

Game Shop

ร้านขายเกมออนไลน์



SC313002

PRINCIPLES OF SOFTWARE DESIGN AND
DEVELOPMENT





สมาชิก

1. 613020218-8 นายธนภัทร กิ่งสาร

2. 613020233-2 นายวัชรະ ศรีต้นวงศ์

3. 613020594-0 นายมีชัย หनुพิศ

ที่มาและความสำคัญ

ในปัจจุบันในการซื้อขายสินค้าส่วนมากจะใช้ผ่านทางอินเทอร์เน็ตมากที่สุด โดยไม่จำเป็นต้องเดินไปหน้าร้านเพื่อทำการซื้อสินค้า โดยวิดีโอเกมก็เป็นอีกหนึ่งสินค้าที่มีการซื้อขายกันอย่างแพร่หลายและ ณ ปัจจุบันนี้มีร้านที่ขายวิดีโอเกม ออนไลน์มากมาย ทางคณะผู้จัดทำเห็นในส่วนนี้จึงได้สร้างเว็บไซต์อย่างง่ายในการซื้อขายเกมออนไลน์โดยใช้หลักการ SOLID ช่วยในการออกแบบ

วัตถุประสงค์



1. เพื่อสร้างระบบฐานข้อมูลร้านขายเกมออนไลน์ โดยใช้ MySQL

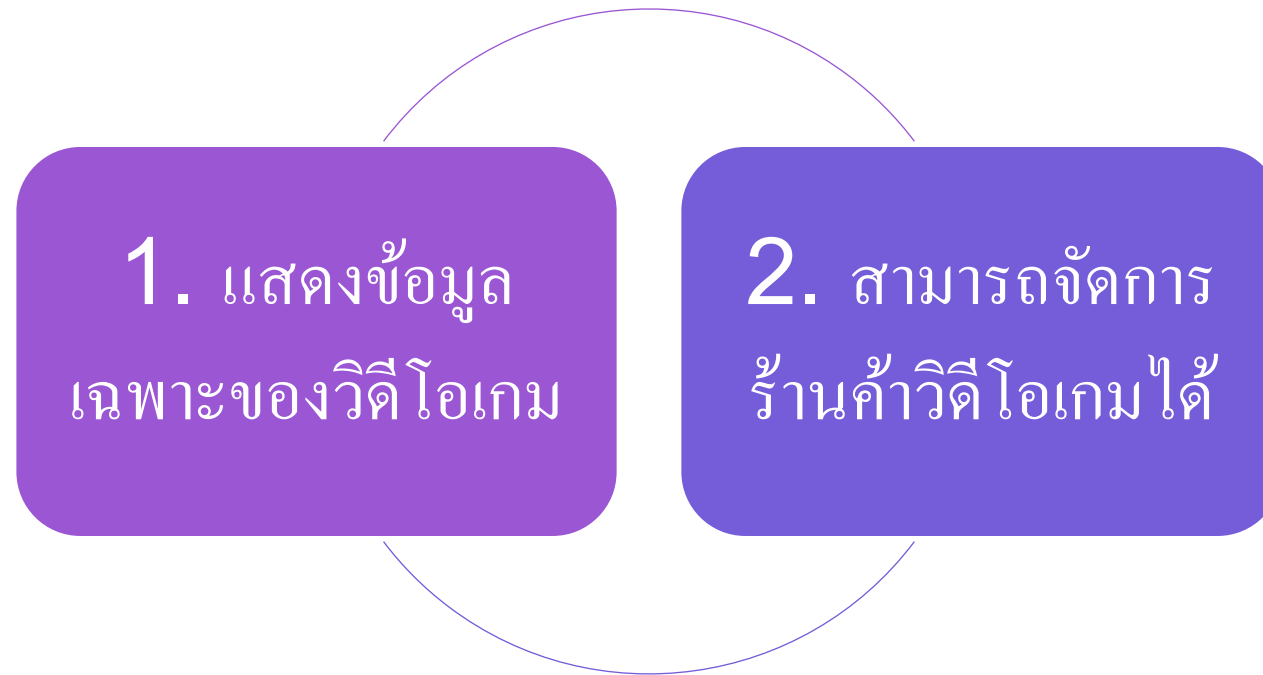


2. เพื่อให้ผู้ใช้งานสามารถซื้อสินค้าผ่านเว็บไซต์ได้



3. เพื่อให้ผู้ใช้งานสามารถเพิ่มหรือแก้ไขข้อมูลวิดีโอเกมที่วางจำหน่ายได้

ขอบเขตของการศึกษา



ประโยชน์ที่คาดว่าจะได้รับ



1. สามารถเข้าใจกระบวนการออกแบบและการทำงานของ Spring boot



2. สามารถเข้าใจหลักการ SOLID



3. โปรแกรมสามารถจัดการเกี่ยวกับวิธีโอเกมได้



4. ลูกค้าสามารถซื้อสินค้าผ่านเว็บไซต์ได้



5. ผู้จัดการสามารถดูแลร้านค้าผ่านเว็บไซต์ได้

วิธีการดำเนินงาน



แผนการดำเนินงาน



1. คิดหัวข้อเรื่อง



2. ศึกษาลักษณะข้อมูล



3. ร่างฐานข้อมูล

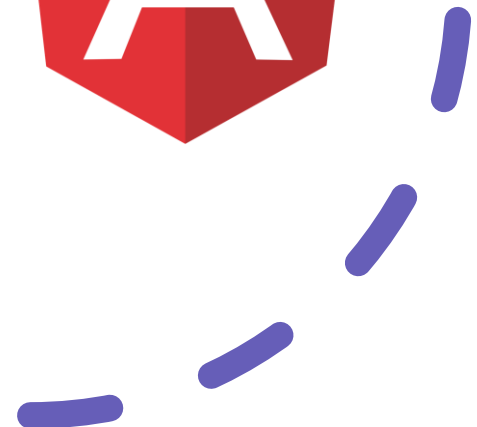
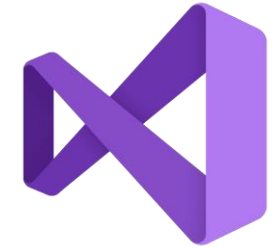


4. สร้างฐานข้อมูล



5. สร้างฟังก์ชันต่าง ๆ ดังนี้

เทคโนโลยีที่ใช้ในการ
ดำเนินงาน



หลักการออกแบบที่ สอดคล้องกับหลักการ **SOLID**

```
test03 [boot] [devtools]
└─ src/main/java
   └─ com.example.demo
      └─ com.example.demo.controller
         └─ LoginController.java
      └─ com.example.demo.controller.admin
         └─ AdminCreateController.java
         └─ AdminDeleteController.java
         └─ AdminGetController.java
