


Tutorial สำหรับปฏิบัติการ 2

สร้าง directory Project ชื่อ sa-64-example

```
cd c:\  
mkdir sa-64-example  
cd sa-64-example
```

1. ติดตั้ง Go compiler โดย download ได้จากที่นี่ <https://golang.org/dl/>

DocumentsPackagesThe ProjectHelpBlogPlay

Downloads

[Featured downloads](#)
[Stable versions](#)
[Unstable version](#)

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.

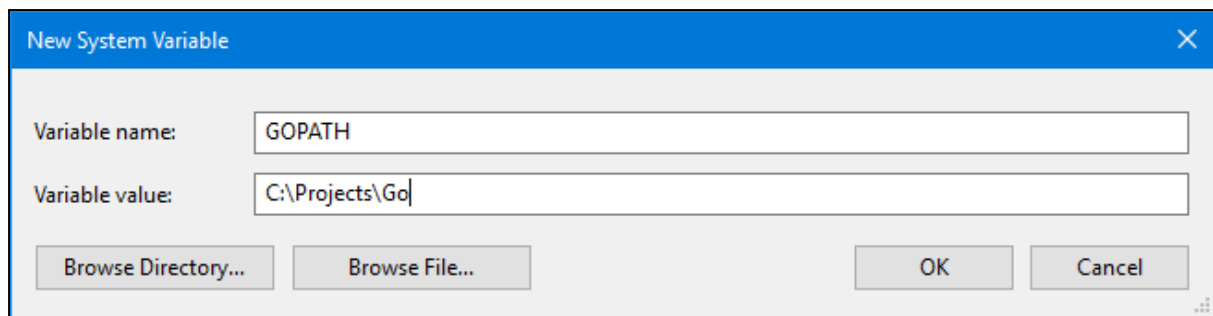
Featured downloads

Microsoft Windows <i>Windows 7 or later, Intel 64-bit processor</i> go1.16.7.windows-amd64.msi (119MB)	Apple macOS <i>macOS 10.12 or later, Intel 64-bit processor</i> go1.16.7.darwin-amd64.pkg (125MB)
Linux <i>Linux 2.6.23 or later, Intel 64-bit processor</i> go1.16.7.linux-amd64.tar.gz (123MB)	Source go1.16.7.src.tar.gz (20MB)

Stable versions

go1.16.7 ▼

ตั้งค่า Environment ดังนี้



GOROOT ตั้งเป็น c:\Go

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

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

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

2. ติดตั้ง VS Code

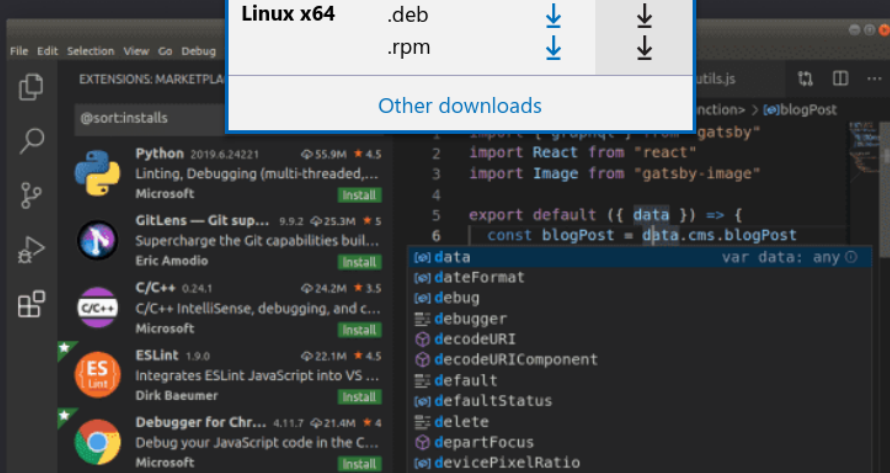
Visual Studio Code Docs Updates Blog API Extensions FAQ

Version 1.47 is now available! Read about the new features and fixes from June.

Code editing. Redefined.

