

## Tutorial สำหรับปฏิบัติการ 1

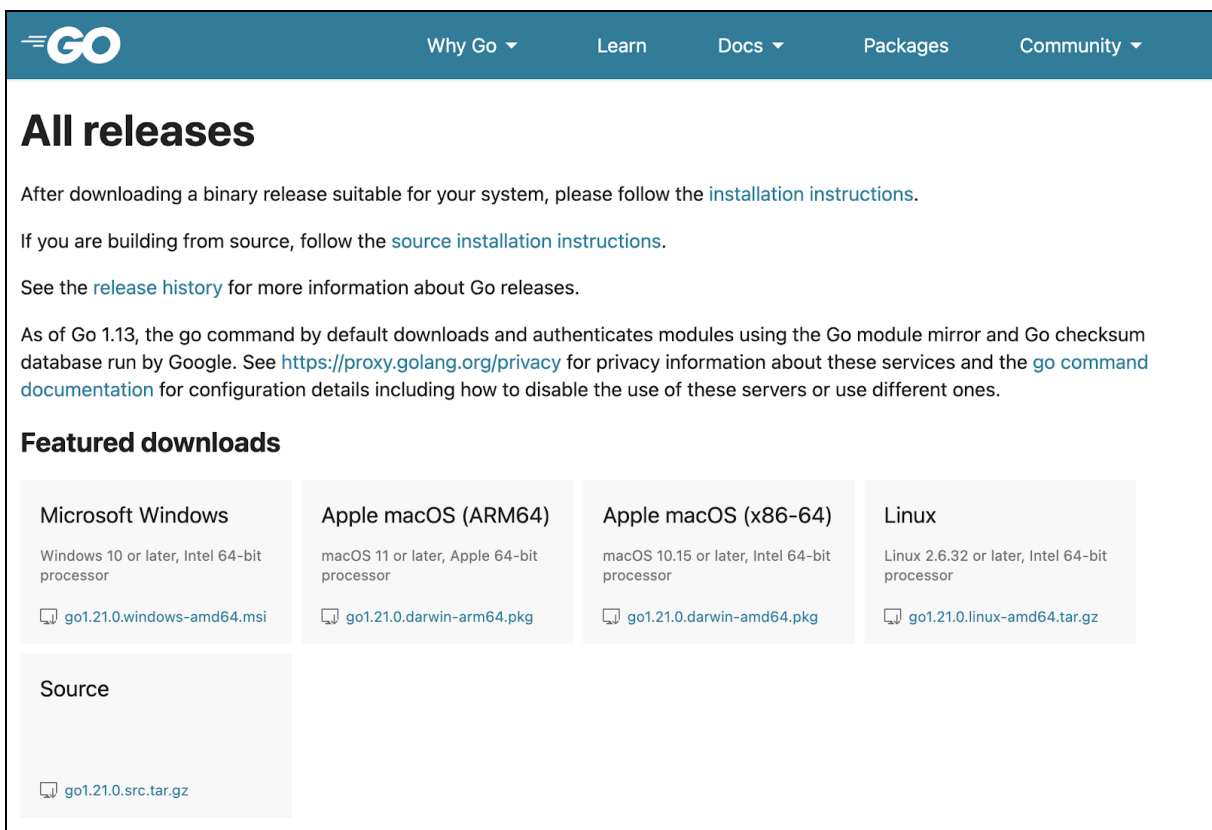
สร้าง directory Project ชื่อ sa-66-example

```
cd c:\
```

```
mkdir sa-66-example
```

```
cd sa-66-example
```

1. ติดตั้ง Go compiler (เราจะใช้ v1.19.x ในเทอม 1/66) โดย download ได้จาก  
ที่นี่ <https://go.dev/dl/go1.19.12.windows-amd64.msi>



The screenshot shows the 'All releases' page on the Go website. The page has a dark blue header with the Go logo and navigation links: 'Why Go', 'Learn', 'Docs', 'Packages', and 'Community'. The main content area is white and contains the following text:

### All releases

After downloading a binary release suitable for your system, please follow the [installation instructions](#).

If you are building from source, follow the [source installation instructions](#).

See the [release history](#) for more information about Go releases.

As of Go 1.13, the go command by default downloads and authenticates modules using the Go module mirror and Go checksum database run by Google. See <https://proxy.golang.org/privacy> for privacy information about these services and the [go command documentation](#) for configuration details including how to disable the use of these servers or use different ones.

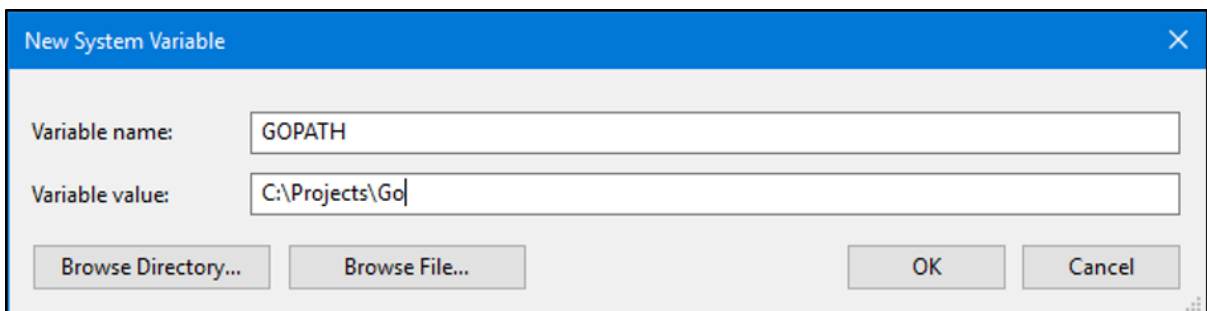
#### Featured downloads

Microsoft Windows	Apple macOS (ARM64)	Apple macOS (x86-64)	Linux
Windows 10 or later, Intel 64-bit processor	macOS 11 or later, Apple 64-bit processor	macOS 10.15 or later, Intel 64-bit processor	Linux 2.6.32 or later, Intel 64-bit processor
<a href="#">go1.21.0.windows-amd64.msi</a>	<a href="#">go1.21.0.darwin-arm64.pkg</a>	<a href="#">go1.21.0.darwin-amd64.pkg</a>	<a href="#">go1.21.0.linux-amd64.tar.gz</a>

#### Source

[go1.21.0.src.tar.gz](#)

ตั้งค่า Environment ดังนี้



The screenshot shows a 'New System Variable' dialog box with a blue title bar and a close button (X) in the top right corner. The dialog contains two input fields:

Variable name:

Variable value:

At the bottom, there are four buttons: 'Browse Directory...', 'Browse File...', 'OK', and 'Cancel'.

