Training Company :

Requirements:

1. Develop endpoints for CRUD operations for courses and trainers.
2. Implement functionality to link a course to a trainer.
3. Include endpoints for managing payments for trainers.
4. Develop basic reporting functionality for course-trainer linkages.
5. Write unit tests to ensure the reliability of the API.
6. Consider scalability and performance optimization in your implementation.

API:

- `Register` **(POST `/api/Auth/Register`)**
  - **Purpose:** Allows a new user to sign up.
  - **Input:** `RegisterDto` (includes email, password, and full name).
  - **Response:**
    - `200 OK` if registration is successful (returns user details and token).
    - `400 Bad Request` if registration fails.

- `Login` **(POST `/api/Auth/Login`)**
  - **Purpose:** Authenticates an existing user and returns a JWT token.
  - **Input:** `LoginDto` (email & password).
  - **Response:**
    - `200 OK` if login is successful (returns user details and token).
    - `400 Bad Request` if login fails.

TrainerController Endpoints

Method	Route	Description	Request Body / Params	Response
POST	`/api/Trainer`	Add a new trainer	`CreateTrainerDto`	`200 OK` (Success) / `400 Bad Request`
PUT	`/api/Trainer/{id}`	Update an existing trainer	`id (int)`, `CreateTrainerDto`	`200 OK` (Updated) / `404 Not Found`
DELETE	`/api/Trainer/{id}`	Delete a trainer	`id (int)`	`200 OK` (Deleted) / `404 Not Found`
GET	`/api/Trainer/{id}`	Get details of a trainer	`id (int)`	`200 OK` (`TrainerViewDto`) / `404 Not Found`

GET	`/api/Trainer`	Get all trainers	None	`200 OK` (List of `TrainerViewDto`)

TrainerCourseController Endpoints

Method	**Route**	**Description**	**Request Body / Params**	**Response**
POST	`/api/TrainerCourse`	Assign a course to a trainer	`AddCourseToTrainerDto`	`200 OK` (Success) / `400 Bad Request`
GET	`/api/TrainerCourse/{id}`	Get courses assigned to a trainer	`trainerId (int)`	`200 OK` (List of `CourseTrainersViewDto`) / `404 Not Found`
GET	`/api/TrainerCourse`	Get all course trainers with payments	None	`200 OK` (List of `CourseTrainersWithPaymentsView`)
GET	`/api/TrainerCourse/export-pdf`	Export course trainers with payments report as PDF	None	`200 OK` (PDF file) / `404 Not Found`

CourseController Endpoints

Method	**Route**	**Description**	**Request Body / Params**	**Response**
POST	`/api/Course`	Add a new course	`CreateCourseDto`	`200 OK` (Success) / `400 Bad Request`
PUT	`/api/Course/{id}`	Update an existing course	`id (int)`, `CreateCourseDto`	`200 OK` (Updated) / `404 Not Found`
DELETE	`/api/Course/{id}`	Delete a course	`id (int)`	`200 OK` (Deleted) / `404 Not Found`
GET	`/api/Course/{id}`	Get details of a course	`id (int)`	`200 OK` (`ViewCourseDto`) / `404 Not Found`
GET	`/api/Course`	Get all courses	None	`200 OK` (List of `ViewCourseDto`)

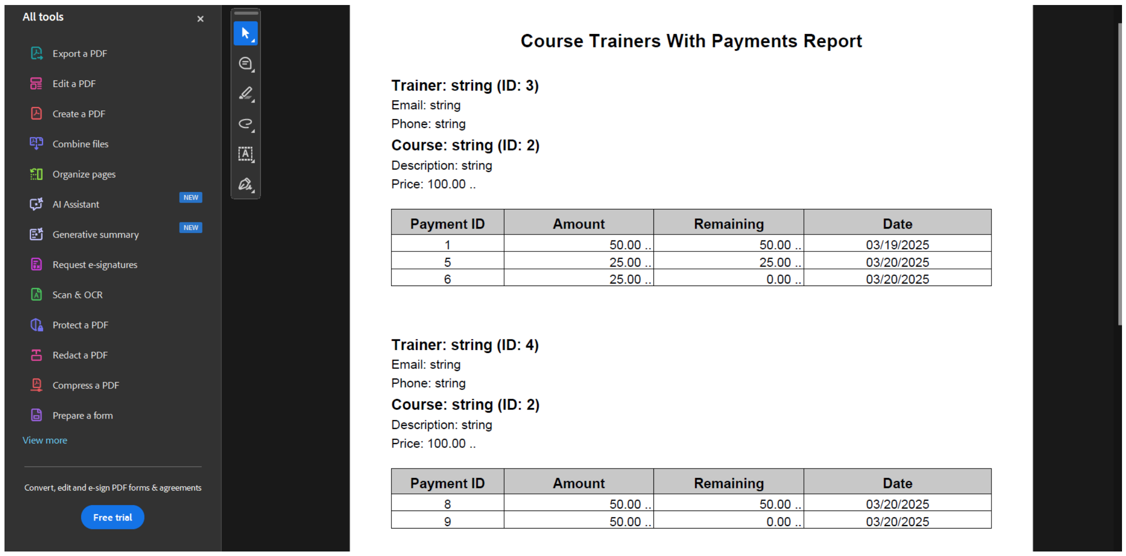
PaymentController Endpoints

Method	**Route**	**Description**	**Request Body / Params**	**Response**
POST	`/api/Payment`	Add a payment for a course	`AddCourseToTrainerDto`	`200 OK` (Success) / `400 Bad Request`
GET	`/api/Payment`	Get payments for a trainer and course	`trainerId (int)`, `courseId (int)` (Query Params)	`200 OK` (List of `TrainerPaymentsDto`)

📌 Technical Stack

- **Backend**: .NET Core

- **Logging**: Serilog
- **Database**: Entity Framework Core
- **Authentication**: JWT
- **Testing**: xUnit (Unit Testing)
- **Caching**: In-Memory Cache
- **Reporting**: PDF Library (for generating reports)
-



Course Trainers With Payments Report

Trainer: string (ID: 3)
Email: string
Phone: string
Course: string (ID: 2)
Description: string
Price: 100.00 ..

Payment ID	Amount	Remaining	Date
1	50.00 ..	50.00 ..	03/19/2025
5	25.00 ..	25.00 ..	03/20/2025
6	25.00 ..	0.00 ..	03/20/2025

Trainer: string (ID: 4)
Email: string
Phone: string
Course: string (ID: 2)
Description: string
Price: 100.00 ..

Payment ID	Amount	Remaining	Date
8	50.00 ..	50.00 ..	03/20/2025
9	50.00 ..	0.00 ..	03/20/2025