

ASP.NET Core assessment



Name: Peter Ezzat Zakher

Table of Contents :

Abstract

Chapter 1 : Introduction

Overview

Objectives

Purpose

Scope

General Constraints

Chapter 2: Project “ Planning and analysis ”

Analysis of the new system

Functional Requirements

Non – Functional Requirements

Chapter 3 : Software Design

Design of database “ ERD Diagram”

Chapter 4: Implementation

Software architecture

Pseudocode , Flowchart or workflow “ workflow ”

Chapter 5 : Web Api

Document controller api code

Priority controller code

Controller tests

Chapter 1 :

Introduction

1.1 overview :

Part one: Creating sample web API for storing documents with some metadata, and accept request to get documents itself or information about it (metadata). Part two Create sample web interface for uploading document(s) or display available documents in table, each row has a link to display document information(s), editing and deleting.

1.2 Objectives :

The system provides a way to upload your documents on an online server so you can access them any time

1.3 Purpose :

The purpose of the project is to create a website that provides the user to upload his/her documents on a web server.

1.4 Scope :

- Coding “ language & tools ”:

I used :

- HTML , Css , Javascript , Bootstrap , JQuery .
- C# , asp .net core MVC.

- datatables ajax control .
- SQL Server , Entity framework core.
- automapper.
- dependency injection.
- Visual Studio 2022 .

- Testing :

We have test cases discuss what conditions the system are success or fail .

1.5 General Constraints :

In this section we will introduce some of the general constrains in our project:

- Time :

I can consider that time is the main constrain, because I had to finish the project in a short time.

Chapter 2:

Project “ Planning & Analysis ”

2,1 Project Planning :

Market analysis:

1. Main target actors:

- Ordinary user

2. Effective points in the project:

- Create new directory on the server for each uploaded document
- Upload document files to its directory

3. Target market:

- **Age** : 10 – 60
- **Gender** : both
- **Place** : Any Place
- **Level of education** : Any level .

Technical Analysis:

- No hardware required for the site.
- I used a lot of technologies such as :
 - Html5 , css3 , JS , Bootstrap , JQuery
 - Asp .net core , C# , Ajax , API
- I used a lot of software tools :
 - Visual studio code
 - Visual studio
 - Microsoft office
 - Microsoft SQL server Management studio
 - Postman.

2.4.4 Functional Requirements

User Functional Requirements:

- **Upload a document:**

Actor: User

Pre: none.

Description: if the user wants to upload a document, the system creates a new directory on the server with name [Documentname_DocumentId] and then upload its file on the same directory.

- **Update document information**

Actor: User

Pre: document already exists on the server

Description: User can update document information by selecting what document he/she want to update from the data table

- **Get document information**

Actor: User

Pre: document already exists on the server

Description: User can View all documents information from the data table.

- **Delete entire document**

Actor: User

Pre: document already exists on the server

Description: User can delete document full record including its files from the server, the server first deletes the

document's directory including its files from the server then deletes the full records from the database.

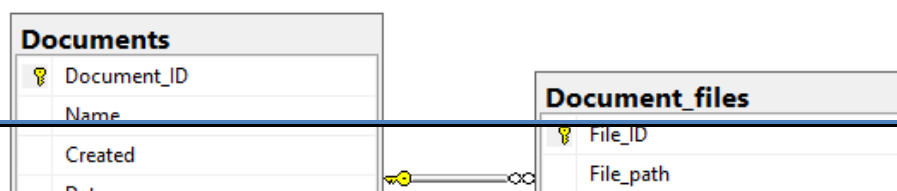
2.4.5 Non- Functional Requirements:

- **Availability :**
The site must be contactable 24 hours .
- **Security :**
Documents are secured on the server and can't be manipulated.
- **Usability :**
 - The site should be suitable for all ages.
 - It should be easy to use so that the user can understand it.
- **Flexibility :**
 - The user can modify his data or declare anything he wants through comments
- **Efficiency :**
The user must get what he wants, whether he is viewing, sharing, or finding.

