Assignment 8: User Management (UM)

Web Intermediate Course

AtlasX Web Application & Web Service Interfacing

- AtlasX Web Application Template มีคลาส AxRequestService เป็น injectable service ที่ประกอบด้วย method สำหรับเรียก API ให้ใช้งาน โดย method หลักๆ ที่สามารถใช้งานได้ ได้แก่
  - AxRequestService.request() สำหรับเรียก API endpoint ทั่วไป อันไหนก็ได้
  - AxRequestService.sp() สำหรับเรียก Stored Procedure ผ่าน /api/appdata โดยตรง
- การตั้งค่า Web Service URL สามารถตั้งค่าได้ที่ไฟล์ environment.ts (local) และไฟล์ environment.prod.ts (production)

```
...

export const environment: ApplicationConfig & AxAuthenticationConfig = {
    production: false,
    webServiceUrl: 'https://atlasx.cdg.co.th/axws-demo',
    ...
}
```

AxRequestService

- ก่อนเริ่มทำการใช้งาน ให้ import และ inject AxRequestService ไปที่ constructor() ของ component ที่จะเรียกใช้ก่อน
- เพื่อความสะดวกในการเรียกใช้งาน ควร inject Web Service URL ที่ตั้งค่าไว้ใน environment เข้ามาใช้งานด้วย โดยการ @Inject(AxWebServiceUrl) เข้ามาที่ component

```
import { AxRequestService, AxWebServiceUrl } from '@atlasx/core/http-service'
...

export class SomeComponent {
  constructor(
    private axRequestService: AxRequestService,
    @Inject(AxWebServiceUrl) private webServiceUrl: string
  ) {}
  ...
}
```

AxRequestService.request()

- ใช้ AxRequestService.request(url, method, body, params) เพื่อเรียก API พร้อมทั้งระบุ body (กรณีที่เรียกด้วย method POST) หรือ params (กรณีที่เรียกด้วย method GET)

```
this.requestService.request(
    this.webServiceUrl + '/api/apptest',
    'GET',
    {}, // POST body is not required since the endpoint is called with GET
    {
        name: 'John Doe',
    }
).subscribe((response) => { console.log(response) });
```

AxRequestService.sp()

- ใช้ AxRequestService.sp(spName, method, body, params) เพื่อเรียก Stored Procedure ผ่าน WebServiceUrl/api/appdata โดยตรง (ไม่ต้องต่อ URL เอง) โดยระบุชื่อของ Stored Procedure ที่ต้องการเรียก, method ที่จะเรียก (รองรับ GET/POST) และ POST body หรือ GET params ตาม method ที่เลือก

```
this.requestService.sp(
    ‘TEMP_USER_Q',
    'GET',
    {}, // POST body is not required since the endpoint is called with GET
    {} // No GET params required in this case
).subscribe((response) => { console.log(response) });
```

การสร้าง API Endpoint เองใน AtlasX Web Service

- ในกรณีที่มี requirement เพิ่มเติม และต้องมีการสร้าง API Endpoint ใหม่ เราสามารถทำได้โดยการเพิ่ม controller ใหม่ไปที่โฟลเดอร์ Controllers และตั้งชื่อไฟล์เป็น <<ชื่อ Controller>>Controller.cs เช่น AppTestController.cs พร้อมระบุโครงสร้างภายในเบื้องต้นดังนี้

```
using Microsoft.AspNetCore.Mvc;

// This will be the endpoint to be called.
// In this case is <web-service-url>/api/apptest
[Route("api/[controller]")]
[ApiController]
public class AppTestController : Controller
{
    public AppTestController() // Constructor's name must be the same as Controller's
    {
    }
    ...
}
```

การสร้าง API Endpoint เองใน AtlasX Web Service (ต่อ)

- สามารถระบุ HTTP method ที่สามารถเข้าถึง endpoint แต่ละอันได้โดยการใส่ [HttpGet] หรือ [HttpPost] ไว้ก่อนชื่อ method
- หากไม่ได้ระบุ segment ใดๆ ต่อจาก API Endpoint ตัวหลัก request จะวิ่งไปที่ Index()
- ใช้ return Ok(response) เพื่อส่ง response กลับไป

```
// <web-service-url>/api/apptest
[Route("api/[controller]")]
[ApiController]
public class AppTestController : Controller
{
    ...
    // GET <web-service-url>/api/apptest
    [HttpGet]
    public IActionResult Index()
    {
        return Ok("Test"); // Return response as plain text
    }
    ...
}
```