         └─ AdminUpdateController.java
      └─ com.example.demo.controller.user
         └─ UserCreateController.java
         └─ UserDeleteController.java
         └─ UserGetController.java
         └─ UserUpdateController.java
      └─ com.example.demo.employee
         └─ EmployeeCreateController.java
         └─ EmployeeDeleteController.java
         └─ EmployeeGetController.java
         └─ EmployeeUpdateController.java
         └─ ProductController.java
      └─ com.example.demo.exception
         └─ ResourceNotFoundException.java
      └─ com.example.demo.model
         └─ Admin.java
         └─ Employee.java
         └─ Product.java
         └─ User.java
      └─ com.example.demo.repository
         └─ AdminRepository.java
         └─ EmployeeRepository.java
         └─ ProductRepository.java
         └─ UserRepository.java
```

1. Single Responsibility

Create admin

```
1 package com.example.demo.controller.admin;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.web.bind.annotation.CrossOrigin;
5 import org.springframework.web.bind.annotation.PostMapping;
6 import org.springframework.web.bind.annotation.RequestBody;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RestController;
9
10 import com.example.demo.model.Admin;
11 import com.example.demo.repository.AdminRepository;
12
13 @CrossOrigin(origins = "http://localhost:4200")
14 @RestController
15 @RequestMapping("/api/admin")
16 public class AdminCreateController {
17
18     @Autowired
19     private AdminRepository adminRepository;
20
21     @PostMapping("/create")
22     public Admin createAdmin(@RequestBody Admin admin) {
23         return this.adminRepository.save(admin);
24     }
25
26 }
27
```

