**ระบบจัดการจองร้านบอร์ดเกม**

**Board game store booking management system**

`

นายภาณุพงศ์ วิเศษชลธาร

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา

หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการข้อมูลและนวัตกรรมซอฟต์แวร์

ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยอุบลราชธานี

ปีการศึกษา 2568

**ลิขสิทธิ์ของมหาวิทยาลัยอุบลราชธานี**

โครงงาน : ระบบจัดการจองร้านบอร์ดเกม

Board game store booking management system

โดย : นายภาณุพงศ์ วิเศษชลธาร

อาจารย์ที่ปรึกษา : อาจารย์ วาโย ปุยะติ

ระดับการศึกษา : วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการข้อมูลและนวัตกรรม

ซอฟต์แวร์

ปีการศึกษา : 2568

ได้รับการพิจารณาให้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการข้อมูลและนวัตกรรมซอฟต์แวร์

คณะกรรมการสอบประเมินความรู้โครงงานของคอมพิวเตอร์

…………………………………………………………. อาจารย์ที่ปรึกษา

( ผศ.ดร.ไพชยนต์ คงไชย)

……………………………………………………………………. กรรมการ

( ดร.วิชิต สมบัติ)

……………………………………………………………………. กรรมการ

( ผศ.ดร.สุภาวดี หิรัญพงศ์สิน)

…………………………………………………………. หัวหน้าภาควิชา

( รศ.ชาญชัย ศุภอรรถกร )

วันที่ ………../……………./…………….

# **กิตติกรรมประกาศ**

ชื่อผู้พัฒนา

วัน เดือน ปี

โครงงาน : ระบบจัดการจองร้านบอร์ดเกม

Board game store booking management system

โดย : นายภาณุพงศ์ วิเศษชลธาร

อาจารย์ที่ปรึกษา : อาจารย์ วาโย ปุยะติ

ระดับการศึกษา : วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการข้อมูลและนวัตกรรมซอฟต์แวร์

ปีการศึกษา : 2568

# **บทคัดย่อ**

Topic : Board game store booking management system

Author : Phanuphong Wisetchluelathan

Advisor :

Degree : Bachelor of Science (Data Science and Software Innovation)

Academic Year : 2025

# **Abstract**

# สารบัญ

หัวข้อ หน้า

กิตติกรรมประกาศ 3

บทคัดย่อ 4

Abstract 5

สารบัญ 6

สารบัญภาพ 6

**บทที่ 1 บทนำ 1**

1.1 ที่มาและความสำคัญของโครงงาน 1

1.2 วัตถุประสงค์ของโครงงาน 2

1.3 ขอบเขตของโครงงาน 2

1.4 ประโยชน์ที่คาดว่าจะได้รับ 3

1.5 แผนการดำเนินงาน 4

**บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 5**

2.1 ระบบการจอง (Booking System) 5

2.2 ภาษา Python และ Django Framework 6

2.3 ระบบจัดการฐานข้อมูล PostgreSQL 10

2.4 การตกแต่งหน้าเว็บไซต์ด้วย Tailwind CSS 12

2.5 โปรแกรม Visual Studio Code 15

2.6 งานวิจัยที่เกี่ยวข้อง 16

**บทที่ 3 วิธีการดำเนินงาน 18**

3.1 การรวบรวมข้อมูลและความต้องการของระบบ 18

3.2 การวิเคราะห์และออกแบบระบบ (System Analysis and Design) 19

3.2.1 แผนภาพแสดงกรณีการใช้งาน (Use Case Diagram) 19

3.2.2 คำอธิบายกรณีการใช้งาน (Use Case Description) 20

3.2.3 แผนภาพกิจกรรม (Activity Diagram) 25

3.2.4 แผนภาพความสัมพันธ์ของข้อมูล (ER Diagram) 30

3.2.5 พจนานุกรมข้อมูล (Data Dictionary) 31

3.3 การออกแบบส่วนติดต่อผู้ใช้งาน (User Interface Design) 35

**บทที่ 4 ผลการดำเนินงาน45**

4.1 ส่วนของผู้ใช้งานทั่วไป (Customer) 45

4.2 ส่วนของผู้ดูแลระบบ (Admin) 55

**บทที่ 5 การทดสอบระบบ 65**

5.1 การทดสอบหน่วยย่อย (Unit Testing) 65

5.1.1 การทดสอบฟังก์ชันคำนวณสต็อกและตรวจสอบข้อมูล 65

5.2 การทดสอบการทำงานร่วมกัน (Integration Testing) 66

5.2.1 การทดสอบกระบวนการตั้งแต่จองนัดหมายจนถึงการบันทึก 66

5.3 ผลการทดสอบและกรณีทดสอบ (Test Cases) 67

5.4 การประเมินความพึงพอใจและประสิทธิภาพของระบบ 70

**บทที่ 6 สรุปและข้อเสนอแนะ 72**

6.1 สรุปความสามารถของระบบ 72

6.2 ปัญหาและอุปสรรคในการพัฒนา 73

6.3 แนวทางในการพัฒนาต่อ 73

**บรรณานุกรม 75**

**ภาคผนวก 77**

ภาคผนวก ก Source Code หรือ โครงสร้างฐานข้อมูล 78

ภาคผนวก ข คู่มือการติดตั้งระบบ 85

ภาคผนวก ค คู่มือการใช้งานของระบบ 88

# 

# 

# 

# 

# 

# 

# สารบัญภาพ

ภาพประกอบที่ หน้า

3.1 แผนภาพแสดงกรณีการใช้งาน (Use Case Diagram) ภาพรวมของระบบ 19

3.2 แผนภาพกิจกรรม (Activity Diagram) ส่วนการสมัครสมาชิกและเข้าสู่ระบบ 25

3.3 แผนภาพกิจกรรม (Activity Diagram) ขั้นตอนการจอง (Booking Process) 26

3.4 แผนภาพกิจกรรม (Activity Diagram) ส่วนของผู้ดูแลระบบ (Admin) 27

3.5 แผนภาพความสัมพันธ์ของข้อมูล (Entity Relationship Diagram) 30

3.6 ผังโครงสร้างหน้าเว็บไซต์ (Sitemap)35ค.1หน้าจอหลัก (Home Page) และเมนูเข้าสู่ระบบ 89