การสร้าง API Endpoint เองใน AtlasX Web Service (ต่อ)

- ใช้ [Route("<segment-name>")] เพื่อระบุ path ของ API ที่ต้องการให้เรียกได้
- ใช้คลาส QueryParameter ที่อยู่ใน AtlasX.Engine.Connector เพื่อรับ parameter ต่างๆ (ทั้งจาก GET และ POST) เข้ามาใช้งานต่อใน method
- ใช้ return BadRequest(message) เพื่อส่ง response เป็น HTTP 400 Bad Request ได้ กรณีที่มีการส่งค่า parameter มาไม่ครบ หรือไม่ถูกต้อง
- ใช้คลาส Dictionary<string, object> เพื่อสร้าง response ให้อยู่ในรูปแบบ key-value pair ก่อนส่งค่ากลับ จะได้ response ออกมาในรูปของ JSON object ที่ง่ายต่อการใช้งาน

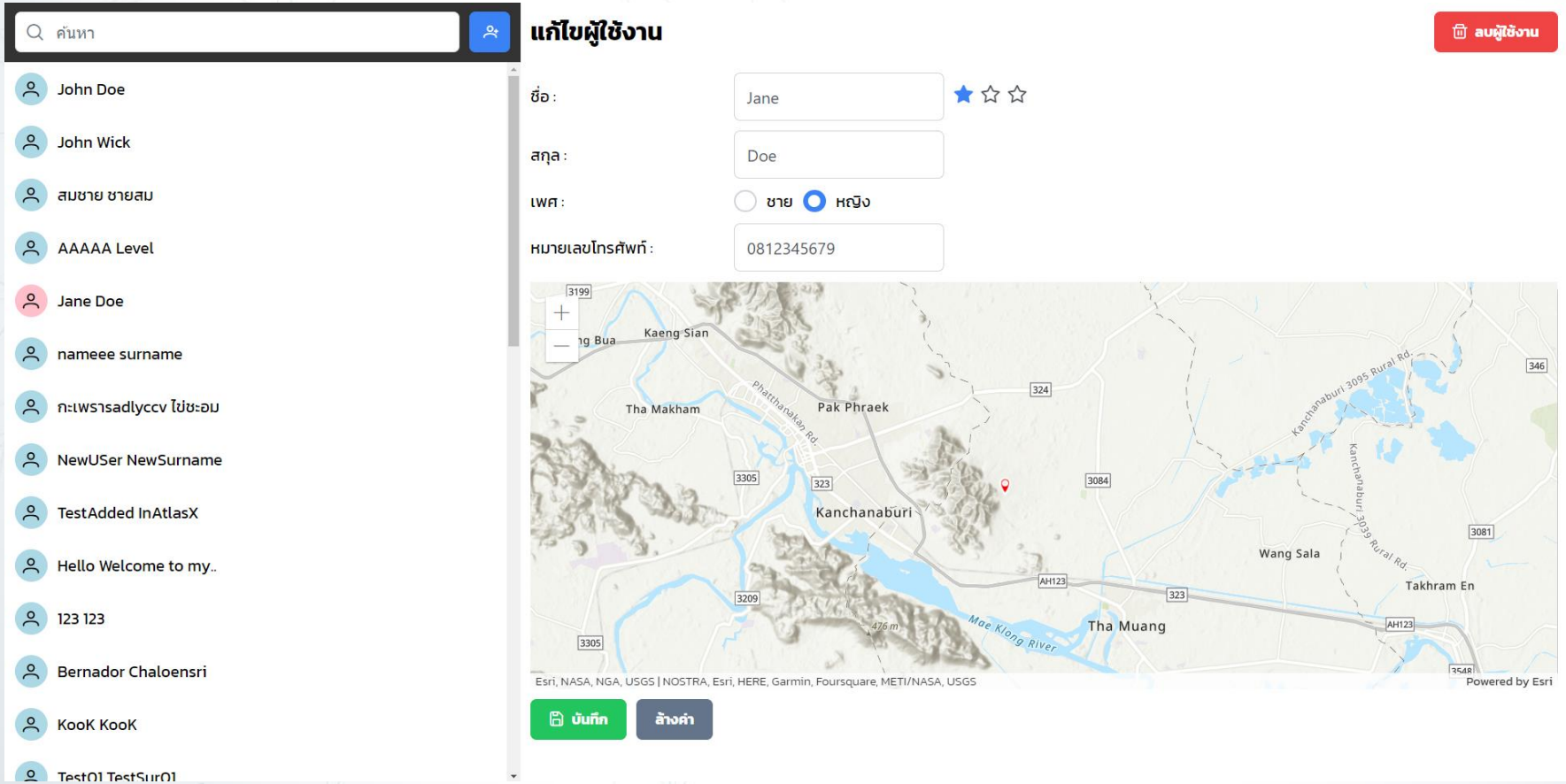
การสร้าง API Endpoint เองใน AtlasX Web Service (ต่อ)

```
...
using AtlasX.Engine.Connector;
using System.Collections.Generic;

// <web-service-url>/api/apptest
[Route("api/[controller]")]
[ApiController]
public class AppTestController : Controller
{
    ...
    // POST <web-service-url>/api/apptest/hello
    [HttpPost]
    [Route("hello")]
    public IActionResult Hello()
    {
        var queryParameter = new QueryParameter(Request);
        if(queryParameter["name"] is null){ // If no parameter “name” exists in the POST body
            return BadRequest("Missing parameter: name");
        }
        string name = queryParameter["name"].ToString();
        var response = new Dictionary<string, object>();
        response.Add("success", true);
        response.Add("message", "Hello, " + name + "!");
        return Ok(response);
    }
    ...
}
```

