Київський національний університет ім. Тараса Шевченка

Кафедра мережевих та інтернет технологій

Лабораторна робота № 9

Дисципліна: Хмарні технології

Тема: Доповнити створений на попередньому занятті Web-застосунок новими сторінками, на яких користувачу пропонується проаналізувати візуальну інформацію різними способами (необхідно застосувати три способи: розпізнавання тексту (Optical character recognition**)**, аналіз зображень (Image Analysis) та Face service.

Виконав: Студент групи МІТ-31

Пугач Назар

Мета: Доповнити створений на попередньому занятті Web-застосунок новими сторінками, на яких користувачу пропонується проаналізувати візуальну інформацію різними способами (необхідно застосувати три способи: розпізнавання тексту (Optical character recognition**)**, аналіз зображень (Image Analysis) та Face service.

Хід роботи

Завдання 9.1: Створити Computer vision сервіс та Face API.

На платформі Azure потрібно знайти Computer vision сервіс та Face API , створити їх, використовуючи безкоштовну версію.

Завдання 8.2: Доповнити створений на попередньому занятті Web-застосунок новими сторінками, на яких користувачу пропонується проаналізувати візуальну інформацію різними способами (необхідно застосувати три способи: розпізнавання тексту (Optical character recognition**)**, аналіз зображень (Image Analysis) та Face service.

Спочатку створимо моделі:

1. Для Optical character recognition (OCR)

```
public class OCRResultModel
{
    public string FullText { get; set; }
    public string ImagePath { get; set; }

}
```

2. Для Image Analysis

```
public class ImageAnalysisResultModel
{
    public string Description { get; set; }
    public List<string> Categories { get; set; }
    public List<string> Tags { get; set; }
    public List<string> Objects { get; set; }
    public string ImagePath { get; set; }

    public ImageAnalysisResultModel()
    {
        Categories = new List<string>();
        Tags = new List<string>();
        Objects = new List<string>();
    }
}
```

3. Для Face service

```
public class FaceDetectionResultModel
{
    public string Message { get; set; }

    public List<FaceInfo> Faces { get; set; } = new List<FaceInfo>();
}

public class FaceInfo
{
    public string Glasses { get; set; }
    public double Pitch { get; set; }
    public double Roll { get; set; }
    public double Yaw { get; set; }
    public List<string> Accessories { get; set; } = new List<string>();
    public string BlurLevel { get; set; }
    public double BlurValue { get; set; }
    public string ExposureLevel { get; set; }
    public double ExposureValue { get; set; }
```

```
    public string NoiseLevel { get; set; }
    public double NoiseValue { get; set; }
    public string FaceImageUrl { get; set; }
}
```

Рисунок 9.1 – Моделі.

Створимо контролери.

1. OCRController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.CognitiveServices.Vision.ComputerVision;
using lab6.Models;
using Microsoft.Azure.CognitiveServices.Vision.ComputerVision.Models;

public class OCRController : Controller
{
    private readonly ComputerVisionClient _computerVisionClient;

    public OCRController(ComputerVisionClient computerVisionClient)
    {
        _computerVisionClient = computerVisionClient;
    }

    [HttpGet]
    public IActionResult Index()
    {
        return View(new OCRResultModel());
    }

    [HttpPost]
    public async Task<IActionResult> Index(IFormFile file)
    {
        var model = new OCRResultModel();

        if (file == null || !file.ContentType.StartsWith("image/"))
        {
            ModelState.AddModelError("", "File must be an image.");
            return View(model);
        }

        var uploadPath = Path.Combine("wwwroot", "uploads");
        Directory.CreateDirectory(uploadPath);
        var filePath = Path.Combine(uploadPath, file.FileName);

        await using (var fs = new FileStream(filePath, FileMode.Create))
        {
            await file.CopyToAsync(fs);
        }

        model.ImagePath = "/uploads/" + file.FileName;

        using var ms = new MemoryStream();
        await file.CopyToAsync(ms);
        ms.Seek(0, SeekOrigin.Begin);

        var ocrResult = await _computerVisionClient.ReadInStreamAsync(ms);
        var operationId = ocrResult.OperationLocation.Split('/').Last();
        var result = await
_computerVisionClient.GetReadResultAsync(Guid.Parse(operationId));

        while (result.Status == OperationStatusCodes.Running)
        {
```

```
            result = await
_computerVisionClient.GetReadResultAsync(Guid.Parse(operationId));
        }

        var textList = new List<string>();

        foreach (var page in result.AnalyzeResult.ReadResults)
        {
            foreach (var line in page.Lines)
            {
                textList.Add(line.Text);
            }
        }

        model.FullText = string.Join(" ", textList);

