AGL-Inspired DES Head Unit Project Structure

Based on Automotive Grade Linux (AGL) architecture, here's the recommended project structure following AGL's 5-layer architecture:

🏗️ AGL Architecture Layers

Layer 1: Operating System & BSP (Board Support Package)

- **Location**: Handled by Yocto Linux (in `yocto/` directory)
- **Purpose**: Linux kernel, device drivers, hardware abstraction

Layer 2: Services Layer

- **Location**: `src/services/`
- **Purpose**: System services, middleware, core functionality

Layer 3: Application Framework Layer

- **Location**: `src/framework/`
- **Purpose**: APIs, bindings, common libraries

Layer 4: Applications Layer

- **Location**: `src/applications/`
- **Purpose**: Main applications (head-unit, media-player, etc.)

Layer 5: HMI (Human Machine Interface) Layer

- **Location**: `src/hmi/`
- **Purpose**: User interface, QML components, styling

📁 Complete Project Structure

```
DES_Head-Unit/
├── CMakeLists.txt              # 🏠 AGL-inspired root build config
├── README.md                   # 📖 Project documentation
├── LICENSE                 # ⚖️ License file
├── FILESTRUCTURE.md            # 📋 This file
│
├── .github/                # 🤖 GitHub Actions & CI/CD
│   ├── workflows/
│   │   ├── ci.yml              # Multi-platform CI
│   │   ├── agl-build.yml       # AGL-style builds
│   │   └── deploy.yml          # Deployment pipeline
│   └── ISSUE_TEMPLATE/         # Issue templates
│
├── scripts/                    # 🔧 Build and setup scripts
│   ├── agl-setup.sh            # AGL environment setup
│   ├── build-profiles.sh       # Build different AGL profiles
│   ├── des-head-unit.desktop.in    # Linux desktop file template
│   └── des-head-unit.manifest.in   # AGL manifest template
│
├── yocto/                      # 🐧 AGL/Yocto Layer 1: OS & BSP
│   ├── meta-des-head-unit/     # Custom Yocto layer
│   ├── conf/                   # Build configuration
│   └── recipes-*/              # Yocto recipes
│
├── src/                        # 💻 Source code (Layers 2-5)
│   ├── CMakeLists.txt          # Source build configuration
│   │
│   ├── services/               # 🔧 Layer 2: Services
│   │   ├── CMakeLists.txt
│   │   ├── can-service/        # CAN bus service
│   │   │   ├── can-binding.cpp     # AGL-style service binding
│   │   │   ├── can-manager.h
│   │   │   └── CMakeLists.txt
│   │   ├── audio-service/      # Audio management service
│   │   ├── network-service/    # Network connectivity
│   │   ├── security-service/   # Security & authentication
│   │   └── ipc-service/        # Inter-process communication
│   │
│   ├── framework/              # 🏗️ Layer 3: Application Framework
│   │   ├── CMakeLists.txt
│   │   ├── agl-bindings/       # AGL service bindings
│   │   │   ├── binding-common.h
│   │   │   ├── des-binding.cpp
│   │   │   └── CMakeLists.txt
│   │   ├── apis/               # Common APIs
│   │   │   ├── vehicle-api.h   # Vehicle data API
│   │   │   ├── media-api.h     # Media control API
│   │   │   └── settings-api.h  # Settings API
│   │   ├── common/             # Shared libraries
│   │   │   ├── utils.cpp
│   │   │   ├── logger.cpp
│   │   │   └── config-manager.cpp
│   │   └── ipc/                 # IPC framework
│   │       ├── dbus-wrapper.cpp   # D-Bus abstraction
│   │       └── message-router.cpp
│   │
│   ├── applications/            # 📱 Layer 4: Applications
│   │   ├── CMakeLists.txt
│   │   ├── head-unit/           # Main head unit app (IVI profile)
│   │   │   ├── main.cpp
│   │   │   ├── head-unit-app.cpp
│   │   │   ├── head-unit-app.h
│   │   │   └── CMakeLists.txt
│   │   ├── instrument-cluster/    # Instrument cluster (IC profile)
│   │   │   ├── cluster-main.cpp
│   │   │   ├── cluster-app.cpp
│   │   │   └── CMakeLists.txt
│   │   ├── media-player/        # Media application
│   │   │   ├── media-main.cpp
│   │   │   ├── media-controller.cpp
│   │   │   └── CMakeLists.txt
│   │   ├── ambient-lighting/     # Ambient lighting control
│   │   │   ├── lighting-main.cpp
│   │   │   ├── lighting-controller.cpp
│   │   │   └── CMakeLists.txt
│   │   └── settings/            # Settings application
│   │       ├── settings-main.cpp
│   │       └── CMakeLists.txt
│   │
│   └── hmi/                     # 🎨 Layer 5: HMI (User Interface)
│       ├── CMakeLists.txt
│       ├── qml/                 # QML user interfaces
│       │   ├── Main.qml         # Main application UI
│       │   ├── Dashboard.qml     # Dashboard interface
│       │   ├── MediaPlayer.qml   # Media player UI
│       │   ├── Settings.qml      # Settings interface
│       │   ├── InstrumentCluster.qml # Cluster UI
│       │   └── components/       # Reusable QML components
│       │       ├── SpeedGauge.qml
│       │       ├── MediaControl.qml
│       │       ├── Navigation.qml
│       │       └── VehicleStatus.qml
│       ├── styles/              # UI styling and themes
│       │   ├── AglTheme.qml     # AGL-inspired theme
│       │   ├── Colors.qml       # Color definitions
│       │   └── Fonts.qml        # Font definitions
│       ├── assets/          # Images, icons, resources
│       │   ├── icons/
│       │   ├── images/
│       │   └── fonts/
│       └── translations/     # Internationalization
│           ├── en_US.ts
│           ├── de_DE.ts
│           └── ja_JP.ts
│
├── tests/                   # 🧪 Testing framework
│   ├── CMakeLists.txt
│   ├── unit/                # Unit tests
│   │   ├── test_services.cpp    # Service layer tests
│   │   ├── test_framework.cpp   # Framework tests
│   │   ├── test_applications.cpp # Application tests
│   │   └── CMakeLists.txt
│   ├── integration/          # Integration tests
│   │   ├── test_can_integration.cpp
│   │   ├── test_ipc_integration.cpp
│   │   └── CMakeLists.txt
│   └── e2e/                 # End-to-end tests
│       ├── test_user_workflows.cpp
│       └── CMakeLists.txt
│
├── docker/                  # 🐳 Container configurations
│   ├── Dockerfile.agl        # AGL-compatible container
│   ├── Dockerfile.dev        # Development container
│   └── docker-compose.yml     # Multi-service setup
│
└── docs/                    # 📚 Documentation
    ├── architecture.md        # System architecture
    ├── agl-integration.md      # AGL integration guide
    ├── api-reference.md       # API documentation
    ├── deployment.md          # Deployment guide
    └── contributing.md        # Contribution guidelines
```