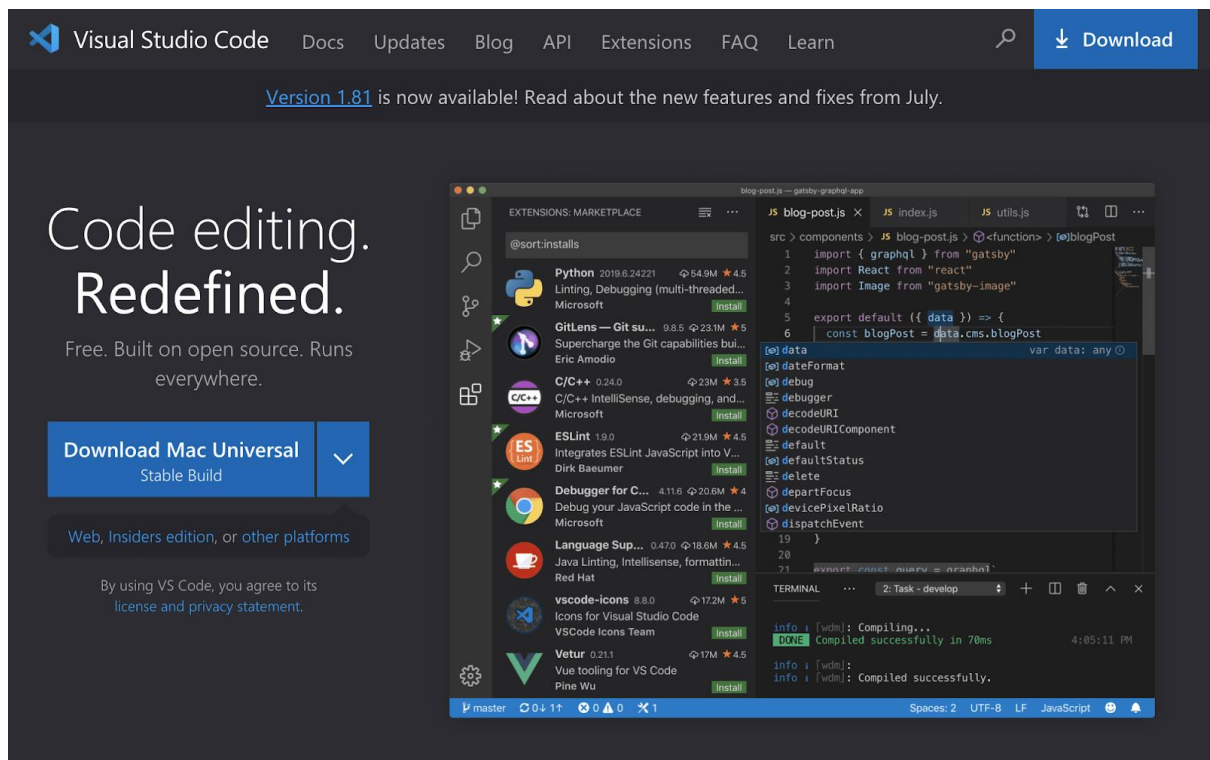
GOROOT ตั้งเป็น C:\Go

GOPATH ตั้งเป็น C:\Users\<ชื่อ user ของตนเอง>\Go

PATH เพิ่ม C:\Users\<ชื่อ user ของตนเอง>\Go\bin;

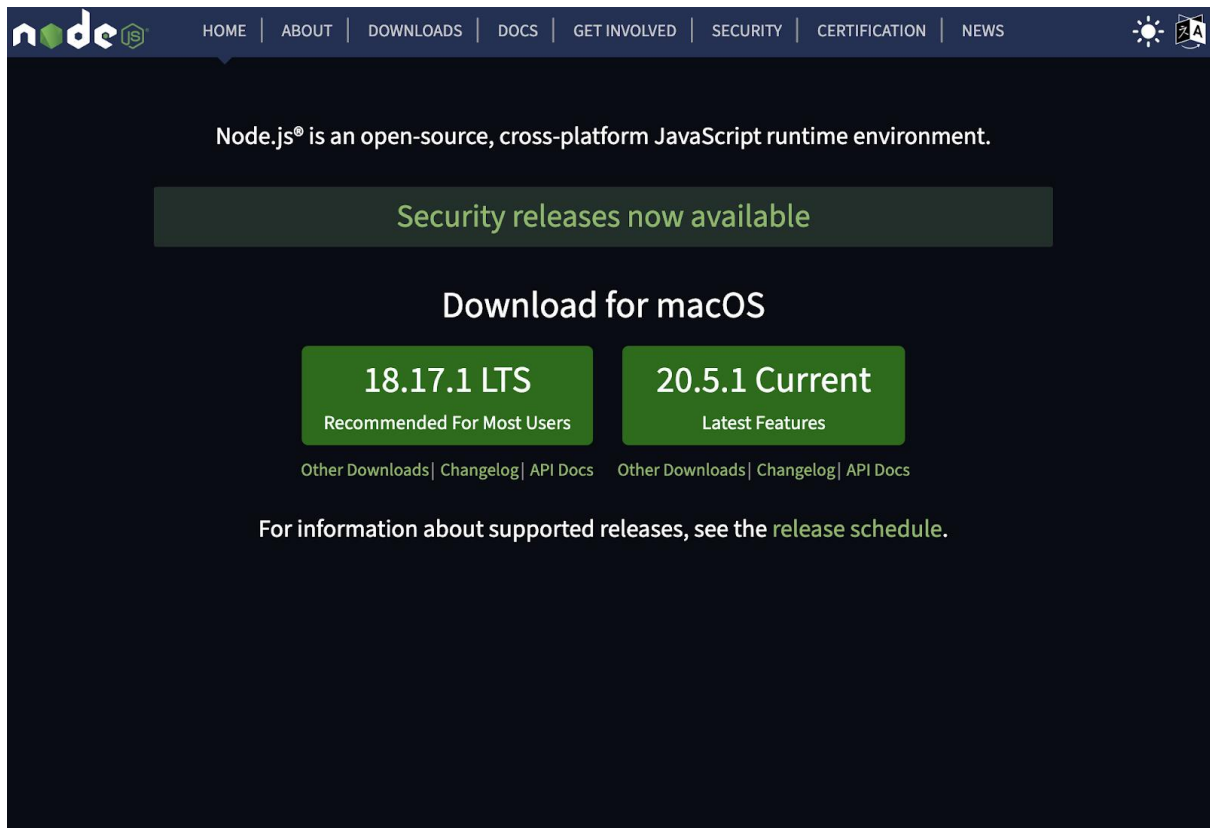
ถ้าเปิด Command Prompt (cmd.exe) หรือ Power Shell อยู่ ให้ปิดแล้วเปิดใหม่ หลังจากตั้งค่า Environment เสร็จ

## 2. ติดตั้ง VS Code



เลือก Extension ติดตั้ง extension สำหรับภาษา Go

## 3. ติดตั้ง nodejs (เราจะใช้ v18.x ในเทอม 1/66)



4. สมัคร Github (<https://github.com>) แล้วจะได้ชื่อ user ของ GitHub มา
5. Install Git <https://git-scm.com/download/win>
6. สร้าง project backend ที่ c:\sa-66-example

```
cd c:\sa-66-example
```

```
mkdir backend
```

```
cd backend
```

```
go mod init github.com/<ชื่อ user ที่ได้มาจากการสมัคร GitHub>/sa-66-example
```

```
เช่น github.com/tanapon395/sa-66-example
```

7. ติดตั้ง GORM, Gin และ SQL Lite (ถ้าเรียก go get ไม่ได้แปลว่าการติดตั้ง Go มีปัญหา)

```
go get -u github.com/gin-gonic/gin
```

```
go get -u gorm.io/gorm
```

```
go get -u gorm.io/driver/sqlite
```

## 8. สร้าง Schema

โครงสร้างไฟล์ที่เราต้องสร้างใน directory backend

<pre>./backend   /...   /controler     user.go   /entity     user.go     setup.go     main.go</pre>	<pre>✓ controller   =GO user.go ✓ entity   =GO setup.go   =GO user.go ≡ go.mod ≡ go.sum ≡ main =GO main.go ≡ sa-66.db</pre>
---	---

mkdir entity

cd entity/

ใน c:\sa-66-example\backend\entity สร้าง schema ของ User ในไฟล์ชื่อ **user.go**

แก้ไขไฟล์ **user.go** ให้เป็นแบบนี้

```
package entity
import (
    "gorm.io/gorm"
)
type User struct {
    gorm.Model
    FirstName string
    LastName string
    Email string
    Phone string
}
```