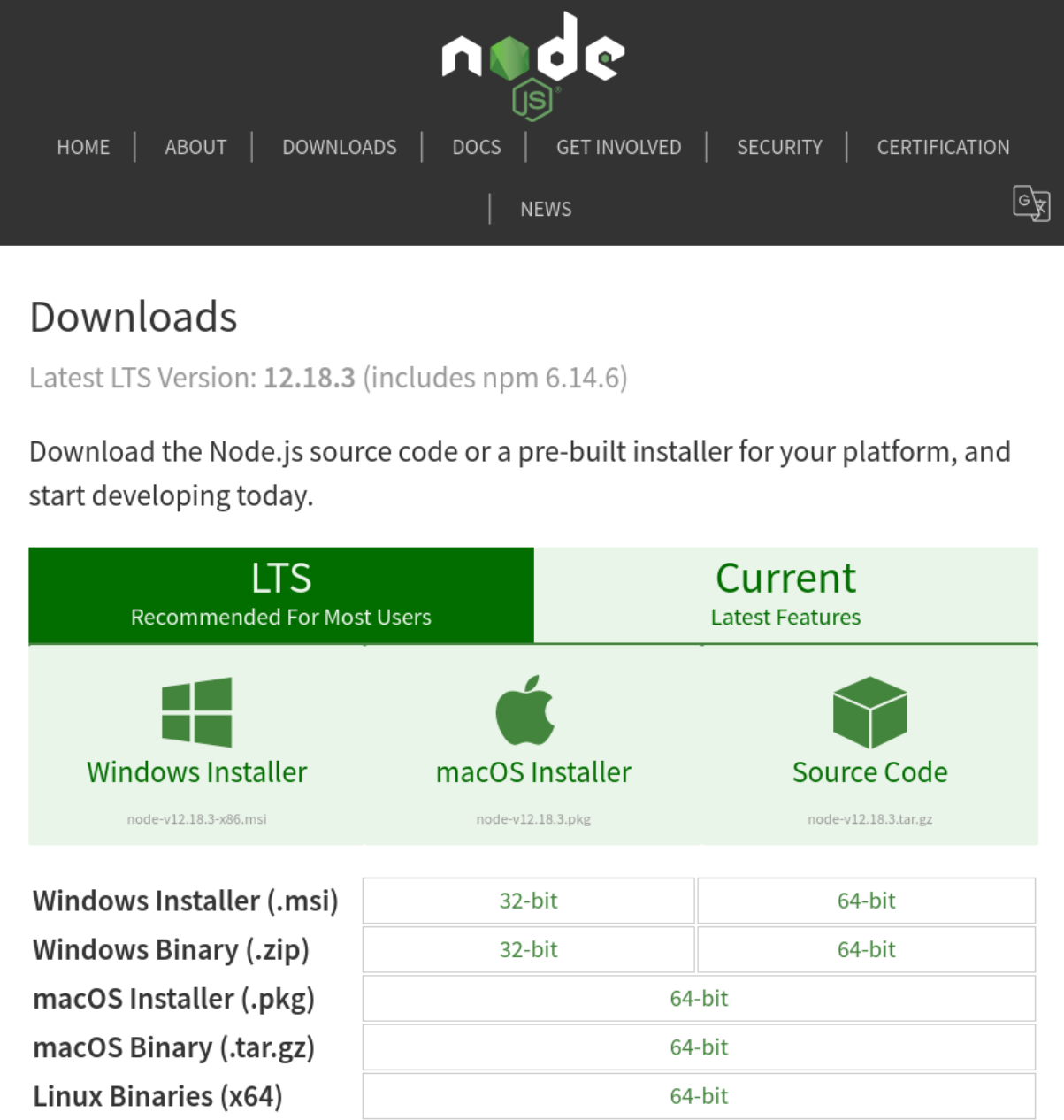
Free. Built on open source. Runs everywhere.

↓ .deb Debian, Ubuntu...		↓ .rpm Red Hat, Fedora...		↓	
		Stable	Insiders		
macOS	Package	↓	↓		
Windows x64	User Installer	↓	↓		
Linux x64	.deb	↓	↓		
	.rpm	↓	↓		
Other downloads					



เลือก Extension ติดตั้ง extension สำหรับภาษา Go




3. ติดตั้ง nodejs (เราจะใช้ v14.x ในเทอม 1/64)



Downloads

Latest LTS Version: **12.18.3** (includes npm 6.14.6)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features	
 Windows Installer node-v12.18.3-x86.msi	 macOS Installer node-v12.18.3.pkg	 Source Code node-v12.18.3.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	
Linux Binaries (x64)	64-bit	

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-64-example

```
cd c:\sa-64-example
```

```
mkdir backend
```

```
cd backend
```

```
go mod init github.com/<ชื่อ user ที่ได้มาจากการสมัคร GitHub>/sa-64-example
```

```
เช่น ของอาจารย์จะเป็น github.com/chanwit/sa-64-example
```

