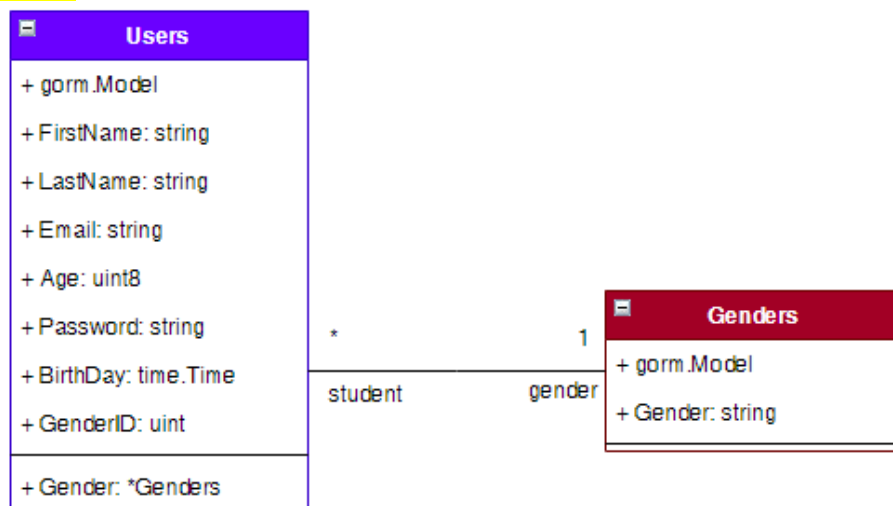


## Class diagram :



## รายละเอียด :

- **Users:**
  - `gorm.Model`: ใช้สำหรับเพิ่มฟิลด์ `ID`, `CreatedAt`, `UpdatedAt` และ `DeletedAt` อัตโนมัติ
  - `FirstName`: ชื่อจริงของผู้ใช้
  - `LastName`: นามสกุลของผู้ใช้
  - `Email`: อีเมลของผู้ใช้
  - `Age`: อายุของผู้ใช้
  - `Password`: รหัสผ่านของผู้ใช้ (ไม่แสดงใน JSON)
  - `BirthDay`: วันเกิดของผู้ใช้
  - `GenderID`: รหัสเพศที่เชื่อมโยงกับ `Genders`
  - `Gender`: ชี้ไปยัง `Genders` ใช้ `foreignKey gender_id`
- **Genders:**
  - `gorm.Model`: ใช้สำหรับเพิ่มฟิลด์ `ID`, `CreatedAt`, `UpdatedAt` และ `DeletedAt` อัตโนมัติ
  - `Gender`: คำอธิบายของเพศ (เช่น ชาย, หญิง)

## ความสัมพันธ์:

- ความสัมพันธ์แบบ One-to-Many ระหว่าง `Genders` และ `Users` โดยใช้ฟิลด์ `GenderID` เป็น foreign key.

สร้าง directory Project ชื่อ `sa-67-example`

```
$ cd c:\
```

```
$ mkdir sa-67-example
```

```
$ cd sa-67-example
```

## Backend :

1. สร้าง project backend ที่ `c:\sa-67-example`

```
$ cd c:\sa-67-example
$ mkdir backend
$ cd backend
$ go mod init example.com/sa-67-example
```

- 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
$ go get -u github.com/dgrijalva/jwt-go
$ go get -u golang.org/x/crypto
```

- โครงสร้างไฟล์ที่เราต้องสร้างใน directory backend จะเป็นดังนี้

```
└─ backend
    ├── README.md
    ├── config
    │   ├── config.go
    │   └── db.go
    ├── controller
    │   ├── genders
    │   │   └── genders.go
    │   └── users
    │       ├── auth.go
    │       └── users.go
    ├── go.mod
    ├── go.sum
    ├── main.go
    ├── middlewares
    │   └── authorization.go
    ├── models
    │   ├── genders.go
    │   └── users.go
    ├── sa.db
    └── services
        └── auth.go
```

#### 4. สร้าง Schema

```
$ mkdir entity
```

```
$ cd entity/
```

ใน c:\sa-67-example\backend\entity เราจะดำเนินการสร้าง schema ของ User และ Genders ซึ่งเป็นส่วนของการกำหนดโครงสร้างของฐานข้อมูล  
สร้างไฟล์ชื่อ users.go เขียนโปรแกรมดังนี้

```
package models
import (
    "time"
    "gorm.io/gorm"
)
type Users struct {
    gorm.Model
    FirstName string `json:"first_name"`
    LastName  string `json:"last_name"`
    Email     string `json:"email"`
    Age       uint8  `json:"age"`
    Password  string `json:"-"`
    BirthDay  time.Time `json:"birthday"`
    GenderID  uint    `json:"gender_id"`
    Gender    *Genders `gorm:"foreignKey: gender_id" json:"gender"`
}
```

สร้างไฟล์ชื่อ genders.go เขียนโปรแกรมดังนี้

```
package models
import "gorm.io/gorm"
type Genders struct {
    gorm.Model
    Gender string `json:"gender"`
}
```

#### 5. ตั้งค่าการเชื่อมต่อฐานข้อมูล ฟังก์ชันสำหรับเข้ารหัส และการตรวจสอบรหัสผ่าน

```
$ mkdir config
```

```
$ cd config/
```

ใน c:\sa-67-example\backend\config เราจะดำเนินการสร้างไฟล์ db.go และ config.go  
สร้างไฟล์ชื่อ db.go เพื่อใช้สำหรับการสร้างฐานข้อมูล (Database) เขียนโปรแกรมดังนี้

```
package config
import (
    "fmt"
    "time"
```

```

"example.com/sa-65-example/models"
"gorm.io/driver/sqlite"
"gorm.io/gorm"
)
var db *gorm.DB
func DB() *gorm.DB {
    return db
}
func ConnectionDB() {
    database, err := gorm.Open(sqlite.Open("sa.db?cache=shared"), &gorm.Config{})
    if err != nil {
        panic("failed to connect database")
    }
    fmt.Println("connected database")
    db = database
}
func SetupDatabase() {
    db.AutoMigrate(
        &models.Users{},
        &models.Genders{},
    )
    GenderMale := models.Genders{Gender: "Male"}
    GenderFemale := models.Genders{Gender: "Female"}
    db.FirstOrCreate(&GenderMale, &models.Genders{Gender: "Male"})
    db.FirstOrCreate(&GenderFemale, &models.Genders{Gender: "Female"})
    hashedPassword, _ := HashPassword("123456")
    BirthDay, _ := time.Parse("2006-01-02", "1988-11-12")
    User := &models.Users{
        FirstName: "Software",
        LastName: "Analysis",
        Email: "sa@gmail.com",
        Age: 80,
        Password: hashedPassword,
        BirthDay: BirthDay,
        GenderID: 1,
    }
    db.FirstOrCreate(User, &models.Users{
        Email: "sa@gmail.com",

```

```
})
}
```

สร้างไฟล์ชื่อ **config.go** ซึ่งเป็นการเตรียมฟังก์ชันไว้ ได้แก่ ฟังก์ชันสำหรับการเข้ารหัส Password และ ฟังก์ชันสำหรับการตรวจสอบความถูกต้องของรหัสผ่าน เขียนโปรแกรมดังนี้

```
package config
import "golang.org/x/crypto/bcrypt"
// hashPassword เป็น function สำหรับการแปลง password
func HashPassword(password string) (string, error) {
    bytes, err := bcrypt.GenerateFromPassword([]byte(password), 14)
    return string(bytes), err
}
// checkPasswordHash เป็น function สำหรับ check password ที่ hash แล้ว ว่าตรงกันหรือไม่
func CheckPasswordHash(password, hash []byte) bool {
    err := bcrypt.CompareHashAndPassword(hash, password)
    return err == nil
}
```

## 6. สร้าง Controller

ขั้นตอนการสร้าง controller จะสร้างที่ folder c:\sa-67-example\backend

```
$ mkdir controller
```

```
$ cd controller
```

สร้างโฟลเดอร์ **users**

- สร้างไฟล์ชื่อ **users.go** เป็น controller สำหรับเชื่อมต่อกับ entity User
- สร้างไฟล์ชื่อ **auth.go** เป็น controller สำหรับการลงทะเบียน (Signup) และการเข้าใช้งาน (Signin)

สร้างโฟลเดอร์ **genders**

- สร้างไฟล์ชื่อ **genders.go** เป็น controller สำหรับเชื่อมต่อกับ entity Gender มีส่วน

โครงสร้างไฟล์ใน c:\sa-67-example\backend\controller

```
backend/
|
|----- controller/
|   |
|   |----- users
|   |
|   |----- users.go
|   |----- auth.go
|   |
|   |----- genders
|   |
|   |----- genders.go
```

Source Code สำหรับไฟล์ genders.go