ค.2 หน้าจอสมัครสมาชิก (Register)89ค.3หน้าจอเข้าสู่ระบบ (Login) 90

ค.4 ขั้นตอนที่ 1 การเลือกวันที่ (Select Date) 90

ค.5 ขั้นตอนที่ 2 การเลือกช่วงเวลา (Select Timeslot) 91

ค.6 ขั้นตอนที่ 3 การเลือกบอร์ดเกม (Select Game) 91

ค.7 ขั้นตอนที่ 4 สรุปรายการจอง (Booking Summary) 92

ค.8 หน้าจอแจ้งผลการจองสำเร็จ (Booking Success) 92

ค.9 หน้าจอประวัติการจอง (Booking History) 93

ค.10 หน้าจอแก้ไขโปรไฟล์ (User Profile) 93

ค.11 หน้าจอแดชบอร์ดผู้ดูแลระบบ (Admin Dashboard) 94

ค.12 หน้าจอรายการการจองทั้งหมด (Booking List) 94

ค.13 หน้าจอแก้ไขสถานะการจอง 95

ค.14 หน้าจอจัดการคลังเกม (Game Management) 95

ค.15 หน้าจอเพิ่มบอร์ดเกมใหม่ (Add Game Form) 96

ค.16 หน้าจอจัดการข้อมูลโต๊ะและช่วงเวลา (Table & Timeslot) 96

ค.17 หน้าจอรายชื่อสมาชิก (Customer List) 97

ค.18 หน้าจอปรับสถานะผู้ใช้งาน (Suspend User) 97

# 

# 

# 

# 

# 

# 

# สารบัญตาราง

ตารางที่ หน้า

3.1 คำอธิบายกรณีการใช้งาน การสมัครสมาชิก (Register) 20

3.2 คำอธิบายกรณีการใช้งาน การเข้าสู่ระบบ (Login) 21

3.3 คำอธิบายกรณีการใช้งาน การจัดการโปรไฟล์ (Manage Profile) 21

3.4 คำอธิบายกรณีการใช้งาน การเลือกวันที่และเวลา (Select Date & Time) 22

3.5 คำอธิบายกรณีการใช้งาน การเลือกโต๊ะและบอร์ดเกม (Select Table & Game) 22

3.6 คำอธิบายกรณีการใช้งาน การยืนยันการจอง (Confirm Booking) 23

3.7 คำอธิบายกรณีการใช้งาน การดูประวัติการจอง (View History) 23

3.8 คำอธิบายกรณีการใช้งาน การยกเลิกการจอง (Cancel Booking) 24

3.9 คำอธิบายกรณีการใช้งาน การจัดการข้อมูลบอร์ดเกม (Manage Game - Admin) 24

3.10 คำอธิบายกรณีการใช้งาน การจัดการข้อมูลการจอง (Manage Booking - Admin) 25

3.11 คำอธิบายกรณีการใช้งาน การจัดการข้อมูลสมาชิก (Manage Customer - Admin) 25

3.12 คำอธิบายกรณีการใช้งาน การดูภาพรวมระบบ (View Dashboard - Admin) 26

3.13 พจนานุกรมตาราง ผู้ใช้งาน (Accounts\_User) 31

3.14 พจนานุกรมตาราง การจอง (Booking\_Booking) 32

3.15 พจนานุกรมตาราง รายการเกมที่จอง (Booking\_BookingItem) 32

3.16 พจนานุกรมตาราง บอร์ดเกม (Booking\_Game) 33

3.17 พจนานุกรมตาราง โต๊ะบอร์ดเกม (Booking\_TableBooking) 33

3.18 พจนานุกรมตาราง ช่วงเวลา (Booking\_Timeslot) 34

5.1 ผลการทดสอบระบบส่วนการจัดการผู้ใช้งาน (User Management) 67

5.2 ผลการทดสอบระบบการจอง (Booking System) 68

5.3 ผลการทดสอบระบบผู้ดูแลระบบ (Admin Management) 69

5.4 ผลการประเมินความพึงพอใจของผู้ใช้งาน 71

# บทที่ 1 บทนำ

## 1.1 ที่มาและเหตุผล

ปัจจุบันการจัดการการจองคิวบริการในร้านบอร์ดเกมยังคงประสบปัญหาด้านประสิทธิภาพและความโปร่งใส เนื่องจากมักมีการจัดการด้วยมือ ทำให้เกิดความซับซ้อนและข้อผิดพลาดได้ง่าย ผู้ประกอบการและลูกค้าจึงต้องเผชิญกับความท้าทายในการติดตามสถานะการจอง โต๊ะที่ว่าง และประวัติการจอง ปัญหาเหล่านี้ส่งผลให้เกิดความยุ่งยากในการบริหารจัดการคิวและทรัพยากร

ด้วยเหตุผลดังกล่าว ผู้พัฒนาจึงได้ศึกษาและพัฒนาระบบจัดการการจองร้านบอร์ดเกมขึ้นมา เพื่อแก้ไขปัญหาที่เกิดขึ้น โดยมีจุดมุ่งหมายเพื่อสร้างสภาพแวดล้อมที่สะดวกสบายสำหรับทั้งผู้ประกอบการและลูกค้า ระบบนี้จะช่วยจัดเก็บและบริหารจัดการข้อมูลการจองคิวแบบรวมศูนย์ เพื่อให้การดำเนินการจองเป็นไปอย่างมีประสิทธิภาพและโปร่งใส ข้อมูลที่ถูกบันทึกในระบบจะสามารถติดตามและตรวจสอบรายละเอียดการจอง เวลาที่ว่าง และประวัติการเข้าใช้บริการของลูกค้าแต่ละรายได้ง่ายดาย ระบบจัดการการจองนี้จะรับรองว่าข้อมูลไม่ถูกซ้ำซ้อนหรือตกหล่น และสามารถตรวจสอบข้อมูลย้อนกลับได้ เพื่อรับประกันความถูกต้องและประสิทธิภาพของข้อมูลการจอง

## 1.2 วัตถุประสงค์

1. เพื่อพัฒนาระบบจัดการการจองคิวร้านบอร์ดเกมผ่านเว็บไซต์ โดยให้ลูกค้าสามารถตรวจสอบตารางเวลาว่าง เลือกโต๊ะ และเลือกเกมที่ต้องการเล่นได้อย่างสะดวก รวดเร็ว และลดข้อผิดพลาดจากการจองซ้ำ
2. เพื่อเพิ่มประสิทธิภาพในการบริหารจัดการการจองโต๊ะของร้าน

## 1.3 ขอบเขตของโครงงาน

1. ขอบเขตการพัฒนาระบบ: โครงงานนี้เป็นการพัฒนาเว็บแอปพลิเคชัน (Web Application) ที่สามารถเข้าถึงและใช้งานได้ผ่านเว็บเบราว์เซอร์บนระบบปฏิบัติการต่างๆ โดยมุ่งเน้นการทำงานบนอุปกรณ์คอมพิวเตอร์ส่วนบุคคล (PC)
2. ชุดข้อมูลที่ศึกษา: ข้อมูลหลักที่ใช้ในการศึกษาและจัดเก็บคือ ข้อมูลการจองคิว (เช่น วันที่, เวลา, โต๊ะที่เลือก), ข้อมูลลูกค้า (ชื่อ, เบอร์โทรศัพท์, อีเมล), ข้อมูลโต๊ะบอร์ดเกม (หมายเลขโต๊ะ, สถานะว่าง/ไม่ว่าง)
3. กลุ่มเป้าหมาย/กลุ่มผู้ใช้:

3.1 ผู้ประกอบการ/ผู้ดูแลร้าน: บุคลากรของร้านบอร์ดเกมที่ต้องการจัดการระบบการจองคิว, ตรวจสอบสถานะโต๊ะ, และดูข้อมูลการจองของลูกค้า