7. ติดตั้ง GORM (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
```

```
mkdir entity
cd entity/
```

8. ใน c:\sa-64-example\backend\entity สร้าง schema ของ User ในไฟล์ชื่อ **user.go**

แก้ไขไฟล์ **user.go** ให้เป็นแบบนี้

```
package entity

import (
    "time"
    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    FirstName string
    LastName  string
    Email     string
    Age       uint8
    BirthDay  time.Time
}
```

จากตำแหน่ง folder เดียวกัน เราจะเตรียมไฟล์ถัดมาคือไฟล์ **setup.go** เพื่อใช้สร้าง database

```
package entity

import (
    "gorm.io/gorm"
    "gorm.io/driver/sqlite"
)

var db *gorm.DB

func DB() *gorm.DB {
    return db
}
```

```

}

func SetupDatabase() {
    database, err := gorm.Open(sqlite.Open("sa-64.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-64-example\backend

1. mkdir controller
2. cd controller
3. ใน c:\sa-64-example\backend\controller สร้างไฟล์ชื่อ **user.go** เพื่อเก็บ controller สำหรับเชื่อมต่อกับ entity User

```

package controller

import (
    "github.com/<ชื่อ github id ของตนเอง>/sa-64-example/entity"
    "github.com/gin-gonic/gin"
    "net/http"
)

```

จากนั้นสร้าง function ต่อไปนี้

1. 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-64-example เป็นชื่ออื่น ๆ ที่ระบุไว้ใน go.mod และใน **user.go**

```
package main

import (
    "github.com/chanwit/sa-64-example/controller"
    "github.com/chanwit/sa-64-example/entity"
    "github.com/gin-gonic/gin"
)

func main() {
    entity.SetupDatabase()

    r := gin.Default()

    // 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()
}
```

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
```

ถ้า error ให้อ่าน 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>

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

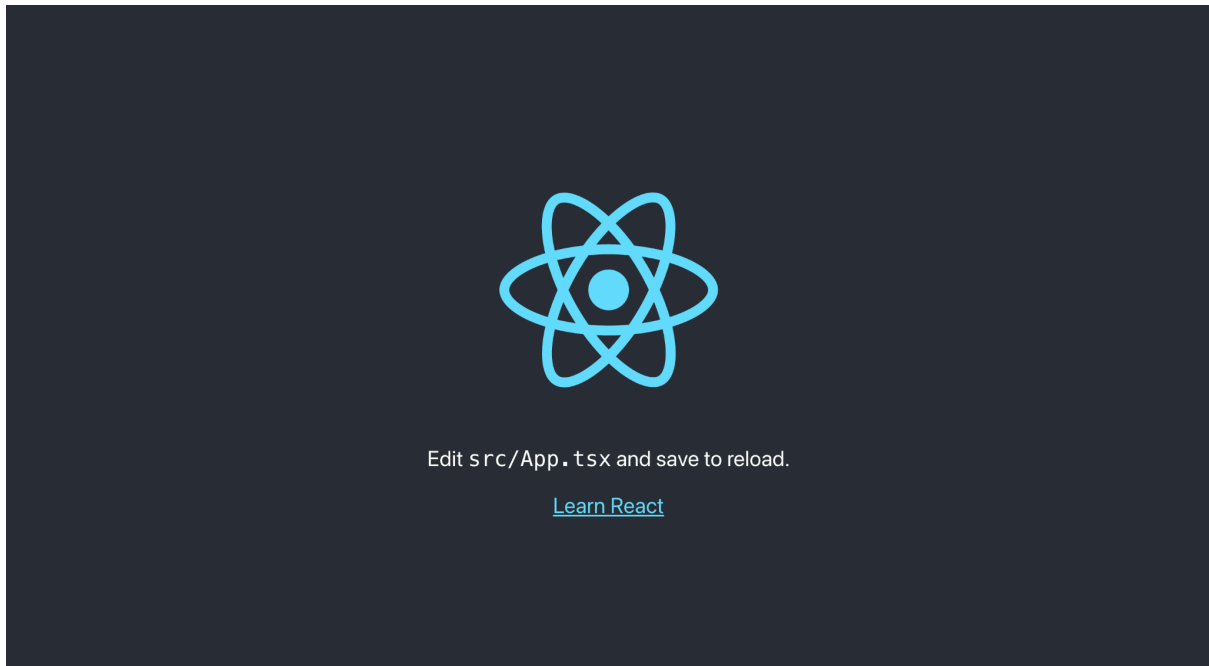
ป้อนคำว่า **frontend** เป็นชื่อ app

