📁 ProdigyHub Code Files Documentation

Quick Navigation: Frontend Files | Backend Files | TMF APIs | Testing Scripts

---

🎯 Project Overview

ProdigyHub is a TMF (TM Forum) compliant API Dashboard system with React frontend and Node.js backend. This document explains what **every single code file** does in the project.

Project Structure:

```
ProdigyHub/
├── ProdigyhubFrontend/Admin frontend/    → React Frontend
└── ProdigyhubFrontend/ProdigyHub-Unified/ → Node.js Backend
```

🖥️ Frontend Files

React application with TypeScript, located in `ProdigyhubFrontend/Admin frontend/`

🚀 Core Application Files

File	What It Does
`client/App.tsx`	🏠 Main React app - sets up routing, authentication, theme providers, and defines all routes (public, user, admin)
`client/main.tsx`	⚡ App entry point - renders the React app with StrictMode enabled
`client/global.css`	🎨 Global styles and Tailwind CSS imports for the entire application
`package.json`	📦 Frontend dependencies - React, TypeScript, Tailwind, UI libraries
`vite.config.ts`	⚙️ Vite build tool configuration for development and production
`tsconfig.json`	🔧 TypeScript compiler configuration and project settings
`tailwind.config.ts`	🎨 Tailwind CSS configuration with custom themes and styling

🔐 Authentication & State Management

File	What It Does
`client/contexts/AuthContext.tsx`	👤 Manages user login state, authentication, and role-based permissions throughout the app

🪝 Custom Hooks (React Logic)

File	What It Does
client/hooks/use-mobile.tsx	📱 Detects if user is on mobile device for responsive design
client/hooks/use-toast.ts	🔔 Manages popup notifications and toast messages
client/hooks/useDashboardData.ts	📊 Fetches and manages main dashboard data and metrics
client/hooks/useFirebaseDashboardData.ts	🔥 Handles Firebase-specific data operations for dashboard
client/hooks/useMongoOfferingsLogic.ts	🛍️ Business logic for managing product offerings in MongoDB
client/hooks/useMongoSpecsLogic.ts	📋 Business logic for product specifications management
client/hooks/usePolling.ts	🔄 Automatically refreshes data at regular intervals
client/hooks/useProductCatalogState.ts	🗓️ Manages product catalog state and updates
client/hooks/useSLTConfiguration.ts	📡 Handles SLT telecom-specific configuration settings
client/hooks/useWebSocket.ts	⚡ Real-time data connection for live updates

📚 Libraries & Utilities

File	What It Does
client/lib/api.ts	🌐 Main HTTP client - handles all API requests to backend
client/lib/firebase.ts	🔥 Firebase setup and authentication configuration
client/lib/firebaseSeeder.ts	🌱 Creates sample data in Firebase for development/testing
client/lib/inventoryUtils.ts	📦 Helper functions for inventory calculations and management
client/lib/logger.ts	📝 Frontend logging system for debugging and monitoring
client/lib/mock-data.ts	🎭 Fake data for development and testing when backend isn't available
client/lib/mongoDataFetcher.ts	🗄️ Specialized functions for fetching MongoDB data
client/lib/orderIdUtils.ts	🔢 Generates and manages unique order identification numbers
client/lib/utils.ts	🛠️ General utility functions used throughout the app

📄 Main Application Pages

Authentication & User Management

File	What It Does
client/pages/Login.tsx	🚪 User login page with email/password and Google Sign-in
client/pages/SignUp.tsx	✍️ New user registration form with validation
client/pages/EmailVerification.tsx	✉️ Email verification page for new accounts
client/pages/UserProfile.tsx	👤 Displays user profile information and account details
client/pages/EditProfile.tsx	🖍️ Form for users to edit their personal information
client/pages/ChangePassword.tsx	🔒 Secure password change functionality

Dashboard Pages

File	What It Does
client/pages/Index.tsx	🏠 Main admin dashboard with overview metrics and charts
client/pages/UserDashboard.tsx	👥 Dashboard for regular users (non-admin)
client/pages/EnhancedDashboard.tsx	📈 Advanced monitoring dashboard with real-time metrics
client/pages/DashboardRedirect.tsx	🔁 Smart redirect - sends users to correct dashboard based on their role