Assignment 8

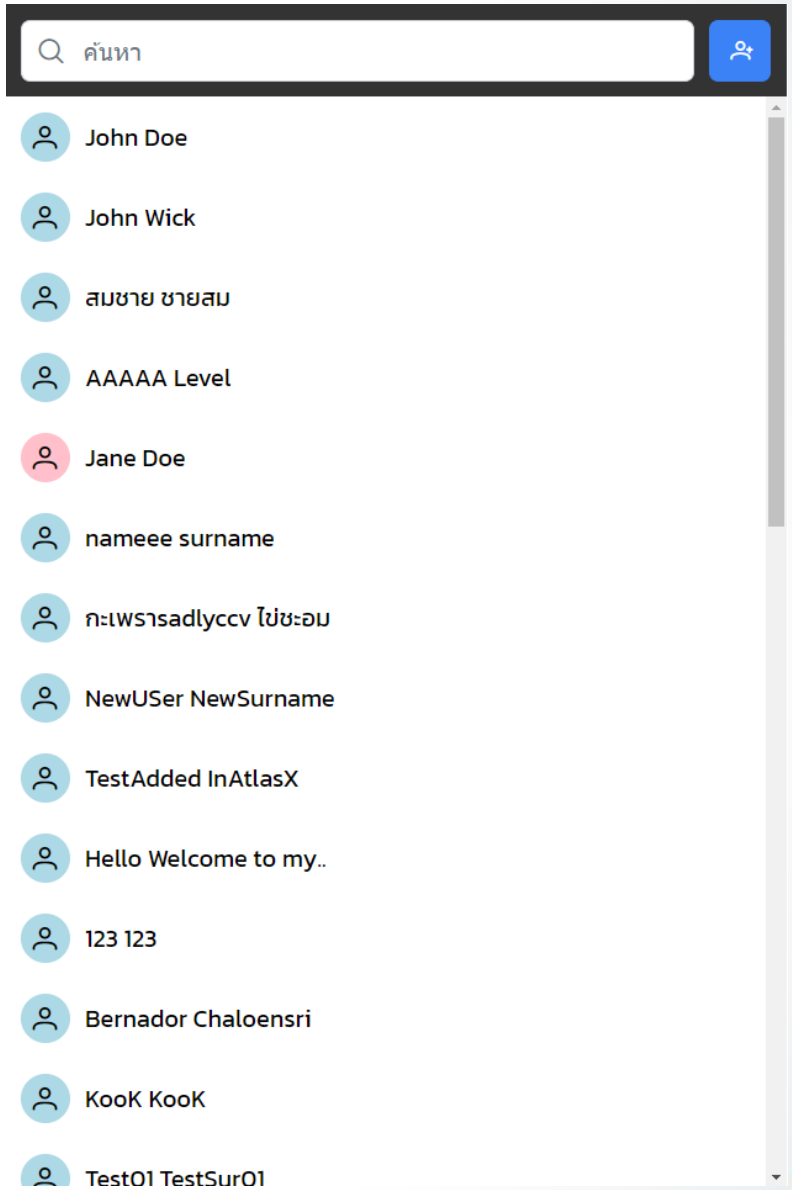
จงสร้าง Web Page ต่อไปนี้ด้วย AtlasX Web Application โดยมี UI เป็นดังนี้



Assignment 8 (ต่อ)