```
npx create-react-app <<ชื่อโปรเจค>> --template typescript
```

```
cd <<ชื่อโปรเจค>>
```

```
npm start
```

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



3. ติดตั้ง **Package** ที่ต้องใช้งาน

ติดตั้ง package สำหรับจัดการ Router

```
npm install --save @types/react-router-dom
```

```
npm install --save react-router-dom
```

4. ติดตั้ง **Material-UI** (<https://material-ui.com/getting-started/installation/>)

สำหรับใช้งาน component ต่างๆ ของ material เช่น button, box, grid เป็นต้น

```
npm install @material-ui/core หรือ yarn add @material-ui/core
```

สำหรับใช้งาน icon

```
npm install @material-ui/icons หรือ yarn add @material-ui/icons
```

สำหรับใช้งาน date picker (<https://material-ui-pickers.dev/>)

```
npm i @material-ui/pickers หรือ yarn add @material-ui/pickers
```

```
npm i @date-io/date-fns@1.x date-fns
```

สำหรับใช้งาน package เสริม เช่น alert

```
npm install @material-ui/lab หรือ yarn add @material-ui/lab
```

สำหรับจัดการ Date & Time

```
npm install moment
```

5. ปรับปรุง source code frontend เพื่อทดสอบการเชื่อมต่อกับ backend สำหรับส่งข้อมูล และรับข้อมูล

หน้าตาเว็บที่เราจะทำเป็นดังนี้

- หน้าแรกเป็นหน้า แสดงข้อมูลผู้ใช้ ที่ GET ข้อมูลจาก Database
- หน้าสำหรับ Insert ข้อมูล User เข้าไปที่ Database

หน้าแรก

System Analysis and Design					
Users					CREATE USER
ID	First	Last	Age	Email	Birth Day
1	ชาญวิทย์	แก้วกลี	100	chanwit@gmail.com	2021-08-15T00:00:00Z
2	ธนพล	คงเจริญสุข	25	tanapon@gmail.com	2021-08-15T21:40:00Z
3	สิทธิชัย	สิริฤทธิกุลชัย	26	sithichai@gmail.com	2021-08-19T21:40:00Z

หน้าเพิ่มข้อมูล User

System Analysis and Design	
<div> <div>Creaget User</div> <div> <div>First Name</div> <div>Last Name</div> </div> <div> <input type="text"/> <input type="text"/> </div> <div>Email</div> <div> <input type="text"/> </div> <div> <div>Age</div> <div>BirthDay</div> </div> <div> <input type="text"/> <div>2021-08-17</div> </div> <div> <div>BACK</div> <div>SUBMIT</div> </div> </div>	

เมื่อเห็นภาพของระบบแล้วเราก็จะเริ่ม การแก้ไข source code ของโปรแกรมเรา

โครงสร้างไฟล์ที่เราต้องสร้างใน directory frontend

```
./frontend
  /...
  /src
    /...
    /App.tsx
```

```
/models
  /IUser.ts
/components
  /Navbar.tsx
  /Users.tsx
  /UserCreate.tsx
```

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

```
import React from "react";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";

import Navbar from "../components/Navbar";
import Users from "../components/Users";
import UserCreate from "../components/UserCreate";

export default function App() {
  return (
    <Router>
      <div>
        <Navbar />
        <Switch>
          <Route exact path="/" component={Users} />
          <Route exact path="/create" component={UserCreate} />
        </Switch>
      </div>
    </Router>
  );
}
```

จะเห็นว่ามี path="/" เรียกใช้ component User ที่ import มาจาก.
/components/Users คือหน้าแสดงข้อมูล user

และ path="/create" เรียกใช้ component UserCreate ที่ import มาจาก.
/components/UserCreate คือหน้าสร้างข้อมูล user

ส่วนของ Navbar เรา import เข้ามาเพื่อใช้ทุกๆหน้า

- สร้าง User Interface เพื่อประกาศโครงสร้างข้อมูลของ User ว่ามี field ชื่ออะไร และชนิดข้อมูลเป็นอะไร

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