```
package genders
```

```

import (
    "net/http"
    "example.com/sa-65-example/config"
    "example.com/sa-65-example/models"
    "github.com/gin-gonic/gin"
)

func GetAll(c *gin.Context) {
    db := config.DB()
    var genders []models.Genders
    db.Find(&genders)
    c.JSON(http.StatusOK, &genders)
}

```

อธิบายการทำงานของฟังก์ชันต่างๆ

- function ถัดมา GetAll จะเป็นการ list รายการของ Genders ออกมา โดยแสดงการใช้ SELECT \* ผลลัพธ์ที่เป็น รายการข้อมูลจะสามารถดึงออกมาได้อย่างถูกต้องเมื่อนำตัวแปรที่เป็น array มารับ ในตัวอย่างนี้ genders เป็นตัวแปรประเภท array ของ entity.Genders (สังเกต []entity.Genders)

Source Code สำหรับไฟล์ users.go

```

package users

import (
    "net/http"
    "github.com/gin-gonic/gin"
    "example.com/sa-65-example/config"
    "example.com/sa-65-example/models"
)

func GetAll(c *gin.Context) {
    var users []models.Users
    db := config.DB()
    results := db.Preload("Gender").Find(&users)
    if results.Error != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": results.Error.Error()})
        return
    }
    c.JSON(http.StatusOK, users)
}

func Get(c *gin.Context) {
    ID := c.Param("id")
    var user models.Users

```

```

    db := config.DB()
    results := db.Preload("Gender").First(&user, ID)
    if results.Error != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": results.Error.Error()})
        return
    }
    if user.ID == 0 {
        c.JSON(http.StatusNoContent, gin.H{})
        return
    }
    c.JSON(http.StatusOK, user)
}

func Update(c *gin.Context) {
    var user models.Users
    UserID := c.Param("id")
    db := config.DB()
    result := db.First(&user, UserID)
    if result.Error != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": "id not found"})
        return
    }
    if err := c.ShouldBindJSON(&user); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Bad request, unable to map payload"})
        return
    }
    result = db.Save(&user)
    if result.Error != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Bad request"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"message": "Updated successful"})
}

func Delete(c *gin.Context) {
    id := c.Param("id")
    db := config.DB()
    if tx := db.Exec("DELETE FROM users WHERE id = ?", id); tx.RowsAffected == 0 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "id not found"})
        return
    }
}

```

```

}
c.JSON(http.StatusOK, gin.H{"message": "Deleted successful"})
}

```

### อธิบายการทำงานของฟังก์ชันต่างๆ

- function ถัดมา GetAll จะเป็นการ list รายการของ Users ออกมา โดยแสดงการใช้ SELECT \* ผลลัพธ์ที่เป็น รายการข้อมูลจะสามารถดึงออกมาได้อย่างถูกต้องเมื่อนำตัวแปรที่เป็น array มารับ ในตัวอย่างนี้ users เป็นตัวแปรประเภท array ของ entity.Users (สังเกต []entity.Users)
- function Get โดยในตัวอย่างเป็นการตั้งใจใช้คำสั่ง SELECT ... WHERE id =... เพื่อดึงข้อมูล user ออกมาตาม primary key ที่กำหนด ผ่าน func DB.Raw(...)
- function สำหรับ update user ก็คือการ UPDATE ... WHERE ID=... ในตัวอย่างใช้คำสั่ง DB.Save() แทน update ของ SQL
- function ถัดมาเป็น function สำหรับลบ user ด้วย ID ก็คือการ DELETE ... WHERE ID=...

### Source Code สำหรับไฟล์ auth.go

```

package users
import (
    "errors"
    "net/http"
    "time"
    "github.com/gin-gonic/gin"
    "golang.org/x/crypto/bcrypt"
    "gorm.io/gorm"
    "example.com/sa-65-example/config"
    "example.com/sa-65-example/models"
    "example.com/sa-65-example/services"
)
type (
    Authen struct {
        Email    string `json:"email"`
        Password string `json:"password"`
    }
    signUp struct {
        FirstName string `json:"first_name"`
        LastName  string `json:"last_name"`
        Email     string `json:"email"`
        Age       uint8  `json:"age"`
    }
)

```



```

    Password string `json:"password"`
    BirthDay  time.Time `json:"birthday"`
    GenderID  uint      `json:"gender_id"`
}
)

func SignUp(c *gin.Context) {
    var payload signUp
    // Bind JSON payload to the struct
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    db := config.DB()
    var userCheck models.Users
    // Check if the user with the provided email already exists
    result := db.Where("email = ?", payload.Email).First(&userCheck)
    if result.Error != nil && !errors.Is(result.Error, gorm.ErrRecordNotFound) {
        // If there's a database error other than "record not found"
        c.JSON(http.StatusInternalServerError, gin.H{"error": result.Error.Error()})
        return
    }
    if userCheck.ID != 0 {
        // If the user with the provided email already exists
        c.JSON(http.StatusConflict, gin.H{"error": "Email is already registered"})
        return
    }
    // Hash the user's password
    hashedPassword, _ := config.HashPassword(payload.Password)
    // Create a new user
    user := models.Users{
        FirstName: payload.FirstName,
        LastName:  payload.LastName,
        Email:     payload.Email,
        Age:       payload.Age,
        Password:  hashedPassword,
        BirthDay:  payload.BirthDay,
        GenderID:  payload.GenderID,
    }
}

```

```

// Save the user to the database
if err := db.Create(&user).Error; err != nil {
    c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
    return
}
c.JSON(http.StatusCreated, gin.H{"message": "Sign-up successful"})
}

func SignIn(c *gin.Context) {
    var payload Authen
    var user models.Users
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    // ค้นหา user ด้วย Username ที่ผู้ใช้กรอกเข้ามา
    if err := config.DB().Raw("SELECT * FROM users WHERE email =
?", payload.Email).Scan(&user).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
    // ตรวจสอบรหัสผ่าน
    err := bcrypt.CompareHashAndPassword([]byte(user.Password), []byte(payload.Password))
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "password is incorrect"})
        return
    }
    jwtWrapper := services.JwtWrapper{
        SecretKey:    "SvNQpBN8y3qlVrsGAYYWoJJk56LtzFHx",
        Issuer:        "AuthService",
        ExpirationHours: 24,
    }
    signedToken, err := jwtWrapper.GenerateToken(user.Email)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "error signing token"})
        return
    }
    c.JSON(http.StatusOK, gin.H{"token_type": "Bearer", "token": signedToken, "id": user.ID})
}

```

## อธิบายการทำงานของฟังก์ชันต่างๆ

- function SignUp เป็นการทำงานแทนคำสั่ง insert ของ SQL โดย function นี้จะคืนค่าเป็น user ที่สร้างเสร็จแล้ว กลับไปเป็น JSON ให้ฝั่ง UI นำไปแสดงผล
- function SignIn เป็นการตรวจสอบการเข้าสู่ระบบโดยการ Query ข้อมูลด้วย email จากนั้นนำรหัสผ่านมาตรวจสอบ

## 7. สร้าง Middlewares

ขั้นตอนการสร้าง Middlewares จะสร้างที่ folder c:\sa-67-example\backend

```
$ mkdir middlewares
```

```
$ cd middlewares
```

```
backend/
```

```
|----- middlewares/  
|      |  
|      |----- authorization.go
```

สร้างไฟล์ชื่อ **authorization.go** เป็นฟังก์ชันตรวจเช็ค Cookie

```
package middlewares  
import (  
    "net/http"  
    "strings"  
    "example.com/sa-67-example/services"  
    "github.com/gin-gonic/gin"  
)  
var HashKey = []byte("very-secret")  
var BlockKey = []byte("a-lot-secret1234")  
// Authorization เป็นฟังก์ชันตรวจเช็ค Cookie  
func Authorizes() gin.HandlerFunc {  
    return func(c *gin.Context) {  
        clientToken := c.Request.Header.Get("Authorization")  
        if clientToken == "" {  
            c.AbortWithStatusJSON(http.StatusUnauthorized, gin.H{"error": "No Authorization  
header provided"})  
            return  
        }  
        extractedToken := strings.Split(clientToken, "Bearer ")  
        if len(extractedToken) == 2 {  
            clientToken = strings.TrimSpace(extractedToken[1])  
        } else {  
            c.AbortWithStatusJSON(http.StatusUnauthorized, gin.H{"error": "Incorrect Format of  
Authorization Token"})  
            return  
        }  
    }  
}
```