Chapter 3 :

Software Design

3.1: Design of database ERD Diagram



Chapter 4 :

Implementation :

4.1 software architecture:

4.1 Folder upload

```
1 reference
public static string CreateNewFolder(string DocumentName, int DocumentId)
{
    string FolderName = DocumentName + "_" + DocumentId;
    var directory = Directory.GetCurrentDirectory() + "/wwwroot/Uploads/" + FolderName;
    if (!Directory.Exists(directory))
    {
        Directory.CreateDirectory(directory);
    }
    return FolderName;
}
```

4.2 Folder delete

```
< references
public static void DeleteFolder(string FolderPath)
{
    if (Directory.Exists(FolderPath))
    {
        // Delete all files from the Directory
        foreach (string filename in Directory.GetFiles(FolderPath))
        {
            File.Delete(filename);
        }
        // Check all child Directories and delete files
        foreach (string subfolder in Directory.GetDirectories(FolderPath))
        {
            DeleteFolder(subfolder);
        }
        Directory.Delete(FolderPath);
    }
}
```

4 software architecture:

4.3 File upload

```
reference
public static string UploadFile(IFormFile File, string PhysicalPath)
{
    string FilePath = Directory.GetCurrentDirectory() + PhysicalPath;
    string FileName = Guid.NewGuid().ToString() + Path.GetFileName(File.FileName) ;
    string finalpath = FilePath + FileName;
    using (var stream = System.IO.File.Create(finalpath))
    {
        File.CopyTo(stream);
    }
    string FileFinalPath = PhysicalPath + FileName;
    return FileFinalPath;
}
```

4.4 Upload document

```

2 references
public void UploadDocument(DocumentVM documentVM)
{
    Documents document = map(documentVM);
    DocumentRepo.Add(document);

    int documentId = DocumentRepo.GetLastOne(document => document.Name.Equals(documentVM.Name)).
        DocumentId;
    string FolderName = FolderManger.CreateNewFolder(documentVM.Name, documentId);

    foreach (var file in documentVM.DocumentFiles)
    {
        DocumentFilesVM DocumentFile = new DocumentFilesVM { DocumentFile = file,
            DocumentId = documentId,
            DocumentFolderName=FolderName };
        documentFilesManger.UploadFile(DocumentFile);
    }
}
1 reference

```

4.5 Search Document

```

1 reference
public IQueryable<Documents> SearchDocument(string SearchVal = null)
{
    if (string.IsNullOrEmpty(SearchVal)){
        return DocumentRepo.GetAll();
    }

    return DocumentRepo.GetMany(doc => doc.Name.Contains(SearchVal) ||
        doc.Created.ToString().Contains(SearchVal) ||
        doc.DueDate.ToString().Contains(SearchVal) ||
        doc.Priority.ToString().Contains(SearchVal) ||
        doc.Created.ToString().Contains(SearchVal) ||
        doc.Date.ToString().Contains(SearchVal));
}
6 references

```

4.6 Get Document By Id

```

6 references
public DocumentVM GetDocumentById(int id)
{
    var data = DocumentRepo.GetOne(doc => doc.DocumentId == id, doc => doc.PriorityNavigation);
    if (data == null)
        return null;
    DocumentVM documentVM =mapper.Map<DocumentVM>(data);
    documentVM.PriorityName = data.PriorityNavigation.Name;
    return documentVM;
}
2 references

```

4.7 Update Document

```

public void UpdateDocument(DocumentVM documentVM)
{
    Documents doc= map(documentVM);
    DocumentRepo.Edit(doc);
}

```

2 references

4.8 Delete Document

```

public void DeleDocument(int Id)
{
    var data = GetDocumentById(Id);
    string DocumentName = data.Name + "_" + data.DocumentId;
    FolderManger.DeleteFolder(DocumentName);
    DocumentRepo.Delete(Id);
}

```

1 reference

4.9 Get All Documents

```

public List<DocumentVM> GetDocuments()
{
    var data = DocumentRepo.GetAll();
    return mapper.Map<IQueryable<DocumentVM>>(data).ToList();
}

```

2 references

4.10 Upload document files

```

public void UploadFile(DocumentFilesVM documentFilesVM)
{
    documentFilesVM.FilePath = FileManger.UploadFile(documentFilesVM.DocumentFile,
        "/wwwroot/Uploads/" + documentFilesVM.DocumentFolderName + "/");
    DocumentFiles data =mapper.Map<DocumentFiles>(documentFilesVM);
    documentFilesRepo.Add(data);
}

```

4.11 Priority crud operations

5 references

```
public IQueryable<Priorities> GetAllPriorities()
```

```
{
```

```
    return PrioritiesRepo.GetAll();
```

```
}
```

1 reference

```
public Priorities SearchPriority(int id)
```

```
{
```

```
    return PrioritiesRepo.Get(id);
```

```
}
```