```
export interface UsersInterface {  
  ID: string,  
  FirstName: string;  
  LastName: string;  
  Email: string;  
  Age: number;  
  BirthDay: Date | null;  
}
```

- สร้าง directory ชื่อ components ใน directory src และสร้างไฟล์ต่อไปนี

สร้างไฟล์ **Navbar.tsx** ดังนี้

```
import React from "react";  
import { makeStyles } from "@material-ui/core/styles";  
import AppBar from "@material-ui/core/AppBar";  
import Toolbar from "@material-ui/core/Toolbar";  
import Typography from "@material-ui/core/Typography";  
import IconButton from "@material-ui/core/IconButton";  
import MenuIcon from "@material-ui/icons/Menu";  
import { Link } from "react-router-dom";  
  
const useStyles = makeStyles((theme) => ({  
  root: {flexGrow: 1},  
  menuButton: {marginRight: theme.spacing(2)},  
  title: {flexGrow: 1},  
  navlink: {color: "white",textDecoration: "none"},  
}));  
  
function Navbar() {  
  const classes = useStyles();  
  return (  
    <div className={classes.root}>  
      <AppBar position="static">  
        <Toolbar>  
          <IconButton  
            edge="start"  
            className={classes.menuButton}
```



```

        color="inherit"
        aria-label="menu"
      >
        <MenuIcon />
      </IconButton>
      <Link className={classes.navlink} to="/">
        <Typography variant="h6" className={classes.title}>
          System Analysis and Design
        </Typography>
      </Link>
    </Toolbar>
  </AppBar>
</div>
);
}
export default Navbar;

```

สร้างไฟล์ **Users.tsx** ดังนี้

```

import React, { useEffect } from "react";
import { Link as RouterLink } from "react-router-dom";
import { createStyles, makeStyles, Theme } from "@material-ui/core/styles";
import Typography from "@material-ui/core/Typography";
import Button from "@material-ui/core/Button";
import Container from "@material-ui/core/Container";
import Paper from "@material-ui/core/Paper";
import Box from "@material-ui/core/Box";
import Table from "@material-ui/core/Table";
import TableBody from "@material-ui/core/TableBody";
import TableCell from "@material-ui/core/TableCell";
import TableContainer from "@material-ui/core/TableContainer";
import TableHead from "@material-ui/core/TableHead";
import TableRow from "@material-ui/core/TableRow";
import { UsersInterface } from "../models/IUser";

import moment from 'moment';

const useStyles = makeStyles((theme: Theme) =>
  createStyles({
    container: {marginTop: theme.spacing(2)},
    table: { minWidth: 650},
    tableSpace: {marginTop: 20},

```

```

    })
  );

function Users() {
  const classes = useStyles();
  const [users, setUsers] = React.useState<UsersInterface[]>([]);

  const getUsers = async () => {
    const apiUrl = "http://localhost:8080/users";
    const requestOptions = {
      method: "GET",
      headers: { "Content-Type": "application/json" },
    };

    fetch(apiUrl, requestOptions)
      .then((response) => response.json())
      .then((res) => {
        console.log(res.data);
        if (res.data) {
          setUsers(res.data);
        } else {
          console.log("else");
        }
      });
  };

  useEffect(() => {
    getUsers();
  }, []);

  return (
    <div>
      <Container className={classes.container} maxWidth="md">
        <Box display="flex">
          <Box flexGrow={1}>
            <Typography
              component="h2"
              variant="h6"
              color="primary"
              gutterBottom
            >
              Users
            </Typography>
          </Box>
          <Box>
            <Button

```

```

        component={RouterLink}
        to="/create"
        variant="contained"
        color="primary"
      >
        Create User
      </Button>
    </Box>
  </Box>
  <TableContainer component={Paper} className={classes.tableSpace}>
    <Table className={classes.table} aria-label="simple table">
      <TableHead>
        <TableRow>
          <TableCell align="center" width="5%">
            ID
          </TableCell>
          <TableCell align="center" width="25%">
            First
          </TableCell>
          <TableCell align="center" width="25%">
            Last
          </TableCell>
          <TableCell align="center" width="5%">
            Age
          </TableCell>
          <TableCell align="center" width="20%">
            Email
          </TableCell>
          <TableCell align="center" width="20%">
            Birth Day
          </TableCell>
        </TableRow>
      </TableHead>
      <TableBody>
        {users.map((user: UsersInterface) => (
          <TableRow key={user.ID}>
            <TableCell align="right">{user.ID}</TableCell>
            <TableCell align="left" size="medium">
              {user.FirstName}
            </TableCell>
            <TableCell align="left">{user.LastName}</TableCell>
            <TableCell align="left">{user.Age}</TableCell>
            <TableCell align="left">{user.Email}</TableCell>
            <TableCell
              align="center">{moment(user.BirthDay).format("DD/MM/YYYY")}</TableCell>
            </TableCell>
          </TableRow>
        ))}
      </TableBody>
    </Table>
  </TableContainer>

```