TMF API Management Dashboards

File	What It Does
client/pages/ProductCatalogDashboard.tsx	🗓️ **TMF620** - Manage product catalogs, categories, and specifications
client/pages/ProductOrderingDashboard.tsx	📋 **TMF622** - Handle customer orders and order lifecycle
client/pages/ProductInventoryDashboard.tsx	📦 **TMF637** - Track product inventory and stock levels
client/pages/ProductQualificationDashboard.tsx	✅ **TMF679** - Check service eligibility and qualifications
client/pages/EventManagementDashboard.tsx	📡 **TMF688** - Manage system events and notifications
client/pages/ProductConfigurationDashboard.tsx	⚙️ **TMF760** - Configure product options and rules

Order Management

File	What It Does
client/pages/CreateOrder.tsx	🛒 Form for creating new product orders
client/pages/OrderDetail.tsx	🔍 Detailed view of a specific order with all information
client/pages/ProductOders.tsx	📊 List and manage all product orders
client/pages/OderOverview.tsx	📋 Summary view of orders with key metrics

Customer-Facing Pages

File	What It Does
client/pages/NewCustomerOnboarding.tsx	🎯 Guided setup process for new SLT customers
client/pages/PublicOfferings.tsx	🌐 Public page showing available products (no login required)
client/pages/BroadbandDetails.tsx	🌐 Detailed information about broadband services
client/pages/LocationCheckDialog.tsx	📍 Check if services are available at customer's address

Specialized Features

File	What It Does
client/pages/CategoryDetails.tsx	🗂️ Detailed view and management of product categories
client/pages/CategorySelector.tsx	🎯 Interface for selecting product categories
client/pages/InventoryProductList.tsx	📋 List view of all products in inventory
client/pages/QualificationOverviewTab.tsx	✅ Overview of product qualification status
client/pages/ProductConfigurationOverview.tsx	⚙️ Overview of product configuration options
client/pages/EventOverview.tsx	📡 Overview of system events and notifications

Utility Pages

File	What It Does
client/pages/Settings.tsx	⚙️ Application settings and user preferences
client/pages/NotFound.tsx	❌ 404 error page for invalid URLs
client/pages/EnhancedViewDialog.tsx	🔍 Popup dialog for detailed data viewing

🧩 Reusable Components

Layout & Navigation

File	What It Does
client/components/Layout.tsx	🏗️ Main app layout with sidebar navigation and header
client/components/ProtectedRoute.tsx	🛡️ Prevents unauthorized access to pages based on user role
client/components/ErrorBoundary.tsx	🚨 Catches and displays errors gracefully without crashing the app

User Interface Components

File	What It Does
client/components/ConnectionIndicator.tsx	🌐 Shows network connection status (online/offline)
client/components/ProfileCard.tsx	👤 Displays user profile information in card format
client/components/ProfilePopup.tsx	🔍 Popup dialog showing detailed user profile
client/components/NewUserOnboardingPopup.tsx	🎯 Welcome popup for new Google sign-in users

Dashboard Tab Components

File	What It Does
client/components/OverviewTab.tsx	📊 General overview tab with key metrics
client/components/InventoryTab.tsx	📦 Inventory management interface
client/components/MessagesTab.tsx	💬 Messages and notifications interface
client/components/RequestsTab.tsx	📋 Service requests management interface
client/components/QualificationTab.tsx	✅ Product qualification interface
client/components/CustomerPackagesTab.tsx	📦 Customer packages management
client/components/CustomerCustomizeTab.tsx	🎨 Customer customization options

Enhanced Features

File	What It Does
client/components/EnhancedOfferingsTab.tsx	🚀 Advanced product offerings management
client/components/EnhancedPricesTab.tsx	💰 Advanced pricing management interface
client/components/EnhancedSpecsTab.tsx	📋 Advanced specifications management

