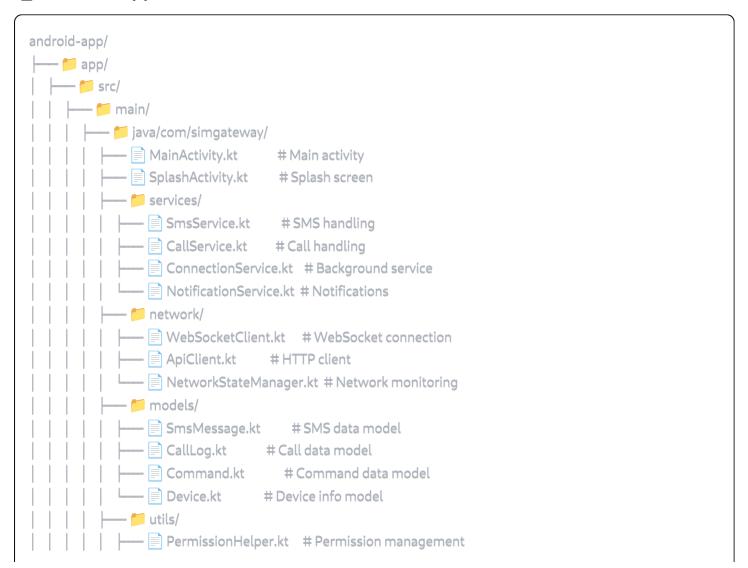
Remote SIM Gateway - Complete Folder Structure

Root Project Structure

```
remote-sim-gateway/
    android-app/
                       # Android bridge application
    backend/
                     # Go backend server
    frontend/
                     # React web dashboard
    docs/
                    # Documentation
    # Deployment configurations
    scripts/
                  # Build and utility scripts
    README.md
                        # Project documentation
    LICENSE
                    # MIT License
    gitignore # Git ignore rules
    docker-compose.yml # Multi-service deployment
    docker-compose.dev.yml # Development environment
    CONTRIBUTING.md # Contribution guidelines
```

