FPT University – PRN232 Practice Exam

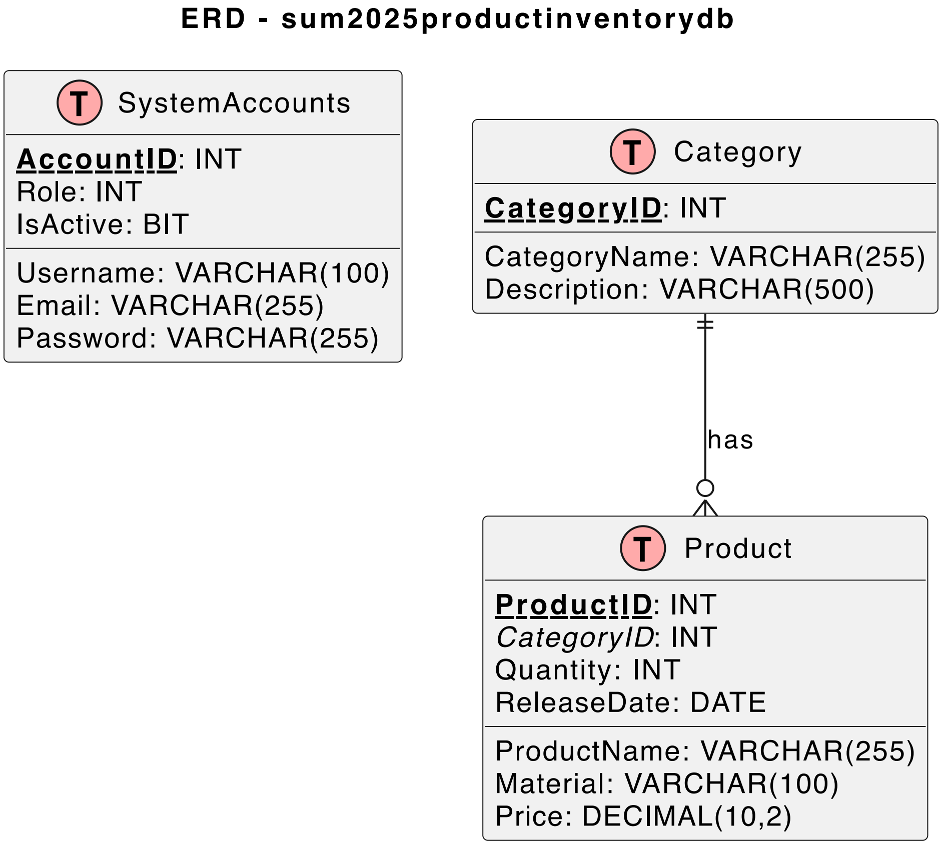
PE_PRN232_SU25_250733

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: `summ2025productinventorydb`
- Entity Framework Core: Configure DbContext via appsettings.json (hardcoded connection strings will receive 0 points)
- Solution Structure:
  - PE_PRN232_SU25_ `<YourStudentID>` _api – API Project
- **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 (1.5 point)

- Implement login using JWT (email + password).
- **Allowed Roles:**
    - `admin, manager` : Full access (CRUD + search)
    - `analyst` : Read + search only
    - `Others` roles: No token issued

Endpoint:

```
POST /api/auth
```

Request:

```
{
  "email": "manager@system.com",
  "password": "123456"
}
```

Response:

```
{
  "token": "<JWT token>",
  "role": "admin"
}
```

2.2. Product API Endpoints (4.0 points)

GET `/api/products`

- List all products with category info
- Roles: All logged-in roles
- Status Codes: `200, 401, 403`

GET `/api/products/{id}`

- Get product by ID
- Status Codes: `200, 404, 401, 403`

POST `/api/products`

- Add a new product
- Roles: `admin, manager only`
- Validation:
  - `productName` : must match `^[A-Z][a-zA-Z0-9\s]{2,50}$`
  - price > 0, quantity ≥ 0
- Status Codes: `201, 400, 401, 403`
- **Request Body:**

```
{
  "productName": "SmartWatch Z3",
  "price": 149.99,
  "quantity": 30,
  "categoryId": 2
}
```

PUT `/api/products/{id}`

- Update a product
- Roles: admin, manager
- Status Codes: `200, 400, 404, 401, 403`

```
{
  "productName": "SmartWatch Z3",
  "price": 149.99,
  "quantity": 30,
  "categoryId": 2
}
```

DELETE `/api/products/{id}`

- Remove a product
- Roles: admin only
- Status Codes: `200, 404, 401, 403`
- **Request Body:**

GET `/api/products/search?name=...&category=...`

- Filter products by name and category
- Results grouped by category
- Roles: All with valid token
- Status Codes: `200, 401, 403`

2.3. Error Code Format (1.0 point)

Use the JSON error response format below:

```
{
  "errorCode": "PR40001",
  "message": "Product name is required"
}
```

Error Code	HTTP Status	Meaning
PR40001	400	Invalid input
PR40101	401	Authentication error
PR40301	403	Not authorized
PR40401	404	Resource not found
PR50001	500	Server error

2.4. Swagger Integration (1.0 point)

- Document all API endpoints
- Allow token-based authentication testing
- Display schema validation rules

2.5. Postman Test Suite (2.5 points)

Create and run at least 6 test cases:

- Login success
- Login failure
- Add product (admin/manager)
- Update product
- Delete product
- Search / Get product by `ID`

Requirements:

- Use `JWT` in request headers
- Validate status codes and message fields
- Use appropriate `HTTP` method/body

3. Submission Guidelines

- ❗ Compilation errors = 0 points
- ❗ No hardcoded SQL / irrelevant logic = 0 points
- File naming must follow `PE_PRN232_SU25_<YourStudentID>_api` convention
- Postman Script `PE_PRN232_SU25_<YourStudentID>_PM.txt`
- Upload `.zip` file containing all code and test files

Good luck!