└── index.html                 # Main HTML file
```

Architecture Pattern

The application follows Angular's **module-based architecture**:

1. **Root Module** (`app.module.ts`) - Core module with essential dependencies
2. **Feature Modules** - Lazy-loaded modules for each feature area
3. **Shared Module** - Reusable components and utilities
4. **Services** - Business logic and API communication
5. **Guards** - Route protection and authorization
6. **Models** - TypeScript interfaces for data structures

Module Loading Strategy

- **Eager Loading**: Core components (layouts, login, profile)
- **Lazy Loading**: All feature modules for optimal performance
- Routes are protected by `AuthGuard` requiring authentication