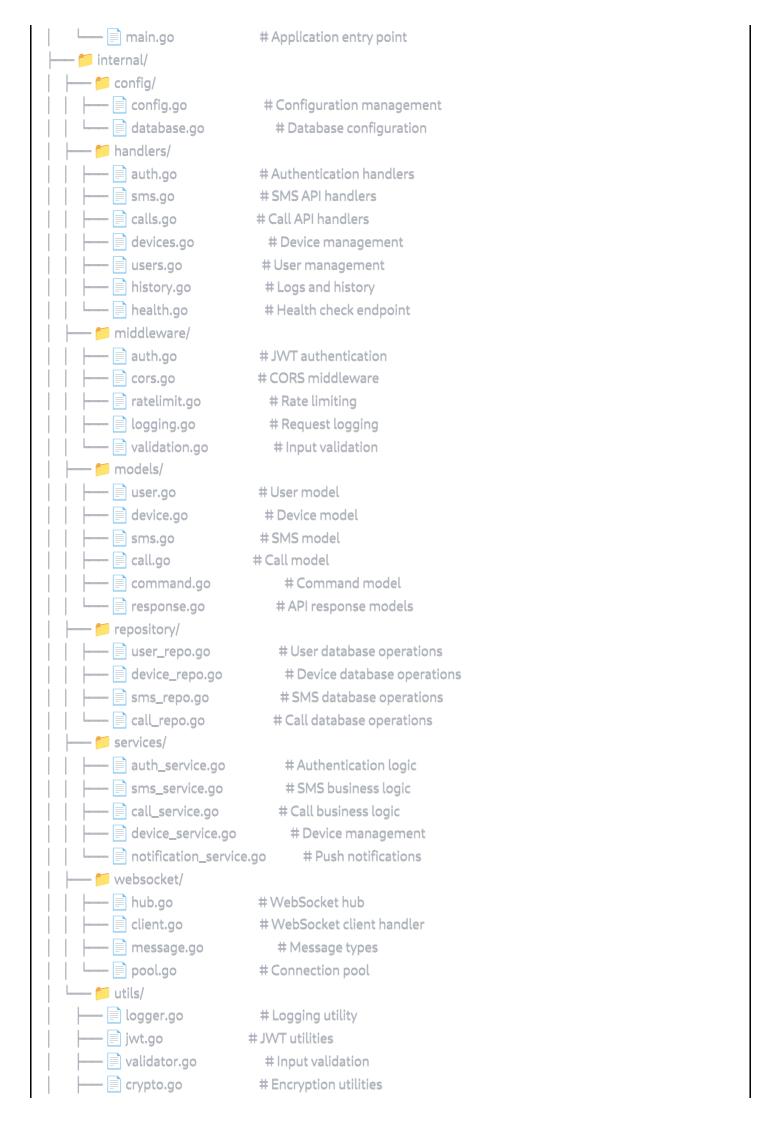
Android App Structure

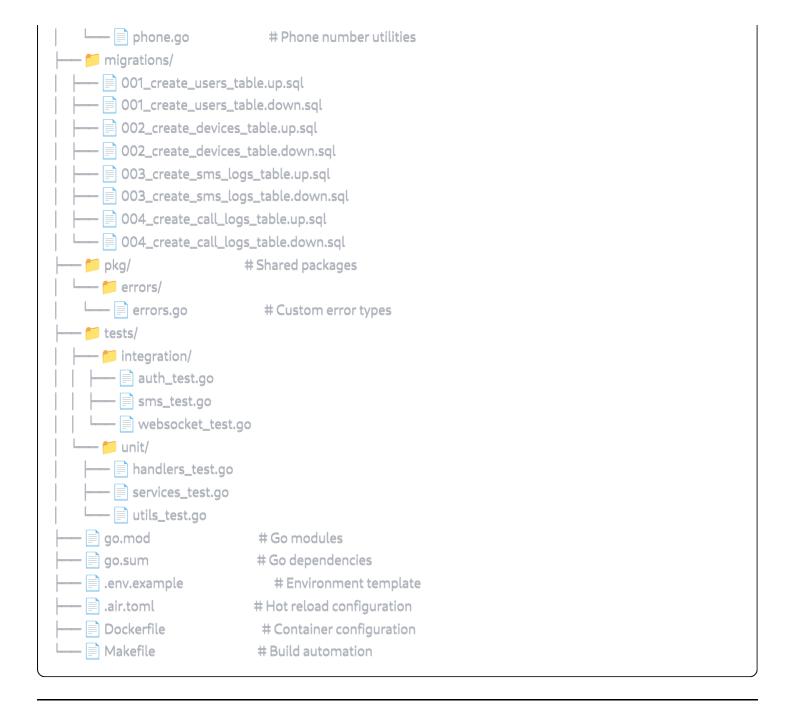


```
DeviceUtils.kt
                               # Device info utilities
             Constants.kt
                               # App constants
             Logger.kt
                              # Logging utility
                                 # SMS broadcast receiver
             SmsReceiver.kt
             CallReceiver.kt
                                # Call state receiver
             BootReceiver.kt
                                 # Auto-start on boot
          – 📄 DashboardFragment.kt 🛮 # Main dashboard
          - SettingsFragment.kt # App settings
          LogsFragment.kt # Activity logs
          - StatusFragment.kt # Connection status
          layout/
            activity_main.xml
             fragment_dashboard.xml
             fragment_settings.xml
           – 📄 fragment_logs.xml
          values/
            strings.xml
             colors.xml
            styles.xml
            config.xml
                             # App configuration
          drawable/
          mipmap/
                              # App icons
        - 📁 xml/
          – 📄 network_security_config.xml
      - AndroidManifest.xml
                                  # App permissions & config
     – 📁 java/com/simgateway/
 build.gradle
                           # App build configuration
 - proguard-rules.pro
                               # Code obfuscation rules
build.gradle
                           # Project build configuration
gradle.properties
                             # Gradle properties
settings.gradle
                            # Project settings
gradlew
                          # Gradle wrapper (Unix)
gradlew.bat
                           # Gradle wrapper (Windows)
gradle/wrapper/
                             # Gradle wrapper files
```