Panel ด้านซ้าย ประกอบด้วย

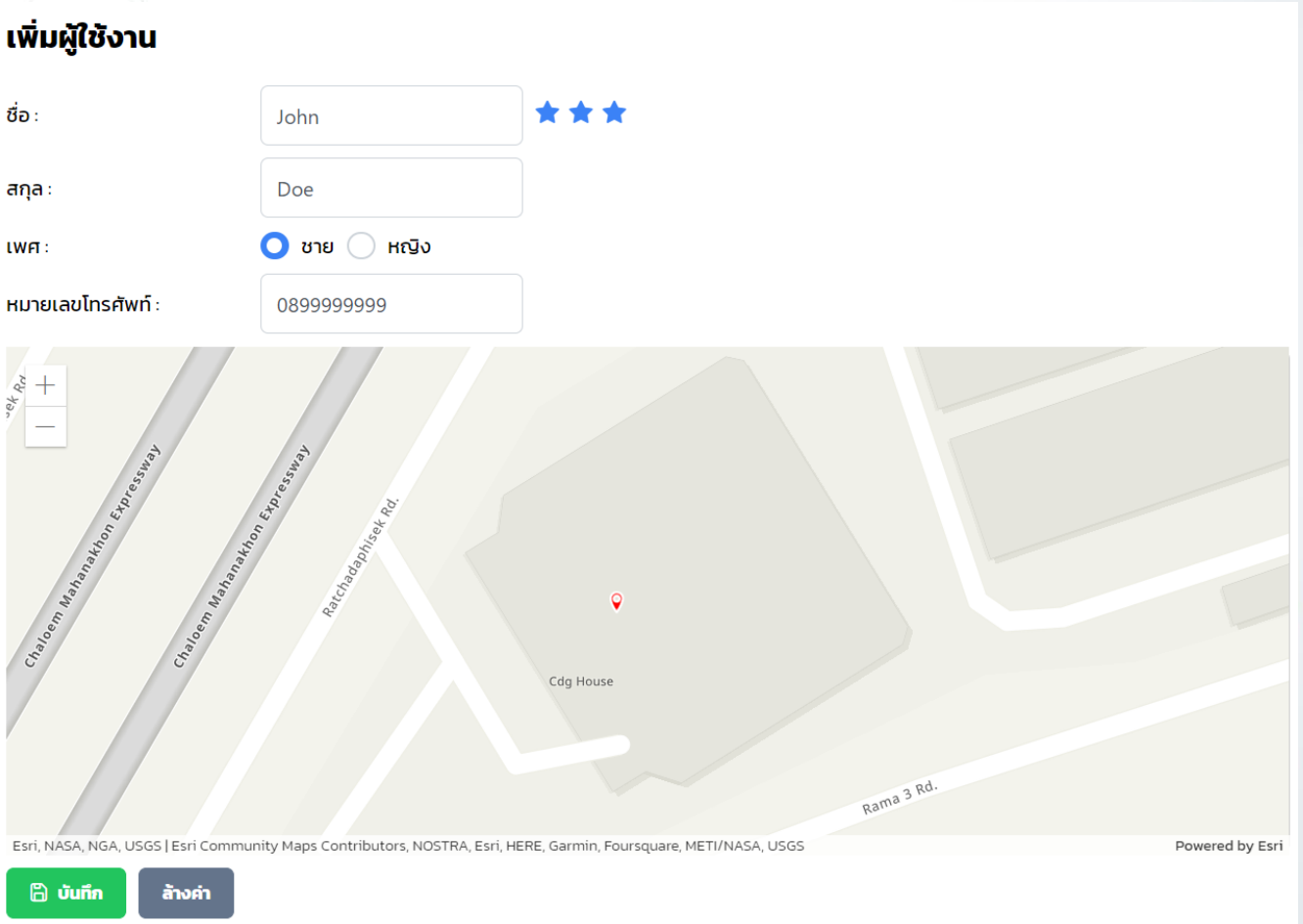
- Search Bar ด้านบน ประกอบด้วย Text Input สำหรับพิมพ์คำค้นหา และปุ่มสำหรับ Add User
- แสดงรายชื่อ User ทั้งหมดที่ดึงได้จาก
  SP: TEMP_USER_Q
  Input Parameters: ไม่มี
- User เพศหญิง (GENDER = 'F') ให้แสดง Avatar สีชมพู ส่วนเพศชาย หรือค่าอื่นๆ ให้แสดง Avatar สีฟ้า
- เมื่อคลิกที่ชื่อ User คนใด ให้นำข้อมูลของ User คนนั้นไปใส่ในแบบฟอร์มที่ Panel ด้านขวา
- เมื่อพิมพ์ค้นหาที่ช่องค้นหา ให้ทำการ Filter รายชื่อ User ให้เหลือเฉพาะที่ชื่อ – สกุล match กับคำค้นหาเท่านั้น
- เมื่อคลิกที่ปุ่มเพิ่มผู้ใช้งาน ให้ทำการ Reset Form ที่ Panel ทางขวาให้ผู้ในโหลดสร้างผู้ใช้งานใหม่