Category Management

File	What It Does
client/components/CategoryConfig.tsx	⚙️ Configuration interface for product categories
client/components/CategoryManagementTab.tsx	🗂️ Main category management interface
client/components/HierarchicalCategorySelector.tsx	🌳 Tree-like category selection with parent/child relationships

Forms & Dialogs

File	What It Does
client/components/CreateDialogs.tsx	➕ Popup forms for creating new items (products, orders, etc.)
client/components/SpecDialogs.tsx	📋 Popup forms for managing product specifications
client/components/SettingsGroup.tsx	⚙️ Organizes settings into logical groups

Charts & Visualization

File	What It Does
client/components/charts/ChartContainer.tsx	📊 Container for displaying various chart types
client/components/charts/MetricsCard.tsx	📈 Card component showing key performance metrics
client/components/charts/ProgressChart.tsx	📊 Visual progress indicators and charts

🎨 UI Component Library (shadcn/ui)

Pre-built, customizable UI components

Basic Components

File	What It Does
client/components/ui/button.tsx	⭕ Button component with different styles and sizes
client/components/ui/input.tsx	⌨️ Text input field component
client/components/ui/label.tsx	🏷️ Form label component
client/components/ui/card.tsx	📄 Card container for grouping content
client/components/ui/badge.tsx	🏷️ Small badge/tag component for status indicators

Navigation Components

File	What It Does
client/components/ui/breadcrumb.tsx	🗺️ Navigation breadcrumb trail
client/components/ui/navigation-menu.tsx	📍 Main navigation menu component
client/components/ui/menubar.tsx	📋 Menu bar for application menus
client/components/ui/sidebar.tsx	📱 Collapsible sidebar navigation

Data Display Components

File	What It Does
client/components/ui/table.tsx	📊 Data table for displaying structured information
client/components/ui/chart.tsx	📈 Chart wrapper for data visualization
client/components/ui/progress.tsx	📊 Progress bar component
client/components/ui/skeleton.tsx	⏳ Loading placeholder component

Form Components

File	What It Does
client/components/ui/form.tsx	📝 Form wrapper with validation
client/components/ui/checkbox.tsx	☑️ Checkbox input component
client/components/ui/radio-group.tsx	⭕ Radio button group component
client/components/ui/select.tsx	📋 Dropdown select component
client/components/ui/textarea.tsx	📝 Multi-line text input
client/components/ui/slider.tsx	🎚️ Range slider component
client/components/ui/switch.tsx	🔄 Toggle switch component

Dialog & Popup Components

File	What It Does
client/components/ui/dialog.tsx	🔲 Modal dialog component
client/components/ui/alert-dialog.tsx	⚠️ Confirmation dialog for important actions
client/components/ui/sheet.tsx	📄 Slide-out panel component
client/components/ui/drawer.tsx	🗂️ Bottom drawer component
client/components/ui/popover.tsx	💭 Small popup tooltip component

Feedback Components

File	What It Does
client/components/ui/toast.tsx	📦 Toast notification component
client/components/ui/alert.tsx	⚠️ Alert message component
client/components/ui/tooltip.tsx	💡 Hover tooltip component

🖥️ Backend Files

Node.js/Express server located in `ProdigyhubFrontend/ProdigyHub-Unified/`

🚀 Main Server Files

File	What It Does
`server.js`	🏗️ Main Express server - sets up all APIs, middleware, database connections, and starts the server
`package.json`	📦 Backend dependencies - Express, MongoDB, authentication libraries, email services

⚙️ Configuration Files

File	What It Does
`src/config/database.js`	🗄️ MongoDB connection setup and database configuration
`src/config/cors.js`	🌐 Cross-origin resource sharing settings for frontend-backend communication
`src/config/email.js`	✉️ Email service configuration for sending notifications and verification emails
`src/config/environment.js`	🌍 Environment variables management (API keys, database URLs, etc.)

