แอปพลิเคชันติดตามยอดจำนวน COVID–19 อย่างง่าย

รายวิชาวิทยาการคำนวณ (Computer Science)

เสนอ

มิสกัญญา สุนทรธัย

ผู้จัดทำ

วริศ ศรีปทุมรักษ์ ม.4/1 เลขที่ 9

ภาคเรียนที่ 1 ปีการศึกษา 2564

โรงเรียนเซนต์คาเบรียล

คำนำ

รายงาน “แอปพลิเคชันติดตามยอดจำนวน COVID–19 อย่างง่าย” ฉบับนี้ เป็นส่วนหนึ่งของวิชาวิทยาการคำนวณ (Computer Science) ชั้นมัธยมศึกษาปีที่ 4 มีจุดประสงค์เพื่อศึกษาทฤษฎีของ IoT, แอปพลิเคชันที่ใช้ในการเขียนโปรแกรม, หลักการทำงานของตัวแอปพลิเคชันและ IoT และ การออกแบบ User Interface (UI) ของเป็นต้น ข้าพเจ้าหวังเป็นอย่างยิ่งว่าเนื้อหาในรายงานฉบับนี้ได้ถูกเรียบเรียงมาเป็นประโยชน์ต่อผู้สนใจเป็นอย่างดี หากมีสิ่งใดในรายงานฉบับนี้ที่ต้องปรับปรุง ข้าพเจ้าจะรับข้อชี้แนะและนำไปแก้ไขพัฒนาให้ถูกต้องสมบูรณ์ต่อไป

วริศ ศรีปทุมรักษ์ ม.4/1 เลขที่ 9
ผู้จัดทำ
5 กรกฎาคม 2564

สารบัญ

บทที่ 1
ที่มาและความสำคัญ

ปัจจุบันในปีค.ศ. 2020–2021 ที่ไม่ผ่านมานี้ มีโรคไวรัสระบาด COVID–19 จากประเทศจีน ทำให้ผู้คนจำนวนมากล้มป่วยเสียชีวิตจากโรคนี้ ต้องระมัดระวัง เว้นระยะห่าง ออกจากบ้านยามจำเป็น ป้องกันตัวเอง และติดตามข่าวสาร ซึ่งการติดตามข่าวสารเป็นหนึ่งในวิธีที่สำคัญในการทำให้ตัวเองปลอดภัยอย่างสายพันธุ์ใหม่ต่าง ๆ การล็อก-ดาวน์ ข่าวสารที่จำเป็น และจำนวนคนผู้ติดเชื้อ

เราจำเป็นต้องปรับตัวเข้ากับสถานการณ์ต่าง ๆ อย่างในสมัยนี้ที่เทคโนโลยีมีความก้าวหน้า ทันสมัย เข้าถึงได้อย่างง่ายดาย และมีความสำคัญสูงอย่างอินเทอร์เน็ต จึงต้องนำมันมาใช้ควบคู่กับแหล่งทรัพยากรข้อมูลที่มีอยู่ให้เกิดประโยชน์สูงสุด ด้วยเหตุนี้ การได้รับข่าวสารจึงเป็นสิ่งที่ง่ายและสามารถเข้าถึงได้เร็วกว่าเมื่อยังไม่มีการใช้บริการอินเทอร์เน็ต

ในที่นี้ การแสดงตัวเลขจำนวนคนผู้ติดเชื้อในแต่ละวันบน Apple Watch ทำให้ข้อมูลนี้เป็นสิ่งที่อยู่ใกล้ตัว และสามารถติดตามได้อยู่ตลอด สามารถเป็นการเตือนโดยนัยได้ว่าโดยนับวันตามตัวเลขและความเสี่ยงที่สูงขึ้นเรื่อย ๆ ทำให้ตระหนักรู้ถึงอันตรายที่อยู่ใกล้ตัวรวมถึงยังสร้างความสะดวกสบายให้กับผู้ใช้งานอีกด้วย โดยการใช้ IoT (Internet of Things; อินเทอร์เน็ตของสรรพสิ่ง) อำนวยความสะดวกในด้านนี้

ผู้จัดทำได้นำความรู้ที่ได้รับและศึกษามา มาประยุกต์ใช้ให้เกิดประโยชน์และประสิทธิภาพสูงสุดเท่าที่จะเป็นไปได้ในโครงงานนี้ด้วยเทคโนโลยีและข้อมูลที่ผู้จัดทำได้นำมาใช้งาน

1.1 ปัญหาที่พบ

บางครั้งเราอาจจะติดตามสถานการณ์ได้ไม่ทัน หรือ ไม่เป็นไปตามสถานการณ์ปัจจุบัน สามารถทำให้ตกข่าวไม่ทันการณ์ได้

1.2 ที่มาของปัญหา

สามารถมาจากหลายปัจจัย เช่น มีเวลาไม่พอ → ความยุ่ง → ความสะดวก–ไม่สะดวกในการรับข้อมูล เป็นต้น

1.3 รายละเอียดของปัญหา

มีเวลาไม่พอ

รีบ มีเวลาน้อย จำกัด และกระชั้นชิด จึงทำให้ไม่มีเวลาให้ความสำคัญกับสิ่งอื่น ๆ ที่สำคัญรองลงมา

ความยุ่ง

เป็นสิ่งที่ส่งผลมาจากการมีเวลาไม่พอ บางคนอาจมีงานที่จำเป็นต้องทำให้เสร็จภายในเวลาที่จำกัดอยู่มาก จึงให้ความสนใจกับงานนั้น ๆ จนจบก่อน จึงค่อยให้ความสำคัญกับงาน หรือ กิจกรรมอื่น ๆ ในภายหลัง

ความสะดวก-ไม่สะดวกในการรับข้อมูล

บางเวลา อุปกรณ์ในการที่เราจะสามารถรับข้อมูลได้อาจขาดแหล่งพลังงาน ไม่ว่างให้ใช้งาน หรือ อยู่ในระยะที่ไกลตัว

1.4 แนวทาง/วิธีการแก้ปัญหา

สร้างสิ่งที่ให้ข้อมูลเหล่านี้ อย่างในที่นี้คืออัตราการติดเชื้อ/หายติดเชื้อ/การตาย จาก ไวรัสโควิด–19 บน Apple Watch เนื่องจากเป็นสิ่งที่ผู้คนส่วนใหญ่ใช้ประกอบการทำงานในชีวิตประจำวันต่าง ๆ และอยู่ติดตัวตลอด

1.5 ชื่อหัวข้อโครงงาน

“แอปพลิเคชันติดตามยอดโควิด–19 อย่างง่าย”

1.6 วัตถุประสงค์ในการพัฒนาโครงงาน

- ทำให้ข้อมูลต่าง ๆ อย่างเช่น โควิด–19 เป็นสิ่งที่อยู่ใกล้ตัว
- สามารถติดตามข้อมูล COVID–19 ได้ทุกที่ทุกเวลา
- เพื่อผู้ที่มีเวลาน้อยและจำกัด

1.7 ขอบเขตการพัฒนาโครงงาน

- เป็นแอปพลิเคชันเกี่ยวกับอัตราการติดเชื้อ/หายติดเชื้อ/การตาย จาก ไวรัสโควิด–19
- สร้างขึ้นมาเพื่อ Apple Watch
- ควรอัปเดตข้อมูลได้ เพื่อให้ตรงตามปัจจุบัน

1.8 ระยะเวลาในการพัฒนาโครงงาน

19 วัน

บทที่ 2
หลักการ ทฤษฎี และงานที่เกี่ยวข้อง

2.1 แอปพลิเคชันที่ใช้ในการเขียนโปรแกรม

ไอคอนแอปพลิเคชัน Xcode

แอปพลิเคชันที่ใช้ในการเขียนโค้ดคือ Xcode ซึ่งเป็นแอปพลิเคชันของบริษัท Apple เพื่อให้ผู้ใช้ macOS พัฒนาซอฟต์แวร์สำหรับ macOS เอง รวมถึง iOS. iPadOS, tvOS, และ watchOS โดยภาษาที่ใช้พัฒนาซอฟต์แวร์เป็นของ Apple เช่นกันโดยมีชื่อว่า Swift

2.1.1 ตัวอย่างการเขียนโค้ดในภาษา Swift มีดังนี้:—

1. `print("Hello world!")` จะแสดงผลออกมาเป็นข้อความ: `Hello world!`

2.
```
var a:Int = 2
var b:Int = 3
b = 4                    มีการเปลี่ยนค่าตัวแปรทีหลังจาก 3 เป็น 4
print(a+b)               จะแสดงผลบวกออกมาเป็น 2 + 4 = 6
```