Assignment 8 (ต่อ)

Panel ด้านขวา ประกอบด้วย

- แบบฟอร์มสำหรับกรอกรายละเอียด User ได้แก่ ชื่อ (NAME) สกุล (SURNAME) เพศ (GENDER) และหมายเลขโทรศัพท์ (MOBILE)
- เมื่อพิมพ์ชื่อเสร็จ ให้ทำการเรียก API /api/apphoro ที่เขียนขึ้นเอง จากนั้นนำเกรดที่ได้จาก response มาแสดงรูปดาวด้านหลังชื่อ
- แสดงแผนที่สำหรับเลือกพิกัด LONGITUDE และ LATITUDE ของ User โดยการคลิกลงไปบนแผนที่เพื่อเลือกพิกัด พร้อมทั้งแสดง Marker
- ปุ่ม บันทึก สำหรับบันทึกค่าไปยัง SP: TEMP_USER_I
- ปุ่ม ล้างค่า สำหรับล้างแบบฟอร์มทั้งหมด พร้อมทั้งลบหมุดบนแผนที่



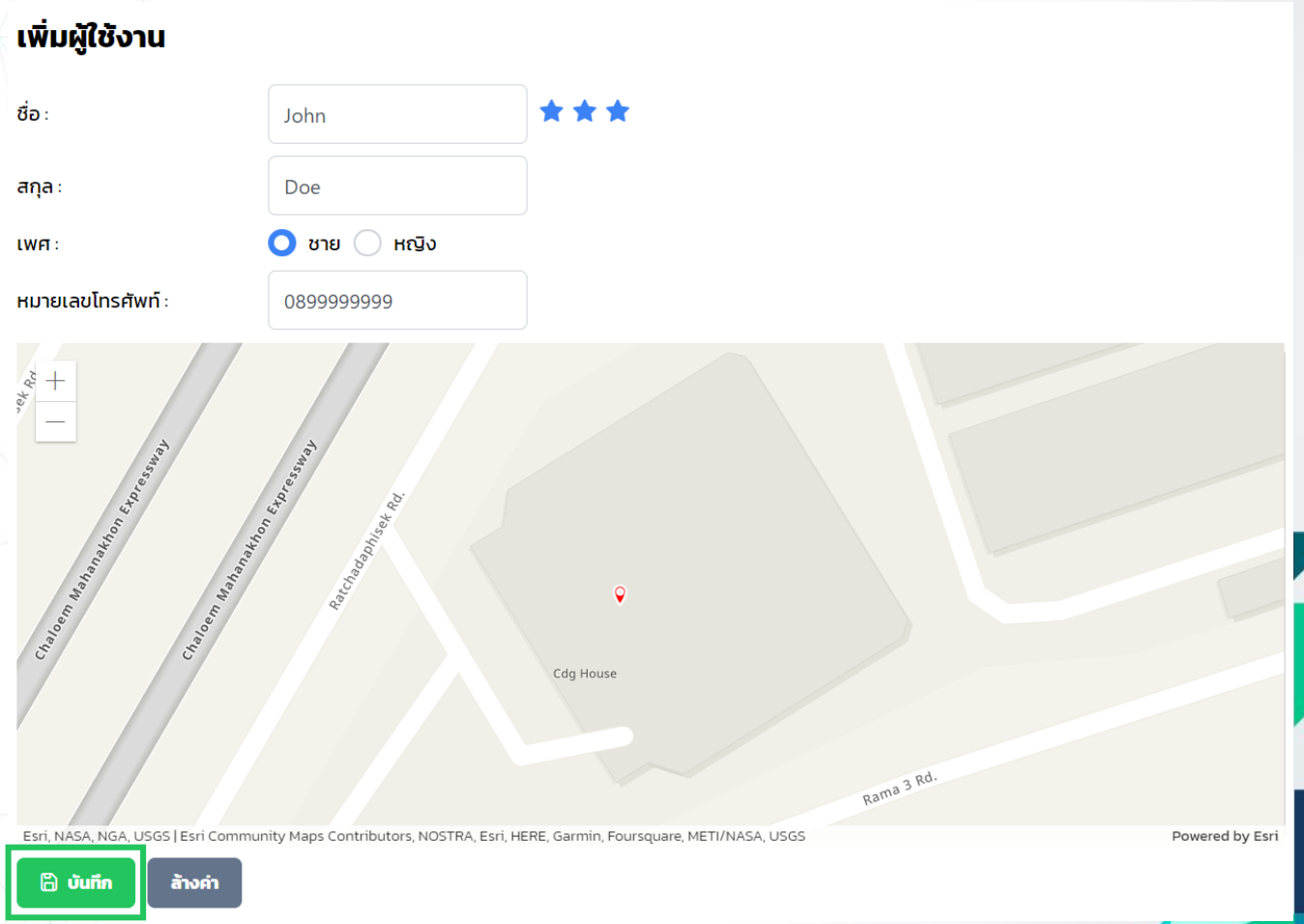
Assignment 8 (ต่อ)