🎛️ Controllers (Business Logic)

File	What It Does
`src/controllers/UnifiedControllers.js`	🔄 Central controller combining all TMF API business logic
`controllers/TMF760Controller.js`	⚙️ Handles TMF760 Product Configuration API operations

🗄️ Database Models

File	What It Does
`src/models/AllTMFModels.js`	📊 MongoDB schema definitions for all TMF entities (products, orders, inventory, etc.)
`src/models/CheckProductConfiguration.js`	✅ Data model for product configuration validation
`src/models/QueryProductConfiguration.js`	🔍 Data model for product configuration queries

🏞️ API Routes

File	What It Does
`routes/tmf760Routes.js`	🏞️ URL routes for TMF760 Product Configuration API endpoints

🔧 Shared Utilities & Middleware

File	What It Does
`src/shared/constants/httpCodes.js`	🔢 HTTP status code constants (200, 404, 500, etc.)
`src/shared/constants/tmfTypes.js`	📋 TMF-specific data type definitions
`src/shared/middleware/auth.js`	🔐 Authentication middleware - verifies user tokens
`src/shared/middleware/cors.js`	🌐 CORS middleware configuration
`src/shared/middleware/errorHandler.js`	🚨 Global error handling and logging
`src/shared/middleware/rateLimit.js`	🚦 Prevents API abuse by limiting requests per user
`src/shared/utils/helpers.js`	🛠️ General utility functions used across the backend
`src/shared/utils/logger.js`	📝 Backend logging system for debugging and monitoring
`src/shared/utils/validation.js`	✅ Data validation utilities to ensure data integrity

🧬 TMF API Implementations

Industry-standard telecom APIs (TM Forum specifications)

📋 TMF620 - Product Catalog Management

Managing products, categories, specifications, and pricing

File	What It Does
`src/api/tmf620/routes/categories.js`	🗂️ API routes for product category management (create, read, update, delete categories)
`src/api/tmf620/routes/productCatalogs.js`	🏪 API routes for product catalog operations
`src/api/tmf620/routes/productOfferings.js`	🛍️ API routes for managing product offerings (services available to customers)
`src/api/tmf620/routes/productOfferingPrices.js`	💰 API routes for product pricing management
`src/api/tmf620/routes/productSpecifications.js`	📋 API routes for product specifications and technical details
`src/api/tmf620/routes/exportJobs.js`	📤 API routes for exporting catalog data
`src/api/tmf620/routes/importJobs.js`	📥 API routes for importing catalog data
`src/api/tmf620/routes/hub.js`	🔄 API routes for TMF620 event notifications
`src/api/tmf620/middleware/errorHandler.js`	🚨 Error handling specific to TMF620 operations

🛒 TMF622 - Product Ordering Management

Handling customer orders and order lifecycle

File	What It Does
`src/api/tmf622/controllers/productOrderController.js`	📋 Business logic for creating, updating, and managing product orders
`src/api/tmf622/controllers/cancelProductOrderController.js`	❌ Business logic for handling order cancellations
`src/api/tmf622/models/ProductOrder.js`	📊 Database model for product orders
`src/api/tmf622/models/CancelProductOrder.js`	❌ Database model for order cancellations
`src/api/tmf622/routes/productOrderRoutes.js`	🛤️ API routes for all order-related operations
`src/api/tmf622/validation/schemas.js`	✅ Validation rules for order data
`src/api/tmf622/middleware/index.js`	🔧 Middleware stack for TMF622 operations
`src/api/tmf622/scripts/sampleData.js`	🎭 Sample order data for testing
`src/api/tmf622/scripts/testAPI.js`	🧪 API testing scripts

📦 TMF637 - Product Inventory Management

Tracking stock levels and inventory

File	What It Does
`src/api/tmf637/controllers/ProductInventoryController.js`	📋 Business logic for inventory management, stock tracking, and allocation
`src/api/tmf637/utils/helpers.js`	🛠️ Utility functions specific to inventory operations