3.2 ลูกค้า: ผู้ที่ต้องการจองคิวเล่นบอร์ดเกมที่ร้าน

## 1.4 ประโยชน์ที่ได้รับ

1. ได้ ระบบจัดการการจองร้านบอร์ดเกมบนเว็บไซต์ ที่ช่วยแก้ไขปัญหาความยุ่งยากและข้อผิดพลาดในการจัดการการจองแบบดั้งเดิม
2. เพิ่มประสิทธิภาพและลดภาระงาน ของบุคลากรในร้านบอร์ดเกมในการบริหารจัดการคิวและโต๊ะ
3. เพิ่มความสะดวกสบายและรวดเร็ว ให้กับลูกค้าในการจองคิว รวมถึงการได้รับข้อมูลและยืนยันการจองผ่านอีเมล
4. เพิ่มความน่าเชื่อถือและความโปร่งใส ในการบริหารจัดการข้อมูลการจอง โดยมีการจัดเก็บข้อมูลแบบรวมศูนย์และสามารถตรวจสอบย้อนหลังได้

## 1.5 เครื่องมือที่ใช้ในการพัฒนา

### 1.5.1 ฮาร์ดแวร์

1. คอมพิวเตอร์ส่วนบุคคล สำหรับพัฒนาและทดสอบระบบ

- CPU: AMD Ryzen 5 5600X

- RAM: 16 GB

- GPU: NVIDIA GeForce GTX 1080 Ti

- SSD: 2 TB

2. เครื่อง Server สำหรับ Deploy ระบบ Production

- สามารถใช้ Cloud Server เช่น AWS EC2, DigitalOcean หรือ Local Server

1.5.2 ซอฟต์แวร์สำหรับการพัฒนา (Software Development Tools)

1. Visual Studio Code (VS Code): เป็นเครื่องมือหลักในการเขียนโค้ด (IDE) รองรับการทำงานร่วมกับภาษา Python และ HTML/CSS ได้ดี
2. Git: สำหรับจัดการเวอร์ชันของซอฟต์แวร์ (Version Control)
3. Docker: สำหรับจำลองสภาพแวดล้อมในการติดตั้งและรันระบบ (Containerization) เพื่อให้มั่นใจว่าระบบสามารถทำงานได้ในทุกสภาพแวดล้อม
4. Google Chrome / Microsoft Edge: เว็บเบราว์เซอร์สำหรับทดสอบการแสดงผลและการทำงานของระบบ

1.5.3 เทคโนโลยีที่ใช้ในการพัฒนาระบบ (Development Technologies)

1. ภาษา Python (เวอร์ชัน 3.x): เป็นภาษาหลักในการประมวลผลฝั่งเซิร์ฟเวอร์ (Back-end) เนื่องจากมีความยืดหยุ่นและมีไลบรารีสนับสนุนจำนวนมาก
2. Django Framework (เวอร์ชัน 5.2.8): เว็บเฟรมเวิร์กสำหรับภาษา Python ที่ใช้สถาปัตยกรรมแบบ MVT (Model-View-Template) ช่วยให้การจัดการฐานข้อมูล ความปลอดภัย และการเชื่อมโยง URL ทำได้ง่ายและรวดเร็ว
3. PostgreSQL: ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS) ที่มีความเสถียรสูง ใช้สำหรับจัดเก็บข้อมูลสมาชิก ข้อมูลการจอง และข้อมูลสต็อกเกม
4. HTML5 และ CSS3: สำหรับสร้างโครงสร้างและจัดรูปแบบหน้าเว็บไซต์
5. Tailwind CSS: เฟรมเวิร์ก CSS แบบ Utility-first ที่ช่วยในการออกแบบหน้าเว็บให้มีความสวยงามและรองรับการแสดงผลบนอุปกรณ์ต่าง ๆ (Responsive Design) ได้อย่างรวดเร็ว
6. JavaScript: ใช้สำหรับการทำงานฝั่งไคลเอนต์ (Client-side) เพื่อเพิ่มลูกเล่นและการตอบสนองของผู้ใช้
7. Chart.js: ไลบรารี JavaScript สำหรับสร้างกราฟและแผนภูมิแสดงสถิติในหน้าผู้ดูแลระบบ (Dashboard)

## 1.6 แผนการดำเนินการ

ในการพัฒนาระบบจองร้านบอร์ดเกม (Board Game Cafe Booking System) มีการแบ่งการดำเนินการออกเป็น 4 ระยะ ดังนี้

ตารางที่ 1.1 แผนการดำเนินการพัฒนาระบบ

| ระยะ (Phase) | กิจกรรมดำเนินงาน | ผลลัพธ์ที่คาดหวัง | ช่วงเวลาดำเนินการ |
| --- | --- | --- | --- |
| 1. การวางแผนและรวบรวมข้อมูล | ศึกษาปัญหาและกระบวนการจองคิวร้านบอร์ดเกม รวบรวมความต้องการจากผู้ใช้ (User Requirements) และศึกษาเครื่องมือที่ใช้พัฒนา (Django, Tailwind CSS) | เอกสารวิเคราะห์ความต้องการและรายละเอียดคุณลักษณะของระบบ (SRS) | สัปดาห์ที่ 1 - 2 |
| 2. การวิเคราะห์และออกแบบระบบ | ออกแบบแผนภาพระบบ (UML) ได้แก่ Use Case, Class Diagram, ออกแบบฐานข้อมูล (ER-Diagram) และออกแบบส่วนติดต่อผู้ใช้ (UI/UX Wireframe) | แผนภาพวิเคราะห์ระบบ, โครงสร้างฐานข้อมูล (Database Schema) และต้นแบบหน้าจอ (Wireframe) | สัปดาห์ที่ 3 - 5 |
| 3. การพัฒนาระบบสารสนเทศ | ดำเนินการติดตั้ง Environment พัฒนาส่วนฐานข้อมูล พัฒนาส่วนหลัง (Back-end) ระบบจอง/ตัดสต็อก และส่วนหน้า (Front-end) | ซอฟต์แวร์ระบบจัดการจองร้านบอร์ดเกมที่สามารถทำงานได้จริงตามขอบเขตงาน | สัปดาห์ที่ 6 - 12 |
| 4. การทดสอบและการจัดทำเอกสาร | ทดสอบการทำงานของระบบ (Unit & Integration Testing) แก้ไขข้อผิดพลาด (Bug Fixing) และจัดทำคู่มือการใช้งานและรายงานโครงงาน | รายงานผลการทดสอบ, ระบบที่สมบูรณ์พร้อมใช้งาน, เล่มรายงานโครงงาน และคู่มือผู้ใช้งาน | สัปดาห์ที่ 13 - 16 |

# บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

ในการพัฒนาระบบจัดการการจองร้านบอร์ดเกม (Board Game Cafe Booking System) ผู้พัฒนาได้ศึกษาทฤษฎี หลักการ และเทคโนโลยีต่างๆ ที่เกี่ยวข้อง เพื่อนำมาประยุกต์ใช้ในการวิเคราะห์ ออกแบบ และพัฒนาระบบให้มีประสิทธิภาพ โดยมีรายละเอียดดังนี้