```

    }
    jwtWrapper := services.JwtWrapper{
        SecretKey: "SvNQpBN8y3qlVrsGAYYWoJJk56LtzFHx",
        Issuer:     "AuthService",
    }
    _, err := jwtWrapper.ValidateToken(clientToken)
    if err != nil {
        c.AbortWithStatusJSON(http.StatusUnauthorized, gin.H{"error": err.Error()})
        return
    }
    c.Next()
}
}

```

## 8. สร้าง Services

ขั้นตอนการสร้าง Services จะสร้างที่ folder c:\sa-67-example\backend

```
$ mkdir services
```

```
$ cd services
```

```
backend/
```

```

|----- services/
|       |
|       |----- auth.go

```

สร้างไฟล์ชื่อ **auth.go** เป็น service ที่ใช้สำหรับ Generate Token และ Validate Token

```

package services
import (
    "errors"
    "time"
    jwt "github.com/dgrijalva/jwt-go"
)
// JwtWrapper wraps the signing key and the issuer
type JwtWrapper struct {
    SecretKey    string
    Issuer       string
    ExpirationHours int64
}
// JwtClaim adds email as a claim to the token
type JwtClaim struct {
    Email string
    jwt.StandardClaims
}

```

```

// Generate Token generates a jwt token
func (j *JwtWrapper) GenerateToken(email string) (signedToken string, err error) {
    claims := &JwtClaim{
        Email: email,
        StandardClaims: jwt.StandardClaims{
            ExpiresAt: time.Now().Local().Add(time.Hour * time.Duration(j.ExpirationHours)).Unix(),
            Issuer:    j.Issuer,
        },
    }
    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    signedToken, err = token.SignedString([]byte(j.SecretKey))
    if err != nil {
        return
    }
    return
}

// Validate Token validates the jwt token
func (j *JwtWrapper) ValidateToken(signedToken string) (claims *JwtClaim, err error) {
    token, err := jwt.ParseWithClaims(
        signedToken,
        &JwtClaim{},
        func(token *jwt.Token) (interface{}, error) {
            return []byte(j.SecretKey), nil
        },
    )
    if err != nil {
        return
    }
    claims, ok := token.Claims.(*JwtClaim)
    if !ok {
        err = errors.New("Couldn't parse claims")
        return
    }
    if claims.ExpiresAt < time.Now().Local().Unix() {
        err = errors.New("JWT is expired")
        return
    }
    return
}

```

```
}
```

## 9. ประกาศเป็น path ของ API ด้วย router

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

```
package main
import (
    "net/http"
    "github.com/gin-gonic/gin"
    "example.com/sa-67-example/config"
    "example.com/sa-67-example/controller/genders"
    "example.com/sa-67-example/controller/users"
    "example.com/sa-67-example/middlewares"
)
const PORT = "8000"
func main() {
    // open connection database
    config.ConnectionDB()
    // Generate databases
    config.SetupDatabase()
    r := gin.Default()
    r.Use(CORSMiddleware())
    // Auth Route
    r.POST("/signup", users.SignUp)
    r.POST("/signin", users.SignIn)
    router := r.Group("/")
    {
        router.Use(middlewares.Authorizes())
        // User Route
        router.PUT("/user/:id", users.Update)
        router.GET("/users", users.GetAll)
        router.GET("/user/:id", users.Get)
        router.DELETE("/user/:id", users.Delete)
    }
    r.GET("/genders", genders.GetAll)
    r.GET("/", func(c *gin.Context) {
        c.String(http.StatusOK, "API RUNNING... PORT: %s", PORT)
    })
}
```

```

})
// Run the server
r.Run("localhost:" + PORT)
}
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,
DELETE")
        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
```

เมื่อโปรแกรม compile ได้ถูกต้องแล้ว

รันโปรแกรมโดยการรันคำสั่ง main

```
.\main.exe
```

หรือ

```
./main บน Linux และ macOS
```

## Frontend :

### 1. สร้างโปรเจก React+ Vite

- ใน Terminal พิมพ์คำสั่งต่อไปนี้เพื่อติดตั้ง React App + Vite

```
$ npm create vite@latest <<ชื่อโปรเจก>>
```

- เลือก Framwork เป็น React (ใช้ลูกศรขึ้นหรือลง ในการเลือก และกด Enter)
- เลือก Variant เป็น TypeScript (ใช้ลูกศรขึ้นหรือลง ในการเลือก และกด Enter)
- รอจนกว่าการติดตั้งจะเสร็จสิ้น จากนั้นพิมพ์คำสั่งต่อไปนี้เพื่อนำทางไปยังโฟลเดอร์โปรเจก

```
$ cd <<ชื่อโปรเจก>>
```

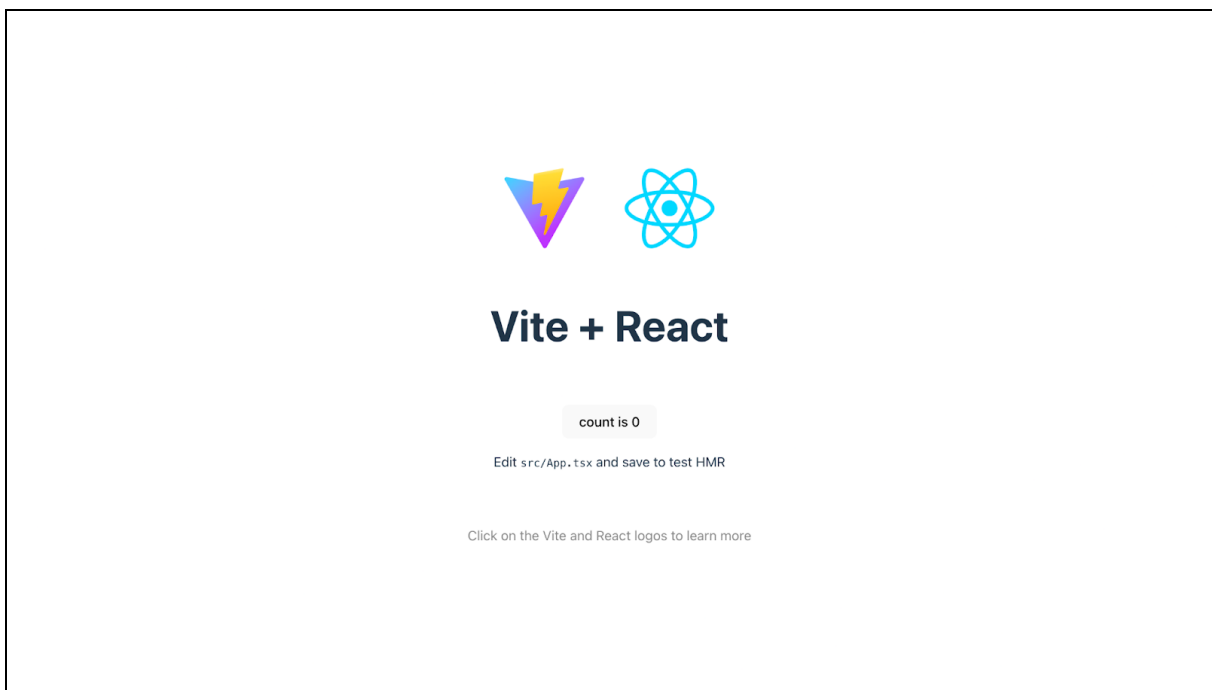
- เปิด Terminal ที่ Visual Studio Code จากนั้นพิมพ์คำสั่งต่อไปนี้เพื่อติดตั้ง Packet

```
$ npm install
```

- รอจนกว่าการติดตั้งจะเสร็จสิ้น จากนั้นพิมพ์คำสั่งต่อไปนี้เพื่อเปิดโปรเจกที่ localhost

```
$ npm run dev
```

เว็บเบราว์เซอร์จะเปิดหน้าเว็บที่อยู่ <http://localhost:5173> แสดงว่าโปรเจก React ของนักศึกษาทำงานได้เรียบร้อยแล้ว





## 2. ติดตั้ง Package ที่ต้องใช้งาน

การติดตั้ง package ต่างๆจะต้องอยู่ที่ directory <<ชื่อโปรเจค>> ที่เราสร้าง

- ติดตั้ง package สำหรับจัดการ Router เป็นไลบรารีสำหรับการจัดการการนำทางในแอปพลิเคชัน

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

- ติดตั้ง Ant Design คือ UI library สำหรับใช้งาน component ต่างๆ เช่น Button, Card, Table เป็นต้น <https://ant.design/docs/react/use-with-create-react-app>