3.
```
let a = 1
if a < 4 {
print("True")
}
else if a > 8 {
print("False")
}                        จะแสดงผลออกมาเป็นข้อความ: True
```

2.2 หลักการการทำงานของโปรแกรม

หลักการการทำงานของโปรแกรมนี้ คือ ใช้ `URLRequest` ในการเรียกใช้งาน API (Application Programming Interface หรือ ส่วนต่อประสานโปรแกรมประยุกต์) ขอข้อมูลมาใช้ ซึ่งข้อมูลนั้นจะอยู่ในรูปแบบของ JSON (จาก JavaScript Object Notation, นามสกุลไฟล์ .json) แล้วใช้ฟังก์ชัน `do` ในการดึงข้อมูล JSON นั้นออกมาให้อยู่ในรูปข้อมูลเฉพาะ แล้วนำข้อมูลนั้นมาแทนค่าใน Label ที่อยู่บน UI ที่ทำไว้

2.3 แหล่งข้อมูลของ COVID–19

นำข้อมูล API มาจากเว็บไซต์ https://covid19.th-stat.com/th/api (แสดงค่าประจำวัน: https://covid19.th-stat.com/json/covid19v2/getTodayCases.json) โดยกรมควบคุมโรค

2.3.1 ตัวอย่างข้อมูล JSON ที่ได้


```
{"Confirmed":254515,"Recovered":205064,"Hospitalized":47481,"Deaths
":1970,"NewConfirmed":4662,"NewRecovered":2793,"NewHospitalized":18
33,"NewDeaths":36,"UpdateDate":"29\/06\/2021 12:13”,"DevBy":"https:
\/\/www.kidkarnmai.com\/"}
```


2.4 ศักยภาพของแอปพลิเคชัน

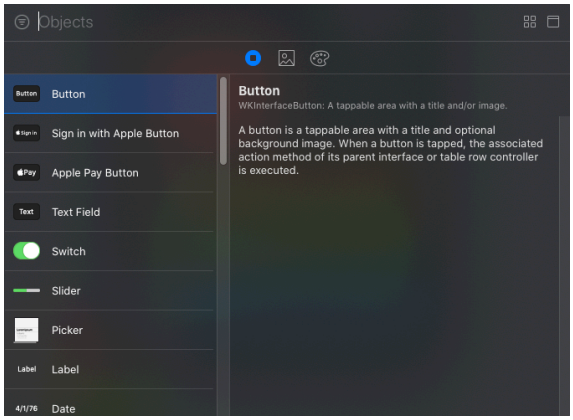
ศักยภาพของแอปพลิเคชันติดตามยอดจำนวน COVID–19 อย่างง่ายมีดังนี้:—

- สามารถเป็นหนึ่งในเครื่องมือที่ทำให้ข้อมูลข่าวสารสถิติผู้ติดเชื้อ ผู้ที่ได้รับการรักษาแล้ว และผู้ที่เสียชีวิตจากเชื้อไวรัส COVID–19 มาอยู่ในพื้นที่ที่มองเห็นและเข้าถึงได้ง่ายที่สุดคือข้อมือที่ใส่ Apple Watch

บทที่ 3
หลักการ ทฤษฎี และงานที่เกี่ยวข้อง

3.1 การสร้างและออกแบบ User Interface (UI)

หน้าจอแอปพลิเคชันแสดงผลยอดจำนวน COVID–19 (Interface.storyboard)

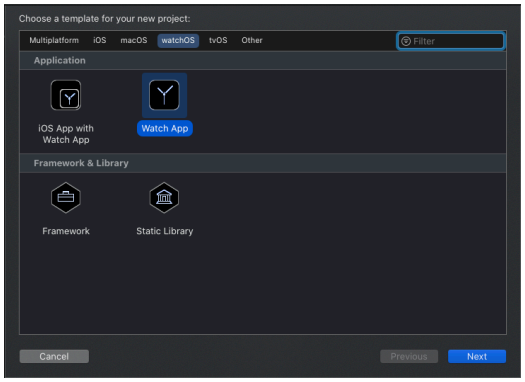


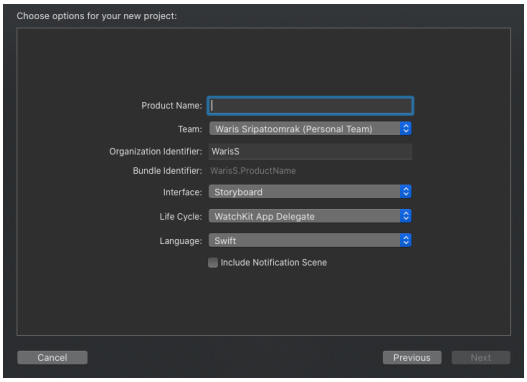
หน้าต่าง Library สำหรับการเพิ่มส่วนต่าง ๆ บนหน้าอินเตอร์เฟซ

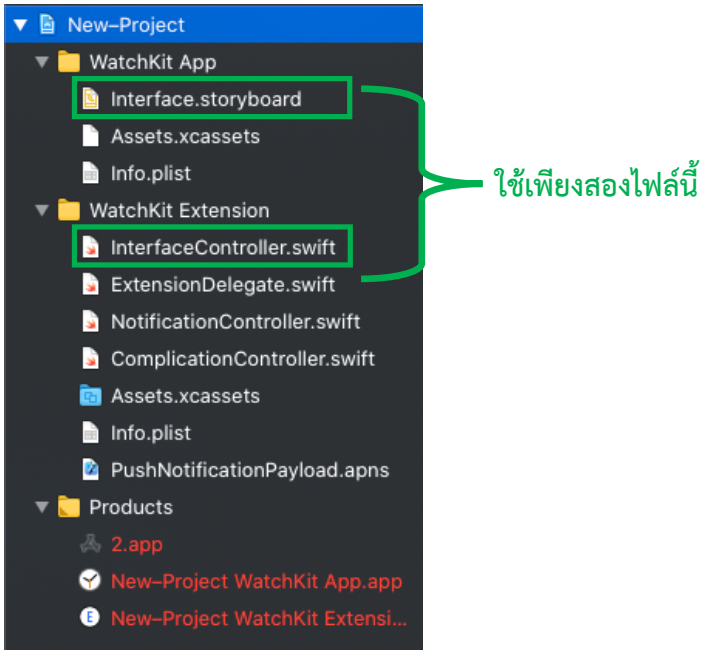
ในการสร้างหน้าจออินเตอร์เฟซใน Xcode สามารถสร้างได้ด้วยการเพิ่ม Object ต่าง ๆ มาใส่ใน Storyboard ของ Xcode ได้โดยไม่จำเป็นต้องเขียนโค้ด จึงทำให้เป็นขั้นตอนที่เรียบง่ายและรวดเร็ว โดยการที่จะให้ข้อมูลแสดงผลตามตัวเลขหัวข้อต่าง ๆ ตามข้อมูลที่ได้มานั้น จะถูกนำมาอธิบายในหัวข้อย่อยถัดไป

3.2 ขั้นตอนการเขียนโปรแกรม

เมื่อสร้างโปรเจกต์โดยการเปิด Xcode ขึ้นมา เลือก Template สำหรับโปรเจกต์เป็น watchOS จากนั้นเลือก Interface เป็น Storyboard และ Life Cycle เป็น WatchKit App Delegate







เมื่อสร้างและ Setting Configurator ในโปรเจกต์ใหม่ Xcode แล้ว จะมีไฟล์ดังนี้ถูกสร้างมาให้โดยอัตโนมัติ โดยในการสร้างแอปพลิเคชันนี้จะใช้ไฟล์ Interface.storyboard และ InterfaceController.swift เท่านั้น โดย Interface.storyboard จะใช้เพื่อสร้างและออกแบบ UI เท่านั้น และ InterfaceController.swift จะใช้เพื่อเขียนโปรแกรม ใส่ข้อมูลและดึงข้อมูลมาแสดงบนแอปพลิเคชันเท่านั้น

Xcode จะมีโค้ดเริ่มต้น (default) มาให้ว่าในแอปพลิเคชันควรมีคลาส (Class) หรือฟังก์ชันใดบ้างโดยใน InterfaceController.swift จะมีโค้ดมาให้เบ็ดเสร็จแล้ว ดังนี้:—

```
import WatchKit
import Foundation

class InterfaceController: WKInterfaceController {

    override func awake(withContext context: Any?) {
    }

    override func willActivate() {
    }

    override func didDeactivate() {
```

```
    }
}
```