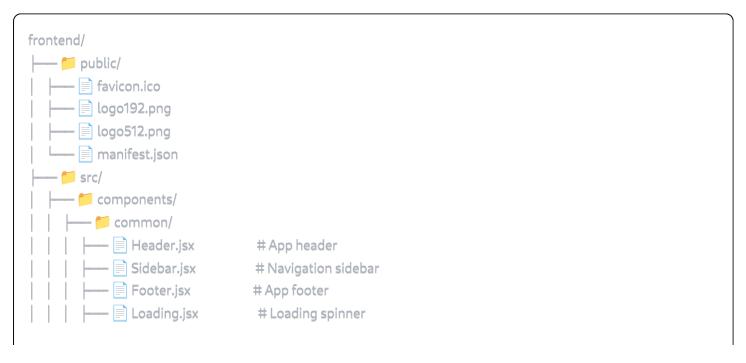
Backend Server Structure



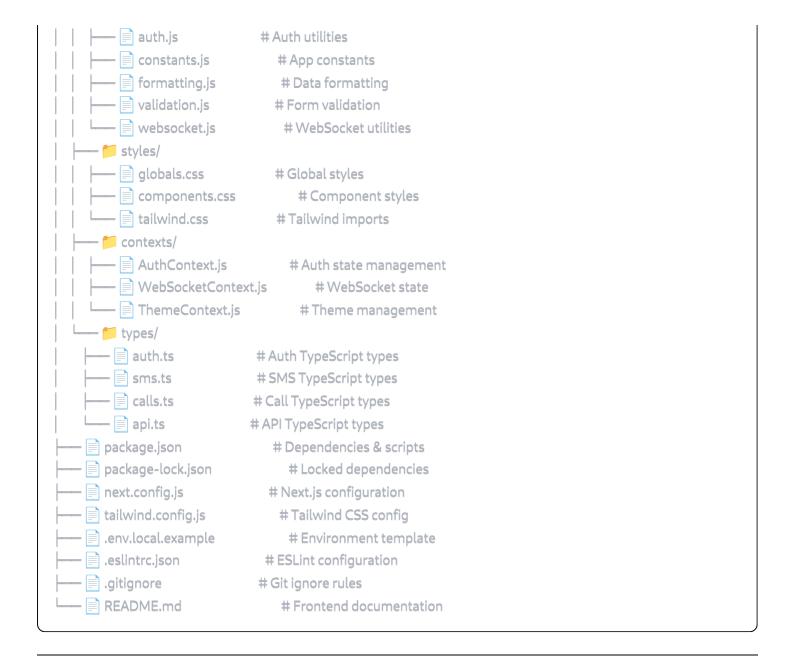




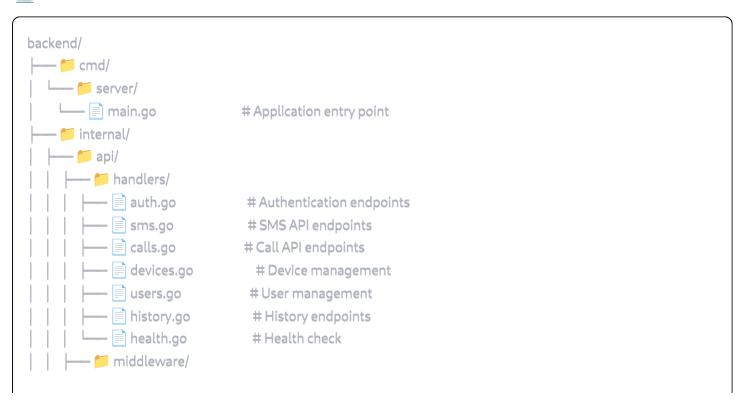
(f) Frontend Web App Structure



│ ├── 🗐 Modal.jsx	# Reusable modal	
Toast.jsx	# Notification toast	
— f auth/		
LoginForm.jsx	# Login component	
RegisterForm.js	x # Registration component	
ProtectedRoute		
dashboard/		
DashboardState	s.jsx # Stats overview	
DeviceStatus.js	# Device connection status	
RecentActivity.j	x # Recent SMS/calls	
QuickActions.js	# Quick send buttons	
sms/		
SmsForm.jsx	# Send SMS form	
SmsHistory.jsx	# SMS history table	
BulkSmsForm.j	sx # Bulk SMS sender	
SmsTemplates.	sx # Message templates	
calls/		
CallForm.jsx	# Make call form	
CallHistory.jsx	# Call history table	
CallStatus.jsx	# Active call status	
devices/		
DeviceList.jsx	# Connected devices	
DeviceCard.jsx	# Individual device card	
AddDevice.jsx	# Add new device	
settings/		
UserSettings.jsx	# User preferences	
SecuritySettings		
ApiSettings.jsx	# API configuration	
— pages/	# Hama/landing page	
index.js	# Home/landing page	
login.js	# Login page	
register.js	# Registration page # Main dashboard	
dashboard.js		
sms.js calls.js	# SMS management	
history.js	# Call management # Combined history	
devices.js	,	
	# Device management	
settings.js	# Settings page	
└── 🗐 _app.js — 📁 hooks/	# Next.js app wrapper	
- nooks/	# Authentication hook	
useWebSocket.js	# WebSocket connection	
useApi.js	# API calls hook	
useApi.js useDeviceStatus.js		
useLocalStorage.js		
— 📁 useLocalStorage.js — 📁 utils/	# Local Storage 1100K	
api.js	# API client	
api.js	# Art client	



Backend Server Structure



auth.go	# JWT validation	
cors.go	# CORS handling	
ratelimit.go	# Rate limiting	
logging.go	# Request logging	
recovery.go	# Panic recovery	
validation.go	# Input validation	
routes/	# Input validation	
api.go	# API route definitions	
websocket.go	# WebSocket routes	
− <mark>™</mark> models/	# WebbocketToutes	
- user.go	# User model & DB methods	
device.go	# Device model	
sms.go	#SMS log model	
— 🖹 call.go	# Call log model	
command.go	# Command model	
base.go	# Base model with common fields	
- services /		
auth_service.go	# Authentication business logic	
sms_service.go	#SMS business logic	
call_service.go	# Call business logic	
device_service.go	# Device management logic	
user_service.go	# User management logic	
notification_service	.go # Push notifications	
— 📁 repository/		
user_repo.go	# User database operations	
device_repo.go	# Device database operations	
sms_repo.go	# SMS database operations	
call_repo.go	# Call database operations	
interfaces.go	# Repository interfaces	
— websocket/		
hub.go	#WebSocket connection hub	
client.go	# WebSocket client handler	
message.go	# Message handling	
— pool.go	# Connection pooling	
commands.go	# Command processing	
database/	11 B 4 1	
connection.go	# Database connection	
migrations.go	# Migration runner	
seeds.go	# Database seeding	
− 📁 utils/	* Character and Longitus	
logger.go	# Structured logging	
jwt.go	# JWT token utilities	
validator.go	# Input validation # Encryption/backing	
crypto.go	# Encryption/hashing # Phone number utilities	
phone.go	# HTTP response helpers	
├── 🗐 response.go └── 🗐 errors.go	# Error handling	
= = = = = = = = = = = = = = = = = = = =	# Ellor Haridalig	