        return View(model);
    }
}
```

2. ImageAnalysisController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.CognitiveServices.Vision.ComputerVision;
using Microsoft.Azure.CognitiveServices.Vision.ComputerVision.Models;
using lab6.Models;

public class ImageAnalysisController : Controller
{
    private readonly ComputerVisionClient _computerVisionClient;

    public ImageAnalysisController(ComputerVisionClient
computerVisionClient)
    {
        _computerVisionClient = computerVisionClient;
    }

    [HttpGet]
    public IActionResult Index()
    {
        return View(new ImageAnalysisResultModel());
    }

    [HttpPost]
    public async Task<IActionResult> Index(IFormFile file)
    {
        var model = new ImageAnalysisResultModel();
        if (file == null || file.Length == 0) return View(model);

        using var ms = new MemoryStream();
        await file.CopyToAsync(ms);
        var imageBytes = ms.ToArray();

        var uploadsPath = Path.Combine("wwwroot", "uploads");
        Directory.CreateDirectory(uploadsPath);

        var fileName = Path.GetFileName(file.FileName);
        var filePath = Path.Combine(uploadsPath, fileName);
        await System.IO.File.WriteAllBytesAsync(filePath, imageBytes);

        model.ImagePath = "/uploads/" + fileName;

        var features = new List<VisualFeatureTypes?>
        {
            VisualFeatureTypes.Description,
```

```
                VisualFeatureTypes.Categories,
                VisualFeatureTypes.Tags,
                VisualFeatureTypes.Objects
            };

            var analysis = await
    _computerVisionClient.AnalyzeImageInStreamAsync(new
    MemoryStream(imageBytes), features);

            model.Description = (analysis.Description?.Captions?.Any() == true)
                ? string.Join("; ", analysis.Description.Captions.Select(c =>
    $"{c.Text} ({c.Confidence:P})"))
                : "No description available.";

            model.Categories = analysis.Categories?.Select(c => c.Name).ToList()
    ?? new();
            model.Tags = analysis.Tags?.Select(t => t.Name).ToList() ?? new();
            model.Objects = analysis.Objects?.Select(o => $"{o.ObjectProperty}
    ({o.Confidence:P})").ToList() ?? new();

            return View(model);
        }
    }
```

3. FaceController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.CognitiveServices.Vision.Face;
using Microsoft.Azure.CognitiveServices.Vision.Face.Models;
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.Formats.Png;
using SixLabors.ImageSharp.Processing;
using lab6.Models;

public class FaceController : Controller
{
    private readonly FaceClient _faceClient;

    public FaceController(FaceClient faceClient)
    {
        _faceClient = faceClient;
    }

    [HttpGet]
    public IActionResult Index()
    {
        return View(new FaceDetectionResultModel());
    }

    [HttpPost]
    public async Task<IActionResult> Index(IFormFile file)
    {
        var model = new FaceDetectionResultModel ();
        if (file == null || file.Length == 0)
        {
            model.Message = "File not loaded";
            return View(model);
        }

        using var ms = new MemoryStream();
        await file.CopyToAsync(ms);
        var imageBytes = ms.ToArray();

        var faceAttributes = new List<FaceAttributeType>
        {
```

```
                FaceAttributeType.Glasses,
                FaceAttributeType.HeadPose,
                FaceAttributeType.Accessories,
                FaceAttributeType.Blur,
                FaceAttributeType.Exposure,
                FaceAttributeType.Noise
            };

            var faces = await _faceClient.Face.DetectWithStreamAsync(
                image: new MemoryStream(imageBytes),
                returnFaceId: false,
                returnFaceLandmarks: false,
                returnFaceAttributes: faceAttributes,
                recognitionModel: "recognition_04",
                detectionModel: "detection_01"
            );

            if (faces == null || !faces.Any())
            {
                model.Message = "Face not found";
                return View(model);
            }

            using var image = Image.Load(imageBytes);
            int faceIndex = 1;

            foreach (var face in faces)
            {
                var rect = face.FaceRectangle;
                var cropped = image.Clone(ctx => ctx.Crop(new
Rectangle(rect.Left, rect.Top, rect.Width, rect.Height)));
                var faceFileName = $"face_{faceIndex}.png";
                var facePath = Path.Combine("wwwroot/images/faces",
faceFileName);
                Directory.CreateDirectory(Path.GetDirectoryName(facePath));
                cropped.Save(facePath, new PngEncoder());

