# Assignment Submission

Generated: 2025-09-28T20:33:07.454157

Source file: documentation\SUBMISSION.md

# Assignment Submission — Employee Management System

Course: BIT21504 - Web Related Frameworks II

Assignment: CRUD App with ASP.NET Core Web API + Angular

Student: <Your name and student ID here>

Submission date: <fill in>

---

## 1) One-line summary

A full-stack Employee Management System with an ASP.NET Core Web API (Entity Framework Core + MSSQL) backend and an Angular frontend (Angular Material). Includes REST endpoints, Angular service + components, pagination/search, snackbar notifications and Dockerfiles for both services.

## 2) Requirements coverage (mapping to `documentation/Instructions.md`)

* Backend
* [x] New ASP.NET Core Web API project (backend/EmployeeManagementSystem)
* [x] Entity Framework Core + MSSQL DbContext (`AppDbContext`)
* [x] Employee model with Id, Name, Position, Department, Salary
* [x] Migrations applied (use `dotnet ef database update`) — database up to date
* [x] `EmployeesController` exposing:
* GET /api/employees
* GET /api/employees/{id}
* POST /api/employees
* PUT /api/employees/{id}
* DELETE /api/employees/{id}
* Frontend
* [x] Angular project (frontend/employee-management)
* [x] Angular Material installed and used
* [x] `employee.service.ts` calling Web API via HttpClient
* [x] Employee List component (table)
* [x] Add / Edit Employee form (Reactive Forms)
* [x] Delete confirmation dialog
* [x] Angular Routing configured
* Deliverables included
* [x] Full source code (this repository)
* [x] Swagger (backend exposes Swagger in Development)
* [x] Database migrations (EF Core migrations are in the backend project; use `dotnet ef migrations list`)
* [x] Dockerfiles for backend and frontend (backend/EmployeeManagementSystem/Dockerfile, frontend/employee-management/Dockerfile)
* Bonus features implemented
* [x] Pagination & Search on employee list
* [x] Angular Material Snackbar notifications for add/update/delete
* [x] Dockerfile for both backend and frontend

## 3) Files of interest (short list)

* backend/EmployeeManagementSystem/
* Program.cs (service registration, CORS configuration for dev)
* Controllers/EmployeesController.cs
* Data/AppDbContext.cs
* Models/Employee.cs
* Properties/launchSettings.json
* Dockerfile
* frontend/employee-management/
* src/app/services/employee.service.ts
* src/app/components/employee-list/\*
* src/app/components/employee-form/\*
* src/app/components/delete-dialog/\*
* app.routes.ts, app.config.ts, main.ts
* Dockerfile
* documentation/
* README.md (how-to/run), Instructions.md (assignment text), SUBMISSION.md (this file)

## 4) How to run (local development)

Notes: These commands are for Windows PowerShell as used in the assignment environment. Ensure prerequisites are installed: .NET 8 SDK, Node.js 20+, npm, SQL Server.

1) Start SQL Server and ensure a database is available for the app. Update connection string if needed in `backend/EmployeeManagementSystem/appsettings.json`.

2) Backend — restore, apply migrations, run:

cd "backend/EmployeeManagementSystem"  
dotnet restore  
dotnet ef database update  
dotnet run

* After `dotnet run` you should see a line like `Now listening on: http://localhost:5192`.
* Swagger (Development): http://localhost:5192/swagger

Important note: For local development we enable a permissive CORS policy for the Angular dev server (http://localhost:4200). This is configured in `Program.cs` and is only enabled in the Development environment. In production revert to a stricter policy.

3) Frontend — install packages and start dev server:

cd "frontend/employee-management"  
npm install --legacy-peer-deps  
npm start

* The Angular app will be available at http://localhost:4200

4) Test the API from the host (PowerShell example):

Invoke-WebRequest -Uri 'http://localhost:5192/api/employees' -UseBasicParsing

## 5) How to run with Docker (recommended for grading)

Prerequisite: Docker Desktop running.

1) Backend image and container (example):

cd backend/EmployeeManagementSystem  
docker build -t employee-backend .  
# Run with host SQL Server connection (example) and expose port 5000  
docker run -e "ConnectionStrings\_\_DefaultConnection=Server=host.docker.internal\\SQLEXPRESS;Database=bit21504\_EmployeeManagement\_c113310;Trusted\_Connection=True;TrustServerCertificate=True;" -p 5000:5000 employee-backend

* API available at: http://localhost:5000
* Swagger: http://localhost:5000/swagger

2) Frontend image and container:

cd frontend/employee-management  
docker build -t employee-frontend .  
docker run -p 4200:80 employee-frontend

* Frontend available at: http://localhost:4200
* Confirm the frontend calls the backend at http://localhost:5000/api/employees if you used the Docker mapping above.

## 6) Testing & API exploration

* Use Swagger (backend in Development) at `/swagger` to test endpoints.
* Or use Postman / curl:

# Get all employees  
Invoke-WebRequest -Uri 'http://localhost:5192/api/employees' -UseBasicParsing

* Frontend exercises the API via UI (Add/Edit/Delete). Pagination and search are available in the list.

## 7) Database script

* EF Core migrations are in the backend project. To create a SQL script for submission (optional):

cd backend/EmployeeManagementSystem  
dotnet ef migrations script -o migrations.sql

* This will produce `migrations.sql` containing the current schema changes.

## 8) Known issues / notes (important for grading)

* CORS: The API sets a development-only CORS policy allowing `http://localhost:4200`. This is intentional for local testing; remove or tighten this for production.
* HTTPS redirect: During development HTTPS redirection may be disabled in `Program.cs` to avoid TLS binding issues on machines without the dev certificate. Re-enable as needed for production.
* Decimal precision warning: EF Core warns that `Salary` (decimal) has no explicit column type/precision — this is harmless for the assignment but can truncate large values; to fix add `HasPrecision(...)` or `HasColumnType("decimal(18,2)")` in `OnModelCreating`.
* If you run backend in Docker on Windows and it needs to connect to a host SQL Server, use `host.docker.internal` in the connection string (see `documentation/README.md`).

## 9) Screenshots / Evidence

Place screenshots in `documentation/screenshots/` and reference them here (examples):

* `screenshots/frontend-list.png` — Employee list view with pagination
* `screenshots/frontend-form.png` — Add/Edit form
* `screenshots/backend-swagger.png` — Swagger UI showing endpoints

(Include screenshots when you prepare the submission zip or GitHub release.)

## 10) Checklist for submission

* [ ] Source code pushed to GitHub
* [ ] `documentation/SUBMISSION.md` (this file) included
* [ ] `documentation/README.md` (run instructions) included
* [ ] Dockerfiles present for both services
* [ ] Migrations / SQL script included or generated
* [ ] Screenshots attached
* [ ] README updated with any environment-specific notes (done)

## 11) Contact / notes to marker

If you need the app running on a specific port or require sample data, I can:

* Provide a small seed data script,
* Reconfigure the backend port to 5000 to match Docker run instructions,
* Add a small Postman collection.

---

If you want, I can now:

* Generate `migrations.sql` and save it under `documentation/` for submission, and
* Create the `documentation/screenshots/` folder with placeholder README explaining what screenshots to include.

Tell me which of these extra items you want me to add next and I'll create them.