โดยสามารถแยกโค้ดต่าง ๆ ที่จำเป็นออกมาได้ ดังนี้:—

1. `class InterfaceController`
2. `func awake`
3. `func willActivate`
4. `func didDeactivate`

หน้าที่ของฟังก์ชัน และ Class ดังกล่าวในเชิงแอปพลิเคชัน มีดังต่อไปนี้

- **`class InterfaceController`** จะทำหน้าที่เพื่อเชื่อมโยงระหว่างตัวโค้ดกับตัวแอปพลิเคชันให้แสดงผลออกมาบนหน้าจอ
- โดยใน `class InterfaceController` จะประกอบด้วยฟังก์ชันที่ชื่อว่า **`awake`** ซึ่งมีหน้าที่แสดงหน้าตาของแอปพลิเคชันเมื่อเริ่มเปิดแอปพลิเคชันขึ้นมาว่าต้องแสดงออกมาเป็นแบบไหน (ไม่ถูกใช้)
- **`override`** คือการกำหนด Function method ใน Class
- **`willActivate`** คือฟังก์ชันที่จะแสดงผลหน้าจอที่ได้ออกแบบจาก Storyboard โดยให้ตัวข้อมูลมาเป็นตัวแสดงผลเมื่อเรียกข้อมูลจาก API เสร็จแล้ว
- **`didDeactivate`** คือฟังก์ชันที่ใช้เมื่อปิดแอปพลิเคชันจะให้ตอบสนองแบบไหน หรือ ให้เชื่อมโยงหรือพาไปหน้าจอไหนบ้าง (ซึ่งฟังก์ชันนี้ไม่ได้ถูกใช้ในการทำแอปพลิเคชันนี้)

เมื่อทำแอปพลิเคชันเสร็จสิ้นแล้วนั้น จะได้โค้ดออกมาเป็นดังต่อไปนี้:—

```
import WatchKit
import Foundation

class InterfaceController: WKInterfaceController {

      @IBOutlet var LastUpdateInfo: WKInterfaceDate!
      @IBOutlet var Case_Label: WKInterfaceLabel!
      @IBOutlet var Recovered_Label: WKInterfaceLabel!
      @IBOutlet var Deaths_Label: WKInterfaceLabel!


override func awake(withContext context: Any?) {
          super.awake(withContext: context)
}
      override func willActivate() {
          super.willActivate()
      var request = URLRequest(url: URL(string: "https://covid19.th-
          stat.com/json/covid19v2/getTodayCases.json")!)
      request.httpMethod = "GET"
      request.addValue("application/json", forHTTPHeaderField:
          "Content-Type")
      let session = URLSession.shared
```

```
        let task = session.dataTask(with: request, completionHandler:
            {data, response, error -> Void in
            print(response!)
do {
            let json = try JSONSerialization.jsonObject(with: data!)
            as! Dictionary<String, AnyObject>

        var Result_AllCase: String
        var Result_AllRecovered: String
        var Result_AllDeath: String

        let result: [String: Any] = json
        if let All_Case = result["NewConfirmed"] as? Int {
            Result_AllCase = String(All_Case)
        }
            else{
            Result_AllCase = "Cannot Access API”
            }

        if let All_Recovered = result["NewRecovered"] as? Int {
            Result_AllRecovered = String(All_Recovered)
        }
            else{
            Result_AllRecovered = "Cannot Access API"
            }

        if let All_Deaths = result["NewDeaths"] as? Int {
            Result_AllDeath = String(All_Deaths)
        }
            else{
            Result_AllDeath = "Cannot Access API"
            }

        let calendar = Calendar.current
        self.Case_Label.setText(Result_AllCase)
        self.Recovered_Label.setText(Result_AllRecovered)
        self.Deaths_Label.setText(Result_AllDeath)
        self.LastUpdateInfo.setCalendar(calendar)


}
        catch {
        self.Case_Label.setText(“X")
        self.Recovered_Label.setText(“X")
        self.Deaths_Label.setText("X")
        }
})

task.resume()

}
override func didDeactivate(){
```

```
        super.didDeactivate()
        }
}
```

3.3 การอธิบายโค้ด

ซึ่งสามารถแยกฟังก์ชันต่าง ๆ ที่จำเป็นและได้ใช้งานออกมาได้ คือ

- **`class InterfaceController`** จะทำหน้าที่เพื่อเชื่อมโยงตัวโค้ดกับตัวแอปพลิเคชันให้แสดงผลออกมาบนหน้าจอ
- โดยใน `class InterfaceController` จะฟังก์ชันที่ชื่อว่า **`awake`** ซึ่งมีหน้าที่แสดงหน้าตาของแอปพลิเคชันเมื่อเพิ่งเริ่มเปิดแอปพลิเคชันว่าต้องแสดงออกมาเป็นแบบไหน
- **`override`** คือเป็นฟังก์ชันที่ต้องมีเพื่อให้แสดงหน้าจอตอนดำเนินโค้ด
- **`willActivate`** คือฟังก์ชันที่จะแสดงหน้าจอต่าง ๆ ต่อไปนี้ โดยให้ตัวข้อมูลมาเป็นตัวแสดงผลเมื่อบรรจุข้อมูล (download; ดาวน์โหลด) เสร็จแล้ว
- **`didDeactivate`** คือฟังก์ชันที่ใช้เมื่อปิดแอปพลิเคชันจะให้ตอบสนองแบบไหน หรือ ให้เชื่อมโยงหรือพาไปหน้าจอไหนบ้าง (ไม่ถูกนำมาใช้งาน)

```
@IBOutlet var LastUpdateInfo: WKInterfaceDate!
@IBOutlet var Case_Label: WKInterfaceLabel!
@IBOutlet var Recovered_Label: WKInterfaceLabel!
@IBOutlet var Deaths_Label: WKInterfaceLabel!
```

- **`@IBOutlet`** (มาจาก Interface Builder Outlet) จะทำหน้าที่เป็นตัวเชื่อมต่อระหว่าง Object ต่าง ๆ ใน Storyboard กับตัวโค้ด หรือ ให้รู้จักตัวแปรที่ได้ประกาศขึ้นมา
- **`var`** คือ ตัวแปร (variables) ที่สามารถเปลี่ยนค่าทีหลังได้หลังจากประกาศตัวแปรไปแล้ว ดังตามตัวอย่างตามหัวข้อที่ 2.1.1 ในหน้าที่ 2
    - ข้อความที่อยู่หลัง **`var`** คือชื่อของตัวแปรนั้น ๆ ที่ได้ตั้งไว้
    - เครื่องหมาย **`:`** (ทวิภาค; colon) สามารถทำหน้าที่เป็นคำว่า “คือ” ได้
    - **`WKInterfaceDate, WKInterfaceLabel`** คือชนิดของ Objects ต่าง ๆ (WK มาจาก WatchKit และ “Date”, “Label” คือชื่อชนิดของ Object)
- เครื่องหมาย **`!`** (และ **`?`**) คือเป็นตัวของ Optionals

    ตามนิยามของ Optionals ในเอกสารทางการของ Apple ได้ระบุไว้ว่า:

    *“...When working with optional values, you can write **`?`** before operations like methods, properties, and subscripting. If the value before the **`?`** is `nil`, everything after the **`?`** is ignored and the value of the whole expression is `nil`. Otherwise, the optional value is unwrapped, and everything after the **`?`** acts on the unwrapped value. In both cases, the value of the whole expression is an optional value. Once you’re sure that the optional does contain a value, you can access its underlying value by adding an exclamation mark (**`!`**) to the end of the optional’s name. The exclamation mark effectively says, “I know that*

this optional definitely has a value; please use it." This is known as forced unwrapping of the optional's value..."