กรณีของการเพิ่มผู้ใช้งานใหม่ ให้ส่งค่าไปที่ SP ต่อไปนี้

SP: TEMP_USER_I

Input Parameters:

- NAME (string)
- SURNAME (string)
- GENDER (string)
  (M = ชาย, F = หญิง)
- MOBILE (string)
  (Format: 0XXXXXXXXX)
- LONGITUDE (string)
- LATITUDE (string)



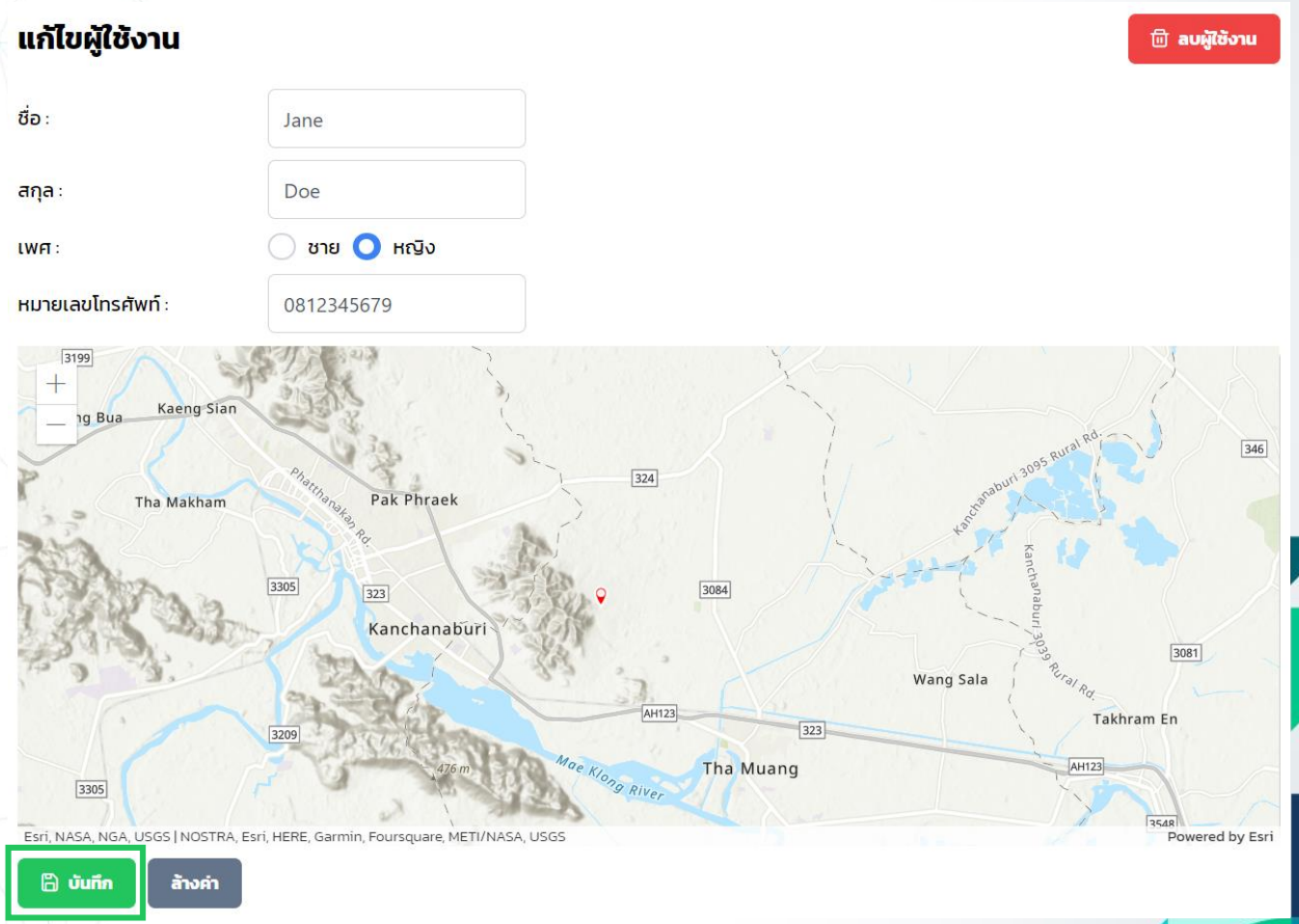
Assignment 8 (ต่อ)