จากตำแหน่ง folder เดียวกัน เราจะเตรียมไฟล์ถัดมาคือไฟล์ `setup.go` เพื่อใช้สร้าง database

```
package entity
import (
    "gorm.io/driver/sqlite"
    "gorm.io/gorm"
)
var db *gorm.DB
func DB() *gorm.DB {
    return db
}
func SetupDatabase() {
    database, err := gorm.Open(sqlite.Open("sa-66.db"), &gorm.Config{})
    if err != nil {
        panic("failed to connect database")
    }
    // Migrate the schema
    database.AutoMigrate(&User{})
    db = database
}
```

เมื่อได้ Entity แล้ว เราจะเตรียมส่วนถัดมาคือ controller

ขั้นตอนการสร้าง controller คือ ที่ folder `c:\sa-66-example\backend`

1. `mkdir controller`
2. `cd controller`
3. ใน `c:\sa-66-example\backend\controller` สร้างไฟล์ชื่อ `user.go` เพื่อเก็บ controller สำหรับเชื่อมต่อกับ entity User

```
package controller
import (
    "net/http"
    "github.com/gin-gonic/gin"
    "github.com/tanapon395/sa-66-example/entity"
)
```

จากนั้นสร้าง function ต่อไปนี้

function CreateUser เป็นการทำงานแทนคำสั่ง insert ของ SQL

```
// POST /users
func CreateUser(c *gin.Context) {
    var user entity.User
    if err := c.ShouldBindJSON(&user); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if err := entity.DB().Create(&user).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": user})
}
```

โดย function นี้จะคืนค่าเป็น user ที่สร้างเสร็จแล้ว กลับไปเป็น JSON ให้ฝั่ง UI นำไปแสดงผล

ถัดมาจะเป็น function GetUser โดยในตัวอย่างเป็นการตั้งใจใช้คำสั่ง SELECT ... WHERE id =... เพื่อดึงข้อมูล user ออกมาตาม primary key ที่กำหนด ผ่าน func DB.Raw(...)

```
// GET /user/:id
func GetUser(c *gin.Context) {
    var user entity.User
    id := c.Param("id")
    if err := entity.DB().Raw("SELECT * FROM users WHERE id = ?", id).Scan(&user).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": user})
}
```

function ถัดมา ListUsers จะเป็นการ list รายการของ User ออกมา โดยแสดงการใช้ SELECT \*

ผลลัพธ์ที่เป็น รายการข้อมูลจะสามารถดึงออกมาได้อย่างถูกต้องเมื่อนำตัวแปรที่เป็น array มารับ ในตัวอย่างนี้ users เป็นตัวแปรประเภท array ของ entity.User (สังเกต []entity.User)

```
// GET /users
func ListUsers(c *gin.Context) {
    var users []entity.User
    if err := entity.DB().Raw("SELECT * FROM users").Scan(&users).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": users})
}
```

function ถัดมาเป็น function สำหรับลบ user ด้วย ID ก็คือการ DELETE ... WHERE ID=...

```
// DELETE /users/:id
func DeleteUser(c *gin.Context) {
    id := c.Param("id")
    if tx := entity.DB().Exec("DELETE FROM users WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user not found"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"data": id})
}
```

และ function สุดท้ายคือ function สำหรับ update user ก็คือการ UPDATE ... WHERE ID=... ในตัวอย่างใช้คำสั่ง DB.Save() แทน update ของ SQL

```
// PATCH /users
func UpdateUser(c *gin.Context) {
    var user entity.User
    if err := c.ShouldBindJSON(&user); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    if tx := entity.DB().Where("id = ?", user.ID).First(&user); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user not found"})
        return
    }
}
```

```

}
if err := entity.DB().Save(&user).Error; err != nil {
c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
return
}
c.JSON(http.StatusOK, gin.H{"data": user})
}

```

เมื่อเราเขียน function ทั้งหมดที่เป็น CRUD แล้วก็นำมาประกาศเป็น path ของ API ด้วย router ในไฟล์ main.go ซึ่งจะอยู่ใน directory นอกสุด (ตำแหน่งเดียวกับที่ go.mod อยู่) สร้างไฟล์ main.go และเขียนโปรแกรมสำหรับสร้าง Server ดังต่อไปนี้

อย่าลืมแก้ [github.com/chanwit/sa-66-example](https://github.com/chanwit/sa-66-example) เป็นชื่ออื่น ๆ ที่ระบุไว้ใน go.mod และใน user.go

```

package main
import (
    "github.com/gin-gonic/gin"
    "github.com/tanapon395/sa-66-example/controller"
    "github.com/tanapon395/sa-66-example/entity"
)
func main() {
    entity.SetupDatabase()
    r := gin.Default()
    r.Use(CORSMiddleware())
    // User Routes
    r.GET("/users", controller.ListUsers)
    r.GET("/user/:id", controller.GetUser)
    r.POST("/users", controller.CreateUser)
    r.PATCH("/users", controller.UpdateUser)
    r.DELETE("/users/:id", controller.DeleteUser)
    // Run the server
    r.Run()
}
func CORSMiddleware() gin.HandlerFunc {
    return func(c *gin.Context) {
        c.Writer.Header().Set("Access-Control-Allow-Origin", "*")
        c.Writer.Header().Set("Access-Control-Allow-Credentials", "true")
    }
}

```



```

c.Writer.Header().Set("Access-Control-Allow-Headers", "Content-Type, Content-Length,
Accept-Encoding, X-CSRF-Token, Authorization, accept, origin, Cache-Control, X-
Requested-With")
c.Writer.Header().Set("Access-Control-Allow-Methods", "POST, OPTIONS, GET, PUT")
if c.Request.Method == "OPTIONS" {
c.AbortWithStatus(204)
return
}
c.Next()
}
}

```

สั่ง download dependency ด้วยคำสั่ง

```
go mod tidy
```

แล้วสั่ง compile โปรแกรม backend ด้วยคำสั่ง

```
go build -o main.exe main.go
```

หรือ

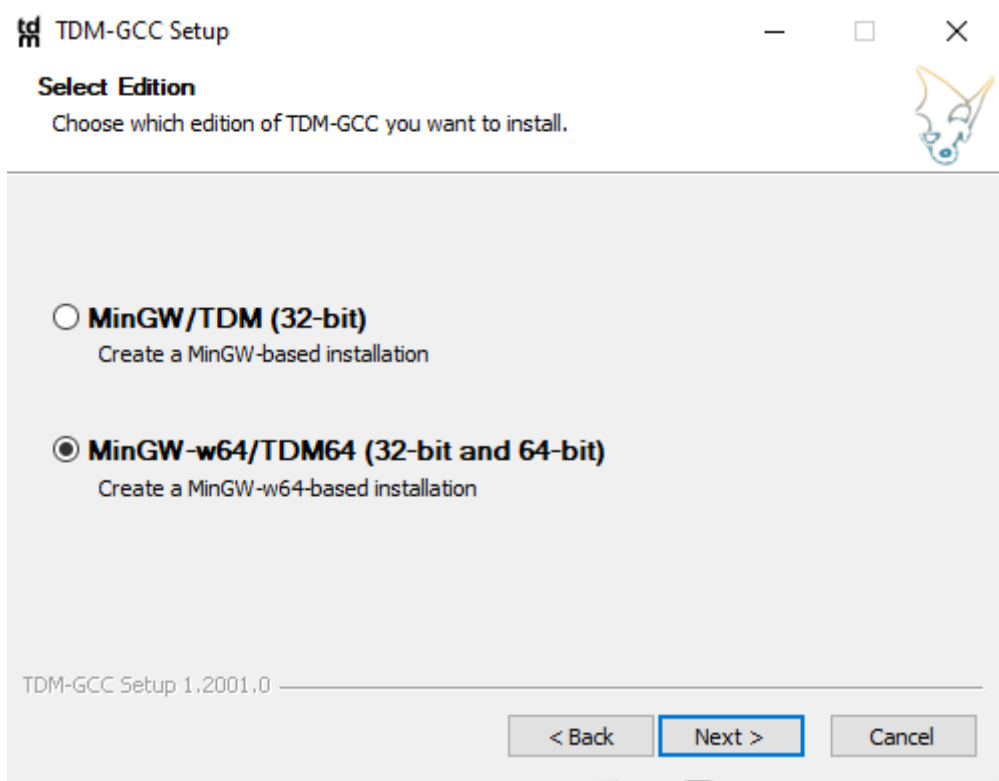
```
go build -o main main.go บน Linux และ macOS
```