```

    )))
  </TableBody>
</Table>
</TableContainer>
</Container>
</div>
);
}

export default Users;

```

สร้างไฟล์ **UserCreate.tsx** ดังนี้

```

import React from "react";
import { Link as RouterLink } from "react-router-dom";
import { makeStyles, Theme, createStyles } from "@material-ui/core/styles";
import TextField from "@material-ui/core/TextField";
import Button from "@material-ui/core/Button";
import FormControl from "@material-ui/core/FormControl";
import Container from "@material-ui/core/Container";
import Paper from "@material-ui/core/Paper";
import Grid from "@material-ui/core/Grid";
import Box from "@material-ui/core/Box";
import Typography from "@material-ui/core/Typography";
import Divider from "@material-ui/core/Divider";
import Snackbar from "@material-ui/core/Snackbar";
import MuiAlert, { AlertProps } from "@material-ui/lab/Alert";
import { UsersInterface } from "../models/IUser";
import { MuiPickersUtilsProvider, KeyboardDatePicker, } from "@material-ui/pickers";
import DateFnsUtils from "@date-io/date-fns";

function Alert(props: AlertProps) {
  return <MuiAlert elevation={6} variant="filled" {...props} />;
}

const useStyles = makeStyles((theme: Theme) =>
  createStyles({
    root: {flexGrow: 1},
    container: {marginTop: theme.spacing(2)},
    paper: {padding: theme.spacing(2),color: theme.palette.text.secondary},

```

```

    })
  );

function UserCreate() {
  const classes = useStyles();
  const [selectedDate, setSelectedDate] = React.useState<Date | null>(
    new Date()
  );
  const [user, setUser] = React.useState<Partial<UsersInterface>>({});
  const [success, setSuccess] = React.useState(false);
  const [error, setError] = React.useState(false);

  const handleClose = (event?: React.SyntheticEvent, reason?: string) => {
    if (reason === "clickaway") {
      return;
    }
    setSuccess(false);
    setError(false);
  };

  const handleDateChange = (date: Date | null) => {
    setSelectedDate(date);
  };

  const handleInputChange = (
    event: React.ChangeEvent<{ id?: string; value: any }>
  ) => {
    const id = event.target.id as keyof typeof UserCreate;
    const { value } = event.target;
    setUser({ ...user, [id]: value });
  };

  function submit() {
    let data = {
      FirstName: user.FirstName ?? "",
      LastName: user.LastName ?? "",
      Email: user.Email ?? "",
      Age: typeof user.Age === "string" ? parseInt(user.Age) : 0,
      BirthDay: selectedDate,
    };

    const apiUrl = "http://localhost:8080/users";
  }
}

```

```

const requestOptions = {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify(data),
};

fetch(apiUrl, requestOptions)
  .then((response) => response.json())
  .then((res) => {
    if (res.data) {
      setSuccess(true);
    } else {
      setError(true);
    }
  });
}

return (
  <Container className={classes.container} maxWidth="md">
    <Snackbar open={success} autoHideDuration={6000} onClose={handleClose}>
      <Alert onClose={handleClose} severity="success">
        บันทึกข้อมูลสำเร็จ
      </Alert>
    </Snackbar>
    <Snackbar open={error} autoHideDuration={6000} onClose={handleClose}>
      <Alert onClose={handleClose} severity="error">
        บันทึกข้อมูลไม่สำเร็จ
      </Alert>
    </Snackbar>
    <Paper className={classes.paper}>
      <Box display="flex">
        <Box flexGrow={1}>
          <Typography
            component="h2"
            variant="h6"
            color="primary"
            gutterBottom
          >
            Create User
          </Typography>
        </Box>
      </Box>
    </Paper>
  </Container>
);

```