- กล่าวคือ Optional **?** จะใช้เมื่อตัวแปรที่กำหนดไว้อาจมีค่าเป็น `nil` (เป็นค่าว่าง) ก็ได้ หากมีค่าเป็น `nil` จริง ๆ ทุกตัวที่เหลือที่อยู่หลัง **?** จะเป็นโมฆะแล้วถือว่า Expression ทั้งอันเป็น `nil` ทันที ส่วน Optional **!** จะใช้เมื่อค่าตัวแปรนั้นไม่สามารถเป็นค่าว่างได้อยู่แล้วอย่างชื่อและอายุของคน ซึ่งในที่นี้คือยอดจำนวน COVID-19 ที่ไม่สามารถเป็นค่าว่างได้

```
override func willActivate() {
        super.willActivate()
}
```

- **super** จะทำหน้าที่ดำเนินโค้ดที่อยู่หลัง **.** โดยในที่นี้ **super** จะดำเนินให้ `willActivate` ทำงาน

```
var request = URLRequest(url: URL(string: "https://covid19.th-
    stat.com/json/covid19v2/getTodayCases.json")!)
    request.httpMethod = "GET"
    request.addValue("application/json", forHTTPHeaderField:
        "Content-Type")
```

- **=** คือตัวดำเนินการกำหนดค่า (assignment operator) ให้กับตัวแปรหรือค่าคงที่
- **URLRequest** ทำหน้าที่เรียกบริการ API นำข้อมูลมาใช้
- **https://covid19.th-stat.com/json/covid19v2/getTodayCases.json** คือข้อมูลที่ต้องใช้
- **httpMethod** คือวิธีการจัดการกับข้อมูลต่าง ๆ
- **GET** เป็นหนึ่งในวิธีการจัดการกับข้อมูล โดย **GET** คือการนำข้อมูลมาเฉย ๆ
- **addValue(__, forHTTPHeaderField)** คือ **addValue** จะเพิ่มค่า (value) (ในที่นี้คือ **"application/json"**) ให้ **forHTTPHeaderField**
    - **forHTTPHeaderField** จะดึงข้อมูล JSON ออกมาทั้งหมดในชุดเดียวใหญ่ ๆ
- **Content-Type** จะใช้ระบุประเภทสื่อต้นแบบของข้อมูล หรือ ทรัพยากร

```
let session = URLSession.shared
    let task = session.dataTask(with: request, completionHandler:
        {data, response, error -> Void in
        print(response!)
```

- **let** คือ ตัวแปรที่ไม่สามารถเปลี่ยนค่าทีหลังได้หลังจากประกาศตัวแปรนั้นไปแล้ว เช่น

```
let c:Float = 13.54
let d       = 52
c:Int       = 19     ไม่สามารถทำได้; Error
print(c*d)           (ไม่แสดงผล)
```

- **dataTask(with:completionHandler:)** จะสร้างงาน (task) ที่จะนำข้อมูลเนื้อหาต่าง ๆ จาก URL ที่ป้อนเข้าไป แล้วค่อยเรียกให้ **completionHandler** (completion = การทำเสร็จ / เสร็จสิ้น,

handler = ผู้ดำเนินการ / ผู้จัดการ ดังนั้น `completionHandler` คือตัวดำเนินการที่ทำให้งานต่าง ๆ เสร็จ-สิ้นได้) ทำงาน

- มีพารามิเตอร์ (parameters) 2 ตัวหลัก ๆ คือ

- **`request`** เป็น Object ที่จะจัดหา URL, cache policy, request type เป็นต้น
- **`completionHandler`** จะถูกมาใช้บริการเมื่องานที่หมอบหมายอื่น ๆ ทำงานเสร็จสิ้นแล้ว
  - ↳ โดย **`completionHandler`** ประกอบด้วยพารามิเตอร์ 3 ตัว คือ
  - **`data`** คือข้อมูลที่ได้รับกลับคืนมาจากเซิร์ฟเวอร์
  - **`response`** คือ Object ที่มอบข้อมูลตอบกลับเป็นข้อมูลแบบอภิพันธุ์ (Metadata) กล่าวคือข้อมูลอภิพันธุ์เป็นข้อมูลรายละเอียดย่อยของข้อมูลใหญ่ เช่น

ชื่อหนังสือ หนังสือเรียนรายวิชาเพิ่มเติมวิทยาศาสตร์และเทคโนโลยี เคมี ชั้นมัธยมศึกษาปีที่ ๔ เล่ม ๑
ครั้งที่พิมพ์ ๓
หมายเลข ISBN 978-616-362-886-2
จัดพิมพ์ สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย

เป็นต้น

  - **`error`** เป็นตัวบ่งว่าทำไมถึง Request ไม่สำเร็จ หรือ จะแสดงค่า `nil` เมื่อสามารถ Request ได้สำเร็จ (คือไม่มี Error)

- **`Void`** (หรือ **`()`**; empty tuple) คือค่าของการตอบกลับของข้อมูลที่ไม่มีการระบุเฉพาะเจาะจงว่าข้อมูลจะมีการตอบกลับมาต้องเป็นข้อมูลชนิดใด
- **`response`** คือการตอบกลับจากเซิร์ฟเวอร์ไปยัง Request ที่กำลังดำเนินอยู่
  - ในที่นี้มีรูปแบบการตอบกลับหลัก ๆ 2 รูปแบบ คือ
    - **`200 OK`** (สำหรับ HTTP method `GET`) คือสามารถรับ และส่งข้อมูลมาได้แล้ว
    - **`500 Internal Server Error`** เกิด Error ในตัวเซิร์ฟเวอร์

```
do {
    let json = try JSONSerialization.jsonObject(with: data!)
        as! Dictionary<String, AnyObject>

    var Result_AllCase: String
    var Result_AllRecovered: String
    var Result_AllDeath: String
... }
```

- **`do`** จะบังคับให้ดึงข้อมูล JSON ออกมาอยู่ในรูปข้อมูล
- **`try`** มักถูกใช้เป็นตัวจัดการกับ Error ที่อาจสามารถเกิดขึ้นได้ในฟังก์ชันใด ๆ ก็ตาม
- **`JSONSerialization`** สามารถแปลง JSON เป็นข้อมูลในรูปแบบต่าง ๆ ได้ ในที่นี้จะถูกแปลงเป็น Dictionary

- **jsonObject** คือชุดการรวมกลุ่มของคู่ key–values ที่เป็น String (ข้อความ) ที่ถูกห่อไว้ด้วย { } (ปีกกา) และ : (ทวิภาค) อยู่ระหว่าง key กับ values และ , (จุลภาค) คั่น key กับ values ด้วยกัน เช่น:—

```
{"Confirmed":254515,"Recovered":205064,"Hospitalized":47481,"Deaths":1970,"NewConfirmed":4662,"NewRecovered":2793,"NewHospitalized":1833,"NewDeaths":36,"UpdateDate":"29\/06\/2021 12:13","DevBy":"https:\/\/www.kidkarnmai.com\/"}
```

- **data** คือข้อมูลที่กักเก็บข้อมูล JSON อยู่

key ↓ value ↓

- **as! Dictionary<String, AnyObject>** ดึงข้อมูลออกมาอยู่ในรูป Dictionary โดยชนิดข้อมูลของ key เป็น String และ values เป็นชนิดข้อมูลใด ๆ

- **var Result_AllCase: String**
  var Result_AllRecovered: String
  var Result_AllDeath: String

  } เป็นการประกาศรองรับกับค่าข้อมูลที่กำลังจะได้กลับมาจากเซิร์ฟเวอร์

```
let result: [String: Any] = json
     if let All_Case = result["NewConfirmed"] as? Int {
          Result_AllCase = String(All_Case)
     }
          else {
          Result_AllCase = "Cannot Access API"
          }
     if let All_Recovered = result["NewRecovered"] as? Int {
          Result_AllRecovered = String(All_Recovered)
     }
          else {
          Result_AllRecovered = "Cannot Access API"
          }
     if let All_Deaths = result["NewDeaths"] as? Int {
          Result_AllDeath = String(All_Deaths)
     }
          else {
          Result_AllDeath = "Cannot Access API"
          }
```

- **[String: Any]** จะมาจากตัว Dictionary เอง โดยจะยัด JSON ให้ไปอยู่ใน `var result` แต่จะมองว่าทั้งอันเป็น String 1 ตัว แล้วนำ `result` ไปหาค่า key ออกมา

- **result["NewConfirmed"]**
  result["NewRecovered"]
  result["NewDeaths"]

  } คือ 3 key หลักที่ต้องใช้มาแสดงแทน Labels ที่ตั้งไว้ แต่จำเป็นต้องรู้ชนิดของข้อมูลก่อน

- **as?** สามารถตรวจสอบได้ว่า Object นั้น ๆ สามารถแปลงเป็นชนิดข้อมูลอื่น ๆ ได้หรือไม่ หากไม่ได้ มันจะส่งค่า `nil` กลับมา

- **Int** คือชนิดข้อมูลที่เป็นจำนวนเต็ม (Integer)