```
$ npm install antd --save
```

- ติดตั้ง Axios เป็นไลบรารี HTTP client <https://www.npmjs.com/package/axios>

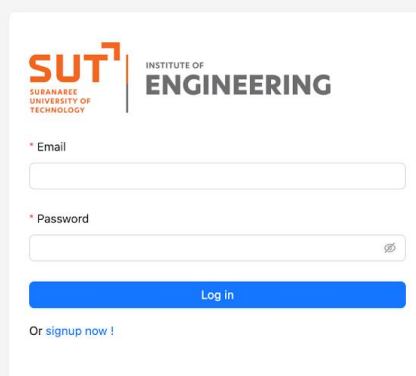
```
$ npm install axios --save
```

- ติดตั้ง Dayjs เป็นไลบรารี JavaScript ที่ช่วยในการจัดการและจัดรูปแบบวันที่และเวลา <https://www.npmjs.com/package/dayjs>

```
$ npm install dayjs --save
```


3. ปรับปรุง source code frontend เพื่อทดสอบการเชื่อมต่อกับ backend สำหรับส่งข้อมูล และรับข้อมูล

- หน้าล็อกอิน



The image shows a login form for the SUT Institute of Engineering. The form is white and centered on a light gray background. It features the SUT logo (Suranaree University of Technology) and the text 'INSTITUTE OF ENGINEERING'. Below the logo, there are two input fields: one for 'Email' and one for 'Password'. The 'Password' field has a small eye icon to toggle visibility. Below the input fields is a blue 'Log in' button. At the bottom of the form, there is a link that says 'Or signup now !'.

- หน้าลงทะเบียน


**SUT** INSTITUTE OF  
ENGINEERING  
SURABHARTI  
UNIVERSITY OF  
TECHNOLOGY

### Sign Up

\* ชื่อจริง

\* นามสกุล

\* อีเมล

\* รหัสผ่าน

\* วัน/เดือน/ปี เกิด


\* อายุ

\* เพศ


Sign up

Or [signin now !](#)

- หน้าแดชบอร์ดเป็นการจำลองข้อมูล


**SUT** INSTITUTE OF  
ENGINEERING  
SURABHARTI  
UNIVERSITY OF  
TECHNOLOGY
 

แดชบอร์ด


 ข้อมูลสมาชิก

ออกจากระบบ

### แดชบอร์ด


จำนวน  
≤ 1,800

จำนวน  
200

จำนวน  
3,000

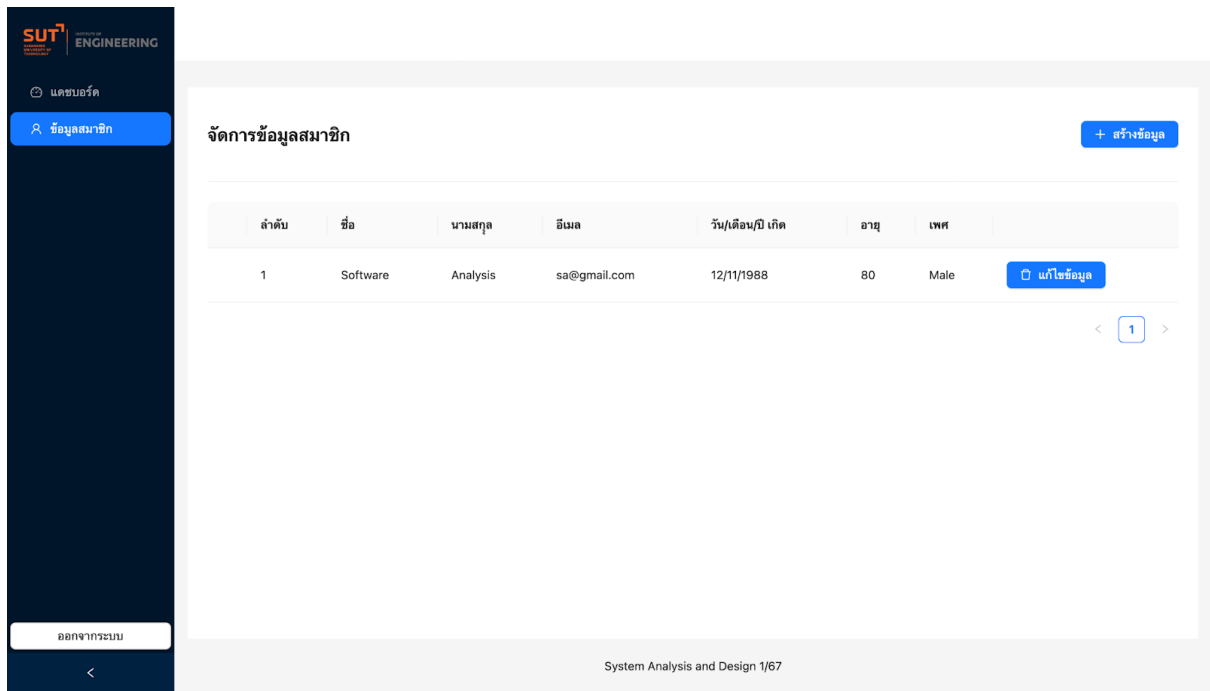
จำนวน  
10

#### ผู้ใช้งานล่าสุด

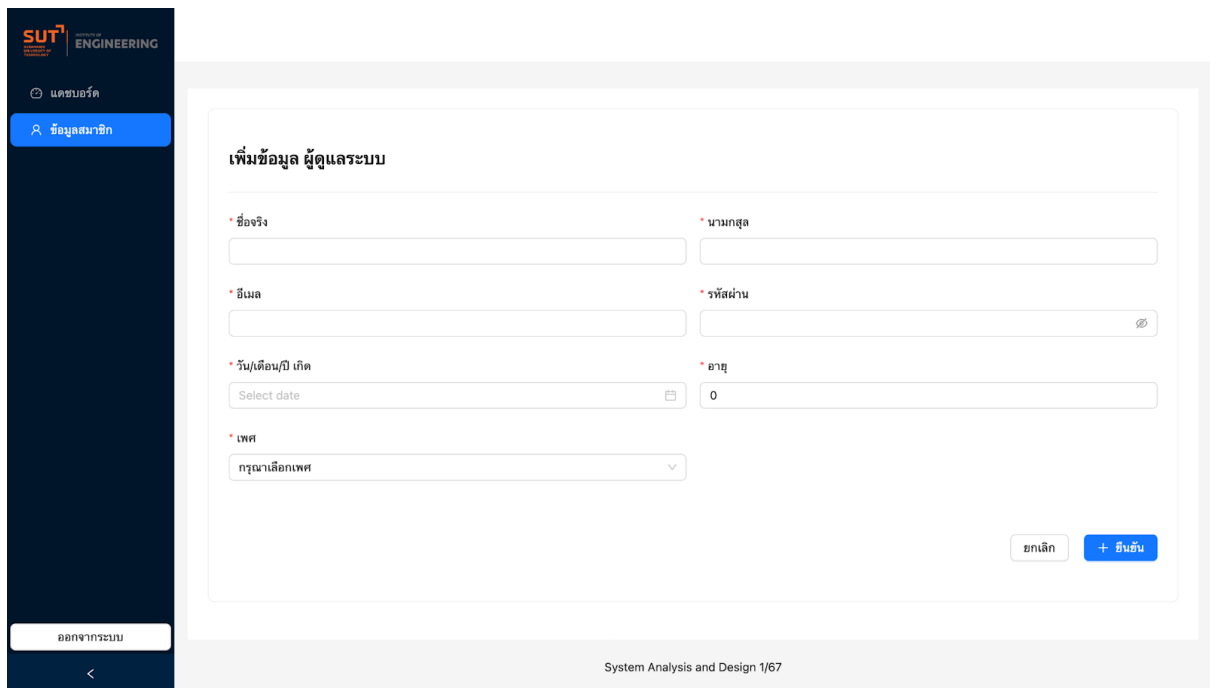
ลำดับ	ชื่อ	นามสกุล	อีเมล	เบอร์โทร
 No data				

System Analysis and Design 1/67

- หน้าข้อมูลสมาชิกเป็นหน้า แสดงข้อมูลผู้ใช้ ที่ GET ข้อมูลจากฐานข้อมูล



- หน้าเพิ่มข้อมูลผู้ใช้เป็นหน้าสำหรับ Insert ข้อมูล User เข้าไปที่ฐานข้อมูล



- หน้าแก้ไขข้อมูลผู้ใช้เป็นหน้าสำหรับ Update ข้อมูล User เข้าไปที่ฐานข้อมูล

SUT  
ENGINEERING

แดชบอร์ด

ข้อมูลสมาชิก

ออกจากระบบ

แก้ไขข้อมูล ผู้ดูแลระบบ

\* ชื่อจริง

Software

\* นามสกุล

Analysis

\* อีเมล

sa@gmail.com

\* วัน/เดือน/ปี เกิด

1988-11-12

\* อายุ

80

\* เพศ

Male

ยกเลิก

+ บันทึก

System Analysis and Design 1/67

เมื่อเห็นภาพตัวอย่างของระบบแล้วเราก็จะเริ่ม การแก้ไข source code ของโปรแกรมเรา

โครงสร้างไฟล์ที่เราต้องสร้างใน directory frontend



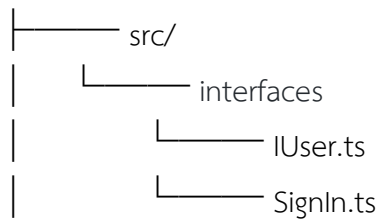
```
1  frontend/
2  |— src/
3  |   |— assets
4  |   |   |— logo.png
5  |   |— components
6  |   |— interfaces
7  |   |   |— IUser.ts
8  |   |   |— SignIn.ts
9  |   |— layout
10 |   |   |— FullLayout
11 |   |   |   |— index.tsx
12 |   |   |— MinimalLayout
13 |   |   |   |— index.tsx
14 |   |— pages
15 |   |   |— authentication
16 |   |   |   |— Login
17 |   |   |   |   |— index.tsx
18 |   |   |   |— Register
19 |   |   |   |   |— index.tsx
20 |   |   |— dashboard
21 |   |   |   |— index.tsx
22 |   |   |— customer
23 |   |   |   |— create
24 |   |   |   |   |— index.tsx
25 |   |   |   |— edit
26 |   |   |   |   |— index.tsx
27 |   |   |   |— index.tsx
28 |   |— routes
29 |   |   |— AdminRoutes.tsx
30 |   |   |— MainRoutes.tsx
31 |   |   |— index.tsx
32 |   |— services
33 |   |   |— https
34 |   |   |   |— index.tsx
```

### 3.1 ทำการบันทึกภาพ ตั้งชื่อ logo.png เก็บไว้ที่โฟลเดอร์ assets



### 3.2 สร้าง Interface เพื่อประกาศโครงสร้างข้อมูลโดยจะเป็นการประกาศชื่อตาราง ชื่อ field ชนิดข้อมูล

frontend/



สร้าง directory ชื่อ interfaces ใน directory src จากนั้น

สร้างไฟล์ **IUser.ts**

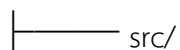
```
export interface UsersInterface {
  ID?: number;
  FirstName?: string;
  LastName?: string;
  Email?: string;
  Phone?: string;
  Age?: number;
  BirthDay?: string;
  GenderID?: number;
  Password?: string;
}
```

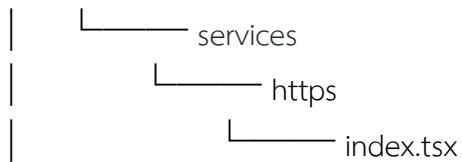
สร้างไฟล์ **SignIn.ts**