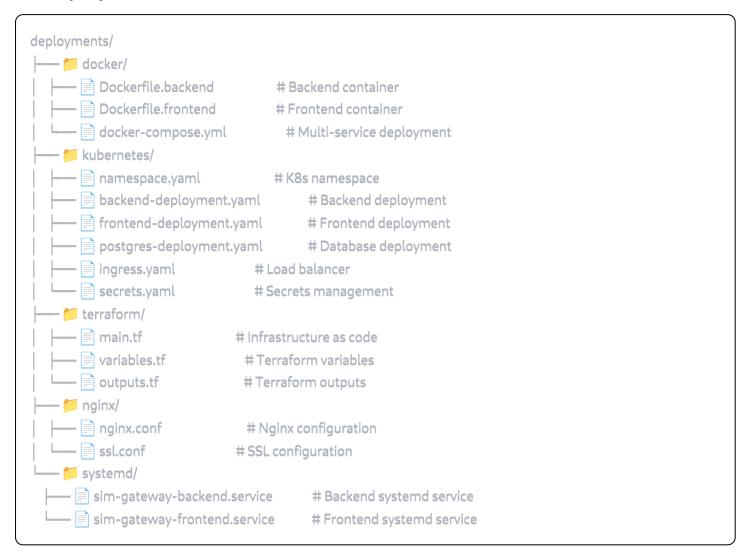


Documentation Structure

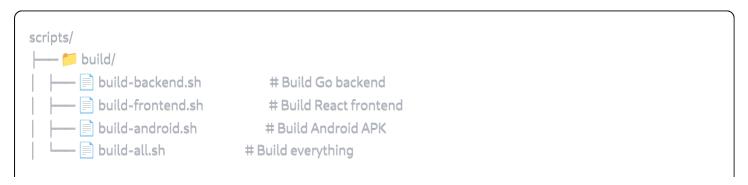


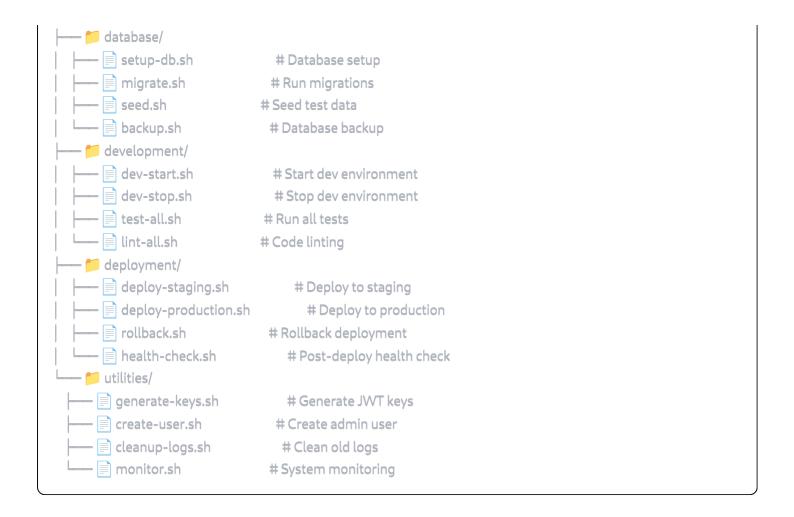
```
| ☐ android-app.png
| ☐ api-docs/
| ☐ openapi.yaml # OpenAPI specification
| ☐ postman-collection.json # Postman collection
| ☐ examples/
| ☐ curl-examples.sh # API examples
| ☐ android-integration.md # Android integration
| ☐ webhook-examples.md # Webhook examples
```

🐳 Deployment Structure



Scripts Structure





Configuration Files

Root Level Config Files



Example .gitignore

gitignore		

```
# Environment files
.env
.env.local
.env.production
# Build outputs
/backend/sim-gateway
/frontend/dist/
/frontend/.next/
/android-app/app/build/
# Dependencies
node_modules/
vendor/
# IDE files
.vscode/
.idea/
*.swp
*.swo
#Logs
*.log
logs/
# Database
*.db
*.sqlite
# Keys and certificates
*.key
*.pem
*.crt
```

The State of the State of the

1. Start Development Environment

```
bash

# Root directory

make dev-start

# or

docker-compose up -d
```

2. Development Order

- 1. Backend First Core API and WebSocket
- 2. Android App SIM bridge functionality
- 3. Frontend Web dashboard
- 4. Integration Testing End-to-end testing

3. Testing Structure

```
# Backend tests

cd backend && go test ./...

# Frontend tests

cd frontend && npm test

# Android tests

cd android-app && ./gradlew test

# Integration tests
./scripts/test-all.sh
```