4.12 Document datatable

,

```
[HttpPost]
```

0 references

```
public IActionResult ListCustomer()
```

```
{
```

```
    var pageSize = int.Parse(Request.Form["length"]);
```

```
    var skip = int.Parse(Request.Form["start"]);
```

```
    var searchValue = Request.Form["search[value]"];
```

```
    var sortColumn = Request.Form[string.Concat("columns[", Request.Form["order[0][column]", "][name]"]];
```

```
    var sortColumnDirection = Request.Form["order[0][dir]"];
```

```
    var Documents = documentManger.SearchDocument(searchValue);
```

```
    if (!(string.IsNullOrEmpty(sortColumn) && string.IsNullOrEmpty(sortColumnDirection)))
```

```
        Documents = Documents.OrderBy(string.Concat(sortColumn, " ", sortColumnDirection));
```

```
    var data = Documents.Skip(skip).Take(pageSize).ToList();
```

```
    var recordsTotal = Documents.Count();
```

```
    var jsonData = new { recordsFiltered = recordsTotal, recordsTotal, data };
```

```
    return Ok(jsonData);
```

```
}
```

Document controller

```

,
[HttpGet]
0 references
public IActionResult Index(string Msg)
{
    if(!string.IsNullOrEmpty(Msg))
    {
        ViewBag.Msg = Msg;
    }
    return View();
}
[HttpGet]
0 references
public IActionResult upload()
{
    ViewBag.PrioritiesSelectList = priorityManger.GetAllPriorities();

    return View();
}
-

```

AspnetCoreAssessment.Controllers.DocumentControll Index(s

```

[HttpPost]
0 references
public IActionResult upload(DocumentVM documentVM)
{
    if(ModelState.IsValid)
    {
        if(documentVM.DocumentFiles == null)
        {
            ModelState.AddModelError("", "At Least Upload one file");
            return View();
        }
        documentManger.UploadDocument(documentVM);
        Redirect("/Document#AvalableDocs");
    }
    ViewBag.PrioritiesSelectList = priorityManger.GetAllPriorities();
    return View();
}

[HttpGet]
0 references
public IActionResult details(int Id=0)
{
    if(Id==0)
    {
        return Redirect("/Document?Msg=NoDocument#Msg");
    }
    var data = documentManger.GetDocumentById(Id);
    return View(data);
}