Update admin

```
1 package com.example.demo.controller.admin;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.CrossOrigin;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.PutMapping;
8 import org.springframework.web.bind.annotation.RequestBody;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import com.example.demo.model.Admin;
13 import com.example.demo.repository.AdminRepository;
14
15 @CrossOrigin(origins = "http://localhost:4200")
16 @RestController
17 @RequestMapping("/api/admin")
18 public class AdminUpdateController {
19
20     @Autowired
21     private AdminRepository adminRepository;
22
23     // Update admin rest api
24     @PutMapping("/update/{username}")
25     public ResponseEntity<Admin> updateAdmin(@PathVariable String username, @RequestBody Admin adminDetails)
26         throws Exception {
27
28         Admin admin = adminRepository.findById(username)
29             .orElseThrow(() -> new Exception("Customer not exist with username : " + username));
30
31         // Update attributes
32         admin.setSalary(adminDetails.getSalary());
33
34         Admin updatedAdmin = adminRepository.save(admin);
35         return ResponseEntity.ok(updatedAdmin);
36     }
37
38 }
39
```

1. Single Responsibility

Delete user

```
1 package com.example.demo.controller.user;
2
3 import java.util.HashMap;
4 import java.util.Map;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.CrossOrigin;
9 import org.springframework.web.bind.annotation.DeleteMapping;
10 import org.springframework.web.bind.annotation.PathVariable;
11 import org.springframework.web.bind.annotation.RequestMapping;
12 import org.springframework.web.bind.annotation.RestController;
13
14 import com.example.demo.model.User;
15 import com.example.demo.repository.UserRepository;
16
17 @CrossOrigin(origins = "http://localhost:4200")
18 @RestController
19 @RequestMapping("/api/user")
20 public class UserDeleteController {
21
22     @Autowired
23     private UserRepository userRepository;
24
25     @DeleteMapping("/delete/{username}")
26     public ResponseEntity<Map<String, Boolean>> deleteUser(@PathVariable String username) throws Exception {
27         User user = userRepository.findById(username)
28             .orElseThrow(() -> new Exception("customer not exist with username : " + username));
29
30         // Delete
31         userRepository.delete(user);
32
33         Map<String, Boolean> response = new HashMap<>();
34         response.put("delete", Boolean.TRUE);
35
36         return ResponseEntity.ok(response);
37     }
38 }
39
40
```

Update user

```
1 package com.example.demo.controller.user;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.CrossOrigin;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.PutMapping;
8 import org.springframework.web.bind.annotation.RequestBody;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import com.example.demo.model.User;
13 import com.example.demo.repository.UserRepository;
14
15 @CrossOrigin(origins = "http://localhost:4200")
16 @RestController
17 @RequestMapping("/api/user")
18 public class UserUpdateController {
19
20     @Autowired
21     private UserRepository userRepository;
22
23     // Update user rest api
24     @PutMapping("/update/{username}")
25     public ResponseEntity<User> updateUser(@PathVariable String username, @RequestBody User userDetails)
26         throws Exception {
27
28         User user = userRepository.findById(username)
29             .orElseThrow(() -> new Exception("Customer not exist with username : " + username));
30
31         // Update attributes
32         user.setName(userDetails.getName());
33         user.setEmail(userDetails.getEmail());
34         user.setMobile(userDetails.getMobile());
35         user.setAddress(userDetails.getAddress());
36         user.setMobile(userDetails.getMobile());
37
38         User updatedUser = userRepository.save(user);
39         return ResponseEntity.ok(updatedUser);
40     }
41 }

```

2. Liskov Substitution & Interface Segregation principle

Admin Repository

```
AdminRepository.java
1 package com.example.demo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.demo.model.Admin;
7
8 @Repository
9 public interface AdminRepository extends JpaRepository<Admin, String>{
10
11 }
12
```

Cart Repository

```
CartRepository.java
1 package com.example.demo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.demo.model.Cart;
7
8 @Repository
9 public interface CartRepository extends JpaRepository<Cart, Long> {
10
11 }
12
```