Build & Deployment

Quick Commands

```
# Build all components
make build-all

# Run development environment
make dev-start

# Deploy to production
make deploy-prod

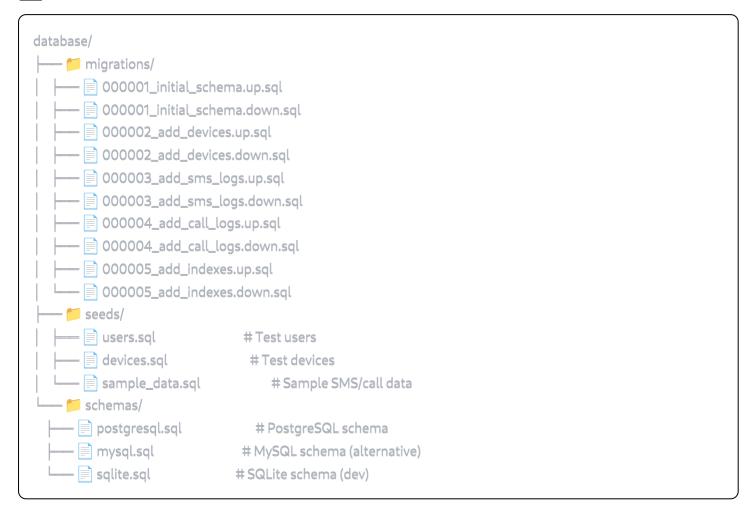
# Run tests
make test-all

# Clean build artifacts
make clean
```

This folder structure provides:

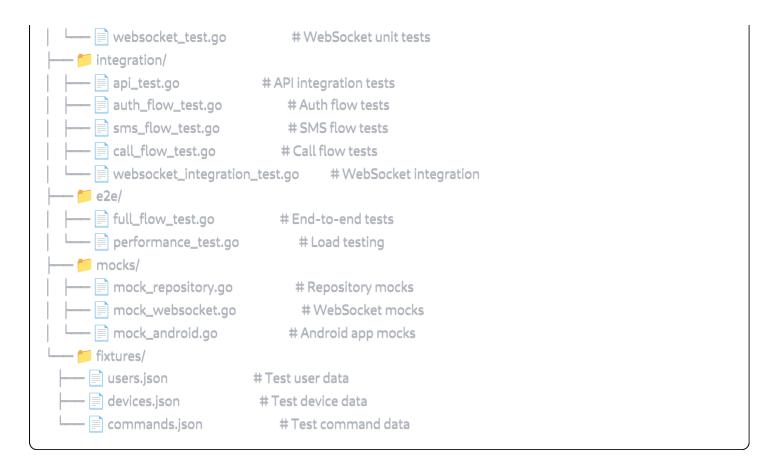
- **Clear separation** of concerns
- **V** Scalable architecture for future features
- Z Easy deployment with Docker/K8s
- **Maintainable codebase** with proper organization
- **Testing support** at all levels