\* ปัญหาที่พบ

- ถ้า error ให้ลง GCC \*\* exec: "gcc": executable file not found in %PATH%

ติดตั้ง GCC compiler: <https://github.com/jmeubank/tdm-gcc/releases/download/v9.2.0-tdm64-1/tdm64-gcc-9.2.0.exe>

Create > Slect MinGW-W64/TDM64 (32-bit and 64-bit) > Browse C:\TDM-GCC-64 > Check gcc, gdb32 > Install



- กรณีเจอปัญหา “unimplimented : 64-bit mode not compiled in”

```
PS C:\sa-66-example\backend> go build -o main.exe main.go
# github.com/mattn/go-sqlite3
cc1.exe: sorry, unimplemented: 64-bit mode not compiled in
PS C:\sa-66-example\backend>
```

ทำตามนี้ video นี้ : <https://youtu.be/Zcy981HhGw0?t=72>

\* หลังจากติดตั้ง MinGW แล้วสำหรับคนที่ไม่มี TDM อยู่ในเครื่องให้ลบ Path ของ TDM

เมื่อโปรแกรม compile ได้อย่างถูกต้องแล้ว  
รันโปรแกรมโดยการรันคำสั่ง main

.\main.exe

หรือ

./main บน Linux และ macOS

## Frontend

update version ของ npm ให้เป็นรุ่นล่าสุด

npm install -g npm@latest

Install : yarn

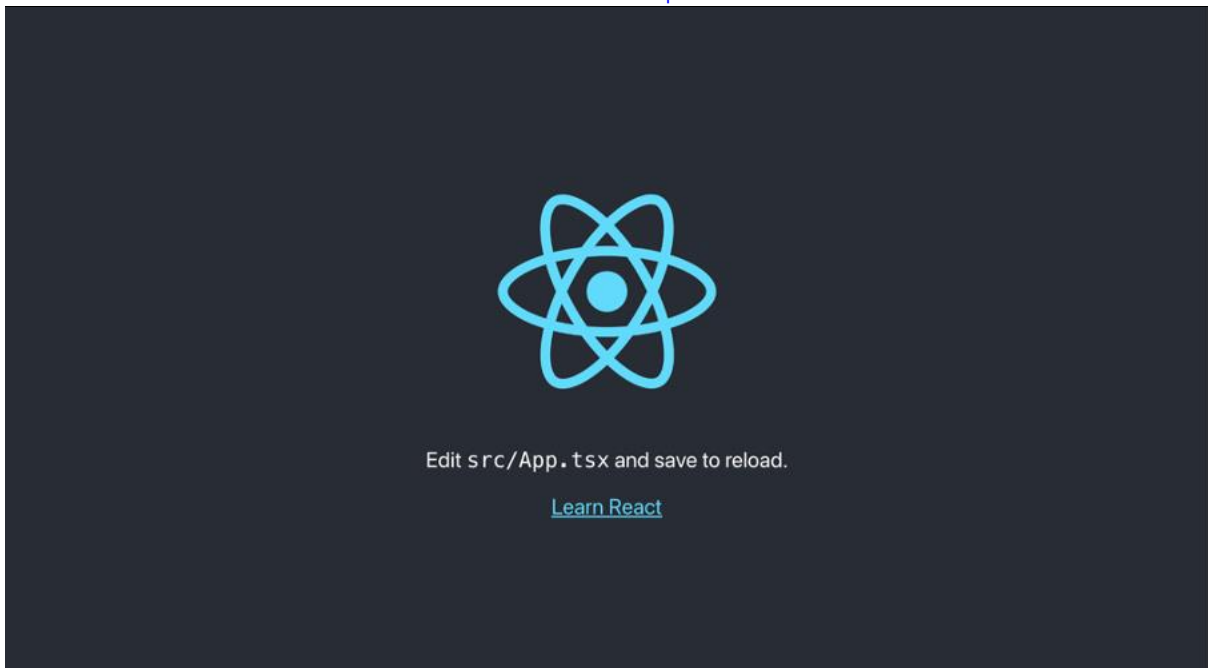
<https://classic.yarnpkg.com/en/docs/install/#windows-stable>

### 1. สร้างโปรเจค react

ป้อนคำว่า **frontend** เป็นชื่อ app

```
npx create-react-app <<ชื่อโปรเจค>> --template typescript  
cd <<ชื่อโปรเจค>>  
npm start
```

### 2. เปิด Browser เพื่อทดสอบผลการ run <http://localhost:3000/>



### 3. ติดตั้ง Package ที่ต้องใช้งาน

การติดตั้ง package ต่างๆจะต้องอยู่ที่ directory <<ชื่อโปรเจค>> ที่เราสร้าง

ติดตั้ง package สำหรับจัดการ Router

```
npm install --save react-router-dom@6.x
```

### 4. ติดตั้ง Ant Design ( <https://ant.design/docs/react/use-with-create-react-app> )

สำหรับใช้งาน component ต่างๆ เช่น Button, Card, Table เป็นต้น

npm install antd --save

หรือ

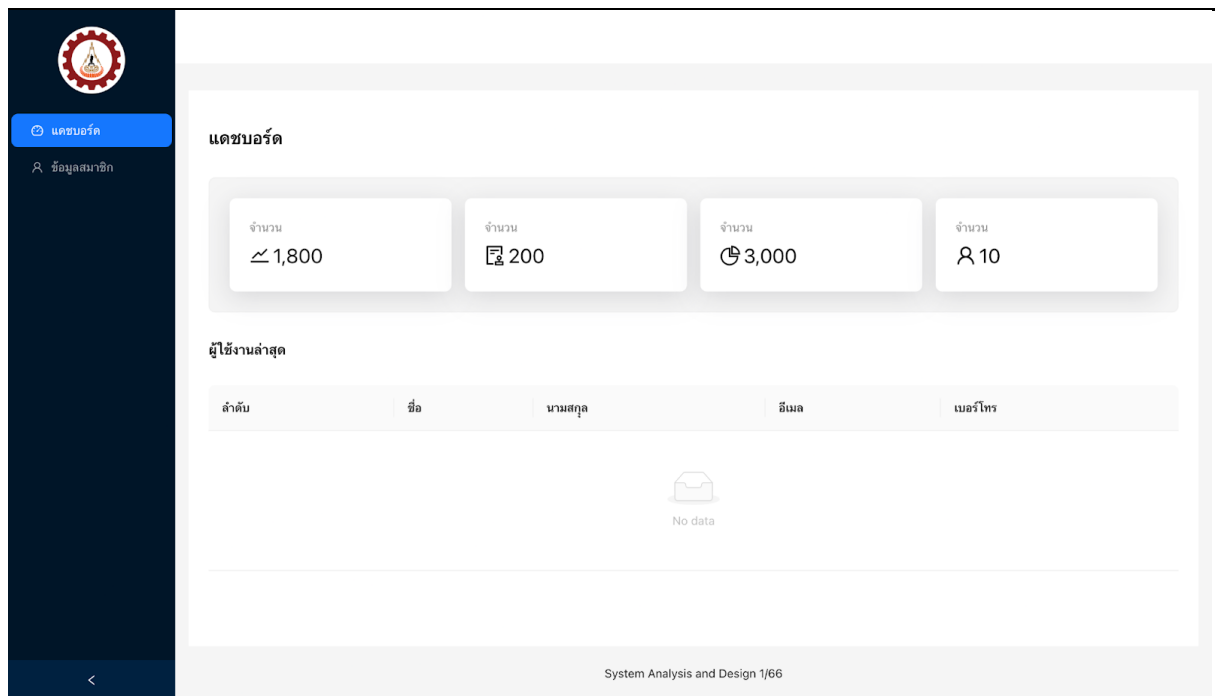
yarn add antd

5. ปรับปรุง source code frontend เพื่อทดสอบการเชื่อมต่อกับ backend สำหรับส่งข้อมูลและรับข้อมูล