```
export interface SignInInterface {
  email?: string;
  password?: string;
}
```

### 3.3 สร้าง Service สำหรับการเชื่อมต่อกับ Backend เพื่อรับส่งข้อมูล

frontend/





สร้าง directory ชื่อ services ต่อจากนั้นสร้าง directory ชื่อ https และ สร้างไฟล์ **index.tsx**

```
import { UsersInterface } from "../../interfaces/IUser";
import { SignInInterface } from "../../interfaces/SignIn";
import axios from "axios";
const apiUrl = "http://localhost:8000";
const Authorization = localStorage.getItem("token");
const Bearer = localStorage.getItem("token_type");
const requestOptions = {
  headers: {
    "Content-Type": "application/json",
    Authorization: `${Bearer} ${Authorization}`,
  },
};
async function SignIn(data: SignInInterface) {
  return await axios
    .post(`${apiUrl}/signin`, data, requestOptions)
    .then((res) => res)
    .catch((e) => e.response);
}
async function GetUsers() {
  return await axios
    .get(`${apiUrl}/users`, requestOptions)
    .then((res) => res)
    .catch((e) => e.response);
}
async function GetUsersById(id: string) {
  return await axios
    .get(`${apiUrl}/user/${id}`, requestOptions)
    .then((res) => res)
    .catch((e) => e.response);
}
async function UpdateUsersById(id: string, data: UsersInterface) {
  return await axios
    .put(`${apiUrl}/user/${id}`, data, requestOptions)
    .then((res) => res)
```

```

    .catch((e) => e.response);
}
async function DeleteUsersById(id: string) {
  return await axios
    .delete(`${apiUrl}/user/${id}`, requestOptions)
    .then((res) => res)
    .catch((e) => e.response);
}
async function CreateUser(data: UsersInterface) {
  return await axios
    .post(`${apiUrl}/signup`, data, requestOptions)
    .then((res) => res)
    .catch((e) => e.response);
}
export {
  SignIn,
  GetUsers,
  GetUsersById,
  UpdateUsersById,
  DeleteUsersById,
  CreateUser,
};

```

3.4 สร้างหน้าสำหรับแสดงข้อมูล ได้แก่ หน้าล็อกอิน หน้าลงทะเบียน หน้าแดชบอร์ด หน้าสร้างข้อมูลผู้ใช้ และหน้าแก้ไขข้อมูล

สร้าง directory ชื่อ pages ใน directory src และสร้างไฟล์ต่อไปนี้

frontend/

```

|____ src/
|   |____ pages

```

สร้างหน้าล็อกอิน ดำเนินการสร้างโฟลเดอร์และสร้างไฟล์ตามโครงสร้างต่อไปนี้

frontend/

```

|____ src/
|   |____ pages
|           |____ authentication
|                   |____ Login
|                           |____ index.tsx

```

```

import { Button, Card, Form, Input, message, Flex, Row, Col } from "antd";
import { useNavigate } from "react-router-dom";

```



```

import { SignIn } from "../../services/https";
import { SignInInterface } from "../../interfaces/SignIn";
import logo from "../../assets/logo.png";
function SignInPages() {
  const navigate = useNavigate();
  const [messageApi, contextHolder] = message.useMessage();
  const onFinish = async (values: SignInInterface) => {
    let res = await SignIn(values);
    if (res.status == 200) {
      messageApi.success("Sign-in successful");
      localStorage.setItem("isLogin", "true");
      localStorage.setItem("page", "dashboard");
      localStorage.setItem("token_type", res.data.token_type);
      localStorage.setItem("token", res.data.token);
      localStorage.setItem("id", res.data.id);
      setTimeout(() => {
        location.href = "/";
      }, 2000);
    } else {
      messageApi.error(res.data.error);
    }
  };
  return (
    <>
    {contextHolder}
    <Flex justify="center" align="center" className="login">
      <Card className="card-login" style={{ width: 500 }}>
        <Row align="middle" justify="center" style={{ height: "400px" }}>
          <Col xs={24} sm={24} md={24} lg={24} xl={24}>
            <img
              alt="logo"
              style={{ width: "80%" }}
              src={logo}
              className="images-logo"
            />
          </Col>
          <Col xs={24} sm={24} md={24} lg={24} xl={24}>
            <Form

```

```
      name="basic"
      onFinish={onFinish}
      autoComplete="off"
      layout="vertical"
    >
      <Form.Item
        label="Email"
        name="email"
        rules={[
          { required: true, message: "Please input your username!" },
        ]}
      >
        <Input />
      </Form.Item>
      <Form.Item
        label="Password"
        name="password"
        rules={[
          { required: true, message: "Please input your password!" },
        ]}
      >
        <Input.Password />
      </Form.Item>
      <Form.Item>
        <Button
          type="primary"
          htmlType="submit"
          className="login-form-button"
          style={{ marginBottom: 20 }}
        >
          Log in
        </Button>
        Or <a onClick={() => navigate("/signup")}>signup now !</a>
      </Form.Item>
    </Form>
  </Col>
</Row>
</Card>
```

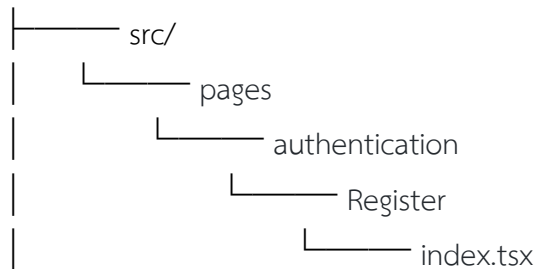
```

    </Flex>
  </>
);
}
export default SignInPages;

```

สร้างหน้าลงทะเบียน ดำเนินการสร้างโฟลเดอร์และสร้างไฟล์ตามโครงสร้างต่อไปนี้

frontend/



```

import {
  Button,
  Card,
  Form,
  Input,
  message,
  Flex,
  Row,
  Col,
  InputNumber,
  DatePicker,
  Select,
} from "antd";
import { useNavigate } from "react-router-dom";
import { CreateUser } from "../../services/https";
import { UsersInterface } from "../../interfaces/IUser";
import logo from "../../assets/logo.png";
function SignUpPages() {
  const navigate = useNavigate();
  const [messageApi, contextHolder] = message.useMessage();
  const onFinish = async (values: UsersInterface) => {
    let res = await CreateUser(values);
    if (res.status === 201) {
      messageApi.open({
        type: "success",

```

