# Manilogix Auth-Service Documentation (Updated)

## 1. Module Overview

The **Auth-Service** module is responsible for managing authentication and authorization for the **Manilogix** platform. It includes the following responsibilities:

* User registration (Signup)
* User login with JWT-based authentication
* Role-based access control (RBAC)
* Token generation and validation
* Secure endpoints using Spring Security

### Technologies Used:

* Java 17
* Spring Boot
* Spring Security
* Spring Data JPA
* MySQL
* JWT (JJWT library)

## 2. Project Structure

auth-service/  
├── src/main/java/com/manilogix/auth/  
│ ├── controller/  
│ │ ├── AuthController.java  
│ │ └── UserController.java  
│ ├── service/  
│ │ ├── AuthService.java  
│ │ ├── JwtService.java  
│ ├── config/  
│ │ ├── SecurityConfig.java  
│ │ └── JwtAuthenticationFilter.java  
│ ├── security/  
│ │ ├── CustomUserDetails.java  
│ │ └── UserDetailsServiceImpl.java  
│ ├── dto/  
│ │ ├── LoginRequest.java  
│ │ ├── SignupRequest.java  
│ │ └── JwtResponse.java  
│ ├── model/  
│ │ ├── User.java  
│ │ ├── Role.java  
│ │ └── RoleType.java  
│ ├── repository/  
│ │ ├── UserRepository.java  
│ │ └── RoleRepository.java  
│ ├── config/DataInitializer.java  
│ └── AuthServiceApplication.java  
└── resources/  
 └── application.properties

## 3. Controllers and Responsibilities

### AuthController.java

**Endpoint:** /api/auth

**Responsibilities:**

* POST /signup: Register a new user
* POST /login: Login user and issue JWT token

### UserController.java

**Endpoint:** /api/user

**Responsibilities:**

* GET /me: Return currently authenticated user’s info

## 4. DTOs (Data Transfer Objects)

### SignupRequest.java

* String firstName
* String lastName
* String email
* String password
* String role

### LoginRequest.java

* String email
* String password

### JwtResponse.java

* String token
* String username
* List roles

## 5. Models

### User.java

* Long id
* String firstName
* String lastName
* String email (unique)
* String password
* Set roles

### Role.java

* Long id
* RoleType name (ENUM)

### RoleType.java (Enum)

public enum RoleType {  
 ADMIN,  
 MANAGER,  
 SUPPLIER  
}

## 6. Services

### AuthService.java

* Handles user registration
* Encodes passwords
* Validates credentials
* Assigns roles on signup

### JwtService.java

* Generates JWT tokens with expiration
* Validates and parses JWT tokens
* Extracts username/email from token

## 7. Security Configuration

### SecurityConfig.java

* Disables CSRF for stateless API
* Configures route access (/auth/\*\* permitted)
* Registers JWT filter
* Defines PasswordEncoder and AuthenticationManager

### JwtAuthenticationFilter.java

* Intercepts all requests
* Extracts JWT from header
* Validates token
* Loads user details
* Sets Spring Security context

### CustomUserDetails.java

* Custom implementation of UserDetails
* Wraps User entity
* Provides authorities (roles) for RBAC

### UserDetailsServiceImpl.java

* Loads User entity from DB by email
* Used by Spring Security during login validation

## 8. Initialization

### DataInitializer.java

* Loads default roles from RoleType enum
* Runs once at application startup
* Inserts missing roles into DB

## 9. Repositories

### UserRepository.java

* findByEmail(String email)
* existsByEmail(String email)

### RoleRepository.java

* findByName(RoleType name)

## 10. Application Properties

server.port=8081  
spring.datasource.url=jdbc:mysql://localhost:3306/authdb  
spring.datasource.username=root  
spring.datasource.password=yourpassword  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
  
jwt.secret=your-secret-key  
jwt.expirationMs=86400000

## 11. API Endpoints Summary

| Method | Endpoint | Description |
| --- | --- | --- |
| POST | /api/auth/signup | Register new user |
| POST | /api/auth/login | Login and return JWT |
| GET | /api/user/me | Get current user info |

## 12. Dependencies

* Spring Web
* Spring Security
* Spring Data JPA
* MySQL Connector
* JJWT (Java JWT)
* Jakarta Validation
* Spring Boot DevTools
* JAXB API (for JWT on Java 11+)

## 13. Notes

* Passwords are stored securely using BCrypt
* JWT tokens are stateless and include user roles
* Service is modular and compatible with microservices architecture
* Can be extended with password reset, verification, or OAuth login

**End of Documentation**