กรณีของการแก้ไขผู้ใช้งานที่มีอยู่เดิม ให้ส่งค่าไปที่ SP ต่อไปนี้

SP: TEMP_USER_U

Input Parameters:

- USER_ID (number)
- NAME (string)
- SURNAME (string)
- GENDER (string)
  (M = ชาย, F = หญิง)
- MOBILE (string)
  (Format: 0XXXXXXXXX)
- LONGITUDE (string)
- LATITUDE (string)

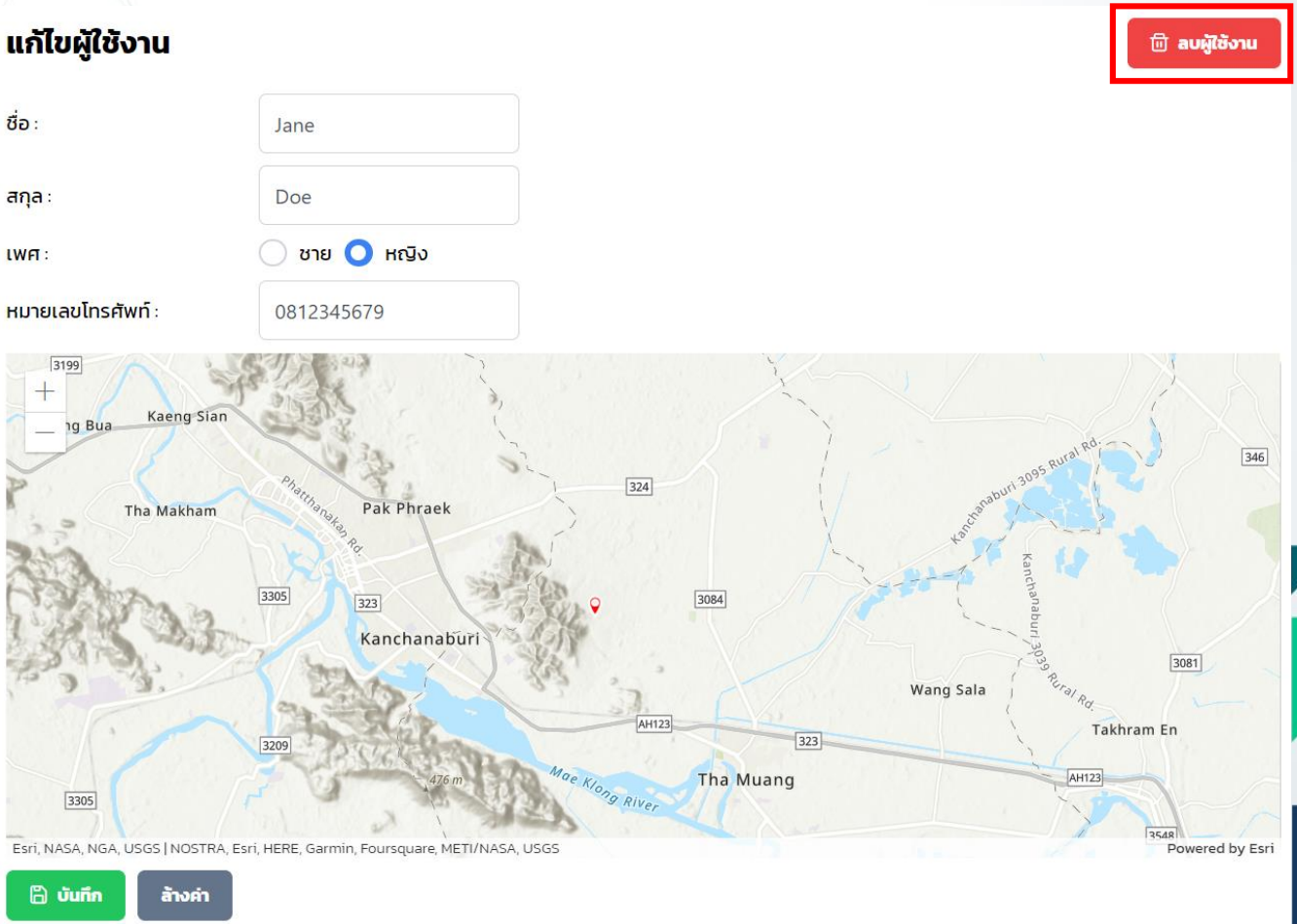


Assignment 8 (ต่อ)

เมื่อคลิกที่ปุ่ม ลบผู้ใช้งาน ให้ส่งค่าไปที่ SP ต่อไปนี้

SP: TEMP_USER_D

Input Parameters:
- USER_ID (number)



Assignment 8 (ต่อ)

ที่ AtlasX Web Service ให้สร้าง Controller ใหม่ชื่อ AppHoroController.cs โดยมีรายละเอียดดังนี้

- สร้าง Index() ให้สามารถเรียก API ผ่าน endpoint https://localhost:5001/api/apphoro ด้วย method GET ได้
- ภายใน Index() ให้รับ Parameter ชื่อ name มาจาก Request จากนั้นนำอักขระแต่ละตัวใน name มาหาผลรวมตามรหัส ASCII ของอักขระ จากนั้น mod (%) ผลรวมที่ได้ด้วย 10
- ตีความค่าคะแนนเป็นเกรดของชื่อ โดยให้ผลลัพธ์ 0 – 3 = “bad”, 4 – 6 = “so so”, 7 – 9 = “good”
- return Response กลับไปเป็นผลลัพธ์ที่เป็นเกรด เพื่อนำไปแสดงเป็นรูปดาวที่ฝั่ง Front-end (“bad” = 1 ดวง, “so so” = 2 ดวง, “good” = 3 ดวง)

Hint

- ใช้ Array.filter() เพื่อทำการกรองผลการค้นหา
- เมื่อมีการเพิ่ม/ลบ/แก้ไขชื่อ-สกุล User ควรอัพเดต User ในรายการทุกครั้ง
- ควรใช้ Reactive Forms ของ Angular ในการผูกกับ <form> เพื่อให้ง่ายต่อการ validate ค่า โดยสามารถศึกษาเพิ่มเติมได้จาก https://angular.io/guide/reactive-forms
