Azure Cognitive Service Workshop @ SP

Artificial Intelligence (AI) is becoming a trending in recent years. It has been used in IT industry, Logistic industry, LawTech, FinTech, EduTech and even Health Care. In line with Singapore's "Smart Nation" initiative, this workshop meant to provide students some exposures to AI technologies and allowing basic practises by developing web application that will utilize the services.

Pre-requisites:

- Microsoft Visual Studio 2017 (VS2017) FREE Community Version * or Professional/Enterprise version installed with:-
  - ASP.NET & Web Development,
  - .NET Core cross platform development,
  - NodeJS Development,
  - .NET Desktop Development
- Microsoft .NET Core 2.1.403 **
- A Microsoft/Hotmail Account ** for Azure Pass Activation later

** If you already have Professional or Enterprise version of VS2017, you will not need to reinstall VS2017 Community version. Search & Run the "Visual Studio installer" to make sure that your VS2017 are already installed with all three-listed component.*

*** Compulsory to have*

Preparation

1. Download the “ACSW” Project from hmheng.github.com
2. Go to the “Before” folder and run the .sln file. It should open the project in Visual Studio. This is the default project generated using ASP.NET Core Angular Template with few utilities classes are being added.
3. Build and Run the project.

Get Started!

1. **Remove** all contents in ***home.component***

   *Head to **ClientApp/src/app/home/** and remove all the html content in home.component.html*

2. **Create** a new content for ***home.component*** with following code. This code includes some Angular variables.

```
<h1>Hello, world!</h1>
<div class="row">
  <div class="col-sm-7">
    <video #video id="video" width="640" height="480"
    muted="muted"></video>
  </div>
  <div class="col-sm-2">
    <div class="col-sm-12">
      <canvas #canvas id="canvas" width="640" height="480"
hidden>
      </canvas>
      <ul>
        <li *ngFor="let c of captures">
          <img src="{{ c }}" height="40" />
        </li>
      </ul>
    </div>
  </div>
  <div class="col-sm-3" style="background-color: #808080">
    <div class="col-sm-12">
      <!--<p>
        Scanned OCR: {{ ocrResult }}
      </p>
      <p>
        Translated: {{ translatedText }}
      </p>-->
    </div>
  </div>
</div>

<div><button id="translate" (click)="capture()">Translate
Text</button></div>
```

<video> tag is a HTML5 component that renders video or live video from Webcam that we will use later.

<canvas> tag allows us to render the image captured by Webcam. Hidden property will hide this from being displayed on UI

**ngFor is Angular directives for FOR loop.*

These two fields will be uncommented in steps 16.

A button click event that will trigger Webcam’s capture() function in home.component.ts later.

3. Create **Video** & **Canvas** elementRef and a **Captures** array in ***home.component.ts***

```
import { ViewChild, ElementRef } from '@angular/core';

export class HomeComponent {
  @ViewChild("video")
  public video: ElementRef;
  @ViewChild("canvas")
  public canvas: ElementRef;

  public captures: Array<any>;
}
```

ViewChild - The change detector looks for the first element or the directive matching the selector in the view DOM.

4. Create a **`ngAfterViewInit()`** function.

```
public ngAfterViewInit() {
    if (navigator.mediaDevices &&
navigator.mediaDevices.getUserMedia) {
      navigator.mediaDevices.getUserMedia({ video:
true }).then(stream => {
        this.video.nativeElement.src =
window.URL.createObjectURL(stream);
        this.video.nativeElement.play();
      });
    }
}
```

The navigator allows our angular application to access Webcam and output the video stream.

5. Create a **`capture`** function.

```
public capture() {

}
```

We will fill this in later.