✅ TMF679 - Product Offering Qualification

Checking service eligibility and availability

File	What It Does
`src/api/tmf679/controllers/queryProductOfferingQualificationController.js`	🔍 Handles qualification queries and eligibility checks
`src/api/tmf679/controllers/checkProductOfferingQualificationMongoController.js`	✅ MongoDB-based qualification validation
`src/api/tmf679/controllers/sltQualificationController.js`	📡 SLT telecom-specific qualification logic
`src/api/tmf679/controllers/addressSyncController.js`	📍 Syncs address data with external systems
`src/api/tmf679/controllers/areaManagementController.js`	🗺️ Manages service coverage areas
`src/api/tmf679/routes/queryProductOfferingQualification.js`	🚦 API routes for qualification queries
`src/api/tmf679/routes/checkProductOfferingQualification.js`	🚦 API routes for qualification checks
`src/api/tmf679/routes/addressSyncRoutes.js`	🚦 API routes for address synchronization
`src/api/tmf679/routes/areaManagementRoutes.js`	🚦 API routes for service area management
`src/api/tmf679/controllers/sltQualificationRoutes.js`	🚦 API routes for SLT-specific qualification
`src/api/tmf679/utils/addressSyncUtils.js`	🏠 Utilities for address data synchronization
`src/api/tmf679/utils/helpers.js`	🛠️ Helper functions for qualification operations

📡 TMF688 - Event Management

System events, notifications, and messaging

File	What It Does
`src/api/tmf688/controllers/eventController.js`	🎯 Manages system events and event lifecycle
`src/api/tmf688/controllers/hubController.js`	🔄 Manages event hubs for publishing/subscribing to events
`src/api/tmf688/controllers/topicController.js`	📢 Manages event topics and categories
`src/api/tmf688/models/Event.js`	📊 Database model for system events
`src/api/tmf688/models/Hub.js`	📊 Database model for event hubs
`src/api/tmf688/models/Topic.js`	📊 Database model for event topics
`src/api/tmf688/routes/events.js`	🛣️ API routes for event operations
`src/api/tmf688/routes/hubs.js`	🛣️ API routes for hub management
`src/api/tmf688/routes/topics.js`	🛣️ API routes for topic management
`src/api/tmf688/middleware/auth.js`	🔐 Authentication for event operations
`src/api/tmf688/middleware/errorHandler.js`	🚨 Error handling for event operations
`src/api/tmf688/middleware/validation.js`	✅ Validation for event data
`src/api/tmf688/utils/helpers.js`	🛠️ Utility functions for event management

⚙️ TMF760 - Product Configuration Management

Product customization and configuration rules

File	What It Does
`src/api/tmf760/app.js`	🏗️ Express app setup specifically for TMF760
`src/api/tmf760/controllers/TMF760Controller.js`	⚙️ Business logic for product configuration management
`src/api/tmf760/models/CheckProductConfiguration.js`	✅ Database model for configuration validation
`src/api/tmf760/models/QueryProductConfiguration.js`	🔍 Database model for configuration queries
`src/api/tmf760/models/index.js`	📦 Exports all TMF760 models
`src/api/tmf760/routes/index.js`	📦 Exports all TMF760 routes
`src/api/tmf760/routes/checkProductConfiguration.js`	🛤️ API routes for configuration validation
`src/api/tmf760/routes/queryProductConfiguration.js`	🛤️ API routes for configuration queries
`src/api/tmf760/routes/hub.js`	🛤️ API routes for TMF760 event hub
`src/api/tmf760/routes/tmf760Routes.js`	🛤️ Main TMF760 API routes
`src/api/tmf760/services/eventService.js`	📡 Event handling services for configuration changes
`src/api/tmf760/middleware/errorHandler.js`	🚨 Error handling for configuration operations
`src/api/tmf760/middleware/validation.js`	✅ Validation for configuration data
`src/api/tmf760/utils/responseHelpers.js`	🧰 Utilities for formatting API responses

👥 User Management API

File	What It Does
`src/api/users/controllers/userController.js`	👤 User account operations - registration, login, profile updates, authentication
`src/api/users/routes/userRoutes.js`	🛤️ API routes for user management operations

🧪 Testing & Development Scripts

Scripts for testing, debugging, and development

🔧 Basic Testing Scripts

File	What It Does
`simple-test.js`	🧪 Basic functionality tests for core API operations
`mongodb-test.js`	🗄️ Tests MongoDB connection and basic database operations
`test-password-hashing.js`	🔒 Tests password encryption and security functions