```

```

}
[HttpGet]
0 references
public IActionResult update(int Id = 0)
{
    ViewBag.PrioritiesSelectList = priorityManger.GetAllPriorities();

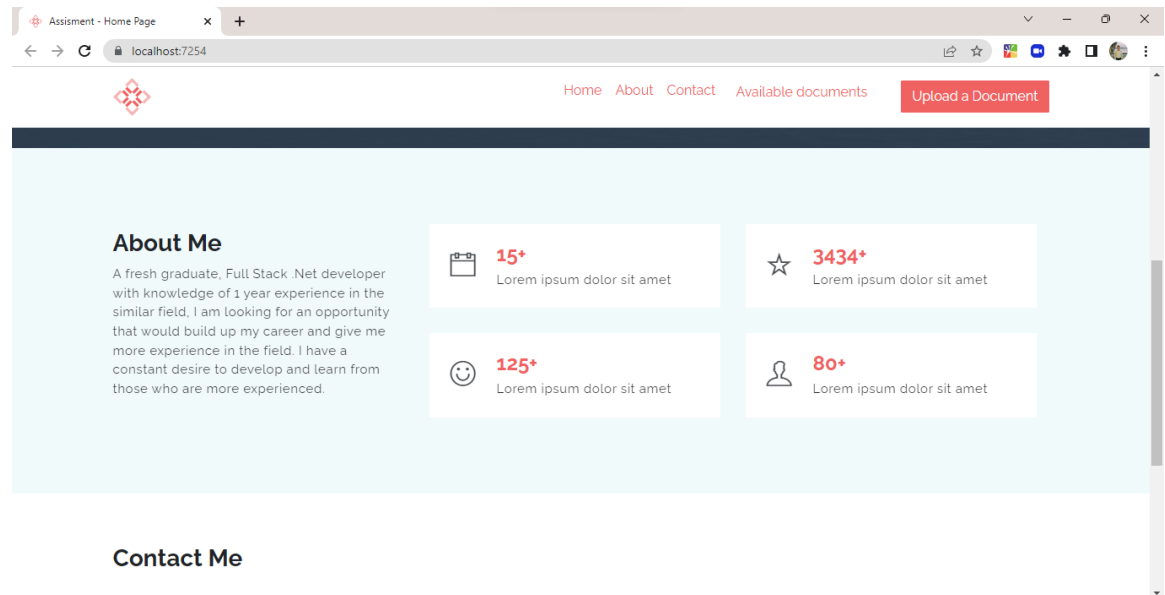
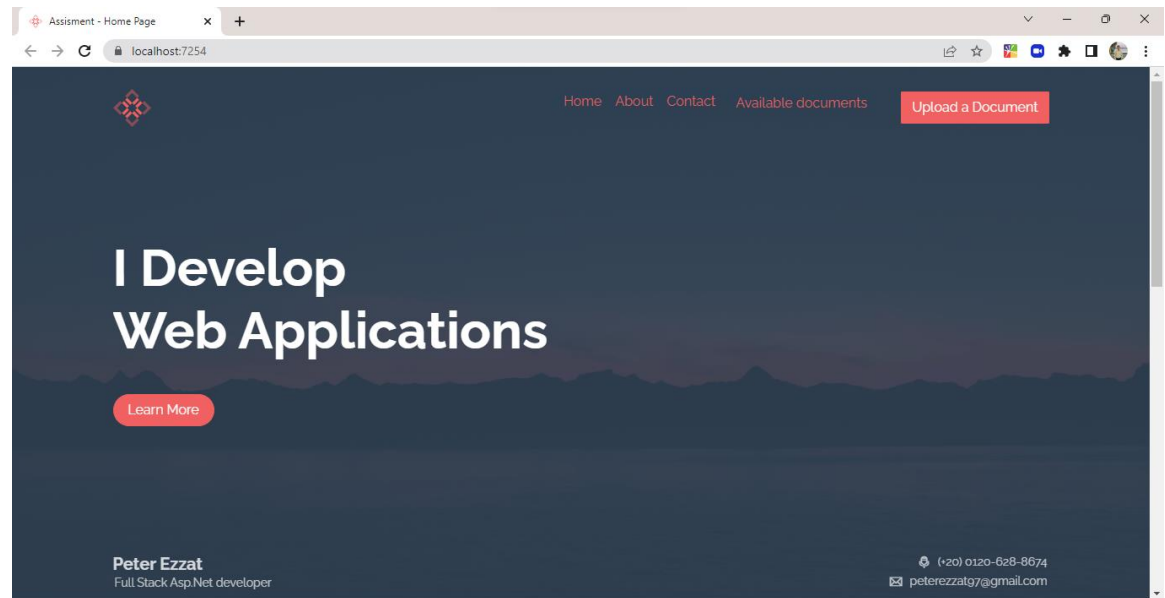
    if (Id == 0)
    {
        return Redirect("/Document?Msg=NoDocument#Msg");
    }
    var data = documentManger.GetDocumentById(Id);
    return View(data);
}
[HttpPost]
0 references
public IActionResult update(DocumentVM documentVM)
{
    ViewBag.PrioritiesSelectList = priorityManger.GetAllPriorities();

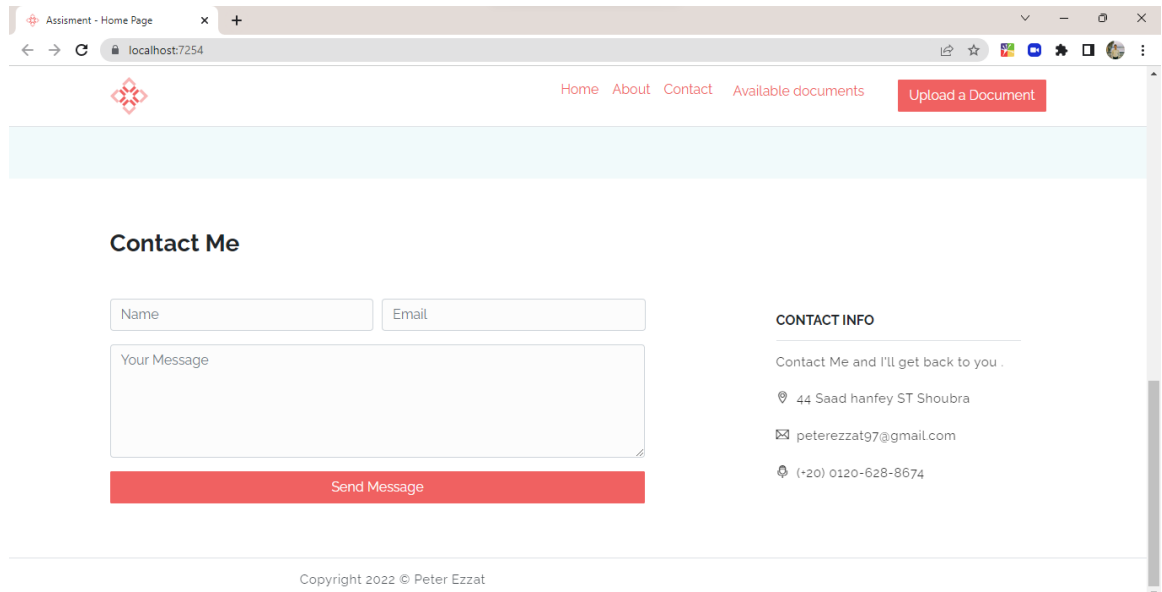
    if (ModelState.IsValid)
    {
        documentManger.UpdateDocument(documentVM);
        return Redirect("/Document?Msg=UpdatedSuccessfully#Msg");
    }
    return View();
}
[HttpGet]
0 references
public IActionResult delete(int Id = 0)
{
    if (Id == 0)
        return Redirect("/Document?Msg=NoDocument#Msg");
    documentManger.DeleDocument(Id);
    return Redirect("/Document?Msg=DeletedSuccessfully#Msg");
}