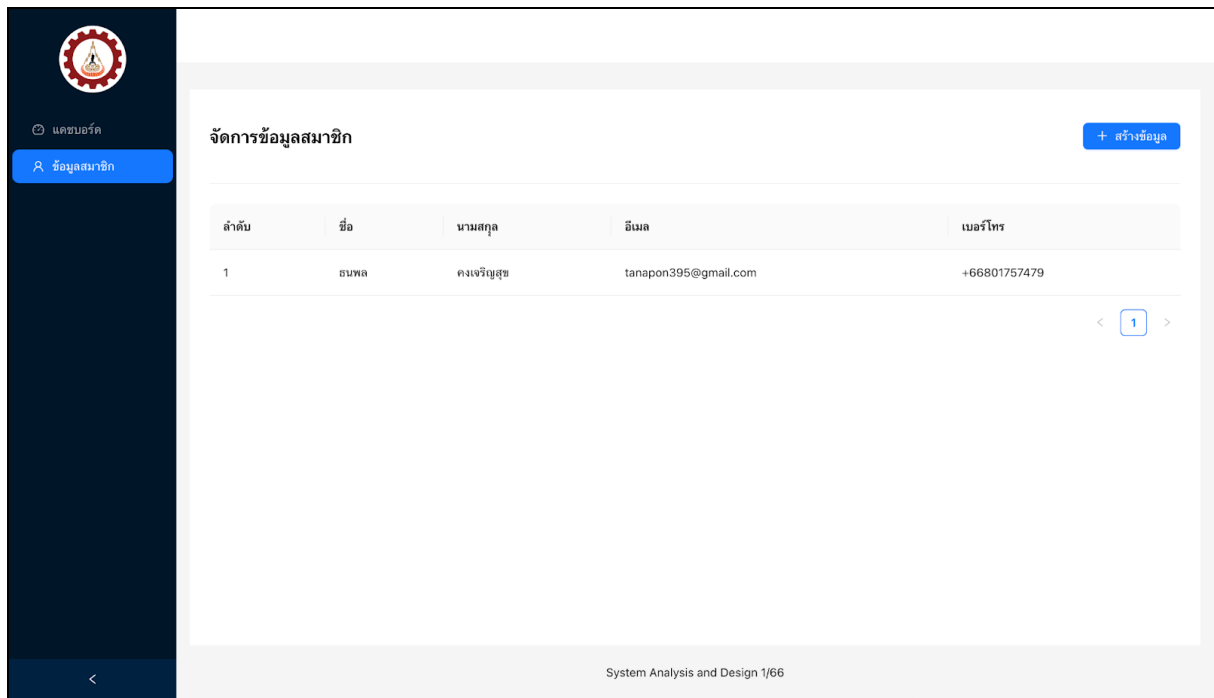
หน้าตาเว็บที่เราจะทำเป็นดังนี้

- หน้าแดชบอร์ดเป็นการจำลองข้อมูล
- หน้าข้อมูลสมาชิกเป็นหน้า แสดงข้อมูลผู้ใช้ ที่ GET ข้อมูลจาก Database
- หน้าเพิ่มข้อมูลผู้ใช้เป็นหน้าสำหรับ Insert ข้อมูล User เข้าไปที่ Database

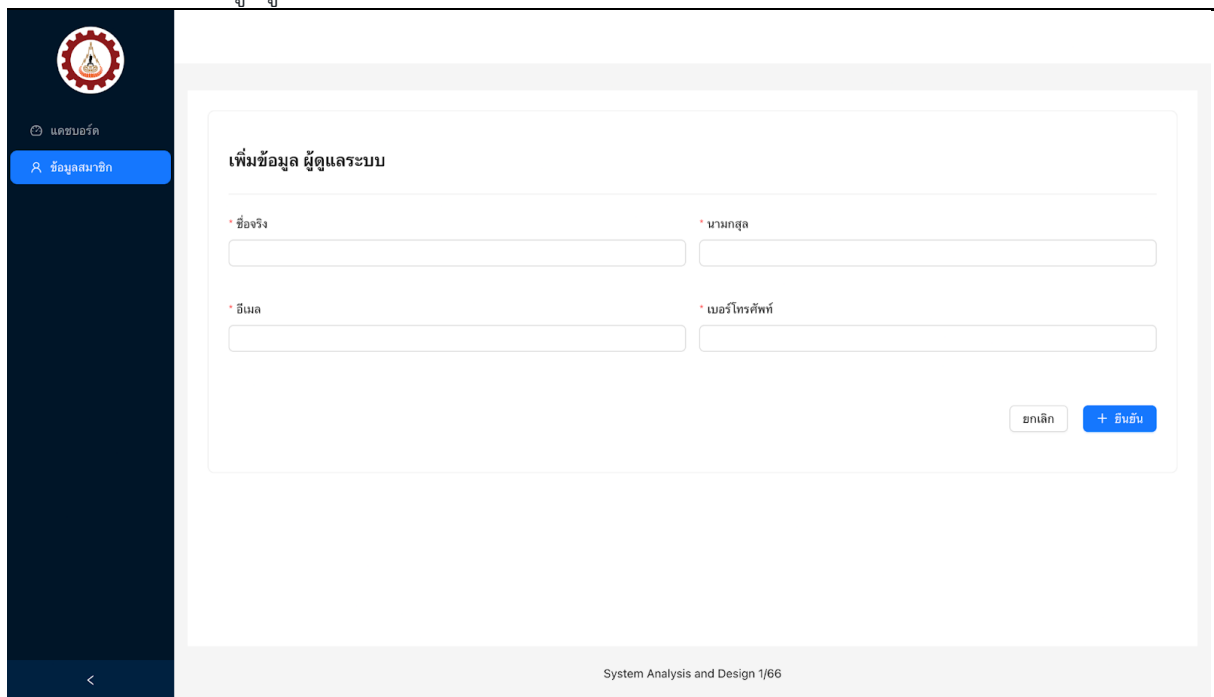
หน้าแดชบอร์ด



หน้าแสดงข้อมูลผู้ใช้แบบตาราง



หน้าเพิ่มข้อมูลผู้ใช้



เมื่อเห็นภาพตัวอย่างของระบบแล้วเราก็จะเริ่ม การแก้ไข source code ของโปรแกรมเรา

โครงสร้างไฟล์ที่เราต้องสร้างใน directory frontend

```

./frontend
  /...
  /src
    /...
    /App.tsx
    /assets
      logo.png
    /interfaces
      IUser.ts
    /pages
      /dashboard
        index.tsx
      /customer
        /create
          index.ts
x
      index.tsx
    /service
      /https
        index.tsx

```

```

✓ src
  ✓ assets
    | 🖼️ logo.png
  ✓ interfaces
    | TS IUser.ts
  ✓ pages
    | ✓ customer
    |   | ✓ create
    |   |   | TS index.tsx
    |   | TS index.tsx
    |   ✓ dashboard
    |   | TS index.tsx
    | ✓ services / https
    |   TS index.tsx