File	What It Does
`test-email-verification.js`	✉️ Tests email sending and verification system
`test-api-endpoint.js`	🎯 Tests individual API endpoints
`test-api-endpoints.js`	📋 Comprehensive test suite for all API endpoints
`test-vercel-endpoint.js`	☁️ Tests API endpoints in Vercel deployment environment

🏠 Address Synchronization Testing

File	What It Does
`test-address-sync.js`	📍 Tests address data synchronization between systems
`test-address-sync-api.js`	🌐 Tests address sync API endpoints
`test-address-sync-complete.js`	📋 Complete address sync workflow testing
`test-address-sync-frontend.html`	🖥️ Frontend testing page for address sync features
`test-address-sync-local.js`	🏠 Local development address sync testing
`test-new-address-sync.js`	🆕 Tests new address sync implementation
`test-street-address-sync.js`	🛣️ Tests street-level address synchronization
`debug-address-sync.js`	🔍 Debugging tools for address sync issues
`debug-address-sync-process.js`	⚙️ Debug the address sync process step by step
`debug-address-sync-specific.js`	🎯 Debug specific address sync problems

👤 User Management Testing

File	What It Does
`test-profile-update.js`	📝 Tests user profile update functionality
`test-user-name-fix.js`	🔧 Tests user name correction features
`debug-users.js`	🔍 Debug user account issues
`create-user-in-deployed-backend.js`	☁️ Create test users in production environment
`create-user-vercel.js`	☁️ Create test users in Vercel deployment environment
`manual-update-user.js`	✏️ Manually update user data in database

✅ Qualification System Testing

File	What It Does
`test-qualification-with-address.js`	📍 Tests service qualification based on customer address
`test-recent-qualification-sync.js`	🔄 Tests synchronization of recent qualification data

File	What It Does
`debug-current-qualification.js`	🔍 Debug current qualification system issues
`sync-existing-qualifications.js`	📊 Sync existing qualification records between systems

🗄️ Database Management Scripts

File	What It Does
`check-collection.js`	🔍 Verify database collections and their structure
`check-users-in-mongo.js`	👥 Check user records in MongoDB database
`fix-collection.js`	🔧 Repair corrupted database collections
`manual-address-update.js`	📍 Manually update address records in database
`manual-address-update-specific.js`	🎯 Update specific address records manually
`manual-sync-current-user.js`	👤 Manually sync current user data
`manual-sync-recent-user.js`	🆕 Manually sync recently created user data
`update-user-address-direct.js`	📍 Directly update user address in database
`trigger-address-sync.js`	🔄 Manually trigger address synchronization process

📚 Documentation & Configuration Files

📖 Setup & Configuration Guides

File	What It Does
`PROJECT_OVERVIEW.md`	📋 Complete project architecture and feature documentation
`FIREBASE_SETUP_GUIDE.md`	🔥 Step-by-step Firebase configuration instructions
`FIREBASE_AUTH_FIX.md`	🔧 Troubleshooting guide for Firebase authentication issues
`GOOGLE_SIGNIN_SETUP.md`	🔑 Google Sign-In integration setup instructions
`EMAIL_VERIFICATION_FIX.md`	✉️ Fix email verification system problems
`PASSWORD_SECURITY_README.md`	🔒 Password security implementation guide

🎯 Feature-Specific Documentation

File	What It Does
`NEW_CUSTOMER_ONBOARDING_README.md`	👋 Documentation for customer onboarding system
`NEW_USER_ONBOARDING_README.md`	🎯 User onboarding flow documentation
`CATEGORY_MANAGEMENT_README.md`	🗂️ Product category management system guide

File	What It Does
SIGNUP_README.md	✍️ User registration process documentation
ADDRESS_SYNC_IMPLEMENTATION.md	📍 Address synchronization system implementation guide

🚀 Deployment & Build Configuration

File	What It Does
scripts/build.sh	🏗️ Production build script for deployment
scripts/start.sh	▶️ Application startup script
scripts/test.sh	🧪 Test execution script
vercel.json	☁️ Vercel cloud deployment configuration
firebase.json	🔥 Firebase project configuration
database.rules.json	🛡️ Firebase database security rules
netlify.toml	🌐 Netlify deployment configuration