2. Liskov Substitution & Interface Segregation principle

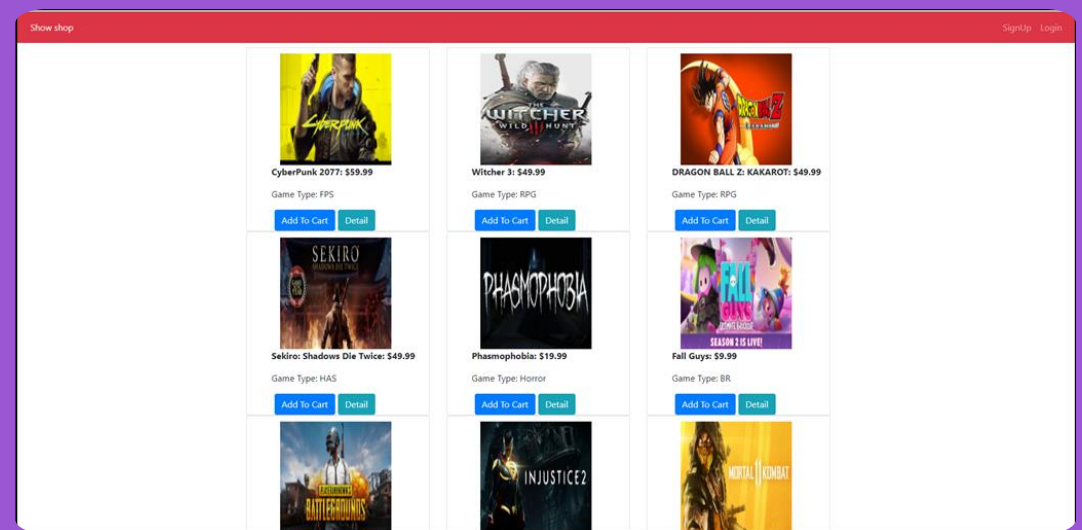
Employee Repository

```
EmployeeRepository.java
1 package com.example.demo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.demo.model.Employee;
7
8 @Repository
9 public interface EmployeeRepository extends JpaRepository<Employee, String> {
10
11 }
12
```

Invoice Repository

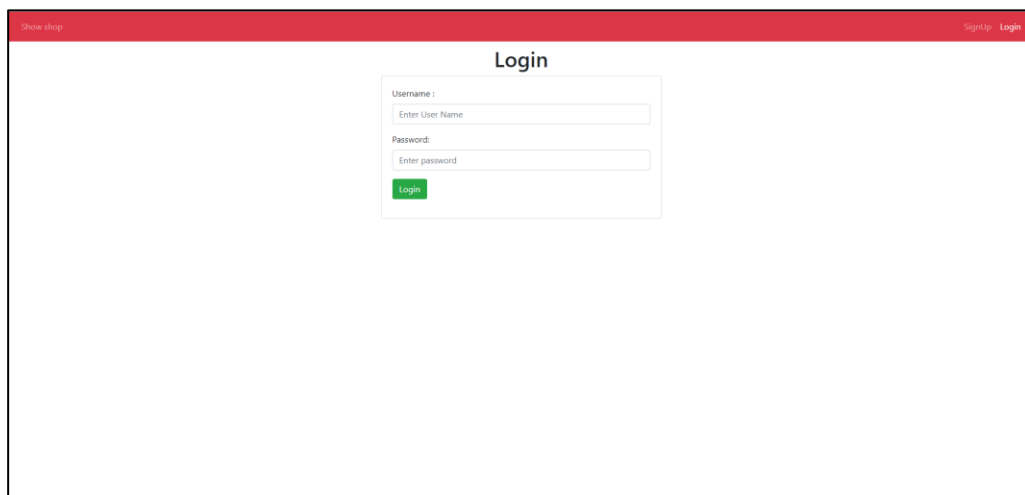
```
InvoiceRepository.java
1 package com.example.demo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.demo.model.Invoice;
7
8 @Repository
9 public interface InvoiceRepository extends JpaRepository<Invoice, Long>{
10
11 }
```