## 2.1 ระบบเว็บแอปพลิเคชัน (Web Application)

เว็บแอปพลิเคชัน คือ โปรแกรมประยุกต์ที่เข้าถึงได้ผ่านโปรแกรมค้นดูเว็บ (Web Browser) ผ่านเครือข่ายอินเทอร์เน็ต หรือ อินทราเน็ต โดยผู้ใช้งานไม่จำเป็นต้องติดตั้งโปรแกรมลงในเครื่องคอมพิวเตอร์ การทำงานของเว็บแอปพลิเคชันจะเป็นรูปแบบ Client-Server โดยฝั่งผู้ใช้ (Client) จะทำการส่งคำร้องขอ (Request) ไปยังเครื่องแม่ข่าย (Server) เพื่อประมวลผลและส่งผลลัพธ์กลับมาแสดงผลในรูปแบบของหน้าเว็บไซต์

สำหรับระบบจองร้านบอร์ดเกมนี้ การพัฒนาในรูปแบบเว็บแอปพลิเคชันช่วยให้ลูกค้าสามารถเข้าถึงระบบได้จากทุกอุปกรณ์ ทั้งคอมพิวเตอร์และสมาร์ตโฟน โดยไม่ต้องติดตั้งแอปพลิเคชันเพิ่มเติม

## 2.2 สถาปัตยกรรม Model-View-Template (MVT)

เนื่องจากระบบนี้พัฒนาด้วย Django Framework จึงใช้รูปแบบสถาปัตยกรรมซอฟต์แวร์แบบ MVT (Model-View-Template) ซึ่งเป็นรูปแบบการออกแบบที่แบ่งส่วนการทำงานของโปรแกรมออกเป็น 3 ส่วนหลัก เพื่อให้ง่ายต่อการพัฒนาและดูแลรักษา

1. Model (M): ส่วนที่ทำหน้าที่จัดการกับข้อมูลและโครงสร้างของฐานข้อมูล (Database Schema) ทำหน้าที่รับส่งข้อมูลระหว่างฐานข้อมูลกับส่วนประมวลผล
2. View (V): ส่วนที่ทำหน้าที่ประมวลผลทางตรรกะ (Business Logic) รับคำสั่งจากผู้ใช้ (User Request) ดึงข้อมูลจาก Model มาประมวลผล และส่งต่อไปยัง Template เพื่อแสดงผล
3. Template (T): ส่วนที่ทำหน้าที่แสดงผลต่อผู้ใช้งาน (User Interface) โดยเขียนด้วยภาษา HTML ที่ผสมผสานกับ Django Template Language (DTL) เพื่อแสดงข้อมูลแบบไดนามิก

## 2.3 ภาษาและเฟรมเวิร์กที่ใช้ในการพัฒนา

### 2.3.1 ภาษา Python

Python เป็นภาษาระดับสูง (High-level programming language) ที่ได้รับความนิยมอย่างแพร่หลาย เนื่องจากมีโครงสร้างทางภาษาที่ไม่ซับซ้อน อ่านและเขียนง่าย มีไลบรารีรองรับการทำงานที่หลากหลาย ในโครงงานนี้ใช้ Python เวอร์ชัน 3.x เป็นภาษาหลักในการพัฒนาระบบฝั่ง Server (Back-end) เพื่อจัดการลอจิกการจอง การคำนวณสต็อกเกม และการจัดการสิทธิ์ผู้ใช้งาน

### 2.3.2 Django Framework

Django เป็น Web Framework ระดับสูงที่เขียนด้วยภาษา Python ซึ่งถูกออกแบบมาเพื่อช่วยให้การพัฒนาเว็บไซต์ทำได้รวดเร็ว ปลอดภัย และดูแลรักษาได้ง่าย (Rapid Development) โดยในโครงงานนี้เลือกใช้ Django เวอร์ชัน 5.2.8 ซึ่งมีจุดเด่นดังนี้:

* Built-in Admin Interface: มีระบบจัดการหลังบ้านสำหรับผู้ดูแลระบบมาให้พร้อมใช้งาน ทำให้ง่ายต่อการจัดการข้อมูลสมาชิก โต๊ะ และเกม
* Security: มีระบบความปลอดภัยพื้นฐานที่ช่วยป้องกันการโจมตีเว็บไซต์ เช่น CSRF (Cross Site Request Forgery) และ SQL Injection
* Authentication System: มีระบบจัดการผู้ใช้งาน (User Authentication) สำหรับการสมัครสมาชิกและเข้าสู่ระบบที่มีความปลอดภัยสูง

## 2.4 ระบบจัดการฐานข้อมูล (Database Management System)

### 2.4.1 PostgreSQL

PostgreSQL เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System - RDBMS) แบบ Open Source ที่มีความเสถียรและประสิทธิภาพสูง รองรับการทำงานกับข้อมูลที่มีความซับซ้อนได้ดี โครงงานนี้เลือกใช้ PostgreSQL ในการจัดเก็บข้อมูลสำคัญ ได้แก่ ข้อมูลสมาชิก (Users), ข้อมูลการจอง (Bookings), ข้อมูลโต๊ะ (Tables) และข้อมูลเกม (Games)

### 2.4.2 ภาษา SQL (Structured Query Language)

เป็นภาษามาตรฐานในการจัดการข้อมูลในฐานข้อมูลเชิงสัมพันธ์ ใช้สำหรับการสร้าง (Create), อ่าน (Read), แก้ไข (Update) และลบ (Delete) ข้อมูล ผ่าน Django ORM (Object-Relational Mapping) ซึ่งช่วยให้เขียนโค้ด Python เพื่อจัดการฐานข้อมูลได้โดยไม่ต้องเขียนคำสั่ง SQL โดยตรง

## 2.5 เทคโนโลยีส่วนติดต่อผู้ใช้งาน (Front-end Technologies)

### 2.5.1 HTML5 และ CSS3

* HTML5 (HyperText Markup Language): ใช้สำหรับสร้างโครงสร้างหลักของหน้าเว็บไซต์
* CSS3 (Cascading Style Sheets): ใช้สำหรับตกแต่งหน้าเว็บไซต์ให้มีความสวยงาม จัดวางองค์ประกอบ และกำหนดสีสัน

### 2.5.2 Tailwind CSS

Tailwind CSS เป็น Utility-first CSS Framework ที่ช่วยให้การออกแบบหน้าเว็บทำได้รวดเร็วและยืดหยุ่น โดยไม่ต้องเขียน CSS เองทั้งหมด โครงงานนี้ใช้ Tailwind CSS ในการจัด Layout และทำให้หน้าเว็บรองรับการแสดงผลบนหน้าจอขนาดต่าง ๆ (Responsive Design) เพื่อความสะดวกในการใช้งานผ่านมือถือ