- **`Result_AllCase = String(All_Case)`**
  `Result_AllRecovered = String(All_Recovered)`
  `Result_AllDeath = String(All_Deaths)`
  } ต้องทำการแปลงชนิดข้อมูลเป็น String เนื่องจากที่เมื่อประกาศเป็นตัวแปรไปสักครู่คือชนิดข้อมูลแบบ String เพราะทั้งสองอย่างนี้ต้องเป็นชนิดเดียวกัน

- ```
  if let All_Case = result["NewConfirmed"] as? Int {
          Result_AllCase = String(All_Case)
      }
          else {
               Result_AllCase = "Cannot Access API”
          }
  ```

คือ “ถ้าพบข้อมูลสำหรับตัวแปร [···] ได้แล้ว ให้เชื่อมข้อมูลนั้นกับ key ใน JSON ที่เหมาะสมซึ่งมีชนิดข้อมูลเป็นจำนวนเต็ม แล้วจึงแปลงตัวแปร [···] ให้เป็น String เนื่องจากทั้งสองอย่างนี้ต้องเป็นชนิดเดียวกัน หากไม่พบข้อมูลให้ขึ้นข้อความว่า ‘Cannot Access API’”

```
let calendar = Calendar.current
     self.Case_Label.setText(Result_AllCase)
     self.Recovered_Label.setText(Result_AllRecovered)
     self.Deaths_Label.setText(Result_AllDeath)
     self.LastUpdateInfo.setCalendar(calendar)
```

- **`calendar = Calendar.current`** ——①
  และ **`self.LastUpdateInfo.setCalendar(calendar)`** ——②
  ใน ① จะต้องทำให้วันที่เป็นปัจจุบันและต้องนำ ② มาเชื่อมข้อมูลกันด้วย เพื่อให้ทั้งคู่มีวันที่เป็นปัจจุบันเหมือนกัน
- **`self`** จะอ้างอิงถึง Object ปัจจุบันที่อยู่หลัง **`.`** อย่างในที่นี้เป็นตัวเชื่อม Label ให้แสดงตาม `setText`
- **`Case_Label`**
  `Recovered_Label`
  `Deaths_Label`
  } คือชื่อ Object
```

- **setText(__)** ให้แสดงข้อมูลที่ได้รับกลับมาจาก API

```
catch {
      self.Case_Label.setText("X")
      self.Recovered_Label.setText("X")
      self.Deaths_Label.setText("X")
      }
```

- **catch** เป็นตัวจัดการ Error ต่าง ๆ ด้วยกลุ่มโค้ด
- **setText("X")** ใน `catch` เมื่อดึงข้อมูลมาไม่ได้ ให้แทน "X" ลงไปใน Label แทน

```
task.resume()
```

- **task** จะดึงข้อมูลมาแสดงผลบนหน้าจอ
- **resume()** จะสั่งให้โค้ดทำงานต่อเมื่อ Tasks ถูกระงับไว้


```
import WatchKit
import Foundation

class InterfaceController: WKInterfaceController {

      @IBOutlet var LastUpdateInfo: WKInterfaceDate!
      @IBOutlet var Case_Label: WKInterfaceLabel!
      @IBOutlet var Recovered_Label: WKInterfaceLabel!
      @IBOutlet var Deaths_Label: WKInterfaceLabel!


override func awake(withContext context: Any?) {
            super.awake(withContext: context)
}
      override func willActivate() {
            super.willActivate()
      var request = URLRequest(url: URL(string: "https://
covid19.th-stat.com/json/covid19v2/getTodayCases.json")!)
      request.httpMethod = "GET"
      request.addValue("application/json",
            forHTTPHeaderField:"Content-Type")
      let session = URLSession.shared
      let task = session.dataTask(with: request,
      completionHandler:
            {data, response, error -> Void in
            print(response!)
do {
            let json = try
            JSONSerialization.jsonObject(with: data!)
            as! Dictionary<String, AnyObject>

      var Result_AllCase: String
      var Result_AllRecovered: String
      var Result_AllDeath: String

      let result: [String: Any] = json
      if let All_Case = result["NewConfirmed"] as? Int {
            Result_AllCase = String(All_Case)
      }
            else{
            Result_AllCase = "Cannot Access API"
            }

      if let All_Recovered = result["NewRecovered"] as? Int {
            Result_AllRecovered = String(All_Recovered)
      }
            else{
            Result_AllRecovered = "Cannot Access API"
            }

      if let All_Deaths = result["NewDeaths"] as? Int {
            Result_AllDeath = String(All_Deaths)
      }
            else{
            Result_AllDeath = "Cannot Access API"
            }

      let calendar = Calendar.current
      self.Case_Label.setText(Result_AllCase)
      self.Recovered_Label.setText(Result_AllRecovered)
      self.Deaths_Label.setText(Result_AllDeath)
      self.LastUpdateInfo.setCalendar(calendar)


}
      catch {
      self.Case_Label.setText("X")
      self.Recovered_Label.setText("X")
      self.Deaths_Label.setText("X")
      }
})

task.resume()