```

        content: res.data.message,
    });
    setTimeout(function () {
        navigate("/");
    }, 2000);
    } else {
        messageApi.open({
            type: "error",
            content: res.data.error,
        });
    }
};
return (
    <>
    {contextHolder}
    <Flex justify="center" align="center" className="login">
        <Card className="card-login" style={{ width: 600 }}>
            <Row align="middle" justify="center">
                <Col xs={24} sm={24} md={24} lg={10} xl={10}>
                    <img alt="logo" src={logo} className="images-logo" />
                </Col>
                <Col xs={24} sm={24} md={24} lg={24} xl={24}>
                    <h2 className="header">Sign Up</h2>
                    <Form
                        name="basic"
                        layout="vertical"
                        onFinish={onFinish}
                        autoComplete="off"
                    >
                        <Row gutter={[16, 0]} align="middle">
                            <Col xs={24} sm={24} md={24} lg={24} xl={24}>
                                <Form.Item
                                    label="ชื่อจริง"
                                    name="first_name"
                                    rules={[
                                        {
                                            required: true,
                                            message: "กรุณากรอกชื่อ !",
                                        },
                                    ]}
                                />
                            </Col>
                        </Row>
                    </Form>
                </Col>
            </Row>
        </Card>
    </Flex>
    </>
);

```

```
    },
  ]
}
>
<Input />
</Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={24}>
  <Form.Item
    label="นามสกุล"
    name="last_name"
    rules={[
      {
        required: true,
        message: "กรุณากรอกนามสกุล !",
      },
    ]}
  >
    <Input />
  </Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={24}>
  <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={12} xl={12}>
  <Form.Item
    label="รหัสผ่าน"
    name="password"
    rules={[
      {
        required: true,
        message: "กรุณากรกรหัสผ่าน !",
      },
    ]}
  >
    <Input.Password />
  </Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={12} xl={12}>
  <Form.Item
    label="วัน/เดือน/ปี เกิด"
    name="birthday"
    rules={[
      {
        required: true,
        message: "กรุณาเลือกวัน/เดือน/ปี เกิด !",
      },
    ]}
  >
    <DatePicker style={{ width: "100%" }} />
  </Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={12} xl={12}>
  <Form.Item
    label="อายุ"
    name="age"
    rules={[
      {
        required: true,
        message: "กรุณากรอกอายุ !",
      },
    ]}
  >
```

```
>
  <InputNumber
    min={0}
    max={99}
    defaultValue={0}
    style={{ width: "100%" }}
  />
</Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={12} xl={12}>
  <Form.Item
    label="เพศ"
    name="gender_id"
    rules={[
      {
        required: true,
        message: "กรุณาเลือกเพศ !",
      },
    ]}
  >
    <Select
      defaultValue=""
      style={{ width: "100%" }}
      options={[
        { value: "", label: "กรุณาเลือกเพศ", disabled: true },
        { value: 1, label: "Male" },
        { value: 2, label: "Female" },
      ]}
    />
  </Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={24}>
  <Form.Item>
    <Button
      type="primary"
      htmlType="submit"
      className="login-form-button"
      style={{ marginBottom: 20 }}
    />
  </Form.Item>
</Col>
```

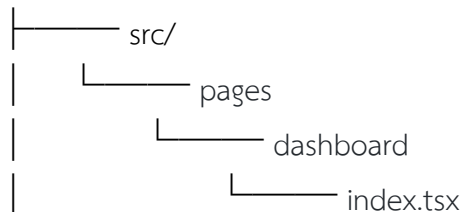
```

        >
        Sign up
      </Button>
      Or <a onClick={() => navigate("/")}>signin now !</a>
    </Form.Item>
  </Col>
</Row>
</Form>
</Col>
</Row>
</Card>
</Flex>
</>
);
}
export default SignUpPages;

```

สร้างหน้าแดชบอร์ด ดำเนินการสร้างโฟลเดอร์และสร้างไฟล์ตามโครงสร้างต่อไปนี้

frontend/



```

import { Col, Row, Card, Statistic, Table } 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}>

```

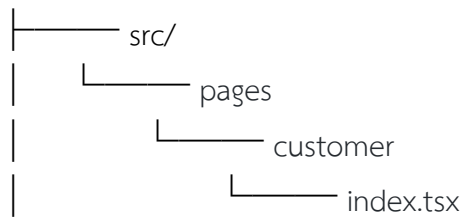
```
<Card
  bordered={false}
  style={{
    boxShadow: "rgba(100, 100, 111, 0.2) 0px 7px 29px 0px",
  }}
>
  <Statistic
    title="จำนวน"
    value={1800}
    prefix={<StockOutlined />}
  />
</Card>
</Col>
<Col xs={24} sm={24} md={12} lg={12} xl={6}>
  <Card
    bordered={false}
    style={{
      boxShadow: "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>
</>
);
}

```

สร้างหน้าแสดงข้อมูลผู้ใช้ ดำเนินการสร้างโฟลเดอร์และสร้างไฟล์ตามโครงสร้างต่อไปนี้  
frontend/



```
import { useState, useEffect } from "react";
import { Space, Table, Button, Col, Row, Divider, message } from "antd";
import { PlusOutlined, DeleteOutlined } from "@ant-design/icons";
import type { ColumnsType } from "antd/es/table";
import { GetUsers, DeleteUsersById } from "../../services/https/index";
import { UsersInterface } from "../../interfaces/IUser";
import { Link, useNavigate } from "react-router-dom";
import dayjs from "dayjs";

function Customers() {
  const navigate = useNavigate();
  const [users, setUsers] = useState<UsersInterface[]>([]);
  const [messageApi, contextHolder] = message.useMessage();
  const myId = localStorage.getItem("id");
  const columns: ColumnsType<UsersInterface> = [
    {
      title: "",
      render: (record) => (
        <>
          {myId == record?.ID ? (
            <></>
          ) : (
            <Button
              type="dashed"
              danger
              icon={<DeleteOutlined />}
              onClick={() => deleteUserById(record.ID)}
            ></Button>
          )}
        </>
      ),
    },
  ],
  {
    title: "ลำดับ",
    dataIndex: "ID",
```

```
    key: "id",
  },
  {
    title: "ชื่อ",
    dataIndex: "first_name",
    key: "first_name",
  },
  {
    title: "นามสกุล",
    dataIndex: "last_name",
    key: "last_name",
  },
  {
    title: "อีเมล",
    dataIndex: "email",
    key: "email",
  },
  {
    title: "วัน/เดือน/ปี เกิด",
    key: "birthday",
    render: (record) => <>{dayjs(record.birthday).format("DD/MM/YYYY")}</>,
  },
  {
    title: "อายุ",
    dataIndex: "age",
    key: "age",
  },
  {
    title: "เพศ",
    key: "gender",
    render: (record) => <>{record?.gender?.gender}</>,
  },
  {
    title: "",
    render: (record) => (
      <>
        <Button
          type="primary"
```

```

        icon={<DeleteOutlined />}
        onClick={() => navigate(`/customer/edit/${record.ID}`)}
      >
        แก้ไขข้อมูล
      </Button>
    </>
  ),
},
];

const deleteUserById = async (id: string) => {
  let res = await DeleteUsersById(id);
  if (res.status === 200) {
    messageApi.open({
      type: "success",
      content: res.data.message,
    });
    await getUsers();
  } else {
    messageApi.open({
      type: "error",
      content: res.data.error,
    });
  }
};

const getUsers = async () => {
  let res = await GetUsers();

  if (res.status === 200) {
    setUsers(res.data);
  } else {
    setUsers([]);
    messageApi.open({
      type: "error",
      content: res.data.error,
    });
  }
};

useEffect(() => {

```

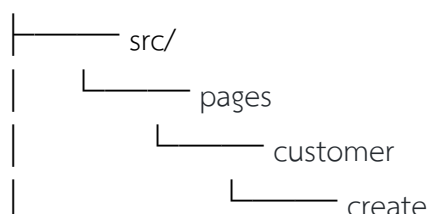
```

    getUsers();
  }, []);
  return (
    <>
      {contextHolder}
      <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}
          style={{ width: "100%", overflow: "scroll" }}
        />
      </div>
    </>
  );
}
export default Customers;

```

หน้าสร้างข้อมูลผู้ใช้ ดำเนินการสร้างโฟลเดอร์และสร้างไฟล์ตามโครงสร้างต่อไปนี้

frontend/