a Database Structure



Testing Structure

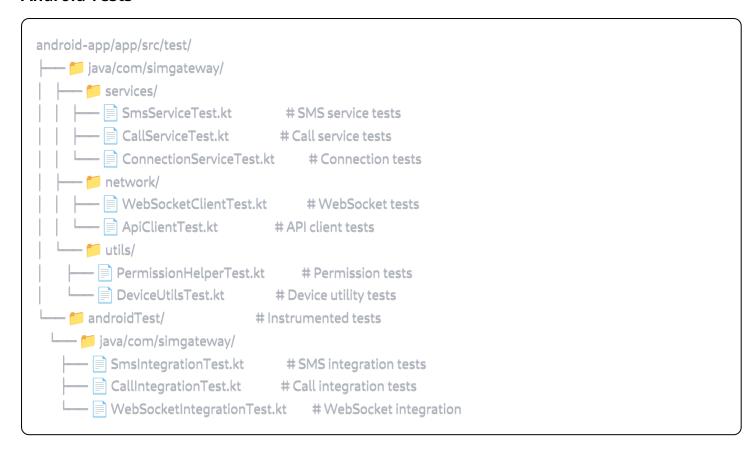
Backend Tests



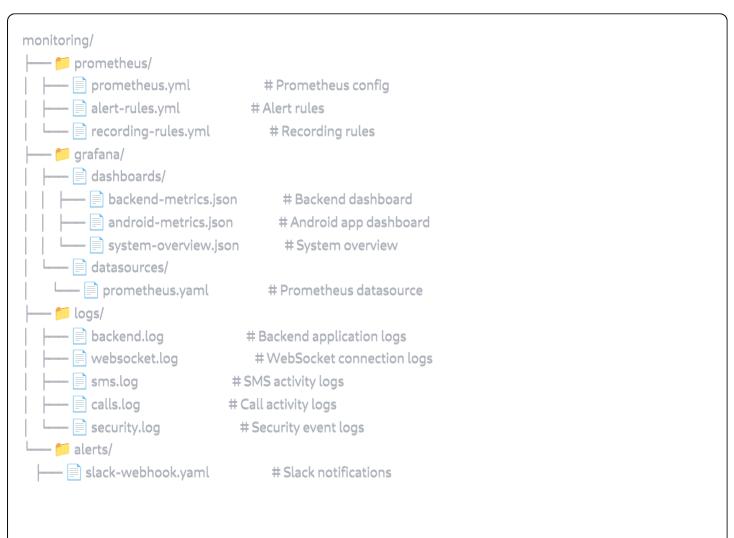
Frontend Tests

```
frontend/tests/
--- components/
  SmsForm.test.jsx
                           # SMS form tests
 CallForm.test.jsx
                            # Call form tests
                            # Device status tests
  — DeviceStatus.test.jsx
    — 📄 Dashboard.test.jsx
                             # Dashboard tests
  – 📁 hooks/
  --- useAuth.test.js
                       # Auth hook tests
  useWebSocket.test.js
                                # WebSocket hook tests
    — useApi.test.js
                    # API hook tests
 --- 📁 utils/
  api.test.js
                         # API utility tests
  --- validation.test.js
                          # Validation tests
  formatting.test.js
                            # Formatting tests
  — 📁 e2e/
  login-flow.spec.js
                            # Login E2E tests
  sms-flow.spec.js
                            # SMS E2E tests
  call-flow.spec.js
                      # Call E2E tests
  – 📁 __mocks__/
   — 📄 api.js
                       # API mocks
 websocket.js
                            # WebSocket mocks
```

Android Tests



III Monitoring & Logs Structure



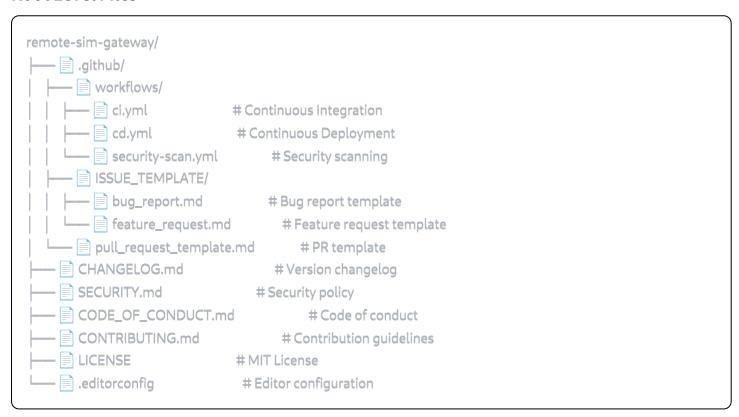
🔒 Security & Secrets Structure

```
security/
 --- 📁 certificates/
    --- server.crt
                                 # SSL certificate
      - server.key
                                 # SSL private key
       - 📄 ca.crt
                               # Certificate authority
       client.crt
                                # Client certificate
     keys/
      – jwt-private.key
                                    # JWT signing key
     — jwt-public.key
                                    # JWT verification key
      encryption.key
                                    # Data encryption key
     — 📄 api-keys.enc
                                   # Encrypted API keys
   – 📁 policies/
     — 📄 rate-limits.yaml
                                    # Rate limiting policies
     – 📄 firewall-rules.yaml
                                     # Firewall configuration
     — 

access-control.yaml
                                      # Access control policies
```

Additional Project Files

Root Level Files

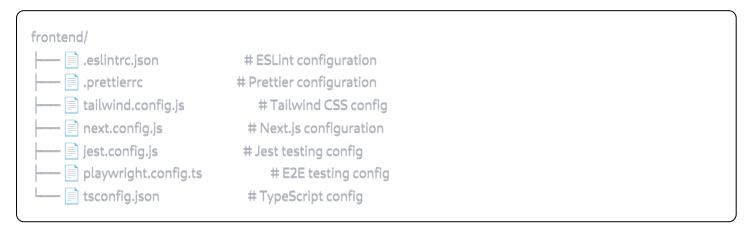


X Development Tools Configuration

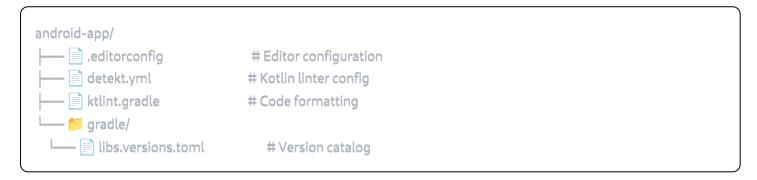
Backend Tools

backend/	
air.toml	# Hot reload config
golangci.yml	# Go linter config
Makefile	# Build automation
L tools/	
generate-docs.go	# API docs generator
imigrate.go	# Migration tool
Seed.go	# Database seeder

Frontend Tools



Android Tools



Section Started Commands

Quick Setup (Recommended)

	 <u> </u>		
bash			

```
#1. Clone repository
git clone https://github.com/yourusername/remote-sim-gateway.git
cd remote-sim-gateway

#2. Start everything with Docker
make dev-start

#3. Open in browser
open http://localhost:3000
```