### 2.5.3 JavaScript และ Chart.js

* JavaScript: ภาษาคอมพิวเตอร์ที่ทำงานฝั่งไคลเอนต์ (Client-side) ใช้เพื่อเพิ่มปฏิสัมพันธ์ (Interactivity) ให้กับหน้าเว็บ
* Chart.js: เป็นไลบรารี JavaScript ที่ใช้สำหรับสร้างกราฟและแผนภูมิข้อมูล ในโครงงานนี้ใช้ Chart.js ในส่วนของ Admin Dashboard เพื่อแสดงสถิติการจองรายวัน และสรุปสถานะการจองในรูปแบบกราฟเส้นและโดนัทกราฟ

## 2.6 เครื่องมือช่วยในการพัฒนา (Development Tools)

### 2.6.1 Docker

Docker เป็นแพลตฟอร์มสำหรับการจำลองสภาพแวดล้อม (Containerization) ช่วยให้นักพัฒนาสามารถรวมแอปพลิเคชันและไลบรารีที่จำเป็นไว้ใน "Container" ทำให้มั่นใจได้ว่าระบบจะสามารถทำงานได้เหมือนกันในทุกเครื่อง ไม่ว่าจะเป็นเครื่องของผู้พัฒนาหรือเครื่องเซิร์ฟเวอร์จริง

### 2.6.2 Git

Git เป็นระบบควบคุมเวอร์ชัน (Version Control System) ที่ใช้สำหรับจัดเก็บประวัติการแก้ไขซอร์สโค้ด ช่วยให้สามารถย้อนกลับไปดูเวอร์ชันก่อนหน้าได้ และอำนวยความสะดวกในการทำงานร่วมกัน

## 

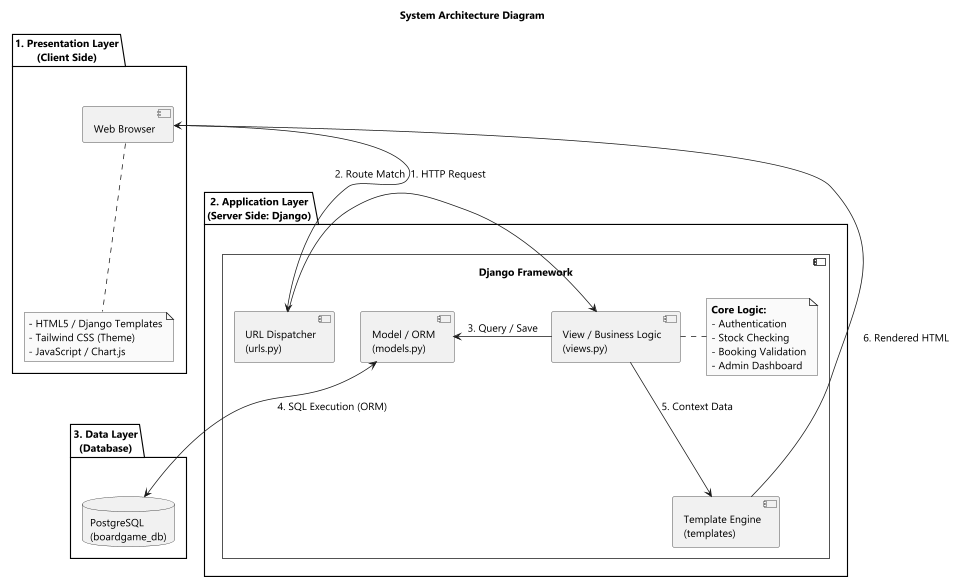
# บทที่ 3 การออกแบบระบบ

การพัฒนาระบบจองโต๊ะบอร์ดเกม ผู้จัดทำได้ดำเนินการวิเคราะห์และออกแบบระบบเพื่อให้ครอบคลุมความต้องการของผู้ใช้งานและผู้ดูแลระบบ โดยมีรายละเอียดดังนี้

## 3.1 System architecture

ระบบจองโต๊ะบอร์ดเกม (Board Game Cafe Booking System) ถูกออกแบบโดยใช้สถาปัตยกรรมแบบ Three-Tier Architecture บนพื้นฐานของ Django Framework (MVT Pattern) เพื่อให้การทำงานเป็นระบบ แยกส่วนการแสดงผลและการประมวลผลออกจากกันอย่างชัดเจน ดังนี้

1. Presentation Layer (Client Side):
   * ส่วนติดต่อผู้ใช้ (User Interface) ทำหน้าที่รับข้อมูลและแสดงผลผ่าน Web Browser
   * พัฒนาด้วย HTML5 ร่วมกับ Tailwind CSS เพื่อให้รองรับการแสดงผลแบบ Responsive Design สามารถใช้งานได้ดีทั้งบนคอมพิวเตอร์และโทรศัพท์มือถือ
2. Application Layer (Server Side):
   * ทำหน้าที่ประมวลผลทางธุรกิจ (Business Logic) ควบคุมการทำงานของระบบ
   * ใช้ Django Framework (Python) เป็นแกนหลัก โดยมีส่วนประกอบสำคัญคือ:
     + URL Dispatcher: จัดการเส้นทาง (Route) ของคำขอ
     + Views: ประมวลผลตรรกะการจอง ตรวจสอบสถานะว่างของโต๊ะ และตรวจสอบสิทธิ์ผู้ใช้งาน (Authentication)
     + Templates: สร้างหน้าเว็บแบบพลวัต (Dynamic HTML) ส่งกลับไปให้ผู้ใช้
3. Data Layer (Database):
   * ทำหน้าที่จัดเก็บข้อมูลทั้งหมดของระบบอย่างมีความสัมพันธ์ (Relational Database)
   * ใช้ PostgreSQL ในการจัดเก็บข้อมูลสมาชิก ข้อมูลโต๊ะ ข้อมูลเกม และรายการจองที่มีความถูกต้องแม่นยำ

รูปที่ 3.1 แผนภาพสถาปัตยกรรมระบบ

### คำอธิบายสถาปัตยกรรมของระบบ

1. ส่วนแสดงผล (Presentation Layer / Client Side) ส่วนที่ทำหน้าที่ติดต่อกับผู้ใช้งาน (User Interface) ทำงานผ่านเว็บเบราว์เซอร์

* Technologies: ใช้ภาษา HTML5 ในการจัดโครงสร้าง และใช้ Tailwind CSS ในการตกแต่งหน้าเว็บให้มีความสวยงามและรองรับการแสดงผลแบบ Responsive
* Interactive: มีการใช้ JavaScript และไลบรารี Chart.js สำหรับแสดงผลกราฟสถิติในหน้า Dashboard ของผู้ดูแลระบบ

2. ส่วนประมวลผล (Application Layer / Server Side) ส่วนหัวใจหลักของระบบที่พัฒนาด้วย Django Framework (Python) โดยใช้รูปแบบการออกแบบ MTV (Model-Template-View)

* URL Dispatcher (urls.py): ทำหน้าที่รับ HTTP Request จากผู้ใช้ และส่งต่อไปยังฟังก์ชัน (View) ที่ถูกต้อง
* Views (views.py): ทำหน้าที่ประมวลผลทางธุรกิจ (Business Logic) เช่น การตรวจสอบสิทธิ์การเข้าใช้งาน (Authentication), การตรวจสอบความว่างของโต๊ะ, การตัดสต็อกเกมตามช่วงเวลา (Timeslot), และการจัดการธุรกรรมฐานข้อมูล (Atomic Transaction)
* Templates: ทำหน้าที่สร้างหน้าเว็บ HTML แบบพลวัต (Dynamic) โดยนำข้อมูลที่ประมวลผลได้จาก View มาผสานกับไฟล์ HTML