```

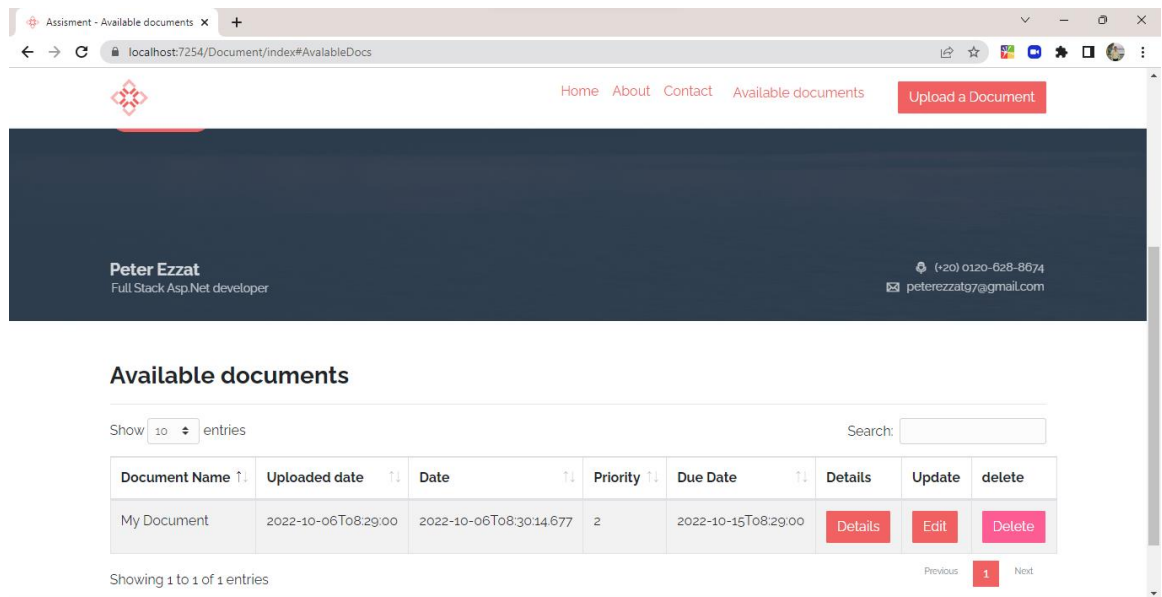

5.0: Pseudocode, Flowchart or workflow :

5.1: Home Page :





5.2: Available documents



5.3 Document Details

The screenshot shows a web browser window with the URL `localhost:7254/Document/details?id=15`. The page has a dark blue header with the Peter Ezzat logo and contact information. The main content area is titled 'Document information' and contains several input fields for document details. The fields are: Document Name (My Document), Created date (10/6/2022 8:30:14 AM), Date (10/6/2022 8:29:00 AM), Priority (High), and Due Date (10/15/2022 8:29:00 AM). There is an 'Upload a Document' button in the top right corner and another one at the bottom right. The footer contains the text 'Copyright 2022 © Peter Ezzat'.

Assisment - Document informati: x +

localhost:7254/Document/details?id=15

Home About Contact Available documents Upload a Document

Peter Ezzat
Full Stack Asp.Net developer
(+20) 0120-628-8674
peterezzat97@gmail.com