🎯 AGL Profile Support

The structure supports multiple AGL profiles:

IVI Profile (In-Vehicle Infotainment)

- **Applications**: head-unit, media-player, settings

- **Services**: audio-service, network-service
- **HMI**: Full dashboard UI with media controls

IC Profile (Instrument Cluster)

- **Applications**: instrument-cluster
- **Services**: can-service (for vehicle data)
- **HMI**: Minimalist cluster UI with gauges

Telematics Profile (Optional)

- **Services**: network-service, security-service
- **Applications**: connectivity features
- **HMI**: Status and diagnostic interfaces

🚀 Benefits of This Structure

1. **AGL Compatibility**: Follows official AGL architecture patterns
2. **Modular Design**: Each layer is independent and testable
3. **Profile Support**: Can build different configurations (IVI, IC, Telematics)
4. **Service-Oriented**: Matches AGL's service-binding architecture
5. **Container Ready**: Supports AGL's container isolation features
6. **Yocto Integration**: Compatible with AGL's Yocto-based builds

🔧 Build Commands

```bash
# Build IVI profile only
cmake -DBUILD_IVI_PROFILE=ON -DBUILD_IC_PROFILE=OFF ..

# Build IC profile only
cmake -DBUILD_IC_PROFILE=ON -DBUILD_IVI_PROFILE=OFF ..

# Build all profiles with AGL features
cmake -DBUILD_IVI_PROFILE=ON -DBUILD_IC_PROFILE=ON -DBUILD_AGL_BINDINGS=ON ..

# Enable container support (AGL advanced feature)
cmake -DENABLE_CONTAINER_SUPPORT=ON ..
```

This structure makes your project AGL-compatible and follows automotive industry standards while maintaining modern development practices! 🚗💻