SwiftLogistics Project File Structure

Root Directory Structure

```
SwiftLogistics/
├── README.md
├── docker-compose.yml
├── .gitignore
├── documentation/
│   ├── architecture-diagrams/
│   ├── api-specs/
│   └── solution-report.pdf
├── backend/
│   ├── api-gateway/
│   ├── mock-cms/
│   ├── mock-ros/
│   ├── mock-wms/
│   ├── order-service/
│   ├── tracking-service/
│   ├── user-service/
│   └── shared-libs/
├── frontend/
│   └── client-portal/
├── mobile/
│   └── driver-app/
└── infrastructure/
    ├── message-broker/
    └── database/
```

Backend Structure (Spring Boot Microservices)

1. API Gateway

```
backend/api-gateway/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/swiftlogistics/gateway/
│       │       ├── SwiftLogisticsGatewayApplication.java
│       │       ├── config/
│       │       │   ├── GatewayConfig.java
│       │       │   └── SecurityConfig.java
│       │       └── filter/
│       │           └── AuthenticationFilter.java
│       └── resources/
│           └── application.yml
├── pom.xml
└── Dockerfile
```

2. Mock CMS (SOAP Service)

```
backend/mock-cms/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/swiftlogistics/cms/
│       │       ├── MockCmsApplication.java
│       │       ├── config/
│       │       │   └── WebServiceConfig.java
│       │       ├── endpoint/
│       │       │   ├── ClientManagementEndpoint.java
│       │       │   └── OrderIntakeEndpoint.java
│       │       ├── model/
│       │       │   ├── Client.java
│       │       │   └── Order.java
│       │       ├── repository/
│       │       │   ├── ClientRepository.java
│       │       │   └── OrderRepository.java
│       │       └── service/
│       │           ├── ClientService.java
│       │           └── OrderService.java
│       └── resources/
│           ├── application.yml
│           └── schema/
│               └── clients.xsd
├── pom.xml
└── Dockerfile
```

3. Mock ROS (REST Service)

```
backend/mock-ros/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/swiftlogistics/ros/
│       │       ├── MockRosApplication.java
│       │       ├── controller/
│       │       │   └── RouteController.java
│       │       ├── model/
│       │       │   ├── Route.java
│       │       │   └── DeliveryPoint.java
│       │       └── service/
│       │           └── RouteOptimizationService.java
│       └── resources/
│           └── application.yml
├── pom.xml
└── Dockerfile
```

4. Mock WMS (TCP/IP Messaging)

```
backend/mock-wms/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/swiftlogistics/wms/
│       │       ├── MockWmsApplication.java
│       │       ├── server/
│       │       │   ├── TcpMessageServer.java
│       │       │   └── MessageHandler.java
│       │       ├── model/
│       │       │   ├── Package.java
│       │       │   └── PackageStatus.java
│       │       ├── service/
│       │       │   └── PackageTrackingService.java
│       │       └── protocol/
│       │           ├── MessageProtocol.java
│       │           └── MessageDecoder.java
│       └── resources/
│           └── application.yml
├── pom.xml
└── Dockerfile
```

5. Order Service

```
backend/order-service/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/swiftlogistics/order/
│       │       ├── OrderServiceApplication.java
│       │       ├── controller/
│       │       │   └── OrderController.java
│       │       ├── model/
│       │       │   └── Order.java
│       │       ├── service/
│       │       │   ├── OrderProcessingService.java
│       │       │   └── OrderOrchestrationService.java
│       │       ├── integration/
│       │       │   ├── CmsIntegration.java
│       │       │   ├── RosIntegration.java
│       │       │   └── WmsIntegration.java
│       │       └── repository/
│       │           └── OrderRepository.java
│       └── resources/
│           └── application.yml
├── pom.xml
└── Dockerfile
```

6. Tracking Service

```
backend/tracking-service/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/swiftlogistics/tracking/
│       │       ├── TrackingServiceApplication.java
│       │       ├── controller/
│       │       │   └── TrackingController.java
│       │       ├── websocket/
│       │       │   └── TrackingWebSocketHandler.java
│       │       ├── model/
│       │       │   └── DeliveryStatus.java
│       │       └── service/
│       │           └── RealtimeTrackingService.java
│       └── resources/
│           └── application.yml
├── pom.xml
└── Dockerfile
```

7. User Service

```
backend/user-service/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/swiftlogistics/user/
│       │       ├── UserServiceApplication.java
│       │       ├── config/
│       │       │   └── SecurityConfig.java
│       │       ├── controller/
│       │       │   ├── AuthController.java
│       │       │   └── UserController.java
│       │       ├── model/
│       │       │   └── User.java
│       │       └── service/
│       │           ├── AuthService.java
│       │           └── UserService.java
│       └── resources/
│           └── application.yml
├── pom.xml
└── Dockerfile
```

Frontend Structure (React - Client Portal)

```
frontend/client-portal/
├── public/
│   ├── index.html
│   └── favicon.ico
├── src/
│   ├── components/
│   │   ├── auth/
│   │   │   └── Login.jsx
│   │   ├── orders/
│   │   │   ├── OrderForm.jsx
│   │   │   ├── OrderList.jsx
│   │   │   └── OrderTracking.jsx
│   │   └── dashboard/
│   │       └── Dashboard.jsx
│   ├── services/
│   │   ├── api.js
│   │   ├── authService.js
│   │   └── orderService.js
│   ├── App.js
│   └── index.js
├── package.json
└── Dockerfile
```

Mobile App Structure (Flutter - Driver App)

```
mobile/driver-app/
├── lib/
│   ├── main.dart
│   ├── screens/
│   │   ├── login_screen.dart
│   │   ├── delivery_list_screen.dart
│   │   ├── delivery_details_screen.dart
│   │   └── route_screen.dart
│   ├── services/
│   │   ├── api_service.dart
│   │   └── auth_service.dart
│   ├── models/
│   │   ├── delivery.dart
│   │   └── route.dart
│   └── widgets/
│       ├── delivery_card.dart
│       └── status_update_form.dart
├── pubspec.yaml
└── android/
```

Infrastructure Structure

```
infrastructure/
├── message-broker/
│   └── rabbitmq/
│       ├── rabbitmq.conf
│       └── Dockerfile
└── database/
    └── postgresql/
        ├── init/
        │   └── 01-create-databases.sql
        └── Dockerfile
```

What You Must Complete (Assignment Requirements Only)

1. Mock Systems ✅

- **Mock CMS**: SOAP-based client management and order intake
- **Mock ROS**: REST-based route optimization
- **Mock WMS**: TCP/IP messaging for warehouse operations

2. Middleware Architecture ✅

- **Protocol Translation**: Bridge SOAP ↔ REST ↔ TCP/IP
- **Service Orchestration**: Coordinate between mock systems
- **Message Broker**: Handle asynchronous communication

3. Client Applications ✅

- **Web Portal**: Client login, order submission, status tracking
- **Mobile App**: Driver manifest, route view, delivery updates

4. Core Functionality ✅

- **Order Processing Workflow**: End-to-end order handling
- **Real-time Updates**: Live tracking using WebSockets
- **High-volume Processing**: Handle multiple orders asynchronously

5. Required Documentation ✅

- **Architecture Diagrams**: Conceptual and implementation
- **Alternative Architectures**: 2 alternatives with rationales
- **Technology Justification**: Why specific tools chosen
- **Security Considerations**: Information security analysis

6. Demonstration ✅

- **Basic Order Flow**: Client submission through all systems
- **Real-time Client UI**: Show live updates
- **10-minute Presentation**: All team members participate

This structure focuses ONLY on what the assignment explicitly requires - no extras.