```
<Divider />

<Grid container spacing={3} className={classes.root}>
  <Grid item xs={6}>
    <p>First Name</p>
    <FormControl fullWidth variant="outlined">
      <TextField
        id="FirstName"
        variant="outlined"
        type="string"
        size="medium"
        value={user.FirstName || ""}
        onChange={handleInputChange}
      />
    </FormControl>
  </Grid>
  <Grid item xs={6}>
    <FormControl fullWidth variant="outlined">
      <p>Last Name</p>
      <TextField
        id="LastName"
        variant="outlined"
        type="string"
        size="medium"
        value={user.LastName || ""}
        onChange={handleInputChange}
      />
    </FormControl>
  </Grid>
  <Grid item xs={12}>
    <FormControl fullWidth variant="outlined">
      <p>Email</p>
      <TextField
        id="Email"
        variant="outlined"
        type="string"
        size="medium"
        value={user.Email || ""}
        onChange={handleInputChange}
      />
    </FormControl>
  </Grid>
  <Grid item xs={6}>
```

```

<FormControl fullWidth variant="outlined">
  <p>Age</p>
  <TextField
    id="Age"
    variant="outlined"
    type="number"
    size="medium"
    InputProps={{ inputProps: { min: 1 } }}
    InputLabelProps={{
      shrink: true,
    }}
    value={user.Age || ""}
    onChange={handleInputChange}
  />
</FormControl>
</Grid>
<Grid item xs={6}>
  <FormControl fullWidth variant="outlined">
    <p>BirthDay</p>
    <MuiPickersUtilsProvider utils={DateFnsUtils}>
      <KeyboardDatePicker
        margin="normal"
        id="BirthDay"
        format="yyyy-MM-dd"
        value={selectedDate}
        onChange={handleDateChange}
        KeyboardButtonProps={{
          "aria-label": "change date",
        }}
      />
    </MuiPickersUtilsProvider>
  </FormControl>
</Grid>
<Grid item xs={12}>
  <Button component={RouterLink} to="/" variant="contained">
    Back
  </Button>
  <Button
    style={{ float: "right" }}
    onClick={submit}
    variant="contained"
    color="primary"
  />
</Grid>

```



```

    >
    Submit
  </Button>
</Grid>
</Grid>
</Paper>
</Container>
);
}

export default UserCreate;

```

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>

System Analysis and Design					
Users					
CREATE USER					
ID	First	Last	Age	Email	Birth Day
1	ชาญวิทย์	แก้วกลี	100	chanwit@gmail.com	2021-08-15T00:00:00Z
2	ธนพล	คงเจริญสุข	25	tanapon@gmail.com	2021-08-15T21:40:00Z
3	สิทธชัย	สิริฤทธิกุลชัย	26	sittichai@gmail.com	2021-08-19T21:40:00Z

*** หมายเหตุ : ข้อมูลที่แสดงเป็นข้อมูลจากการ create user**

ให้ดำเนินการเพิ่มข้อมูล โดยกดที่ปุ่ม create user -> กรอกข้อมูลให้ครบถ้วนจากนั้นกด submit

System Analysis and Design

Creaet User

First Name
Last Name

demo
demo

Email

demo@gmail.com

Age
BirthDay

22
2004-08-22

BACK
SUBMIT

แสดงผลข้อมูลที่เพิ่ม กดปุ่ม back เพื่อกลับไปหน้าแรก

System Analysis and Design

Users
CREATE USER

ID	First	Last	Age	Email	Birth Day
1	ชาญวิทย์	แก้วกลี	100	charnwit@gmail.com	2021-08-15T00:00:00Z
2	ธนพล	คงเจริญสุข	25	tanapon@gmail.com	2021-08-15T21:40:00Z
3	สิทธิชัย	สิริฤทธิกุลชัย	26	sitthichai@gmail.com	2021-08-19T21:40:00Z
4	demo	demo	22	demo@gmail.com	2004-08-22T14:18:40.463Z

- กรณีติด **Access-Control-Allow-Origin** เพิ่มข้อมูลไม่ได้ ให้เพิ่มฟังก์ชัน **CORSMiddleware()** ในไฟล์ **main.go** ของ backend

```
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()
}
}

```

และเรียกใช้ CORSMiddleware() ในฟังก์ชัน main

```

func main() {
    ...
    r.Use(CORSMiddleware())
    ...
}

```

สั่ง `go build -o main.exe main.go` ใหม่อีกครั้ง
และ run backend ใหม่ ด้วยคำสั่ง

`.\main.exe`

หรือ

`./main` บน Linux และ macOS

----- จบ -----