ACME Medical RESTful API System

CST8277 Java Enterprise Project – Summer 2025  
Developed using Jakarta EE, Payara Server, JPA, and JAX-RS.

**Group Members and Task Distribution**

| **Member** | **Contributions** | **Average Peer Grade (Grade Provided by Group #27)** |
| --- | --- | --- |
| Ruchen Ding | T1, T2, T3 and T9 | 100 % |
| Ryan Xu | T4, T5, T6 and T9 | 100 % |
| Yizhen Xu | T7, T8 and T9 | 100% |

* T1: Finish the JPA Annotations for Entities
* T2: Finish Custom Authentication Mechanism
* T3: Relationship Between SecurityUser and Physician
* T4: Building REST API
* T5: Securing REST Endpoints
* T6: POSTMAN Based API Validation
* T7: Building JUnit Tests
* T8: Documentation
* T9: Evaluation

# 1. Overview

This project models the medical training and certification system of ACME Medical Corp.  
It exposes RESTful APIs for managing Physicians, Patients, Medical Certificates, Prescriptions, Medicines, and more.

# 2. Technologies Used

* Java 21.0.1 2023-10-17 LTS
* Jakarta EE 10
* Payara Server 6.2024.4
* JPA / Hibernate
* MySQL 8.0.42
* Postman for Windows 11.55.1
* Eclipse IDE (with Maven support) 2025-03 (4.35.0)
* JUnit 5 + Maven Surefire

# 3. Setup & Deployment Instructions

## 3.1. Prerequisites

* Java JDK 17+
* Payara Server 6.2024+
* MySQL Server (or local MySQL Workbench)
* Eclipse IDE with Maven plugin
* Postman

## 3.2. Database Setup

The database schema undergoes a "drop-and-create" operation each time the application is deployed or restarted on Payara Server.

Required in persistence.xml by this property:

The three SQL were provided in the project skeleton; don't modify them: acmemedical-create.sql acmemedical-data.sql acmemedical-drop.sql

## 3.3. Payara Configuration

1. Run "localhost:4848" in browser
2. JDBC -> JDBC Connection Pools, create a new pool named "acmemedicalPool"
   * Resource Type: javax.sql.DataSource
   * Datasource Classname: com.mysql.cj.jdbc.MysqlDataSource
   * Additional Properties:
     + password: 8277
     + databaseName: acmemedical
     + serverName: 127.0.0.1
     + user: cst8277
     + networkProtocal: tcp
     + portNumber:3306
     + url: jdbc:mysql://localhost:3306/acmemedical?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC
3. Save
4. Ping test

## 3.4. Security Roles

| **Role** | **Username** | **Password** |
| --- | --- | --- |
| ADMIN\_ROLE | admin | admin |
| USER\_ROLE | cst8277 | 8277 |

## 3.5. REST Endpoints

Base URL: <http://localhost:8080/rest-acmemedical/api/v1/>

# 4. Testing with Postman

## 4.1 Steps

1. Import the REST-ACMEMedical-Sample.postman\_collection.json file in POSTMAN.
2. Basic Auth can be changed during testing
   * Admin: admin:admin
   * User: cst8277:8277
3. Ensure correct port and endpoint paths

## 4.2 Special Notes

1. **MedicalTraining Entity**: embedded "DurationAndStatus"
2. **POST of Prescription**: The Prescription entity uses a **composite primary key** (Physician + Patient). By default, the database contains:

* Physician ID = 1
* Patient IDs = 1 and 2
* Prescriptions for combinations (1,1) and (1,2) Any attempt to create duplicate combinations will result in a primary key violation.

To ensure POST of Prescription testing succeeds, our tests first create a new Physician (ID = 2),  
then use the combination Physician = 2, Patient = 1 for the new Prescription. Please run the requests **in order**, starting with the POST Physician request.

1. **POST of MedicalCertificate**: The MedicalCertificate POST API implements business validation to prevent duplicate certificates for the same physician-training combination. Due to existing seed data, POST requests typically return 409 Conflict with a JSON error message, which is the expected and correct behavior. This demonstrates proper data integrity protection and structured error handling.

# 5. JUnit Test

## 5.1 Test Suite Overview

Total Tests: 91 tests across 4 test classes

• EntityTests.java: 24 tests - Entity validation and JPA relationships

• ServiceTest.java: 10 tests - Business logic and database operations

• ResourceTest.java: 56 tests - REST API endpoints and security

• TestACMEMedicalSystem.java: 1 test - System integration

## 5.2 Test Coverage

**Entity Layer (24 tests)**

• Entity creation and properties

• Inheritance (PublicSchool, PrivateSchool)

• Composite keys (PrescriptionPK)

• Bidirectional relationships

• Equals/hashCode validation

**Service Layer (10 tests)**

• Direct EntityManager operations

• CRUD functionality

• Database transactions

• Business logic validation

**REST API Layer (56 tests)**

• HTTP methods (GET, POST, PUT, DELETE)

• Security roles (ADMIN, USER)

• Error handling (400, 401, 403, 404, 409, 415)

• Authentication and authorization

**Integration (1 test)**

• End-to-end system validation

**Security Testing**

Test Users:

• Admin: admin / admin (full access)

• User: cst8277 / 8277 (limited access)

**Security Scenarios:**

• Admin role: Full CRUD operations

• User role: Restricted access to owned resources

• Unauthorized access: 401 errors

• Forbidden access: 403 errors

## 5.3 Test Execution

**Prerequisites**

1. Payara Server running on localhost:8080

2. Database with ACME Medical schema

3. Application deployed and accessible

**Run Tests**

Maven command:

mvn clean install test surefire-report:report site -DgenerateReports=true

**Report location:**

target/site/surefire-report.html

**Execution Order**

1. EntityTests (independent)

2. ServiceTest (creates data)

3. ResourceTest (uses created data)

4. TestACMEMedicalSystem (final validation)

**HTTP Status Codes Validated**

• 200 (OK), 201 (Created), 400 (Bad Request)

• 401 (Unauthorized), 403 (Forbidden), 404 (Not Found)

• 405 (Method Not Allowed), 409 (Conflict), 415 (Unsupported Media Type)

## 5.4 Key Technologies

• JUnit 5: Testing framework

• Hamcrest: Assertions

• JAX-RS Client: REST testing

• JPA/EntityManager: Database testing

• Jersey Client: HTTP requests

Comprehensive test suite ensuring security, functionality, and reliability of the ACME Medical REST API system.