```

- เริ่มต้นด้วยการจัดการ Router โดยการแก้ไขไฟล์ **App.tsx** จะอยู่ใน directory ชื่อ src

```

import React, { useState } from "react";
import { UserOutlined, DashboardOutlined } from "@ant-design/icons";
import type { MenuProps } from "antd";
import {
  BrowserRouter as Router,
  Routes,
  Route,
  Link,
} from "react-router-dom";
import { Breadcrumb, Layout, Menu, theme } from "antd";

```

```

import logo from "./assets/logo.png";
import Dashboard from "./pages/dashboard";
import Customer from "./pages/customer";
import CustomerCreate from "./pages/customer/create";
const { Header, Content, Footer, Sider } = Layout;
type MenuItem = Required<MenuProps>["items"][number];
function getItem(
  label: React.ReactNode,
  key: React.Key,
  icon?: React.ReactNode,
  children?: MenuItem[]
): MenuItem {
  return {
    key,
    icon,
    children,
    label,
  } as MenuItem;
}
const items: MenuItem[] = [
  getItem("แดชบอร์ด", "1", <DashboardOutlined />),
  getItem("ข้อมูลสมาชิก", "2", <UserOutlined />),
];
const App: React.FC = () => {
  const page = localStorage.getItem("page");
  const [collapsed, setCollapsed] = useState(false);
  const {
    token: { colorBgContainer },
  } = theme.useToken();
  const setCurrentPage = (val: string) => {
    localStorage.setItem("page", val);
  };
  return (
    <Router>
    <Layout style={{ minHeight: "100vh" }}>
    <Sider
      collapsible
      collapsed={collapsed}

```

```
onCollapse={(value) => setCollapsed(value)}
>
<div
style={{
display: "flex",
justifyContent: "center",
marginTop: 20,
marginBottom: 20,
}}
>
<img
src={logo}
alt="Logo"
style={{ width: "40%", borderRadius: "50%" }}
/>
</div>
<Menu
theme="dark"
defaultSelectedKeys={[page ? page : "dashboard"]}
mode="inline"
>
<Menu.Item key="dashboard" onClick={() => setCurrentPage("dashboard")}>
<Link to="/">
<DashboardOutlined />
<span>แดชบอร์ด</span>
</Link>
</Menu.Item>
<Menu.Item key="customer" onClick={() => setCurrentPage("customer")}>
<Link to="/customer">
<UserOutlined />
<span>ข้อมูลสมาชิก</span>
</Link>
</Menu.Item>
</Menu>
</Sider>
<Layout>
<Header style={{ padding: 0, background: colorBgContainer }} />
<Content style={{ margin: "0 16px" }}>
```



```

<Breadcrumb style={{ margin: "16px 0" }} />
<div
style={{
padding: 24,
minHeight: "100%",
background: colorBgContainer,
}}
>
<Routes>
<Route path="/" element={<Dashboard />} />
<Route path="/customer" element={<Customer />} />
<Route path="/customer/create" element={<CustomerCreate />} />
</Routes>
</div>
</Content>
<Footer style={{ textAlign: "center" }}>
System Analysis and Design 1/66
</Footer>
</Layout>
</Layout>
</Router>
);
};
export default App;

```

จะเห็นว่ามี path="/" เรียกใช้ component Dashboard ที่ import มาจาก ./pages/dashboard คือหน้าแสดงข้อมูลแดชบอร์ด

path="/customer" เรียกใช้ component Customer ที่ import มาจาก ./pages/customer คือหน้าแสดงข้อมูลผู้ใช้แบบตาราง

และ path="/customer/create" เรียกใช้ component CustomerCreate ที่ import มาจาก ./pages/customer/create คือหน้าสร้างข้อมูลผู้ใช้

ส่วนของ logo เรา import จาก ./assets/logo.png



---

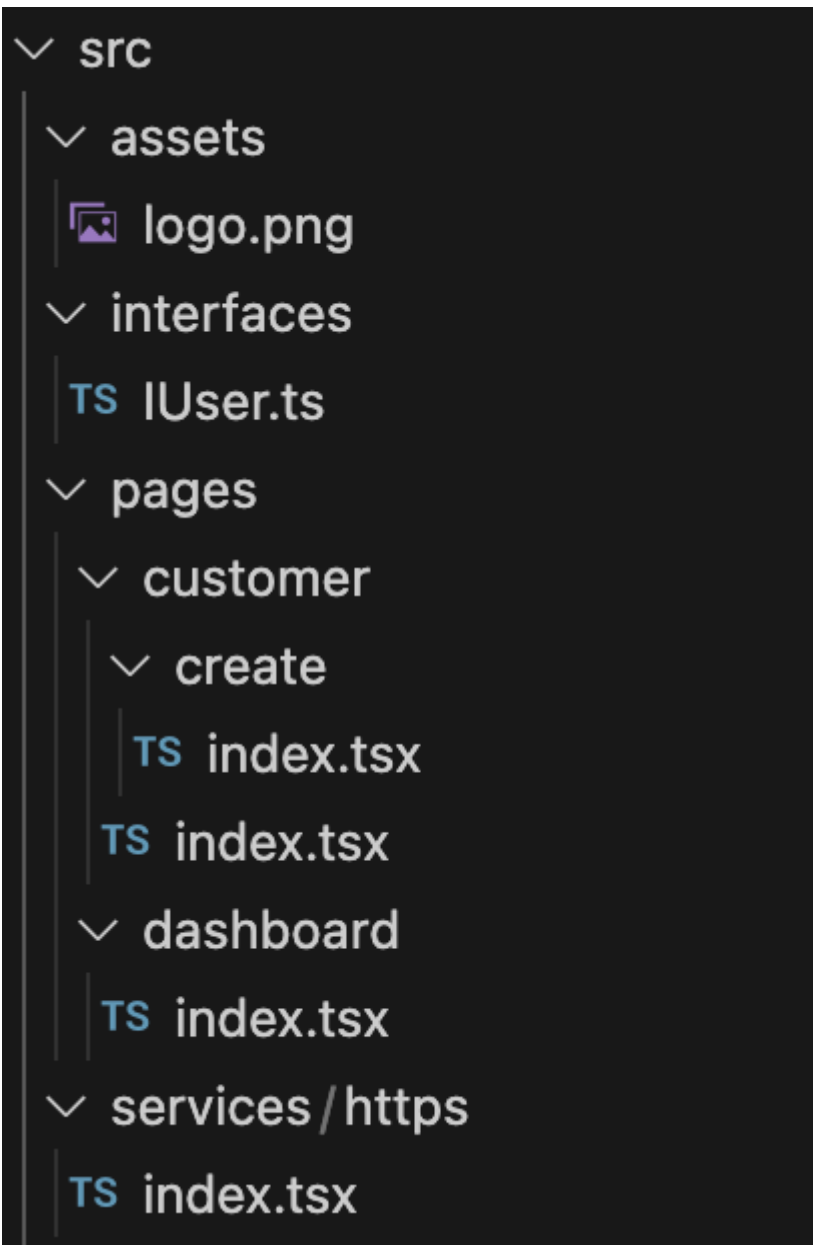
ทำการบันทึกภาพ ตั้งชื่อ logo.png เก็บไว้ที่โฟลเดอร์ assets

- สร้าง User Interface เพื่อประกาศโครงสร้างข้อมูลของ User ว่ามี field ชื่ออะไร และชนิดข้อมูลเป็นอะไร

สร้าง directory ชื่อ interfaces ใน directory src จากนั้นสร้างไฟล์ **IUser.ts**

```
export interface UsersInterface {  
  ID?: number;  
  FirstName?: string;  
  LastName?: string;  
  Email?: string;  
  Phone?: string;  
}
```