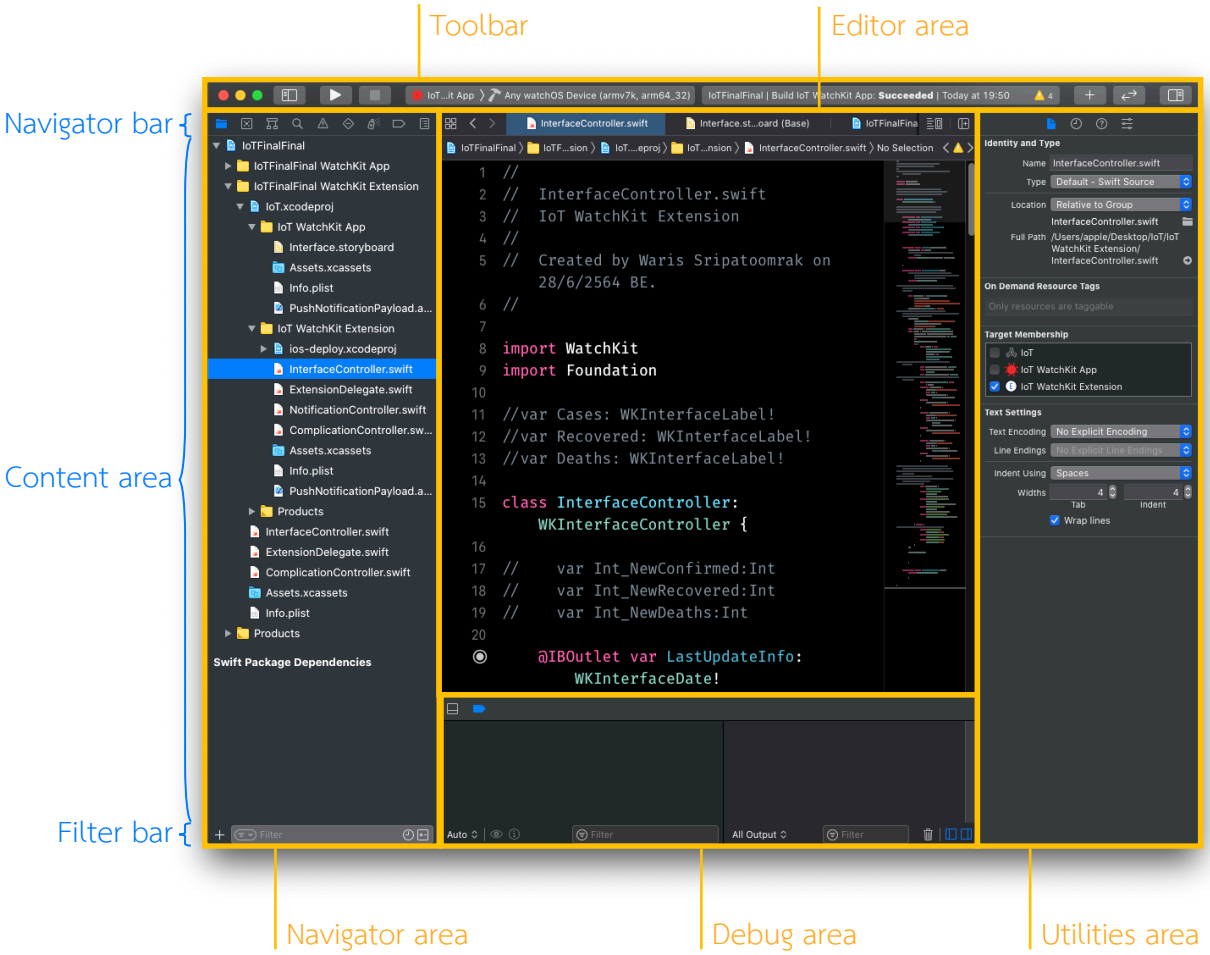
}
override func didDeactivate(){
      super.didDeactivate()
      }
}
```

บทที่ 4

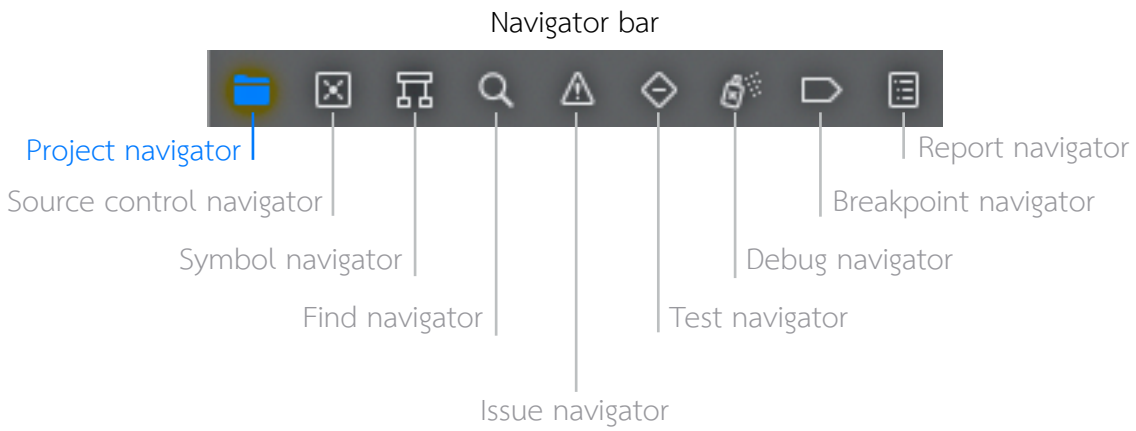
การทดลอง และ ผลการทดลอง

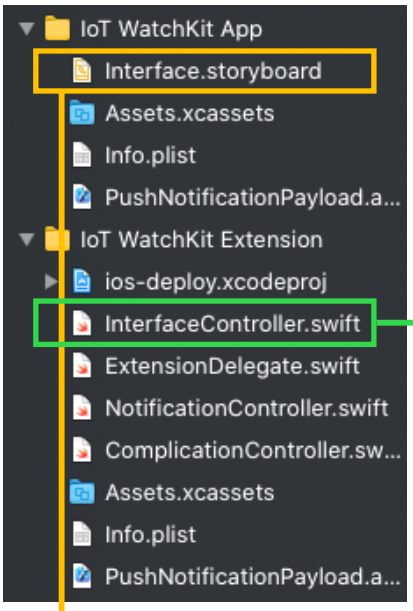
4.1 การทำงานในแต่ละหน้าจอ

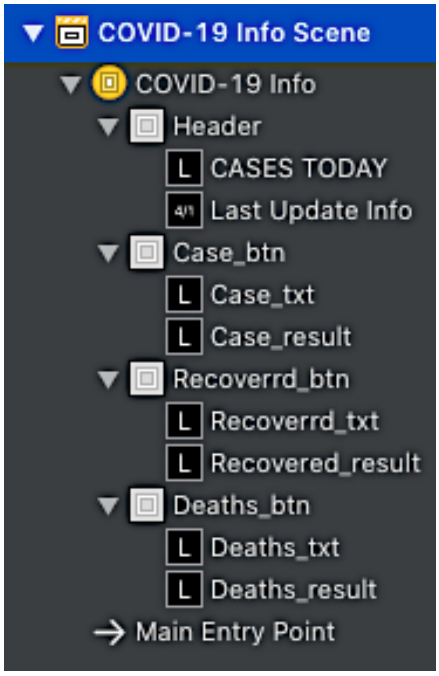
ในหน้าต่างของ Xcode มีส่วนประกอบ ดังนี้:—



ในส่วนของ Project Navigator จะสามารถทำงานกับไฟล์ที่ต้องการได้ ดังนี้:—








```
import WatchKit
import Foundation

class InterfaceController: WKInterfaceController {

      @IBOutlet var LastUpdateInfo: WKInterfaceDate!
      @IBOutlet var Case_Label: WKInterfaceLabel!
      @IBOutlet var Recovered_Label: WKInterfaceLabel!
      @IBOutlet var Deaths_Label: WKInterfaceLabel!


override func awake(withContext context: Any?) {
            super.awake(withContext: context)
}
      override func willActivate() {
            super.willActivate()
      var request = URLRequest(url: URL(string: "https://
covid19.th-stat.com/json/covid19v2/getTodayCases.json")!)
      request.httpMethod = "GET"
      request.addValue("application/json",
            forHTTPHeaderField:"Content-Type")
      let session = URLSession.shared
      let task = session.dataTask(with: request,
      completionHandler:
            (data, response, error -> Void in
            print(response!)
do {
            let json = try
            JSONSerialization.jsonObject(with: data!)
            as! Dictionary<String, AnyObject>

      var Result_AllCase: String
      var Result_AllRecovered: String
      var Result_AllDeath: String

      let result: [String: Any] = json
      if let All_Case = result["NewConfirmed"] as? Int {
            Result_AllCase = String(All_Case)
      }
            else{
            Result_AllCase = "Cannot Access API"
            }

      if let All_Recovered = result["NewRecovered"] as? Int {
            Result_AllRecovered = String(All_Recovered)
      }
            else{
            Result_AllRecovered = "Cannot Access API"
            }

      if let All_Deaths = result["NewDeaths"] as? Int {
            Result_AllDeath = String(All_Deaths)
      }
            else{
            Result_AllDeath = "Cannot Access API"
            }

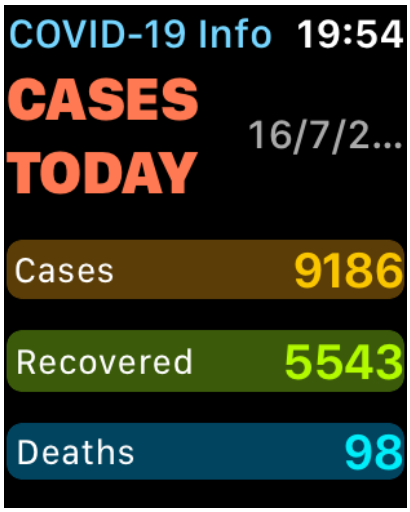
      let calendar = Calendar.current
      self.Case_Label.setText(Result_AllCase)
      self.Recovered_Label.setText(Result_AllRecovered)
      self.Deaths_Label.setText(Result_AllDeath)
      self.LastUpdateInfo.setCalendar(calendar)

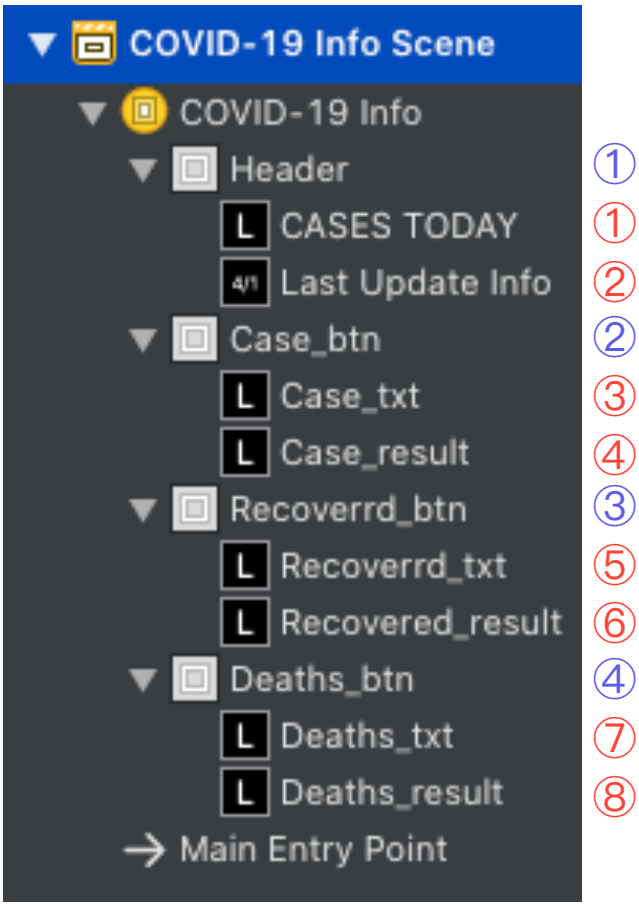

}
      catch {
      self.Case_Label.setText("X")
      self.Recovered_Label.setText("X")
      self.Deaths_Label.setText("X")
      }
})