```
import {
  Space,
  Button,
  Col,
  Row,
  Divider,
  Form,
  Input,
  Card,
  message,
  DatePicker,
  InputNumber,
  Select,
} from "antd";
import { PlusOutlined } from "@ant-design/icons";
import { UsersInterface } from "../../interfaces/IUser";
import { CreateUser } from "../../services/https";
import { useNavigate, Link } 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 === 201) {
      messageApi.open({
        type: "success",
        content: res.data.message,
      });
      setTimeout(function () {
        navigate("/customer");
      }, 2000);
    } else {
      messageApi.open({
        type: "error",
        content: res.data.error,
      });
    }
  };
}
```



```

    }
};
return (
  <div>
    {contextHolder}
    <Card>
      <h2>เพิ่มข้อมูล ผู้ดูแลระบบ</h2>
      <Divider />
      <Form
        name="basic"
        layout="vertical"
        onFinish={onFinish}
        autoComplete="off"
      >
        <Row gutter=[[16, 0]]>
          <Col xs={24} sm={24} md={24} lg={24} xl={12}>
            <Form.Item
              label="ชื่อจริง"
              name="first_name"
              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="last_name"
              rules=[[
                {
                  required: true,
                  message: "กรุณากกรอกนามสกุล !",
                },
              ]]
            >
              <Input />
            </Form.Item>
          </Col>
        </Row>
      </Form>
    </Card>
  </div>
);

```

```
    ]]  
  >  
    <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: "กรุณกรอกอีเมล !",  
      },  
    ]]  
  >  
    <Input />  
  </Form.Item>  
</Col>  
<Col xs={24} sm={24} md={24} lg={24} xl={12}>  
  <Form.Item  
    label="รหัสผ่าน"  
    name="password"  
    rules=[  
      {  
        required: true,  
        message: "กรุณกรอกรหัสผ่าน !",  
      },  
    ]]  
  >  
    <Input.Password />  
  </Form.Item>  
</Col>  
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
```

```
<Form.Item
  label="วัน/เดือน/ปี เกิด"
  name="birthday"
  rules={[
    {
      required: true,
      message: "กรุณาเลือกวัน/เดือน/ปี เกิด !",
    },
  ]}
>
  <DatePicker style={{ width: "100%" }} />
</Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
  <Form.Item
    label="อายุ"
    name="age"
    rules={[
      {
        required: true,
        message: "กรุณากรอกอายุ !",
      },
    ]}
  >
    <InputNumber
      min={0}
      max={99}
      defaultValue={0}
      style={{ width: "100%" }}
    />
  </Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
  <Form.Item
    label="เพศ"
    name="gender_id"
    rules={[
      {
```

```
      required: true,
      message: "กรุณาเลือกเพศ!",
    },
  ]
}
>
<Select
  defaultValue=""
  style={{ width: "100%" }}
  options={[
    { value: "", label: "กรุณาเลือกเพศ", disabled: true },
    { value: 1, label: "Male" },
    { value: 2, label: "Female" },
  ]}
/>
</Form.Item>
</Col>
</Row>
<Row justify="end">
  <Col style={{ marginTop: "40px" }}>
    <Form.Item>
      <Space>
        <Link to="/customer">
          <Button htmlType="button" style={{ marginRight: "10px" }}>
            ยกเลิก
          </Button>
        </Link>
        <Button
          type="primary"
          htmlType="submit"
          icon={<PlusOutlined />}
        >
          ยืนยัน
        </Button>
      </Space>
    </Form.Item>
  </Col>
</Row>
</Form>
```

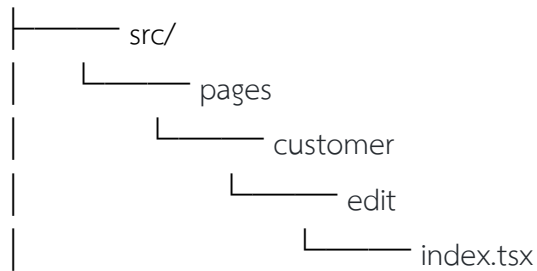
```

    </Card>
  </div>
);
}
export default CustomerCreate;

```

หน้าแก้ไขข้อมูลผู้ใช้ ดำเนินการสร้างโฟลเดอร์และสร้างไฟล์ตามโครงสร้างต่อไปนี้

frontend/



```

import { useEffect } from "react";
import {
  Space,
  Button,
  Col,
  Row,
  Divider,
  Form,
  Input,
  Card,
  message,
  DatePicker,
  InputNumber,
  Select,
} from "antd";
import { PlusOutlined } from "@ant-design/icons";
import { UsersInterface } from "../../interfaces/IUser";
import { GetUsersById, UpdateUsersById } from "../../services/https/index";
import { useNavigate, Link, useParams } from "react-router-dom";
import dayjs from "dayjs";
function CustomerEdit() {
  const navigate = useNavigate();
  const { id } = useParams<{ id: any }>();
  const [messageApi, contextHolder] = message.useMessage();
  const [form] = Form.useForm();

```

```
const getUserById = async (id: string) => {
  let res = await GetUsersById(id);
  if (res.status == 200) {
    form.setFieldsValue({
      first_name: res.data.first_name,
      last_name: res.data.last_name,
      email: res.data.email,
      birthday: dayjs(res.data.birthday),
      age: res.data.age,
      gender_id: res.data.gender?.ID,
    });
  } else {
    messageApi.open({
      type: "error",
      content: "ไม่พบข้อมูลผู้ใช้",
    });
    setTimeout(() => {
      navigate("/customer");
    }, 2000);
  }
};

const onFinish = async (values: UsersInterface) => {
  let payload = {
    ...values,
  };
  const res = await UpdateUsersById(id, payload);
  if (res.status == 200) {
    messageApi.open({
      type: "success",
      content: res.data.message,
    });
    setTimeout(() => {
      navigate("/customer");
    }, 2000);
  } else {
    messageApi.open({
      type: "error",
      content: res.data.error,
    });
  }
};
```

```

    });
  }
};

useEffect(() => {
  getUserById(id);
}, []);

return (
  <div>
    {contextHolder}
    <Card>
      <h2>แก้ไขข้อมูล ผู้ดูแลระบบ</h2>
      <Divider />
      <Form
        name="basic"
        form={form}
        layout="vertical"
        onFinish={onFinish}
        autoComplete="off"
      >
        <Row gutter=[[16, 0]]>
          <Col xs={24} sm={24} md={24} lg={24} xl={12}>
            <Form.Item
              label="ชื่อจริง"
              name="first_name"
              rules={[
                {
                  required: true,
                  message: "กรุณกรอกชื่อ!",
                },
              ]}
            >
              <Input />
            </Form.Item>
          </Col>
          <Col xs={24} sm={24} md={24} lg={24} xl={12}>
            <Form.Item
              label="นามสกุล"
              name="last_name"
            >

```

```
rules=[  
  {  
    required: true,  
    message: "กรุณากรอกนามสกุล !",  
  },  
]  
>  
  <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="วัน/เดือน/ปี เกิด"  
    name="birthday"  
    rules=[  
      {  
        required: true,  
        message: "กรุณาเลือกวัน/เดือน/ปี เกิด !",  
      },  
    ]  
  >
```



```

>
  <DatePicker style={{ width: "100%" }} />
</Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
  <Form.Item
    label="อายุ"
    name="age"
    rules={[
      {
        required: true,
        message: "กรุณากรอกอายุ !",
      },
    ]}
  >
    <InputNumber
      min={0}
      max={99}
      defaultValue={0}
      style={{ width: "100%" }}
    />
  </Form.Item>
</Col>
<Col xs={24} sm={24} md={24} lg={24} xl={12}>
  <Form.Item
    label="เพศ"
    name="gender_id"
    rules={[
      {
        required: true,
        message: "กรุณาเลือกเพศ !",
      },
    ]}
  >
    <Select
      defaultValue=""
      style={{ width: "100%" }}
      options=[

```

```

        { value: "", label: "กรุณาเลือกเพศ", disabled: true },
        { value: 1, label: "Male" },
        { value: 2, label: "Female" },
      ]}
    />
  </Form.Item>
</Col>
</Row>
<Row justify="end">
  <Col style={{ marginTop: "40px" }}>
    <Form.Item>
      <Space>
        <Link to="/customer">
          <Button htmlType="button" style={{ marginRight: "10px" }}>
            ยกเลิก
          </Button>
        </Link>
        <Button
          type="primary"
          htmlType="submit"
          icon={<PlusOutlined />}
        >
          บันทึก
        </Button>
      </Space>
    </Form.Item>
  </Col>
</Row>
</Form>
</Card>
</div>
);
}
export default CustomerEdit;