- สร้าง directory ชื่อ pages ใน directory src และสร้างไฟล์ต่อไปนี้



/src/pages

สร้าง directory ชื่อ dashboard และ สร้างไฟล์ **index.tsx** ดังนี้

```
import { Col, Row, Card, Statistic, Space, Table, Tag } from "antd";
import {
  AuditOutlined,
  UserOutlined,
  PieChartOutlined,
  StockOutlined,
} from "@ant-design/icons";
import type { ColumnsType } from "antd/es/table";
interface DataType {
```

```
key: string;
name: string;
age: number;
address: string;
tags: string[];
}

const columns: ColumnsType<DataType> = [
{
  title: "ลำดับ",
  dataIndex: "ID",
  key: "id",
},
{
  title: "ชื่อ",
  dataIndex: "FirstName",
  key: "firstname",
},
{
  title: "นามสกุล",
  dataIndex: "LastName",
  key: "lastname",
},
{
  title: "อีเมล",
  dataIndex: "Email",
  key: "email",
},
{
  title: "เบอร์โทร",
  dataIndex: "Phone",
  key: "phone",
},
];

const data: DataType[] = [];
export default function index() {
  return (
    <>
    <Row gutter={[16, 16]}>
```

```

<Col xs={24} sm={24} md={24} lg={24} xl={24}>
<h2>แดชบอร์ด</h2>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={24}>
<Card style={{ backgroundColor: "#F5F5F5" }}>
<Row gutter=[[16, 16]]>
<Col xs={24} sm={24} md={12} lg={12} xl={6}>
<Card
bordered={false}
style={{
boxShadow: "rgba(100, 100, 111, 0.2) 0px 7px 29px 0px",
}}
>
<Statistic
title="จำนวน"
value={1800}
prefix={<StockOutlined />}
/>
</Card>
</Col>
<Col xs={24} sm={24} md={12} lg={12} xl={6}>
<Card
bordered={false}
style={{
boxShadow: "rgba(100, 100, 111, 0.2) 0px 7px 29px 0px",
}}
>
<Statistic
title="จำนวน"
value={200}
valueStyle={{ color: "black" }}
prefix={<AuditOutlined />}
/>
</Card>
</Col>
<Col xs={24} sm={24} md={12} lg={12} xl={6}>
<Card
bordered={false}

```

```

style={{
  boxShadow: "rgba(100, 100, 111, 0.2) 0px 7px 29px 0px",
}}
>
<Statistic
  title="จำนวน"
  value={3000}
  valueStyle={{ color: "black" }}
  prefix={<PieChartOutlined />}
/>
</Card>
</Col>
<Col xs={24} sm={24} md={12} lg={12} xl={6}>
<Card
  bordered={false}
  style={{
    boxShadow: "rgba(100, 100, 111, 0.2) 0px 7px 29px 0px",
  }}
  >
    <Statistic
      title="จำนวน"
      value={10}
      valueStyle={{ color: "black" }}
      prefix={<UserOutlined />}
    />
  </Card>
</Col>
</Row>
</Card>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={24}>
<h3>ผู้ใช้งานล่าสุด</h3>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={24}>
<Table columns={columns} dataSource={data} />
</Col>
</Row>
</>

```

```
);  
}
```

/src/pages

สร้าง directory ชื่อ customer และ สร้างไฟล์ **index.tsx** ดังนี้

```
import React, { useState, useEffect } from "react";  
import { Space, Table, Button, Col, Row, Divider } from "antd";  
import { PlusOutlined } from "@ant-design/icons";  
import type { ColumnsType } from "antd/es/table";  
import { GetUsers } from "../../services/https";  
import { UsersInterface } from "../../interfaces/IUser";  
import { Link } from "react-router-dom";  
const columns: ColumnsType<UsersInterface> = [  
  {  
    title: "ลำดับ",  
    dataIndex: "ID",  
    key: "id",  
  },  
  {  
    title: "ชื่อ",  
    dataIndex: "FirstName",  
    key: "firstname",  
  },  
  {  
    title: "นามสกุล",  
    dataIndex: "LastName",  
    key: "lastname",  
  },  
  {  
    title: "อีเมล",  
    dataIndex: "Email",  
    key: "email",  
  },  
  {  
    title: "เบอร์โทร",  
    dataIndex: "Phone",  
    key: "phone",  
  },  
];
```

```

},
];
function Customers() {
  const [users, setUsers] = useState<UsersInterface[]>([]);
  const getUsers = async () => {
    let res = await GetUsers();
    if (res) {
      setUsers(res);
    }
  };
  useEffect(() => {
    getUsers();
  }, []);
  return (
    <>
    <Row>
    <Col span={12}>
    <h2>จัดการข้อมูลสมาชิก</h2>
    </Col>
    <Col span={12} style={{ textAlign: "end", alignSelf: "center" }}>
    <Space>
    <Link to="/customer/create">
    <Button type="primary" icon={<PlusOutlined />}>
    สร้างข้อมูล
    </Button>
    </Link>
    </Space>
    </Col>
    </Row>
    <Divider />
    <div style={{ marginTop: 20 }}>
    <Table rowKey="ID" columns={columns} dataSource={users} />
    </div>
    </>
  );
}
export default Customers;

```



/src/pages/customer

สร้าง directory ชื่อ create และ สร้างไฟล์ **index.tsx** ดังนี้

```
import React from "react";
import {
  Space,
  Button,
  Col,
  Row,
  Divider,
  Form,
  Input,
  Card,
  message,
} from "antd";
import { PlusOutlined } from "@ant-design/icons";
import { UsersInterface } from "../../interfaces/IUser";
import { CreateUser } from "../../services/https";
import { useNavigate } from "react-router-dom";
function CustomerCreate() {
  const navigate = useNavigate();
  const [messageApi, contextHolder] = message.useMessage();
  const onFinish = async (values: UsersInterface) => {
    let res = await CreateUser(values);
    if (res.status) {
      messageApi.open({
        type: "success",
        content: "บันทึกข้อมูลสำเร็จ",
      });
      setTimeout(function () {
        navigate("/customer");
      }, 2000);
    } else {
      console.log(res);
    }
  };
  return (
    <div>
```

```
{contextHolder}
<Card>
<h2> เพิ่มข้อมูล ผู้ดูแลระบบ</h2>
<Divider />
<Form
name="basic"
layout="vertical"
onFinish={onFinish}
autoComplete="off"
>
<Row gutter=[[16, 16]]>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
<Form.Item
label="ชื่อจริง"
name="FirstName"
rules=[[
{
required: true,
message: "กรุณากกรอกชื่อ !",
},
]]
>
<Input />
</Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
<Form.Item
label="นามสกุล"
name="LastName"
rules=[[
{
required: true,
message: "กรุณากกรอกนามสกุล !",
},
]]
>
<Input />
</Form.Item>
```

```
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
<Form.Item
label="อีเมล"
name="Email"
rules={[
{
type: "email",
message: "รูปแบบอีเมลไม่ถูกต้อง !",
},
{
required: true,
message: "กรุณากรอกอีเมล !",
},
]}
>
<Input />
</Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
<Form.Item
label="เบอร์โทรศัพท์"
name="Phone"
rules={[
{
required: true,
message: "กรุณากรอกเบอร์โทรศัพท์ !",
},
]}
>
<Input />
</Form.Item>
</Col>
</Row>
<Row justify="end">
<Col style={{ marginTop: "40px" }}>
<Form.Item>
<Space>
```

```

<Button htmlType="button" style={{ marginRight: "10px" }}>
ยกเลิก
</Button>
<Button
type="primary"
htmlType="submit"
icon={<PlusOutlined />}
>
ยืนยัน
</Button>
</Space>
</Form.Item>
</Col>
</Row>
</Form>
</Card>
</div>
);
}
export default CustomerCreate;