3. ส่วนจัดการข้อมูล (Data Layer / Database) ส่วนที่ทำหน้าที่จัดเก็บและเรียกใช้ข้อมูล

* Database: เลือกใช้ PostgreSQL เป็นระบบจัดการฐานข้อมูลหลัก เนื่องจากมีความเสถียรและรองรับข้อมูลที่มีความซับซ้อนได้ดี
* Django ORM: การเข้าถึงข้อมูลจะทำผ่าน Django Object-Relational Mapping (ORM) ในไฟล์ models.py ซึ่งช่วยให้สามารถจัดการข้อมูลในรูปแบบ Python Object (เช่น Booking, Game, Timeslot) โดยไม่ต้องเขียนคำสั่ง SQL โดยตรง เพิ่มความปลอดภัยและความสะดวกในการพัฒนา

## 3.2 System requirement

จากการวิเคราะห์ระบบจองร้านบอร์ดเกม สามารถสรุปความต้องการด้านฟังก์ชัน (Functional Requirements) และความต้องการด้านอื่น ๆ (Non-Functional Requirements) ได้ดังนี้

### 3.2.1 ความต้องการด้านฟังก์ชัน (Functional Requirements)

FR-01 ระบบสมัครสมาชิกและยืนยันตัวตน (Authentication) ผู้ใช้งานทั่วไปสามารถสมัครสมาชิก (Register) และเข้าสู่ระบบ (Login) เพื่อยืนยันตัวตนก่อนใช้งานระบบจองได้ โดยรหัสผ่านจะถูกเข้ารหัส (Hashing) เพื่อความปลอดภัยตามมาตรฐาน Django

FR-02 ระบบจัดการข้อมูลส่วนตัว (Profile Management) สมาชิกสามารถดูและแก้ไขข้อมูลส่วนตัว เช่น ชื่อที่แสดง (Display Name) และอีเมลได้ เพื่อให้ข้อมูลเป็นปัจจุบันสำหรับการติดต่อ

FR-03 ระบบตรวจสอบสถานะโต๊ะเรียลไทม์ (Real-time Availability) ผู้ใช้สามารถเลือกวันที่ต้องการจอง และระบบจะแสดงสถานะของโต๊ะในแต่ละช่วงเวลา (Timeslot) ว่า "ว่าง" หรือ "จองแล้ว" ได้ทันที เพื่อช่วยในการตัดสินใจ

FR-04 ระบบจองโต๊ะและตัดสต็อกเกม (Booking & Stock Logic) ผู้ใช้สามารถจองโต๊ะและเลือกเกมที่ต้องการเล่นได้ โดยระบบจะทำการตรวจสอบจำนวนเกมคงเหลือในช่วงเวลานั้น ๆ และตัดสต็อกเกมชั่วคราวเมื่อการจองสำเร็จ

FR-05 ระบบแดชบอร์ดผู้ดูแลระบบ (Admin Dashboard) ผู้ดูแลระบบสามารถดูภาพรวมสถิติของร้านผ่านกราฟ (Chart.js) ได้ เช่น จำนวนการจองรายวัน และสัดส่วนสถานะการจอง (Pending/Confirmed/Cancelled)

FR-06 ระบบจัดการการจอง (Booking Management) ผู้ดูแลระบบสามารถดูรายการจองทั้งหมด แก้ไขสถานะการจอง (อนุมัติ/ยกเลิก) และดูรายละเอียดข้อมูลลูกค้า (ชื่อ/เบอร์โทร/อีเมล) ได้อย่างครบถ้วน

FR-07 ระบบจัดการทรัพยากรภายในร้าน (Store Management) ผู้ดูแลระบบสามารถเพิ่ม ลบ หรือแก้ไขข้อมูลโต๊ะ (Tables) และข้อมูลเกม (Games) รวมถึงการอัปเดตจำนวนสต็อกเกมและรูปภาพประกอบได้

FR-08 ระบบประวัติการจอง (Booking History) ระบบมีการบันทึกประวัติการจองทั้งหมดของลูกค้า เพื่อให้ลูกค้าสามารถเรียกดูประวัติการใช้งานย้อนหลัง และผู้ดูแลระบบสามารถตรวจสอบข้อมูลเพื่อการบริหารจัดการได้

### 3.2.2 ความต้องการด้านอื่น ๆ (Non-Functional Requirements)

NFR-01 ประสิทธิภาพการประมวลผล (Performance) ระบบต้องสามารถประมวลผลและแสดงผลหน้าเว็บไซต์ได้รวดเร็ว (Response Time ต่ำ) และรองรับการทำรายการจองพร้อมกันได้โดยข้อมูลไม่ผิดพลาด

NFR-02 ความถูกต้องของข้อมูล (Data Integrity) ระบบต้องมีการจัดการธุรกรรมฐานข้อมูล (Atomic Transaction) ในขั้นตอนการจอง เพื่อป้องกันปัญหาการจองซ้ำ (Double Booking) หรือการจองเกินจำนวนสต็อกที่มีอยู่

NFR-03 ความปลอดภัย (Security) แบบฟอร์มทุกจุดในระบบต้องมีการป้องกันการโจมตีแบบ CSRF (Cross-Site Request Forgery) และมีการกำหนดสิทธิ์การเข้าถึงข้อมูลระหว่างผู้ใช้ทั่วไปและผู้ดูแลระบบอย่างชัดเจน

NFR-04 การรองรับการแสดงผล (Responsive Design) ส่วนติดต่อผู้ใช้ (User Interface) ต้องสามารถปรับขนาดการแสดงผลให้เหมาะสมกับอุปกรณ์ต่าง ๆ ได้ ทั้งคอมพิวเตอร์ตั้งโต๊ะ แท็บเล็ต และสมาร์ตโฟน

## 3.3 การออกแบบหน้าจอ

การออกแบบส่วนติดต่อผู้ใช้ (UI) เน้นความเรียบง่าย ใช้งานสะดวก และรองรับการแสดงผลบนอุปกรณ์ที่หลากหลาย (Responsive) โดยแบ่งออกเป็นส่วนการใช้งานของลูกค้าทั่วไป และส่วนการจัดการของผู้ดูแลระบบ

#### **ส่วนที่ 1: หน้าจอสำหรับผู้ใช้งานทั่วไป (Public & Customer Pages)**

**1. หน้าแรก (Home Page)** หน้าเริ่มต้นของเว็บไซต์ แสดงข้อมูลร้านและปุ่มเข้าสู่ระบบ/สมัครสมาชิก หรือปุ่มจองทันที



ภาพประกอบที่ 3.2 แสดง Wireframe ของ หน้าแรก (Home Page)