                model.Faces.Add(new FaceInfo
                {
                    Glasses = face.FaceAttributes.Glasses.ToString(),
                    Pitch = face.FaceAttributes.HeadPose.Pitch,
                    Roll = face.FaceAttributes.HeadPose.Roll,
                    Yaw = face.FaceAttributes.HeadPose.Yaw,
                    Accessories = face.FaceAttributes.Accessories?.Select(a =>
a.Type.ToString()).ToList() ?? new(),
                    BlurLevel = face.FaceAttributes.Blur.BlurLevel.ToString(),
                    BlurValue = face.FaceAttributes.Blur.Value,
                    ExposureLevel =
face.FaceAttributes.Exposure.ExposureLevel.ToString(),
                    ExposureValue = face.FaceAttributes.Exposure.Value,
                    NoiseLevel =
face.FaceAttributes.Noise.NoiseLevel.ToString(),
                    NoiseValue = face.FaceAttributes.Noise.Value,
                    FaceImageUrl = $"/images/faces/{faceFileName}"
                });

                faceIndex++;
            }
            return View(model);
        }
    }
```

Рисунок 9.2 – Контролери.

Створимо відображення.

1. OCR

```
@model lab6.Models.OCRResultModel

<div class="container py-4">
    <h2 class="mb-4 text-center">Text Recognition</h2>

    <form asp-action="Index" method="post" enctype="multipart/form-data"
class="card shadow-sm p-4 mb-5">
        <div class="mb-3">
            <label for="file" class="form-label">Upload an image:</label>
            <input type="file" class="form-control" id="file" name="file" />
        </div>
        <button type="submit" class="btn btn-success">Analyze</button>
    </form>

    @if (!string.IsNullOrEmpty(Model.FullText))
    {
        <h3 class="mb-4 text-center">Recognition Result</h3>

        <div class="card shadow-sm">
            <div class="row g-0">
                <div class="col-md-5 d-flex align-items-center justify-
content-center p-3">
                    <img src="@Model.ImagePath" alt="Uploaded image"
class="img-fluid rounded border" style="max-height: 250px;" />
                </div>
                <div class="col-md-7">
                    <div class="card-body">
                        <h5 class="card-title">Extracted Text</h5>
                        <p class="card-text">@Model.FullText</p>
                    </div>
                </div>
            </div>
        </div>
    }
</div>
```

2. Image Analysis

```
@model lab6.Models.ImageAnalysisResultModel

<div class="container py-4">
    <h2 class="mb-4 text-center">Image Analysis</h2>

    <form asp-action="Index" method="post" enctype="multipart/form-data"
class="card shadow-sm p-4 mb-5">
        <div class="mb-3">
            <label for="file" class="form-label">Upload an image:</label>
            <input type="file" class="form-control" id="file" name="file" />
        </div>
        <button type="submit" class="btn btn-success">Analyze</button>
    </form>

    @if (Model != null && !string.IsNullOrEmpty(Model.ImagePath))
    {
        <div class="card shadow-sm p-4">
            <div class="text-center mb-4">
                <img src="@Model.ImagePath" alt="Analyzed image" class="img-
fluid rounded border" style="max-height: 500px; max-width: 100%;" />
            </div>

            <h4 class="mb-3 text-center">Analysis Results</h4>
```

```
                <div class="mb-3">
                    <strong>Description:</strong>
                    <p>@Model.Description</p>
                </div>

                @if (Model.Categories.Any())
                {
                    <div class="mb-3">
                        <strong>Categories:</strong>
                        <p>@string.Join(", ", Model.Categories)</p>
                    </div>
                }

                @if (Model.Tags.Any())
                {
                    <div class="mb-3">
                        <strong>Tags:</strong>
                        <p>@string.Join(", ", Model.Tags)</p>
                    </div>
                }

                @if (Model.Objects.Any())
                {
                    <div class="mb-0">
                        <strong>Detected Objects:</strong>
                        <p>@string.Join(", ", Model.Objects)</p>
                    </div>
                }
            </div>
        }
    </div>
```

3. Face API

```
@model lab6.Models.FaceDetectionResultModel

<div class="container py-4">
    <h2 class="mb-4 text-center">Face Detection</h2>

    <form asp-action="Index" method="post" enctype="multipart/form-data"
class="card shadow-sm p-4 mb-5">
        <label for="file" class="form-label">Upload an image:</label>
        <input type="file" class="form-control mb-3" id="file" name="file"
/>
        <button type="submit" class="btn btn-success">Analyze</button>
    </form>

    @if (!string.IsNullOrEmpty(Model.Message))
    {
        <div class="alert alert-info shadow-sm text-center">
            <strong>@Model.Message</strong>
        </div>
    }

    @if (Model.Faces != null && Model.Faces.Any())
    {
        <h3 class="mb-4 text-center">Analysis Results</h3>