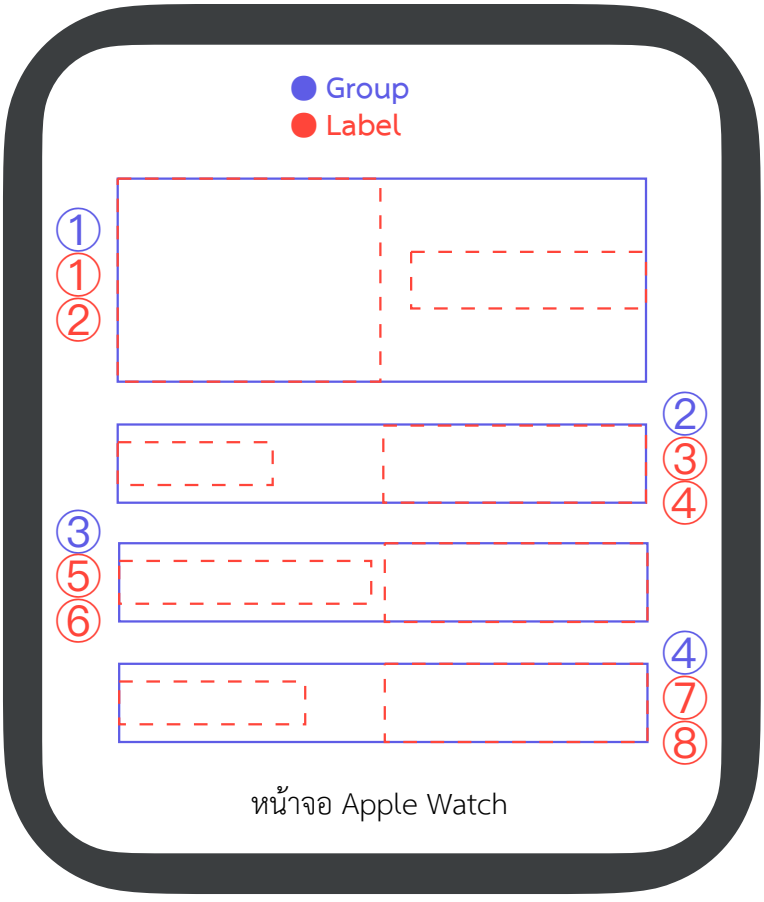
task.resume()