ตัวอย่างเว็บไซต์



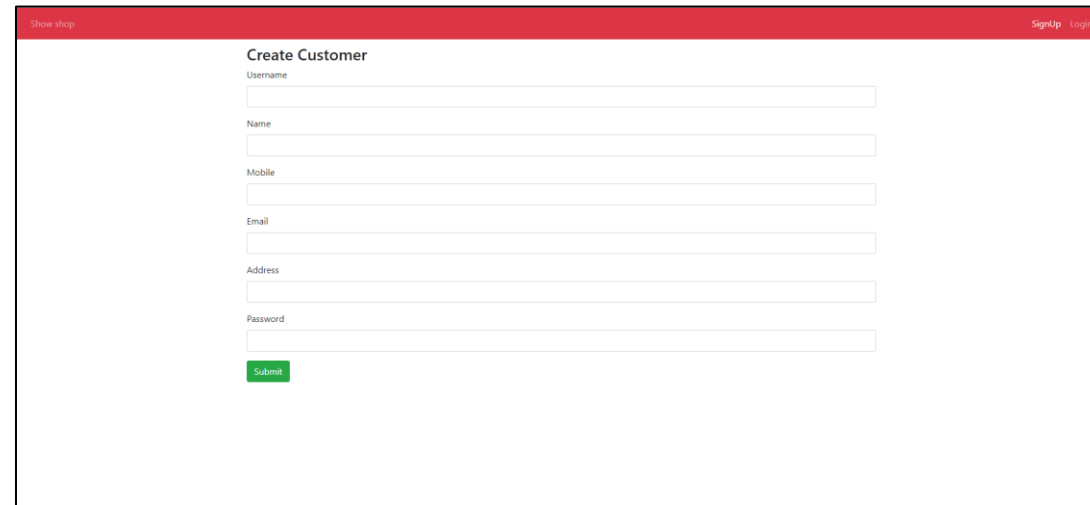
ตัวอย่างเว็บไซต์

1. หน้าเข้าสู่ระบบ



A mockup of a login page. It features a red header bar with the text "Show shop" on the left and "SignUp Login" on the right. The main content area is white and contains a centered "Login" form. The form has a title "Login" and two input fields: "Username:" with a placeholder "Enter User Name" and "Password:" with a placeholder "Enter password". Below the password field is a green "Login" button.