Document information

Document Name
My Document

Created date
10/6/2022 8:30:14 AM

Date
10/6/2022 8:29:00 AM

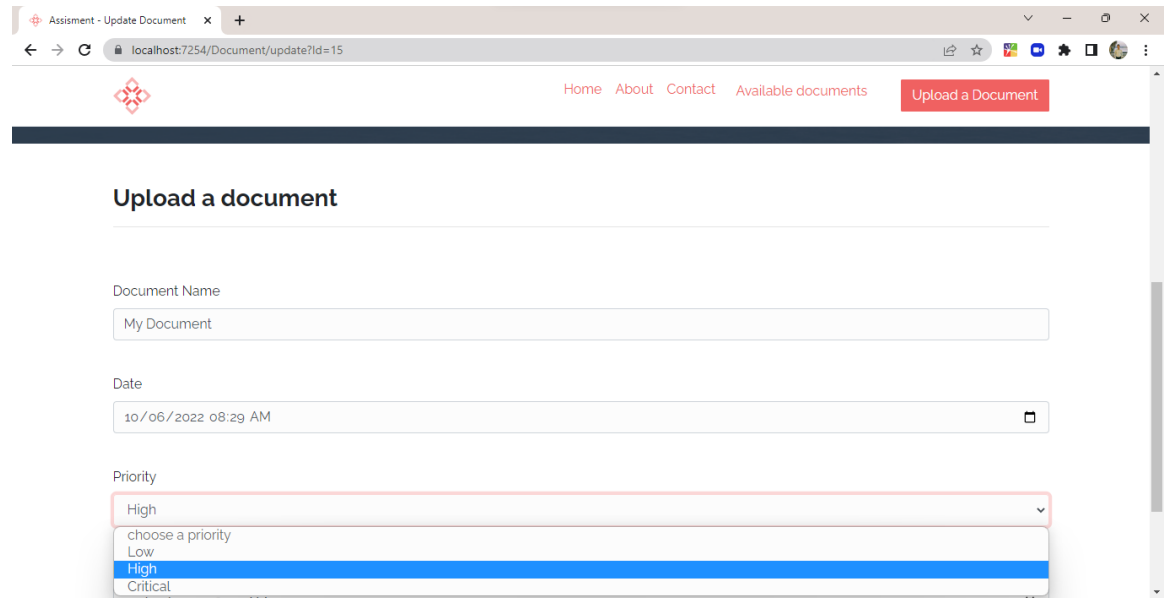
Priority
High

Due Date
10/15/2022 8:29:00 AM

Upload a Document

Copyright 2022 © Peter Ezzat

5.4 Update document



Assisment - Update Document x +

localhost:7254/Document/update?id=15

Home About Contact Available documents Upload a Document

Upload a document

Document Name

My Document

Date

10/06/2022 08:29 AM

Priority

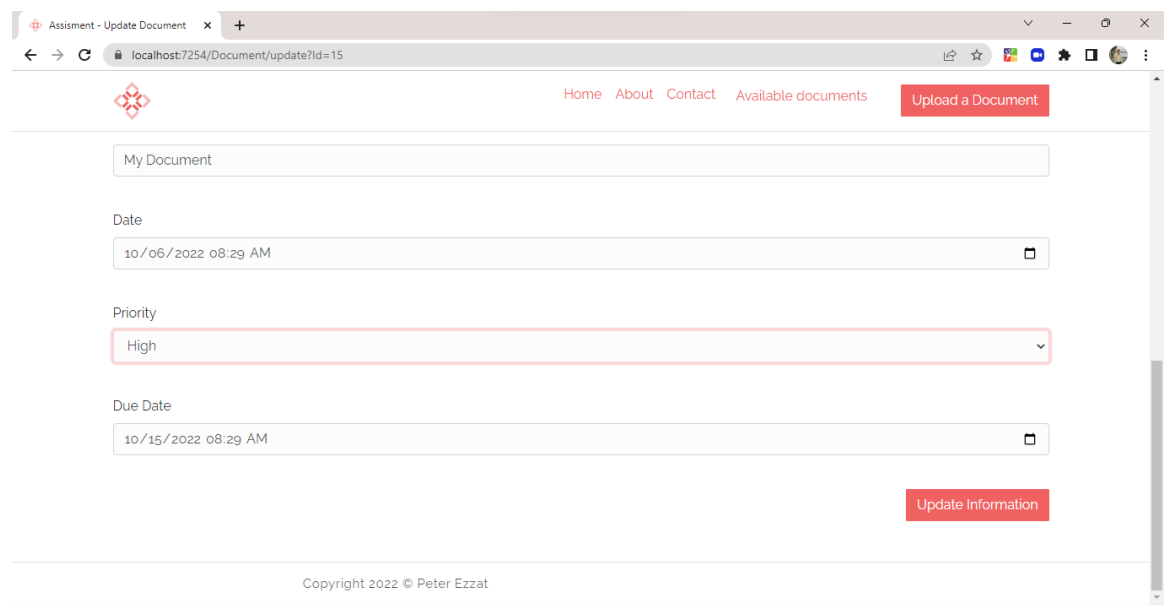
High

choose a priority

Low

High

Critical



Assisment - Update Document x +

localhost:7254/Document/update?id=15

Home About Contact Available documents Upload a Document

My Document

Date

10/06/2022 08:29 AM

Priority

High

choose a priority

Low

High

Critical

Due Date

10/15/2022 08:29 AM

Update Information

Copyright 2022 © Peter Ezzat

Chapter 5:

WEB Api

Document Api Link:

<https://localhost:7254/api/DocumentApi>

Priority Api Link :

<https://localhost:7254/api/Priorities>

5.1 Document controller Api code:

5.1.1 Upload a document

[HttpPost]

0 references

```
public IActionResult upload(DocumentVM documentVM)
{
    if (documentVM.DocumentFiles == null)
        return BadRequest();
    var priority = priorityManger.SearchPriority(documentVM.Priority);
    if(priority == null)
        return NotFound();
    documentManger.UploadDocument(documentVM);
    return Ok();
}
```

5.1.2 Document details

```
,  
[HttpGet("{Id}")]  
0 references  
public IActionResult details(int Id = 0)  
{  
    if (Id == 0)  
        return BadRequest();  
    var data = documentManger.GetDocumentById(Id);  
    if (data == null)  
        return NotFound();  
    return Ok(data);  
}  
-  
-
```

5.1.3 Update document

```
,  
[HttpPut]  
0 references  
public IActionResult update(DocumentVM documentVM)  
{  
    var data = documentManger.GetDocumentById(documentVM.DocumentId);  
    if (data == null)  
        return NotFound();  
  
    documentManger.UpdateDocument(documentVM);  
    return NoContent();  
}
```

5.1.4 Delete a document

```
}  
[HttpDelete("{Id}")]  
0 references  
public IActionResult delete(int Id = 0)  
{  
    if (Id == 0)  
        return BadRequest();  
    var data = documentManger.GetDocumentById(Id);  
    if (data == null)  
        return NotFound();  
    documentManger.DeleteDocument(Id);  
    return NoContent();  
}
```

5.1.5 Get All Documents

```
,  
[HttpGet]  
0 references  
public IActionResult GetAllDocuments()  
{  
    var data = documentManger.GetDocuments();  
    if (data == null)  
        return NoContent();  
    return Ok(data);  
}
```

5.2 Priority controller Api code:

5.2.1 Get All Priorities

```
[HttpGet]  
0 references  
public IActionResult GetAllPriorities()  
{  
    var data= priorityManger.GetAllPriorities();  
    return Ok(data);  
}
```

5.3 Api tests:

5.3.1 Documents api

5.3.1.1 HTTPGET

The screenshot shows the Postman interface with a GET request to `https://localhost:7254/api/DocumentApi`. The response is a JSON object with the following structure:

```
{
  "documentId": 15,
  "name": "My Document",
  "created": "2022-10-06T08:30:14.677",
  "date": "2022-10-06T08:29:00",
  "priority": 2,
  "dueDate": "2022-10-15T08:29:00",
  "priorityNavigation": null,
  "documentFiles": []
}
```

The response status is 200 OK, with a response time of 138 ms and a body size of 343 B.

5.3.1.2 HTTPGET({Id}) If id is found (on success)

The screenshot shows the Postman interface with a GET request to `https://localhost:7254/api/DocumentApi/15`. The response is a JSON object with the following structure:

```
{
  "documentId": 15,
  "name": "My Document",
  "created": "2022-10-06T08:30:14.677",
  "date": "2022-10-06T08:29:00",
  "priority": 2,
  "priorityName": "High",
  "dueDate": "2022-10-15T08:29:00",
  "documentFiles": []
}
```

The response status is 200 OK, with a response time of 160 ms and a body size of 337 B.

If id not found

The screenshot shows the Postman interface with a GET request to `https://localhost:7254/api/DocumentApi/16`. The response is a 404 Not Found status with a response time of 35 ms and a body size of 324 B. The response body is displayed in JSON format:

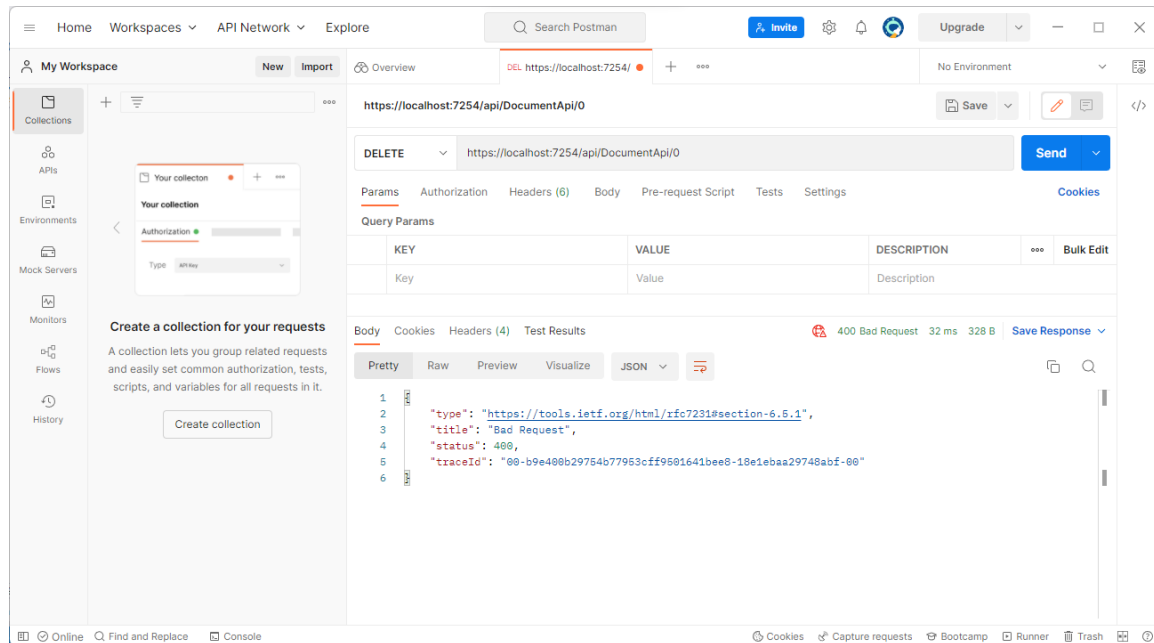
```
{
  "type": "https://tools.ietf.org/html/rfc7231#section-6.6.4",
  "title": "Not Found",
  "status": 404,
  "traceId": "00-9f5d0772f1c1cb3389c96533c7786bee-1c3186e3b298ea7d-00"
}
```

5.3.1.2 HttpDelete

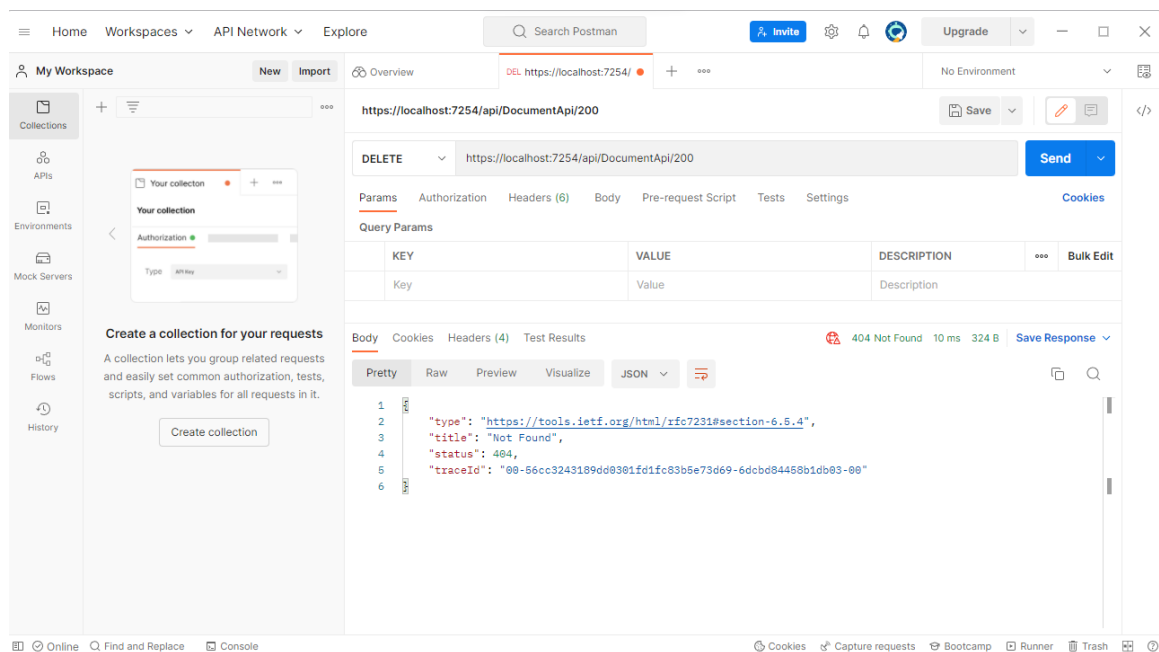
On success return no content

The screenshot shows the Postman interface with a DELETE request to `https://localhost:7254/api/DocumentApi/15`. The response is a 204 No Content status with a response time of 95 ms and a body size of 81 B. The response body is empty, indicating successful deletion.

If id is 0 return bad request



If user not found



Thanks