Manual Setup

```
bash
#1. Backend
cd backend
cp.env.example.env
go mod download
make migrate-up
go run cmd/server/main.go
#2. Frontend (new terminal)
cd frontend
npm install
cp.env.local.example.env.local
npm run dev
#3. Android (Android Studio)
cd android-app
# Open in Android Studio
# Build and install on device
```

File Creation Checklist

✓ Must Create First (Core Files)

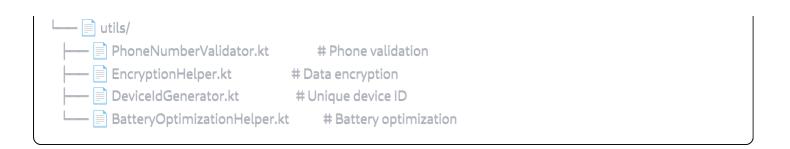
- □ (backend/cmd/server/main.go)
- (backend/internal/config/config.go)
- android-app/app/src/main/java/com/simgateway/MainActivity.kt
- (frontend/pages/index.js
- docker-compose.yml

✓ Essential Configuration Files
backend/.env.example
frontend/.env.local.example
android-app/app/src/main/AndroidManifest.xml
README.md
 ✓ Database Files □ (backend/migrations/000001_initial_schema.up.sql) □ Database connection configuration □ Seed data files
✓ Security Files
☐ JWT key generation
□ SSL certificates
☐ Environment variable templates

Mobile App File Details

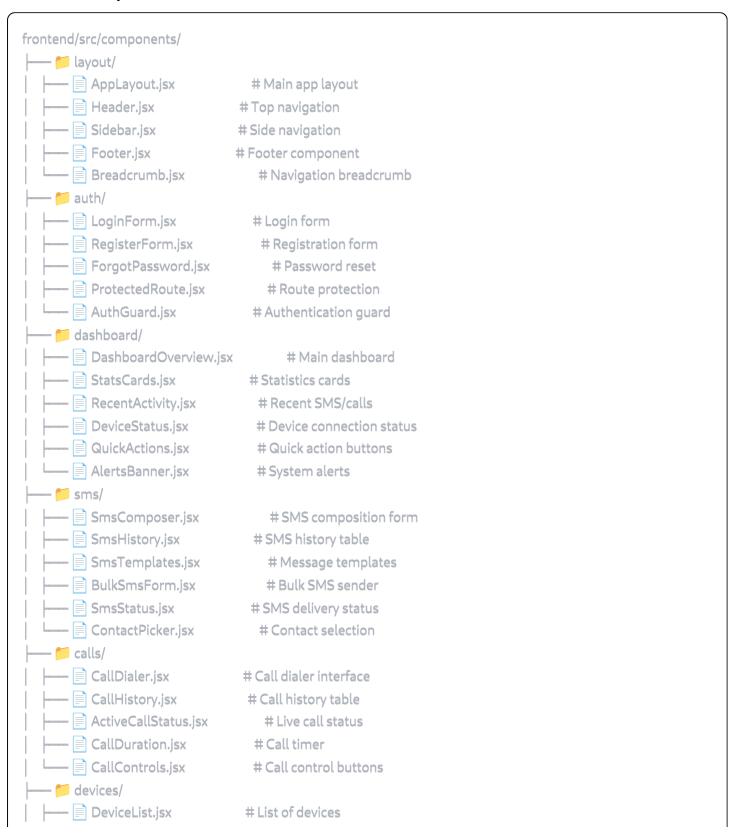
Key Android Files to Create

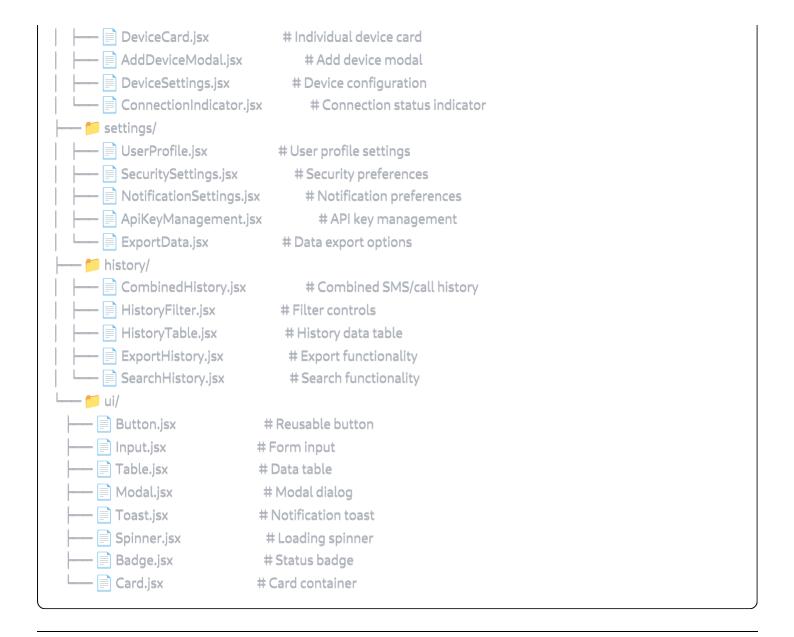
android-app/app/src/main/java/com	/simgateway/
SimGatewayApplication.kt	
MainActivity.kt	# Main activity
SettingsActivity.kt	# Settings screen
PermissionActivity.kt	# Permission requests
services/	
ForegroundService.kt	# Keep-alive service
SmsHandlerService.kt	# SMS operations
CallHandlerService.kt	# Call operations
WebSocketService.kt	# WebSocket management
network/	
WebSocketManager.kt	#WebSocket connection
RetryPolicy.kt	# Connection retry logic
HeartbeatManager.kt	# Keep connection alive
MessageQueue.kt	# Offline message queue
database/	
AppDatabase.kt	# Room database
LogDao.kt	# Local logs DAO
entities/	
LocalSmsLog.kt	# Local SMS log entity
LocalCallLog.kt	# Local call log entity