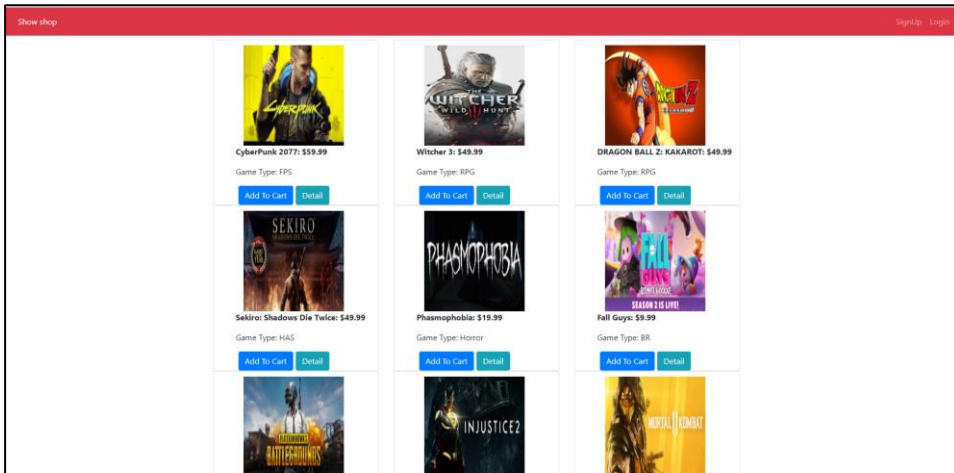
2. หน้าลงทะเบียน



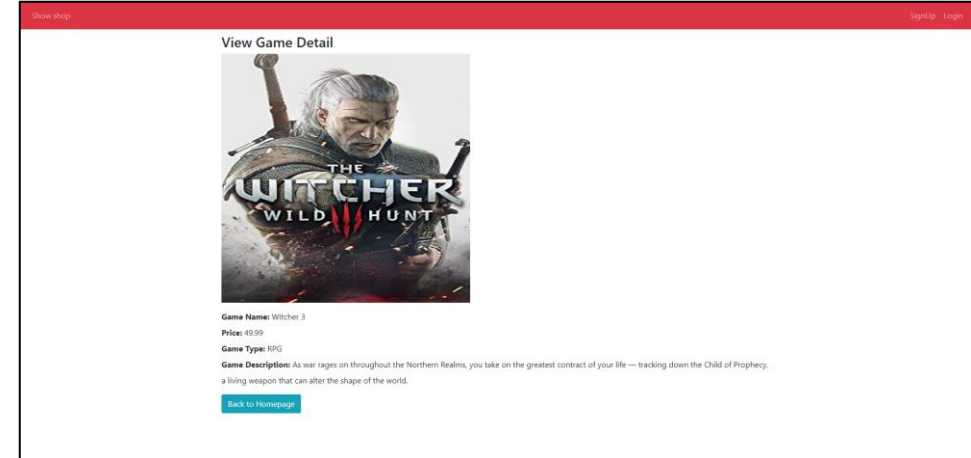
A mockup of a "Create Customer" registration page. It features a red header bar with the text "Show shop" on the left and "SignUp Login" on the right. The main content area is white and contains a centered "Create Customer" form. The form has a title "Create Customer" and six input fields: "Username", "Name", "Mobile", "Email", "Address", and "Password". Below the password field is a green "Submit" button.

ตัวอย่างเว็บไซต์

3. หน้าแสดงรายการสินค้า



3. หน้าแสดงรายละเอียดสินค้า








ตัวอย่างเว็บไซต์ (ส่วนจัดการหลังบ้านของ **admin**)

4. ส่วนจัดการกับรายการสินค้า

6130205942
meehaci

Game List

Game	Game Name	Game Price	Game Description	Game Type	Game Quantity	Action
	Cyberpunk 2077	59.99	Cyberpunk 2077 is an open-world, action-adventure story set in Night City, a megalopolis obsessed with power, glamour and body modification. You play as V, a mercenary outlaw going after a one-of-a-kind implant that is the key to immortality.	FPS	1	Update Delete
	Witcher 3	49.99	As war rages on throughout the Northern Realms, you take on the greatest contract of your life — tracking down the Child of Prophecy, a living weapon that can alter the shape of the world.	RPG	1	Update Delete
	DRAGON BALL Z: KAKAROT	49.99	Relive the story of Goku and other Z Fighters in DRAGON BALL Z: KAKAROT! Beyond the epic battles, experience life in the DRAGON BALL Z world as you fight, fish, eat, and train with Goku, Gohan, Vegeta and others.	RPG	1	Update Delete
	Sekiro: Shadows Die Twice	49.99	Carve your own clever path to vengeance in the award winning adventure from developer FromSoftware, creators of Bloodborne and the Dark Souls series. Take Revenge. Restore Your Honor. Kill Ingeniously.	HAS	1	Update Delete
	Phasmophobia	19.99	Phasmophobia is a 4 player online co-op psychological horror. Paranormal activity is on the rise and it's up to you and your team to use all the ghost hunting equipment at your disposal in order to gather as much evidence as you can.	Horror	1	Update Delete

5. หน้าแสดงส่วนจัดการต่าง ๆ ของ admin

6130205942
meehaci

Admin Home
Manage Store
User List
Employee List
Admin List

ตัวอย่างเว็บไซต์ (ส่วนจัดการหลังบ้านของ **admin**)

4. การเพิ่มพนักงาน

6130205942
meehaci

Create Employee

Username

Name

Mobile

Email

Address

Password

Salary

5. หน้าแสดงส่วนจัดการ User

6130205942
meehaci

User List

Username	Name	Password	Role	Mobile	Email	Address	Action
6130205940	meehaci nineo	123456	employee	0622233174	meechainuphit@kkumail.com	Khon Kaen	Update Detail Delete
6130205941	meehaci	123456	admin	0622233174	meechainuphit@kkumail.com	Khon Kaen	Update Detail Delete
6130205942	meehaci	12355	admin	0622233174	meechainuphit@kkumail.com	Khon Kaen	Update Detail Delete
6130205943	มีชัย ชาญพิศ 6130205940	dfgdfg	admin	0622233174	meechainuphit@kkumail.com	Khon Kaen	Update Detail Delete
6130205944	meehaci nineo	12355	admin	0622233174	meechainuphit@kkumail.com	Khon Kaen	Update Detail Delete
aaaa	meehaci nineo	12355	employee	0622233174	meechainuphit@kkumail.com	Khon Kaen	Update Detail Delete

ขอบคุณครับ
:)

