# FPT University – PRN232 Practice Exam

**Course:** PRN232 - Building Web API with ASP.NET Core

**Duration:** 85 minutes

## 1. General Information

* Environment: Visual Studio 2019+ & SQL Server 2012+
* Architecture: 3-layer (Presentation, Business Logic, Data Access)
* Database: summer2025handbagdb
* Entity Framework Core: Configure DbContext via appsettings.json (hardcoded connection strings will receive 0 points)
* Solution Structure:
  + PRN231\_SU25\_SE<YourStudentID>.api – API Project
  + PRN231\_SU25\_SE<YourStudentID>.client – Frontend (optional, focus on login page)
* Notes:
  + All data operations must go through the Web API
  + No direct DB access from frontend
  + No hardcoded SQL allowed

## 2. Implementation Tasks

## 2.1. Authentication & Authorization (1.0 point)

* Implement login using JWT (email + password).
* Allowed Roles:
  + administrator, moderator: Full access (CRUD + search)
  + developer, member: Read + search only
  + Other roles: No token issued

Endpoint:  
POST /api/auth  
Request:

json

{

"email": "admin@store.com",

"password": "123456"

}

Response:

json

{

"token": "<JWT token>",

"role": "administrator"

}

## 2.2. Handbag API Endpoints (3.0 points)

* GET /api/handbags
  + List all handbags with brand info
  + Roles: administrator, moderator, developer, member
  + Status: 200, 401, 403
* GET /api/handbags/{id}
  + Get handbag by ID
  + Roles: administrator, moderator, developer, member
  + Status: 200, 404, 401, 403
* POST /api/handbags
  + Create a new handbag
  + Roles: administrator, moderator
  + Body:
* json

{

"modelName": "Elegant #2024",

"material": "Leather",

"price": 250.5,

"stock": 10,

"brandId": 1

}

* + Validation:
    - modelName: Regex ^([A-Z0-9][a-zA-Z0-9#]\*\s)\*([A-Z0-9][a-zA-Z0-9#]\*)$
    - price, stock > 0
  + Status: 201, 400, 401, 403
* PUT /api/handbags/{id}
  + Update an existing handbag
  + Roles: administrator, moderator
  + Status: 200, 400, 404, 401, 403
* DELETE /api/handbags/{id}
  + Delete a handbag
  + Roles: administrator, moderator
  + Status: 200, 404, 401, 403
* GET /api/handbags/search?modelName=...&material=...
  + Search handbags by modelName and material (OData supported)
  + Results grouped by brand name
  + Roles: all roles with token
  + Status: 200, 401, 403

## 2.3. Error Code Format (1.0 point)

Always return JSON error messages in the following format:

json

{

"errorCode": "HB40001",

"message": "modelName is required"

}

| **errorCode** | **HTTP Status** | **Meaning** |
| --- | --- | --- |
| HB40001 | 400 | Missing/invalid input |
| HB40101 | 401 | Token missing/invalid |
| HB40301 | 403 | Permission denied |
| HB40401 | 404 | Resource not found |
| HB50001 | 500 | Internal server error |

## 2.4. Swagger (1.0 point)

* Must expose all endpoints listed above
* Must allow JWT token insertion for testing

## 2.5. Postman Test Cases (2.5 points)

Write and execute at least 6 test cases using Postman:

1. Login success
2. Login failed
3. Create handbag (authorized roles only)
4. Update handbag
5. Delete handbag
6. Get list / Get by ID

Requirements:

* Include JWT in Header
* Use correct method/body
* Validate response status & message

## 3. Submission Notes

* Syntax errors, uncompiled code = 0 points
* Any hardcoded/irrelevant code = 0 points
* Naming must strictly follow instruction format

Good luck!