        <div class="row row-cols-1 row-cols-md-2 g-4">
            @foreach (var face in Model.Faces)
            {
                <div class="col">
                    <div class="card shadow h-100 p-3 d-flex flex-row">
                        @if (!string.IsNullOrEmpty(face.FaceImageUrl))
                        {
```

```
                                <img src="@face.FaceImageUrl" alt="Face"
        class="img-thumbnail me-3" style="max-width: 100px; max-height: 100px;" />
                            }

                            <div>
                                <h5 class="card-title">Face
        @(Model.Faces.IndexOf(face) + 1)</h5>
                                <p><strong>Glasses:</strong> @face.Glasses</p>

                                <strong>Head Pose:</strong>
                                <ul class="ps-3 mb-2">
                                    <li>Pitch: @face.Pitch°</li>
                                    <li>Roll: @face.Roll°</li>
                                    <li>Yaw: @face.Yaw°</li>
                                </ul>

                                <strong>Additional Attributes:</strong>
                                <ul class="ps-3 mb-0">
                                    <li>Accessories: @string.Join(", ",
        face.Accessories)</li>
                                    <li>Blur: @face.BlurLevel (value:
        @face.BlurValue)</li>
                                    <li>Exposure: @face.ExposureLevel (value:
        @face.ExposureValue)</li>
                                    <li>Noise: @face.NoiseLevel (value:
        @face.NoiseValue)</li>
                                </ul>
                            </div>
                        </div>
                    </div>
                }
            </div>
        }
        </div>
```

Рисунок 9.3 – Відобарження.

Додамо ключі в appsettings.json.

```
"ComputerVision": {
  "Endpoint": "",
  "Key": ""
},
"FaceApi": {
  "Endpoint": "",
  "Key": ""
},
```

Рисунок 9.4 – Ключі.

Завдання 9.3: Перевірка працездатності.

Face Detection

Upload an image:

Вибрати файл | Файл не вибрано

Analyze

Analysis Results

Face 1

Glasses: ReadingGlasses

Head Pose:

- Pitch: 1,1°
- Roll: -3,1°
- Yaw: 2,5°

Additional Attributes:

- Accessories: Glasses
- Blur: Low (value: 0)
- Exposure: GoodExposure (value: 0,58)
- Noise: Low (value: 0)

Face 2

Glasses: NoGlasses

Head Pose:

- Pitch: -3,7°
- Roll: -10,3°
- Yaw: -8,2°

Additional Attributes:

- Accessories:
- Blur: Low (value: 0,05)
- Exposure: GoodExposure (value: 0,68)
- Noise: Low (value: 0)

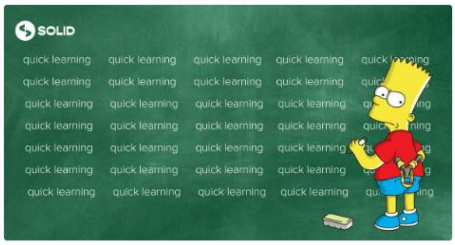
Text Recognition

Upload an image:

Вибрати файл | Файл не вибрано

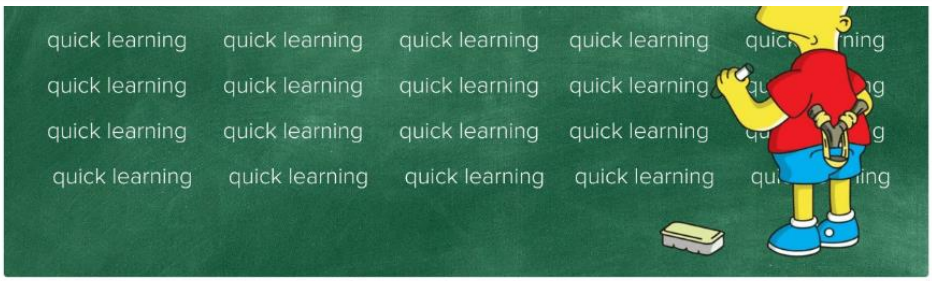
Analyze

Recognition Result



Extracted Text

SOLID quick learning quick learning quick learning quick learning quick learning quick learning quick learning quick learning quick learning quic' quick learning quick learning quick learning quick learning Ing quick learning quick learning quick learning quick learning quich Ining quick learning quick learning quick learning quick learning g quick learning quick learning quick learning g quick learning quick learning quick learning quick learning qui ling



Analysis Results

Description:
text (89,14%)

Categories:
others_, text_, text_menu

Tags:
text, illustration, cartoon

Рисунок 9.5 – Результати.

Висновок: Під час виконання лабораторної робити я доповнив створений на попередньому занятті Web-застосунок новими сторінками, на яких користувачу пропонується проаналізувати візуальну інформацію різними способами.