```

- สร้าง directory ชื่อ services ใน directory src และสร้างไฟล์ต่อไปนี้

/src/services

สร้าง directory ชื่อ http และ สร้างไฟล์ **index.tsx** ดังนี้

```

import { UsersInterface } from "../../interfaces/IUser";
const apiUrl = "http://localhost:8080";
async function GetUsers() {
const requestOptions = {
method: "GET",
headers: {
"Content-Type": "application/json",
},
};
let res = await fetch(`${apiUrl}/users`, requestOptions)
.then((response) => response.json())
.then((res) => {
if (res.data) {
return res.data;
}
}
}

```

```

} else {
return false;
}
});
return res;
}

async function CreateUser(data: UsersInterface) {
const requestOptions = {
method: "POST",
headers: { "Content-Type": "application/json" },
body: JSON.stringify(data),
};
let res = await fetch(`${apiUrl}/users`, requestOptions)
.then((response) => response.json())
.then((res) => {
if (res.data) {
return { status: true, message: res.data };
} else {
return { status: false, message: res.error };
}
});
return res;
}

export {
GetUsers,
CreateUser,
};

```

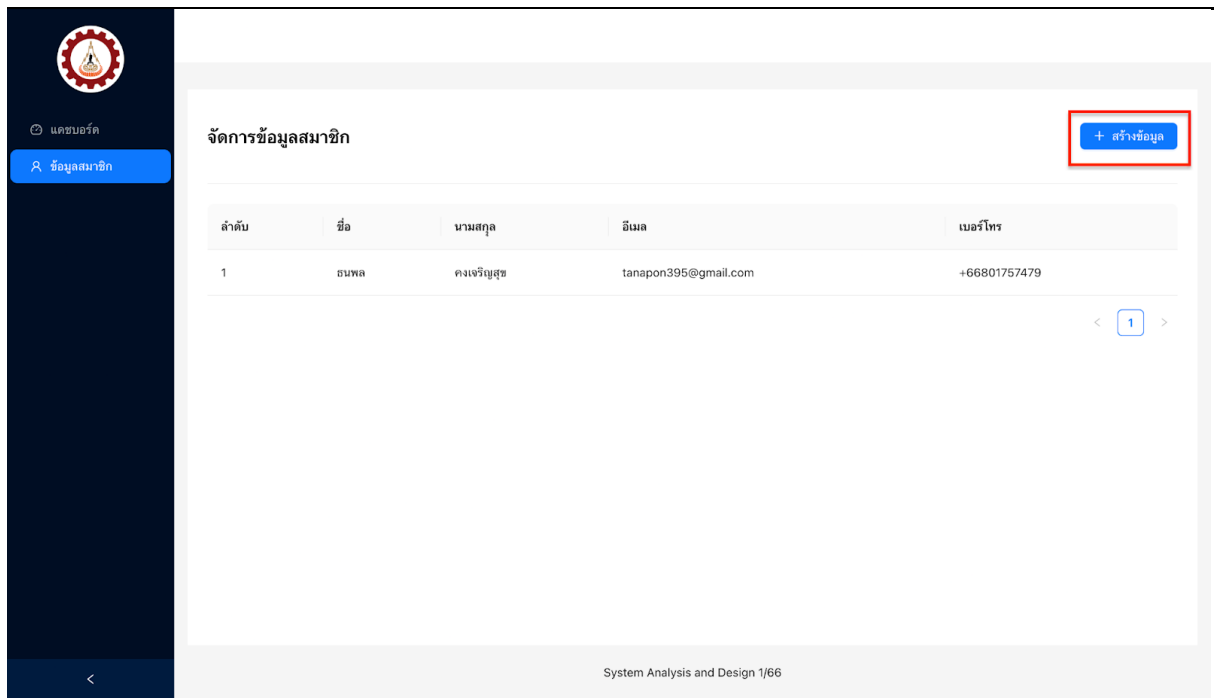
6. สั่ง npm start ที่ directory frontend เพื่อ start frontend

กดปุ่ม Create User เพื่อเพิ่มข้อมูล User จากนั้นกลับมาที่หน้าหลัก ก็จะพบกับข้อมูลที่เพิ่มเข้าไป

\*\*\*\*อย่าลืม run backend\*\*\*\*

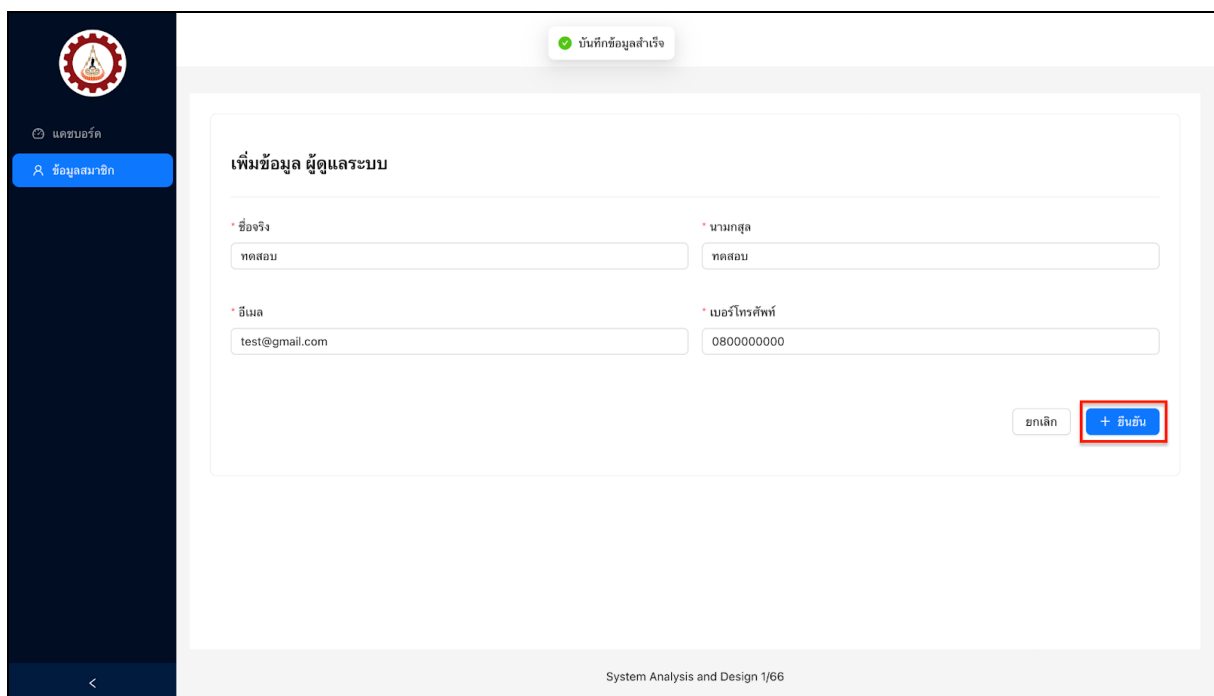
7. ทดสอบเพิ่มข้อมูล และแสดงผลข้อมูล

เปิดเว็บเบราว์เซอร์ที่ <http://localhost:3000>




\*หมายเหตุ : ข้อมูลที่แสดงเป็นข้อมูลจากการ create user

ให้ดำเนินการเพิ่มข้อมูล โดยกดที่ปุ่ม create user -> กรอกข้อมูลให้ครบถ้วนจากนั้นกด submit



เมื่อบันทึกสำเร็จ ระบบจะ navigate ไปยังหน้าแสดงข้อมูลทั้งหมด



แดชบอร์ด

ข้อมูลสมาชิก

จัดการข้อมูลสมาชิก

+ สร้างข้อมูล

ลำดับ	ชื่อ	นามสกุล	อีเมล	เบอร์โทร
1	ธนพล	คงเจริญสุข	tanapon395@gmail.com	+66801757479
2	ทดสอบ	ทดสอบ	test@gmail.com	0800000000

< 1 >

System Analysis and Design 1/66

----- จบ -----