Full-Stack Coding Assessment — ASP.NET Core Web API (EF Core) + Angular + Azure

Total Marks: 100 (Backend 50, Frontend 50)
Max Duration: ~12 hours (2 working days)

Scenario: Build and Deploy a **Product Catalog** application where products are stored in **Azure SQL Database** and Product Images in **Azure Blob Storage**. Provide a production-ready CI/CD for both apps.

---

1) Outcomes Assessed

- Create a **Web API** with **Entity Framework Core** using code-first migrations.
- Create a **frontend Angular** application.
- **Integrate** Angular with the Web API.
- Implement **CRUD** for `Product { id, name, price, imageUrl }`.
- Persist data in **Azure SQL** and images in **Azure Blob Storage** (public read or SAS-based access).
- Configure **Azure DevOps (YAML)** pipeline to deploy **Web API** to **Azure App Service**.
- Configure **GitHub Actions** to deploy **Angular** app to **Azure App Service**.

---

2) Tech Stack & Constraints

- **.NET**: ASP.NET Core **.NET 8 LTS or 9** (Controllers).
- **EF Core**: Code-First with Migrations.
- **DB**: Azure SQL (Single Database or Elastic Pool; S0 or free trial acceptable).
- **Storage**: Azure Blob Storage (Hot Tier).
- **Angular**: v16+ (prefer latest LTS you have installed).
- **Node**: 18+ LTS.
- **CI/CD**:
  - Backend: **Azure DevOps Pipelines (YAML/Classic)** → Azure App Service (Linux/Windows).
  - Frontend: **GitHub Actions** → Azure App Service (Linux/Windows).
- **Source Control**:
  - Backend in **Azure Repos** (or GitHub mirrored to Azure DevOps).
  - Frontend in **GitHub**.
- **Security/Secrets**: Store secrets in **App Service Configuration** or GitHub/Azure DevOps **Secrets/Variables** (no secrets in code).
- **CORS**: Restrict to your Angular app's URL.

---

3) Deliverables

1. **Deployed URLs**
   - Backend API base URL (e.g., `https://<apiapp>.azurewebsites.net`).
   - Swagger/OpenAPI URL.
   - Angular frontend URL (e.g., `https://<webapp>.azurewebsites.net`).
2. **Repositories**
   - Azure Repos link (backend) with `azure-pipelines.yml`.
   - GitHub repo link (frontend) with `.github/workflows/deploy.yml`.
3. **Postman collection** or `http` files to test API endpoints.
4. **README** in both repos with setup, build, run, deploy instructions.
5. **Screenshots** of successful pipeline runs and app running.
6. **Architecture diagram** (simple block diagram).
7. **Evidence of no secrets in code** (explain where you stored them).

---

4) Functional Requirements

- **Product entity**
  - `id` (GUID or int identity), `name` (required), `price` (decimal > 0), `imageUrl` (string; URL to blob).
- **API Endpoints** (sample contract)
  - `GET /api/products` → list all.
  - `GET /api/products/{id}` → details.
  - `POST /api/products` → create product (name, price). Returns created entity.
  - `POST /api/products/{id}/image` → upload image (multipart/form-data: file). Updates `imageUrl` and returns entity.
  - `PUT /api/products/{id}` → update name/price.
  - `DELETE /api/products/{id}` → delete product and its image in blob.
- **Frontend**
  - Products list (table or cards): id, name, price, image preview.
  - Create/Edit product form (reactive forms + validation).
  - Upload image to the product (with progress bar).
  - Delete product (confirm dialog).
  - Error handling, loading states, and basic toasts/snackbars.

---

5) Non-Functional Requirements

- Proper **HTTP status codes**, model **validation**, and **global error handling**.
- **Logging** (console/app insights optional).
- **CORS** correctly configured.
- No sensitive data in source code.
- Clean folder structure and readable code.

---

6) Environment Setup (What you may use)

- Azure Subscription (student/trial acceptable).
- Azure resources: **App Service Plan (Linux)**, **2 App Services** (api + web), **Azure SQL** (server + database), **Storage Account** (Blob), optionally **App Insights**.
- Service connections:
  - Azure DevOps → Azure subscription (ARM).
  - GitHub → Publish profile or OpenID Connect with Azure.

---

7) Step-by-Step Checklist (with Marks)

Tick each item as you complete it. Marks per item are shown in **[brackets]**.

A) Backend API (50 marks)

1. **Project Setup & EF Core [8]**
   ☐ Create ASP.NET Core Web API (.NET 8).
   ☐ Add EF Core packages; define `Product` entity and `AppDbContext`.
   ☐ Create initial migration and update DB locally.
2. **CRUD Endpoints & Validation [10]**
   ☐ Implement `ProductsController` with GET/GET(id)/POST/PUT/DELETE.
   ☐ Use DTOs + DataAnnotations for validation.
   ☐ Return appropriate codes (201, 400, 404, 204).
3. **Azure SQL Integration [6]**
   ☐ Connection string via `DefaultConnection` (User-Managed Identity or SQL Auth).
   ☐ Apply migrations at startup to Azure DB.
4. **Blob Storage for Images [8]**
   ☐ Create container `product-images` (private or blob public read).
   ☐ Service to upload/delete blobs; generate `imageUrl`.
   ☐ Endpoint `POST /api/products/{id}/image` (multipart) updates product.
5. **Swagger & Testing [4]**
   ☐ Enable Swagger in Prod (with auth off, okay) and document endpoints.
   ☐ Provide Postman collection.
6. **Errors, Logging, CORS [6 → 4 (errors/logging) + 2 (CORS)]**
7. ☐ Global exception handler and logging.
   ☐ CORS restrict to frontend origin.
8. **CI (Build/Test/Publish) with Azure DevOps [4]**
   ☐ YAML pipeline: restore, build, test, publish artifact.
9. **CD to Azure App Service [4]**
   ☐ Deploy artifact to App Service using service connection; config app settings (connection string, blob info); smoke test `/swagger`.

Backend Total: 50

B) Frontend Angular (50 marks)

1. **Scaffold & Routing [5]**
   ☐ New Angular app with routing; environments for `apiBaseUrl`.
2. **UI & Forms [10]**
   ☐ Product list page with image thumbnails.
   ☐ Create/Edit page with reactive forms + validation (required name, positive price).
3. **API Integration Service [8]**
   ☐ `ProductService` with CRUD using `HttpClient`.
   ☐ Image upload via `FormData` + progress reporting.
4. **UX Polish [9 → 5 (UX) + 4 (errors/empty)]**
5. ☐ Loading spinners, toasts/snackbars, confirm delete, empty/404 states.
6. **Config & Build [5 → 3 (env) + 2 (build)]**
   ☐ Environment-based `apiBaseUrl`; production build with AOT.
7. **CI with GitHub Actions [5]**
   ☐ Workflow to install, build, run tests/lint, upload artifact.
8. **CD to Azure App Service [5]**
   ☐ Deploy build `/dist` to Web App; smoke test homepage.

Frontend Total: 50

---

8) Timebox & Milestones (Ideal Timing — 8 hours)

- **Azure resources & wiring**: 1.0 h
- **Backend dev (EF + CRUD + Blob)**: 3.0 h
- **Frontend dev (UI + integration)**: 2.5 h
- **Backend CI/CD (ADO)**: 0.75 h
- **Frontend CI/CD (GitHub Actions)**: 0.5 h
- **Buffer & testing**: 0.25 h