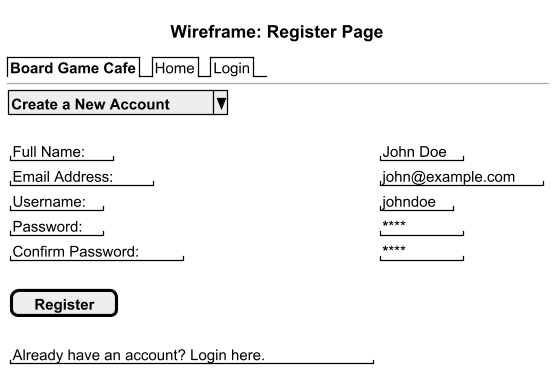
คำอธิบายภาพ: จากภาพที่ 3.2 แสดง Wireframe ของ หน้าแรก (Home Page) ซึ่งเป็นหน้าจอหลักของระบบจองร้านบอร์ดเกม โดยแบ่งออกเป็น 3 ส่วนหลัก ได้แก่ ส่วนหัว (Header) แสดงโลโก้ร้านและเมนูนำทาง (Navigation Bar) สำหรับเชื่อมโยงไปยังหน้าจองและหน้าประวัติ ส่วนเนื้อหาหลัก (Hero Section) ประกอบด้วยข้อความต้อนรับและคำเชิญชวน พร้อมปุ่ม "Book a Table Now" (Call-to-Action) ขนาดใหญ่เพื่อให้ผู้ใช้เข้าสู่กระบวนการจองได้ทันที และ ส่วนท้าย (Footer) แสดงลิขสิทธิ์และข้อมูลติดต่อของร้าน

**2. หน้าเข้าสู่ระบบ (Login Page)** แบบฟอร์มสำหรับสมาชิกเพื่อเข้าใช้งานระบบ



ภาพประกอบที่ 3.3 แสดง Wireframe ของ หน้าเข้าสู่ระบบ (Login)

**3. หน้าสมัครสมาชิก (Register Page)** แบบฟอร์มสำหรับผู้ใช้ใหม่เพื่อสร้างบัญชี

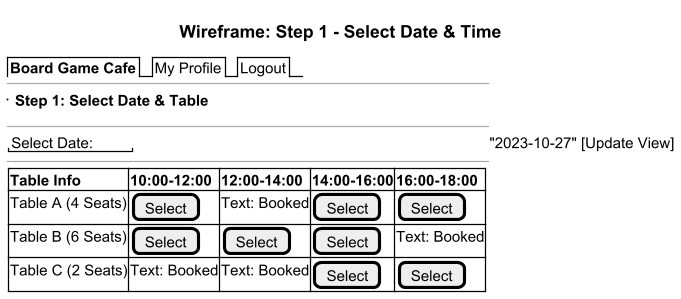


ภาพประกอบที่ 3.4 แสดง Wireframe ของหน้าสมัครสมาชิก (Register)

คำอธิบายภาพ: จากภาพที่ 3.3 และ 3.4 แสดง Wireframe ของ หน้าเข้าสู่ระบบ (Login) และหน้าสมัครสมาชิก (Register) ซึ่งเป็นด่านแรกสำหรับการยืนยันตัวตนก่อนเข้าใช้งานฟังก์ชันการจอง โดยหน้าเข้าสู่ระบบประกอบด้วยช่องกรอก Username และ Password พร้อมปุ่มยืนยัน ส่วนหน้าสมัครสมาชิกจะมีฟอร์มสำหรับกรอกข้อมูลที่จำเป็น ได้แก่ ชื่อ-นามสกุล, อีเมล, ชื่อผู้ใช้, และรหัสผ่าน พร้อมการยืนยันรหัสผ่านอีกครั้ง เพื่อสร้างบัญชีผู้ใช้ใหม่ในระบบ

#### **ส่วนที่ 2: ขั้นตอนการจองของลูกค้า (Customer Booking Flow)**

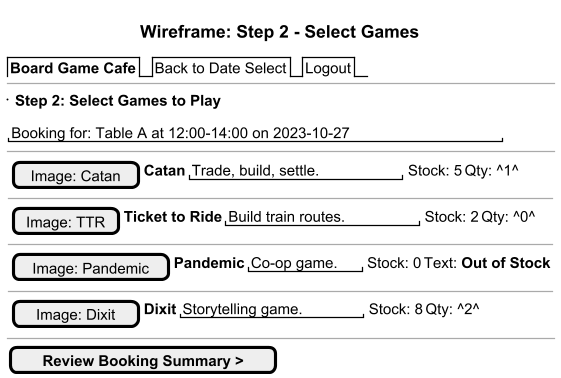
**4. หน้าเลือกวันที่และเวลา (Select Date & Timeslot)** ลูกค้าเลือกวันที่ที่ต้องการ และระบบจะแสดงตารางเวลาของแต่ละโต๊ะ พร้อมสถานะว่าง/ไม่ว่าง



ภาพประกอบที่ 3.5 แสดง Wireframe ของ หน้าเลือกวันที่และช่วงเวลา (Select Date & Timeslot)

คำอธิบายภาพ: จากภาพที่ 3.5 แสดง Wireframe ของ หน้าเลือกวันที่และช่วงเวลา (Select Date & Timeslot) ซึ่งเป็นขั้นตอนแรกของการจอง โดยส่วนบนสุดให้ผู้ใช้เลือกวันที่ที่ต้องการผ่านปุ่มปฏิทิน (Date Picker) ถัดลงมาเป็นส่วนแสดงรายการโต๊ะ (Table List) โดยแต่ละโต๊ะจะแสดงช่วงเวลา (Timeslot) ที่แบ่งเป็นรอบๆ ระบบจะแสดงสถานะของแต่ละช่วงเวลาอย่างชัดเจน หากว่างจะสามารถเลือกได้ แต่หากถูกจองแล้วจะแสดงสถานะ "Booked" และไม่สามารถเลือกได้ เพื่อป้องกันการจองซ้ำซ้อน

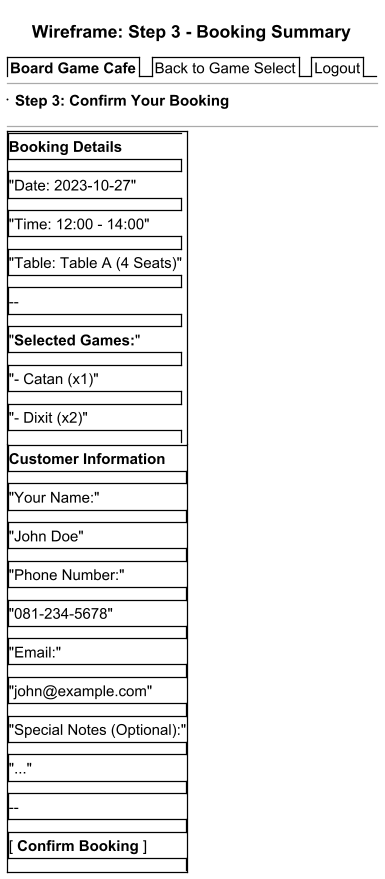
**5. หน้าเลือกเกม (Select Game)** แสดงรายการเกมที่มีในร้าน พร้อมรูปภาพ คำอธิบาย และจำนวนสต็อกคงเหลือในช่วงเวลานั้นๆ ให้ลูกค้าเลือกจำนวนที่ต้องการ