🔧 Development Configuration

File	What It Does
vite.config.server.ts	⚙️ Vite configuration for server-side rendering
postcss.config.js	🎨 PostCSS configuration for CSS processing
components.json	🧩 shadcn/ui component library configuration
vite-env.d.ts	🔧 TypeScript definitions for Vite environment

🌐 API Integration Files

File	What It Does
client/pages/api-service.ts	🔗 General API service functions for frontend
client/pages/sltApiClient.ts	📡 SLT telecom-specific API client
client/pages/sltQualificationApi.ts	✅ SLT qualification API service functions

🌐 Server & API Files

File	What It Does
api/index.ts	🎯 Main API endpoint definitions
server/index.ts	🖥️ Express server configuration and startup
server/node-build.ts	🔧 Node.js build configuration

File	What It Does
`server/routes/`	🚏 Server-side route definitions
`shared/api.ts`	🔗 Shared API utilities between client and server
`shared/product-order-types.ts`	📋 Shared TypeScript types for product orders

☁️ Serverless Functions

File	What It Does
`netlify/functions/api.ts`	⚡ Serverless function for Netlify deployment

🧪 Testing Utilities

File	What It Does
`client/lib/utils.spec.ts`	🧪 Unit tests for frontend utility functions
`client/components/types/SLTTypes.ts`	📋 TypeScript type definitions for SLT-specific data

🎯 Quick File Finder

🔍 Looking for Authentication?

- **Login/Signup:** `client/pages/Login.tsx`, `client/pages/SignUp.tsx`
- **Auth Logic:** `client/contexts/AuthContext.tsx`
- **User API:** `src/api/users/controllers/userController.js`
- **Firebase Setup:** `client/lib/firebase.ts`

🔍 Looking for Dashboard Pages?

- **Admin Dashboard:** `client/pages/Index.tsx`
- **User Dashboard:** `client/pages/UserDashboard.tsx`
- **TMF Dashboards:** `client/pages/Product*Dashboard.tsx` files

🔍 Looking for API Logic?

- **TMF APIs:** `src/api/tmf*/` folders
- **Controllers:** `src/api/*/controllers/` folders
- **Routes:** `src/api/*/routes/` folders
- **Models:** `src/api/*/models/` or `src/models/` folders

🔍 Looking for UI Components?

- **Custom Components:** `client/components/*.tsx`
- **UI Library:** `client/components/ui/*.tsx`
- **Layout:** `client/components/Layout.tsx`

🔍 Looking for Configuration?

- **Frontend Config:** `vite.config.ts`, `tailwind.config.ts`
- **Backend Config:** `src/config/*.js` files
- **Deployment:** `vercel.json`, `netlify.toml`, `firebase.json`

🔍 Looking for Testing?

- **Frontend Tests:** `*.spec.ts` files
- **Backend Tests:** `test-*.js`, `debug-*.js` files
- **API Tests:** `src/api/*/scripts/` folders

🎪 Key Technology Summary

🖥️ Frontend Stack

- **React 18** + **TypeScript** - Modern, type-safe user interface
- **Vite** - Lightning-fast development and build tool
- **Tailwind CSS** - Utility-first styling framework
- **shadcn/ui** - Beautiful, accessible component library
- **Firebase Auth** - Secure authentication system
- **TanStack Query** - Powerful data fetching and caching

🗄️ Backend Stack

- **Node.js** + **Express.js** - Fast, scalable server
- **MongoDB** + **Mongoose** - Flexible document database
- **JWT** - Secure token-based authentication
- **bcryptjs** - Secure password hashing
- **Winston** - Comprehensive logging system

🌐 Deployment Options

- **Vercel** - Serverless deployment for full-stack apps
- **Netlify** - Static site hosting with serverless functions
- **Firebase Hosting** - Google's web hosting platform

📡 TMF API Compliance

Full implementation of 6 TM Forum Open API specifications for telecom industry standards.

This documentation covers every single code file in the ProdigyHub project. Use the Quick File Finder section to navigate to specific functionality you're looking for!