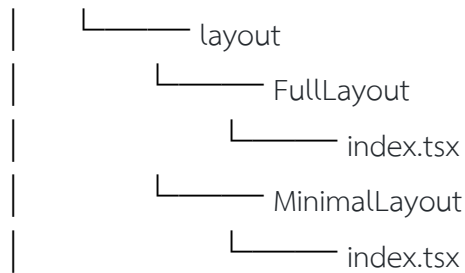
```

### 3.5 สร้าง Layout สำหรับแยกส่วนของใช้กับหน้าก่อน-หลัง เข้าสู่ระบบ

สร้าง directory ชื่อ layout ใน directory src และสร้างไฟล์ต่อไปนี้

frontend/

└── src/



#### FullLayout/ index.tsx

```
import React, { useState } from "react";
import { Routes, Route, Link } from "react-router-dom";
import "../App.css";
import { UserOutlined, DashboardOutlined } from "@ant-design/icons";
import { Breadcrumb, Layout, Menu, theme, Button, message } from "antd";
import logo from "../assets/logo.png";
import Dashboard from "../pages/dashboard";
import Customer from "../pages/customer";
import CustomerCreate from "../pages/customer/create";
import CustomerEdit from "../pages/customer/edit";
const { Header, Content, Footer, Sider } = Layout;
const FullLayout: React.FC = () => {
  const page = localStorage.getItem("page");
  const [messageApi, contextHolder] = message.useMessage();
  const [collapsed, setCollapsed] = useState(false);
  const {
    token: { colorBgContainer },
  } = theme.useToken();
  const setCurrentPage = (val: string) => {
    localStorage.setItem("page", val);
  };
  const Logout = () => {
    localStorage.clear();
    messageApi.success("Logout successful");
    setTimeout(() => {
      location.href = "/";
    }, 2000);
  };
  return (
    <Layout style={{ minHeight: "100vh" }}>
      {contextHolder}
```

```

<Sider
  collapsible
  collapsed={collapsed}
  onCollapse={(value) => setCollapsed(value)}
>
  <div
    style={{
      display: "flex",
      flexDirection: "column",
      justifyContent: "space-between",
      height: "100%",
    }}
  >
    <div>
      <div
        style={{
          display: "flex",
          justifyContent: "center",
          marginTop: 20,
          marginBottom: 20,
        }}
      >
        <img
          src={logo}
          alt="Logo"
          style={{ width: "80%" }}
        />
      </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>
</div>
<Button onClick={Logout} style={{ margin: 4 }}>
  ออกจากระบบ
</Button>
</div>
</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 />} />
        <Route path="/customer/edit/:id" element={<CustomerEdit />} />
      </Routes>
    </div>
  </Layout>
</div>

```

```

    </Content>
    <Footer style={{ textAlign: "center" }}>
      System Analysis and Design 1/67
    </Footer>
  </Layout>
</Layout>
);
};
export default FullLayout;

```

MinimalLayout/index.tsx

```

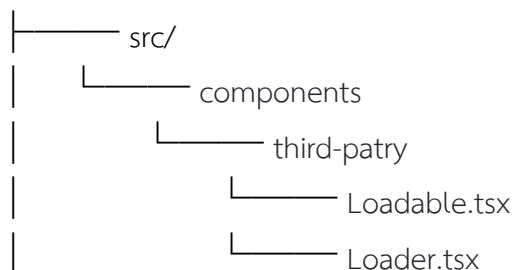
import { Outlet } from "react-router-dom";
const MinimalLayout: React.FC = () => (
  <>
    <Outlet />
  </>
);
export default MinimalLayout;

```

### 3.6 สร้าง Components สำหรับไว้เรียกใช้งานในโปรเจคของเรา

ในตัวอย่างเป็นการทำ components สำหรับการ loading ให้ดำเนินการสร้าง directory ชื่อ components ใน directory src และสร้างไฟล์ต่อไปนี้

frontend/



Loadable.tsx

```

import { Suspense, ComponentType } from "react";
import Loader from "./Loader";
const Loadable =
  <P extends object>(Component: ComponentType<P>): ComponentType<P> =>
    (props: P) =>
      (
        <Suspense fallback={<Loader />}>
          <Component {...props} />
        </Suspense>
      );

```

```

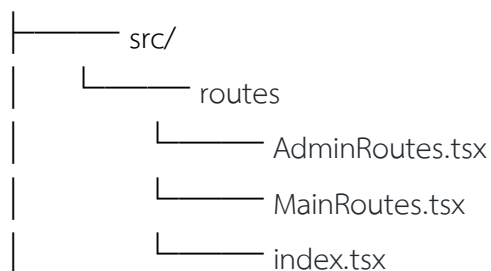
export default Loadable;
Loader.tsx
import { LoadingOutlined } from "@ant-design/icons";
const Loader: React.FC = () => (
  <div
    style={{
      position: "fixed",
      top: "50%",
      left: "50%",
      zIndex: 2000,
      width: "100%",
      height: "100%",
    }}
  >
    <LoadingOutlined
      style={{
        fontSize: 100,
        color: "#180731",
      }}
      spin
    />
  </div>
);
export default Loader;

```

### 3.7 สร้าง Router สำหรับการจัดการการนำทางในแอปพลิเคชัน

สร้าง directory ชื่อ routes ใน directory src และสร้างไฟล์ต่อไปนี้

frontend/



AdminRoutes.tsx

```

import { lazy } from "react";
import { RouteObject } from "react-router-dom";
import Loadable from "../components/third-party/Loadable";
import FullLayout from "../layout/FullLayout";

```

```

const MainPages = Loadable(lazy(() => import("../pages/authentication/Login")));
const Dashboard = Loadable(lazy(() => import("../pages/dashboard")));
const Customer = Loadable(lazy(() => import("../pages/customer")));
const CreateCustomer = Loadable(lazy(() => import("../pages/customer/create")));
const EditCustomer = Loadable(lazy(() => import("../pages/customer/edit")));
const AdminRoutes = (isLoggedIn : boolean): RouteObject => {
  return {
    path: "/",
    element: isLoggedIn ? <FullLayout /> : <MainPages />,
    children: [
      {
        path: "/",
        element: <Dashboard />,
      },
      {
        path: "/customer",
        children: [
          {
            path: "/customer",
            element: <Customer />,
          },
          {
            path: "/customer/create",
            element: <CreateCustomer />,
          },
          {
            path: "/customer/edit/:id",
            element: <EditCustomer />,
          },
        ],
      },
    ],
  },
];
};
export default AdminRoutes;

```

MainRoutes.tsx

```

import { lazy } from "react";
import React from "react";

```



```

import { RouteObject } from "react-router-dom";
import MinimalLayout from "../layout/MinimalLayout";
import Loadable from "../components/third-party/Loadable";
const MainPages = Loadable(lazy(() => import("../pages/authentication/Login")));
const Registerages = Loadable(
  lazy(() => import("../pages/authentication/Register"))
);
const MainRoutes = (): RouteObject => {
  return {
    path: "/",
    element: <MinimalLayout />,
    children: [
      {
        path: "/",
        element: <MainPages />,
      },
      {
        path: "/signup",
        element: <Registerages />,
      },
      {
        path: "*",
        element: <MainPages />,
      },
    ],
  };
};
export default MainRoutes;

```

index.tsx

```

import { useRoutes, RouteObject } from "react-router-dom";
import AdminRoutes from "../AdminRoutes";
import MainRoutes from "../MainRoutes";
function ConfigRoutes() {
  const isLoggedIn = localStorage.getItem("isLoggedIn") === "true";
  let routes: RouteObject[] = [];
  if (isLoggedIn) {
    routes = [AdminRoutes(isLoggedIn), MainRoutes()];
  } else {

```

```

    routes = [MainRoutes()];
  }
  return useRoutes(routes);
}
export default ConfigRoutes;

```

จากนั้นแก้ไขไฟล์ **App.tsx** จะอยู่ใน directory ชื่อ src

```

import React from "react";
import { BrowserRouter as Router } from "react-router-dom";
import ConfigRoutes from "./routes";
import "./App.css";
const App: React.FC = () => {
  return (
    <Router>
      <ConfigRoutes />
    </Router>
  );
};
export default App;

```

4. สั่ง npm run dev ที่ directory frontend เพื่อ start frontend

ทดสอบเพิ่มข้อมูล User จากนั้นกลับมาที่หน้าหลัก หากพบข้อมูลที่เพิ่มเข้าไป แสดงว่าระบบสามารถทำงานได้

\*\*\*\*อย่าลืม run backend\*\*\*\*

5. ทดสอบเพิ่มข้อมูล และแสดงผลข้อมูล

เปิดเว็บเบราว์เซอร์ที่ <http://localhost:5173>

ทดสอบเข้าใช้งานด้วยข้อมูลดังนี้

Email : [sa@gmail.com](mailto:sa@gmail.com)

Password: 123456

-----