6. Now, we need to create a ***config.ts*** typescript file under ***app*** that stores our API endpoints and API keys at once place.

```
export class Config {
  public static COGNITIVE_HOST = 'https://api.cognitive.microsoft.com/';

  public static OCR_API_ENDPOINT =
'https://southeastasia.api.cognitive.microsoft.com/vision/v2.0/ocr'; //
https://[location].api.cognitive.microsoft.com/vision/v1.0/ocr[?language]
[&detectOrientation ]
  public static OCR_API_KEY = '';

  public static TRANSLATE_API_ENDPOINT =
'https://api.cognitive.microsofttranslator.com/translate?api-version=3.0';
  public static TRANSLATE_API_KEY = '';
}
```

7. Then, create a folder ***models*** under ***app***. We will place all the model classes under this folder.
8. Create an ***OcrModel.ts*** under ***models*** folder will following interfaces.

```
class Word {
  boundingBox: number[];
  text: string;
}
```

```
class Line {
  boundingBox: number[];
  text: string;
  words: Word[];
}

class Region {
  boundingBox: string;
  lines: Line[];
}

export class OcrResponse {
  language: string;
  textAngle: number;
  orientation: string;
  regions: Region[];
}
```

9. Create another model class ***TranslateModel.ts***

```
export class TranslateRequest {
  Text : string;
}

export class TranslateResponse {
  detectedLanguage: DetectedLanguage;
  translations: Translation[];
}

class DetectedLanguage {
  language: string;
  score: number;
}

class Translation {
  text: string;
  to: string;
}
```

10. Create a folder ***services*** under ***app***. We will place all the services under this folder later.
11. Create an ***OcrService.ts*** file under ***services*** folder. This injectable service will help to call the Microsoft Cognitive Services' OCR API.

```
import { Component, Inject, Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { OcrResponse } from '../models/OcrModel';
import { Config } from '../config';
import { Observable } from 'rxjs/Observable';


const httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/octet-stream',
    'Ocp-Apim-Subscription-Key': Config.OCR_API_KEY
  })
};

@Injectable()
export class OcrService {
  private response: OcrResponse;
  private _httpClient: HttpClient;

  constructor(http: HttpClient, @Inject('BASE_URL') baseUrl: string) {
    this._httpClient = http;
  }

  public CallOcrAPI(blob: any, language?: string): any {
    return this._httpClient.post<OcrResponse>((Config.OCR_API_ENDPOINT), blob,
httpOptions);
  }
}
```

12. Create another service for calling Microsoft's Translation API.

```
import { Component, Inject, Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { TranslateRequest, TranslateResponse } from '../models/TranslateModel';
import { Config } from '../config';
import { Observable } from 'rxjs/Observable';


const httpOptions = {
  headers: new HttpHeaders({ 'Content-Type': 'application/json', 'Ocp-Apim-Subscription-
Key': Config.TRANSLATE_API_KEY })
};

@Injectable()
export class TranslateService {
  private response: TranslateResponse;
  private _httpClient: HttpClient;

  constructor(http: HttpClient, @Inject('BASE_URL') baseUrl: string) {
    this._httpClient = http;
  }

  public CallTranslateAPI(text: string, language: string): any{
    var translateRequests: TranslateRequest[] = [];
    var request = new TranslateRequest;
    request.Text = text;
    translateRequests.push(request);
    return this._httpClient.post<TranslateResponse>((Config.TRANSLATE_API_ENDPOINT +
'&to=' + language), translateRequests, httpOptions);
  }
}
```

13. Finally, in order to make injectable services work, we have to register at ***app.module.ts***. Under providers section, let's include both ***OcrService***. and ***TranslateService***.

```
import { TranslateService } from
'./services/TranslateService';
import { OcrService } from './services/OcrService';

. . .

providers: [TranslateService, OcrService],

. . .
```

14. Browse back to ***home.component.ts*** to create new properties and function.

First imports all services and models.

```
import { TranslateService } from
'../services/TranslateService';
import { TranslateResponse } from
'../models/TranslateModel';
import { OcrService } from '../services/OcrService';
import { OcrResponse } from '../models/OcrModel';

public language: string = "zh";
public ocrResult: string;
public translatedText: string;
```

Initialize all the services & array.

```
export class HomeComponent {

  constructor(private translateService: TranslateService,
private ocrService: OcrService) {
    this.captures = [];
}
```