Frontend Component Hierarchy

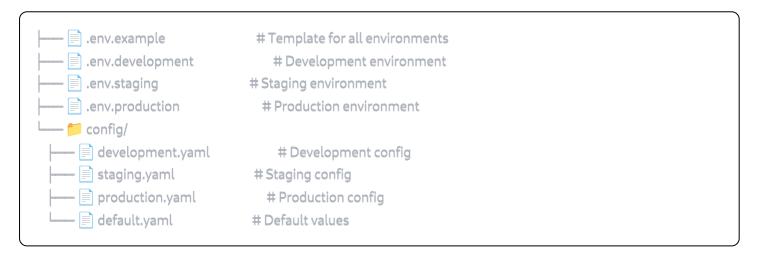
Detailed Component Structure



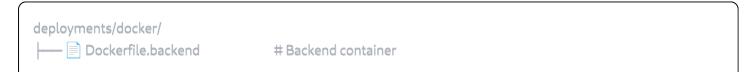


Configuration Files Structure

Environment Files



Docker Configuration



— Dockerfile.frontend	# Frontend container
— Dockerfile.nginx	# Nginx reverse proxy
docker-compose.yml	# Development environment
docker-compose.prod.ym	# Production environment
docker-compose.test.yml	# Testing environment
Configs/	
nginx.conf	# Nginx configuration
postgres.conf	# PostgreSQL configuration
redis.conf	# Redis configuration

■ Sample File Sizes (Estimated)

Component	Files	Lines of Code	Size	
Backend	~45 files	~8,000 LOC	~2.5 MB	
Frontend	~60 files	~12,000 LOC	~15 MB (with node_modules)	
Android	~35 files	~6,000 LOC	~5 MB	
Documentation	~15 files	~3,000 lines	~500 KB	
Configuration	~25 files	~1,000 lines	~200 KB	
Total	~180 files	~30,000 LOC	~25 MB	
▼				

@ Priority File Creation Order

Phase 1: Core Backend (Days 1-2)

bash

- 1. backend/cmd/server/main.go
- 2. backend/internal/config/config.go
- 3. backend/internal/models/user.go
- 4. backend/internal/handlers/auth.go
- 5. backend/internal/websocket/hub.go

Phase 2: Android App (Days 3-4)

bash

- 1. android-app/app/src/main/AndroidManifest.xml
- 2. android-app/app/src/main/java/com/simgateway/MainActivity.kt
- 3. android-app/app/src/main/java/com/simgateway/services/SmsService.kt
- 4. android-app/app/src/main/java/com/simgateway/network/WebSocketClient.kt

Phase 3: Frontend (Days 5-6)

bash

- 1. frontend/pages/_app.js
- 2. frontend/pages/index.js
- 3. frontend/components/auth/LoginForm.jsx
- 4. frontend/components/sms/SmsForm.jsx
- 5. frontend/utils/api.js

Phase 4: Integration (Days 7-8)

bash

- 1. docker-compose.yml
- 2. Database migrations
- 3. Testing files
- 4. Documentation updates



Quick Start File Generation

Want to generate the entire folder structure quickly?

bash

Create all directories

mkdir -p remote-sim-gateway/{android-app,backend,frontend,docs,deployments,scripts}

mkdir -p remote-sim-gateway/backend/{cmd/server,internal/{api/handlers,middleware,models,services,repository,we mkdir -p remote-sim-gateway/frontend/{src/{components/{auth,dashboard,sms,calls,devices,settings,ui},pages,hook mkdir -p remote-sim-gateway/android-app/app/src/main/{java/com/simgateway/{services,network,models,utils,recei

Create basic files

touch remote-sim-gateway/{README.md,LICENSE,.gitignore,docker-compose.yml}

touch remote-sim-gateway/backend/{go.mod,.env.example,Makefile}

touch remote-sim-gateway/frontend/{package.json,.env.local.example,next.config.js}

This structure gives you:

- Professional organization
- Clear separation of concerns
- Scalable architecture
- **Easy navigation** for developers
- **Deployment ready** structure

• **V** Testing support built-in

Ready to start building? Begin with the backend server structure and work your way up!