- กรณีที่ไม่ได้กรอกค่า หรือกรอกค่าไม่ถูกต้อง ควรมีการ highlight สีแดงในฟิลด์ที่กรอกไม่ถูกต้อง โดยตรวจสอบสถานะได้จาก formGroup.controls['ชื่อฟิลด์'].valid
- เมื่อมีการ request SP สำเร็จหรือไม่สำเร็จ ควรแสดงข้อความให้ผู้ใช้งานทราบ (แนะนำ Toast ของ PrimeNG)
- เมื่อคลิกที่ปุ่มลบผู้ใช้งาน ควรมีการ Confirm กับผู้ใช้งานก่อนทำการลบ (แนะนำ ConfirmDialog ของ PrimeNG)
- รูปดาว จะใช้ Component Rating ของ PrimeNG ก็ได้ หรือใช้แค่ Prime Icon รูปดาวก็ได้

Hint (ต่อ)

- ในกรณีที่ไม่สามารถเชื่อมต่อ Database ด้วย AtlasX Web Service ที่รันบน Local ให้เปลี่ยน web service URL ให้ยิงไปที่ https://atlasx.cdg.co.th/axws-demo เพื่อเรียกใช้งาน SP ส่วน /api/apphoro ที่เขียนเอง ให้ยิงไปที่ Local
- ศึกษาการสร้าง Web API ด้วย .NET Core ได้ที่ https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-5.0
- รับค่า Parameter จาก Request โดยใช้
  `var queryParameter = new QueryParameter(Request);`
  จากนั้นเข้าถึงด้วย
  `queryParameter["key"].ToString();`
- ในกรณีที่อยากได้ Response จาก Web Service เป็น JSON object แนะนำให้ปั้น response เป็น Dictionary ก่อน return Ok() กลับมา
- ในกรณีที่ไม่ได้ส่ง Parameter ที่จำเป็นต้องใช้มา ควร return BadRequest("Error message") กลับไปหา Client

Reference

- https://angular.io/guide/reactive-forms
- https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-5.0

End of Assignment