}
override func didDeactivate(){
      super.didDeactivate()
      }
}
```

โค้ดแอปพลิเคชัน



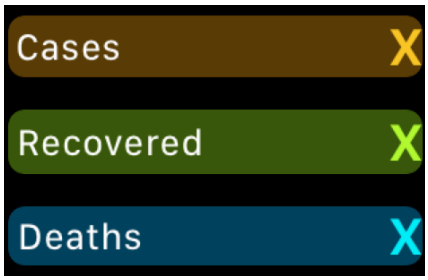

COVID-19 Info Scene
COVID-19 Info
Header
CASES TODAY
Last Update Info
Case_btn
Case_txt
Case_result
Recoverrd_btn
Recoverrd_txt
Recovered_result
Deaths_btn
Deaths_txt
Deaths_result
Main Entry Point
①
①
②
②
③
④
③
⑤
⑥
④
⑦
⑧


Group
Label
①
①
②
②
③
④
③
⑤
⑥
④
⑦
⑧
หน้าจอ Apple Watch

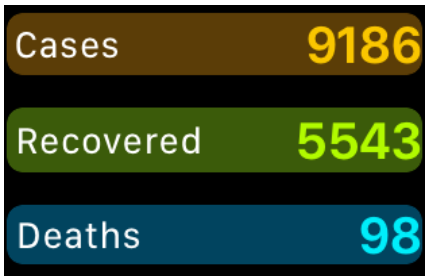
การที่จะให้ข้อมูลที่ได้มาจากเซิร์ฟเวอร์มาปรากฏแทนที่ Labels จะต้องทำความรู้จักชื่อตัวแปรของ Labels ต่าง ๆ ด้วย `@IBOutlet` โดย `@IBOutlet` จะทำกับ Labels ใน Storyboard ซึ่งตัวที่มีหน้าที่รองรับข้อมูลที่จะได้มาจากเซิร์ฟเวอร์คือ ตัวแปร `Result_AllCase`, `Result_AllRecovered`, และ `Result_AllDeath` จากนั้น ข้อมูลพวกนี้จะถูกส่งต่อไปหาฟังก์ชัน `self` โดยจะเชื่อมต่อพวกข้อมูลนี้กับ Labels เข้าด้วยกัน ส่งผลมาเป็น Labels ใหม่ที่ถูกแทนค่าด้วยข้อมูลที่ได้รับมาแล้วนั่นเอง

4.2 ผลสุดท้าย

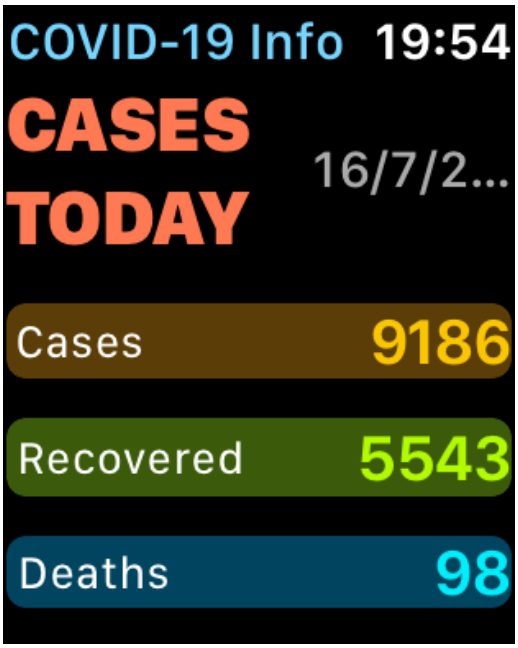
เมื่อใดที่ API ยังสามารถทำงานได้อยู่ โดยไม่คำนึงถึงว่าตัวข้อมูลในเซิร์ฟเวอร์เป็นปัจจุบันแล้วหรือไม่ ข้อมูลที่ได้รับกลับมายัง Apple Watch จะต้องทำการอัปเดตข้อมูลอยู่ตลอดเวลา



API ไม่ทำงาน



API ทำงานได้ปกติ



ผลสำเร็จ

4.3 ทดลองให้ใช้

4.3.1 สรุปความของการทดลองใช้งานแอปพลิเคชัน

จากการทดลองใช้งานแอปพลิเคชันสรุปความได้ว่า

แอปพลิเคชันนี้เป็นการให้บริการข่าวสารทางอินเทอร์เน็ตเกี่ยวกับการสืบค้นหาข้อมูลเฉพาะทาง ช่วยทำให้รับข่าวสารได้สะดวกรวดเร็วมากยิ่งขึ้น

ประโยชน์ของการนำมาใช้:

1. ด้านประสิทธิภาพ ช่วยให้ได้รับข้อมูลที่ต้องการได้เที่ยงตรงและรวดเร็ว
2. ประหยัด ประหยัดเวลาในการค้นหาข้อมูล

บทที่ 5

สรุปผล วิเคราะห์ และข้อเสนอแนะ

5.1 สรุปผลตอบรับแอปพลิเคชัน

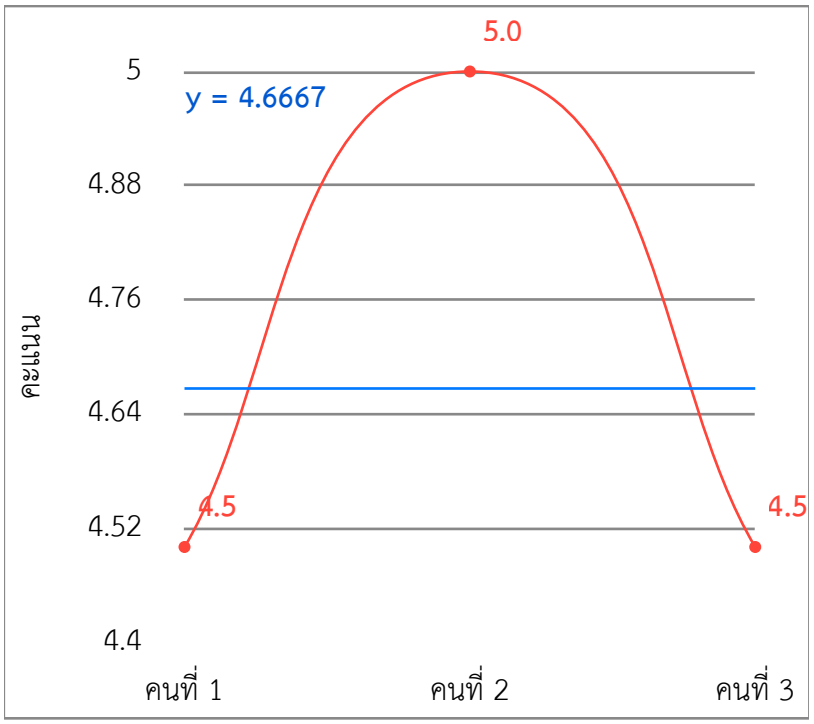
5.1.1 ค่า X̄

จากการสำรวจความพึงพอใจของผู้ใช้แอปพลิเคชันในกลุ่มคนสามคนจากคะแนน 5 คะแนน พบว่าดังนี้:—

$$\bar{x} = \frac{4.5 + 5 + 4.5}{3}$$

$$\bar{x} = \frac{14}{3}$$

$$\bar{x} = 4.67$$



ดังนั้น ผลตอบรับแอปพลิเคชันจากการสำรวจความพึงพอใจมีค่าเท่ากับ 4.67 จาก 5.00 คะแนน

5.1.2 ค่าส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation, SD)

$$\sigma = \sqrt{\frac{(4.5 - 4.67)^2 + (5 - 4.67)^2 + (4.5 - 4.67)^2}{3 - 1}}$$

$$\sigma = \sqrt{\frac{0.1667}{2}}$$

$$\sigma = \sqrt{0.08335}$$

$$\sigma = 0.288704$$

ดังนั้น ค่าส่วนเบี่ยงเบนมาตรฐานมีค่าเท่ากับ 0.288704

บรรณานุกรม

ควบคุมโรค, กรม. **Open API :) สำหรับนักพัฒนา**. [ออนไลน์]. เข้าถึงได้จาก : https://covid19.th-stat.com/th/api. (วันที่ค้นข้อมูล : 29 มิถุนายน 2564).

ควบคุมโรค, กรม. **แสดงค่าประจำวัน**. [ออนไลน์]. เข้าถึงได้จาก : https://covid19.th-stat.com/json/covid19v2/getTodayCases.json. (วันที่ค้นข้อมูล : 29 มิถุนายน 2564).

Cocoacasts. **What Is Void in Swift**. [ออนไลน์]. เข้าถึงได้จาก : https://cocoacasts.com/what-is-void-in-swift. (วันที่ค้นข้อมูล : 11 กรกฎาคม 2564).

Apple Developer Documentation. **dataTask(with:completionHandler:)**. [ออนไลน์]. เข้าถึงได้จาก : https://developer.apple.com/documentation/foundation/urlsession/1407613-datatask. (วันที่ค้นข้อมูล : 11 กรกฎาคม 2564).

Cocoacasts. **What Is The Difference Between Try, Try?, And Try!**. [ออนไลน์]. เข้าถึงได้จาก : https://cocoacasts.com/what-is-the-difference-between-try-try-and-try. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Hacking with Swift. **How to parse JSON using JSONSerialization**. [ออนไลน์]. เข้าถึงได้จาก : https://www.hackingwithswift.com/example-code/system/how-to-parse-json-using-jsonserialization. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

JSONObject. [ออนไลน์]. เข้าถึงได้จาก : https://processing.github.io/processing-javadocs/core/processing/data/JSONObject.html. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Apple Inc.. **Error Handling**. [ออนไลน์]. เข้าถึงได้จาก : https://docs.swift.org/swift-book/LanguageGuide/ErrorHandling.html. (วันที่ค้นข้อมูล : 15 กรกฎาคม 2564).

Thai Swift Class. **ตัวแปรชนิด Array**. [ออนไลน์]. เข้าถึงได้จาก : http://www.thaiswiftclass.com/2014/09/07/ตัวแปรชนิด-array/. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Apple Inc.. **Workspace Window Overview**. [ออนไลน์]. เข้าถึงได้จาก : https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/TheWorkspaceWindow.html#//apple_ref/doc/uid/TP40010215-CH25-SW1. (วันที่ค้นข้อมูล : 16 กรกฎาคม 2564).

ครูต้อม ออนไลน์. **X-Bar ค่าเฉลี่ย and S.D. ส่วนเบี่ยงเบนมาตรฐาน**. [ออนไลน์]. เข้าถึงได้จาก : https://krutomonline.wordpress.com/statistical-packages/x-bar-and-s-d/. (วันที่ค้นข้อมูล : 18 กรกฎาคม 2564).

Apple Developer Documentation. **Void**. [ออนไลน์]. เข้าถึงได้จาก : https://developer.apple.com/documentation/swift/void. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Apple Developer Documentation. **resume()**. [ออนไลน์]. เข้าถึงได้จาก : https://developer.apple.com/documentation/foundation/urlsessiontask/1411121-resume. (วันที่ค้นข้อมูล : 15 กรกฎาคม 2564).

LearnAppMaking.com. **Self and self in Swift**. [ออนไลน์]. เข้าถึงได้จาก : https://learnappmaking.com/self-swift-how-to/#self-as-an-object-in-swift. (วันที่ค้นข้อมูล : 15 กรกฎาคม 2564).

Fresh Beginning. **Swift 3.0 - What's is, as, as? and as! operators?**. [ออนไลน์]. เข้าถึงได้จาก : https://jayeshkawli.ghost.io/swift-3-0-whats-is-as-as-and-as-operators/. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Apple Inc.. **Basic Operators**. [ออนไลน์]. เข้าถึงได้จาก : https://docs.swift.org/swift-book/LanguageGuide/BasicOperators.html. (วันที่ค้นข้อมูล : 11 กรกฎาคม 2564).

CamPus. **[Swift] Optionals เจ้าพวก !? คืออะไรกันเนี่ย**. [ออนไลน์]. เข้าถึงได้จาก : https://bit.ly/3evvon4. (วันที่ค้นข้อมูล : 11 กรกฎาคม 2564).

Holy Knight Ohm. **รูปแบบของ Array ในภาษา Swift**. [ออนไลน์]. เข้าถึงได้จาก : https://bit.ly/3z9Lzyo. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Apple Inc.. **The Swift Programming Language (Swift 4)**. [ออนไลน์]. เข้าถึงได้จาก : https://bit.ly/2Ua9azY. (วันที่ค้นข้อมูล : 13 กรกฎาคม 2564).

Libraryhub. **METADATA คืออะไร ถามกันจัง???**. [ออนไลน์]. เข้าถึงได้จาก : http://www.libraryhub.in.th/2009/11/27/what-is-metadata/. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Dataedo. **What is Metadata (with examples)**. [ออนไลน์]. เข้าถึงได้จาก : https://dataedo.com/kb/data-glossary/what-is-metadata. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

MDN Web Docs. **HTTP response status codes**. [ออนไลน์]. เข้าถึงได้จาก : https://developer.mozilla.org/en-US/docs/Web/HTTP/Status. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Wasith T. (Bai-Phai). **ว่าด้วย function ในภาษา Swift**. [ออนไลน์]. เข้าถึงได้จาก : https://bit.ly/3Batj9Q. (วันที่ค้นข้อมูล : 3 กรกฎาคม 2564).

Ichi.pro. **วิธีทำงานกับ JSON ใน Swift**. [ออนไลน์]. เข้าถึงได้จาก : https://ichi.pro/th/withi-thangan-kab-json-ni-swift-9057435550590. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Apple Developer Documentation. **Dictionary**. [ออนไลน์]. เข้าถึงได้จาก : https://developer.apple.com/documentation/swift/dictionary. (วันที่ค้นข้อมูล : 14 กรกฎาคม 2564).

Stack Overflow. **Difference Between ? and ! in Swift Language? [duplicate]**. [ออนไลน์]. เข้าถึงได้จาก : https://developer.apple.com/documentation/swift/dictionary. (วันที่ค้นข้อมูล : 11 กรกฎาคม 2564).

Hacking with Swift. **What is an IBOutlet?**. [ออนไลน์]. เข้าถึงได้จาก : https://www.hackingwithswift.com/example-code/xcode/what-is-an-iboutlet. (วันที่ค้นข้อมูล : 11 กรกฎาคม 2564).