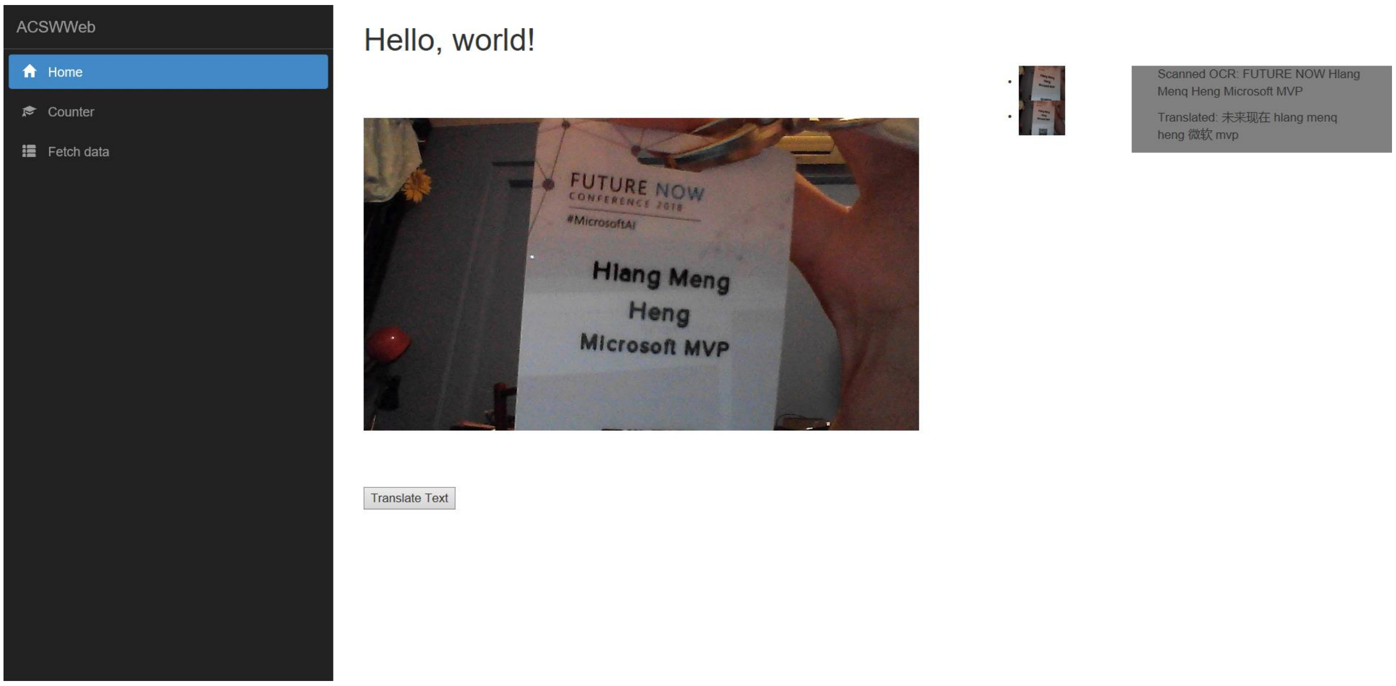
Create function for translation.

```
  translate(text:string) {
    this.translateService.CallTranslateAPI(text,
this.language).subscribe(result => {
      this.translatedText =
result[0].translations[0].text;
    }, error => console.error(error)
    );
 }
}
```

15. Continue writing the **capture** function.

```
public capture() {
    var context =
this.canvas.nativeElement.getContext("2d").drawImage(this.video.nativeElement, 0, 0, 640,
480);
    var url = this.canvas.nativeElement.toDataURL("image/jpg");
    this.captures.push(url);
    var blob = BlobUtilties.makeblob(url);
    var message = "";
    this.ocrService.CallOcrAPI(blob, "").subscribe((ocrResponse: OcrResponse)=> {
      //this.ocrResult = JSON.stringify(result);
      if (ocrResponse && ocrResponse.regions) {
        for (let region of ocrResponse.regions) {
          for (let line of region.lines) {
            for (let word of line.words) {
              message += word.text + " ";
            }
          }
        }
        this.ocrResult = message;
        this.translate(message);
      }
  });
}
```

16. Now, go back ***<u>home.component.html</u>*** to uncomment the line for ***ocrResult*** and ***translatedText.***
17. ***Try to build and run now!***



Hello, world!

Scanned OCR: FUTURE NOW Hlang Menq Heng Microsoft MVP

Translated: 未来现在 hlang menq heng 微软 mvp