ภาพประกอบที่ 3.5 แสดง Wireframe ของ หน้าเลือกเกม (Select Game)

คำอธิบายภาพ: จากภาพที่ 3.5 แสดง Wireframe ของ หน้าเลือกเกม (Select Game) ซึ่งเป็นหน้าที่ช่วยให้ผู้ใช้สำรองบอร์ดเกมล่วงหน้า โดยแสดงรายการเกมในรูปแบบการ์ด (Card Layout) ประกอบด้วยรูปภาพหน้าปกเกม ชื่อเกม และคำอธิบายย่อ จุดสำคัญคือการแสดง จำนวนสต็อกคงเหลือ (Stock Availability) ณ ช่วงเวลาที่เลือกไว้แบบ Real-time และมีช่องสำหรับระบุจำนวนชุดที่ต้องการเล่น หากเกมใดสต็อกหมด ระบบจะแสดงสถานะ "Out of Stock" แจ้งเตือนทันที

**6. หน้าสรุปการจอง (Booking Summary)** แสดงรายละเอียดทั้งหมดที่ลูกค้าเลือกเพื่อให้ตรวจสอบความถูกต้อง และกรอกข้อมูลติดต่อเพิ่มเติมก่อนยืนยัน



ภาพประกอบที่ 3.6 แสดง Wireframe ของ หน้าสรุปการจอง (Booking Summary)

คำอธิบายภาพ: จากภาพที่ 3.6 แสดง Wireframe ของ หน้าสรุปการจอง (Booking Summary) ซึ่งเป็นหน้าสุดท้ายก่อนบันทึกข้อมูล โดยแบ่งหน้าจอเป็น 2 ส่วนหลัก ฝั่งซ้ายแสดงสรุปรายละเอียดการจองทั้งหมดที่ผู้ใช้เลือกมา (วันที่, เวลา, โต๊ะ, และรายชื่อเกม) เพื่อให้ตรวจสอบความถูกต้อง ฝั่งขวาเป็นแบบฟอร์มสำหรับกรอกข้อมูลผู้ติดต่อ (ชื่อ, เบอร์โทร, อีเมล) และช่องกรอกหมายเหตุเพิ่มเติม พร้อมปุ่ม "Confirm Booking" เพื่อยืนยันการทำรายการ

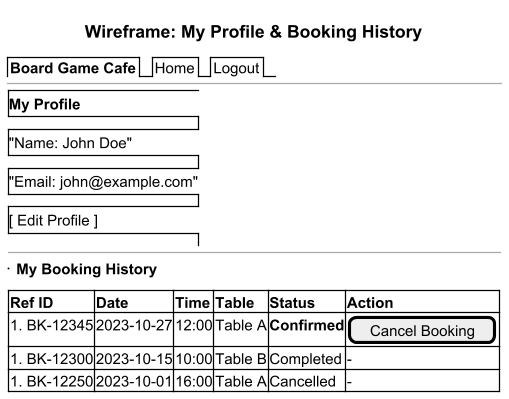
**7. หน้าจองสำเร็จ (Booking Success)** แสดงข้อความยืนยันเมื่อการจองเสร็จสมบูรณ์ พร้อมหมายเลขอ้างอิง



ภาพประกอบที่ 3.7 แสดง Wireframe ของ หน้าจองสำเร็จ (Booking Success)

#### **ส่วนที่ 3: หน้าข้อมูลส่วนตัวและประวัติ (Customer Profile & History)**

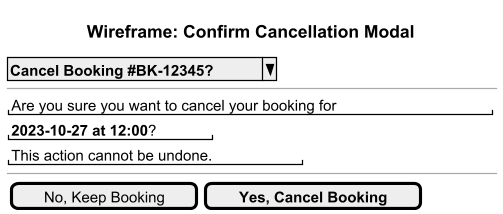
**8. หน้าโปรไฟล์และประวัติการจอง (Profile & Booking History)** แสดงข้อมูลส่วนตัวของสมาชิกและตารางประวัติการจองทั้งหมด พร้อมปุ่มสำหรับยกเลิกการจองที่ยังมาไม่ถึง



ภาพประกอบที่ 3.8 แสดง Wireframe ของ หน้าโปรไฟล์และประวัติการจอง (Profile & Booking History)

คำอธิบายภาพ: จากภาพที่ 3.8 แสดง Wireframe ของ หน้าโปรไฟล์และประวัติการจอง (Profile & Booking History) หน้าจอนี้แบ่งเป็นส่วนบนสำหรับแสดงและแก้ไขข้อมูลส่วนตัวสมาชิก และส่วนล่างเป็นตารางแสดงรายการจองย้อนหลังทั้งหมด (Booking History Table) โดยระบุเลขที่อ้างอิง วันที่ เวลา และสถานะการจอง (Pending, Confirmed, Cancelled) รวมถึงมีปุ่ม "Cancel Booking" ในกรณีที่ต้องการยกเลิกการจองที่ยังมาไม่ถึงตามเงื่อนไขที่กำหนด

**9. หน้าต่างยืนยันการยกเลิก (Confirm Cancel Modal)** หน้าต่างแจ้งเตือนเพื่อให้ลูกค้ายืนยันอีกครั้งก่อนทำการยกเลิกการจอง

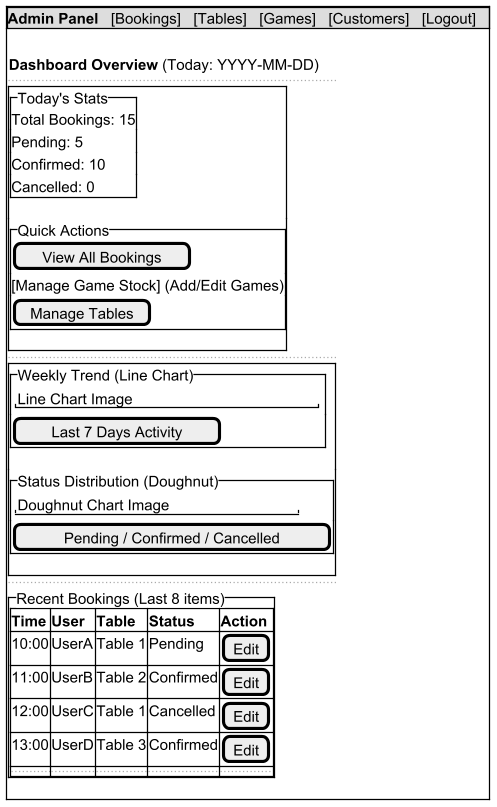


ภาพประกอบที่ 3.9 แสดง Wireframe ของ หน้าต่างยืนยันการยกเลิก (Confirm Cancel Modal)

คำอธิบายภาพ: จากภาพที่ 3.9 แสดง Wireframe ของ หน้าต่างยืนยันการยกเลิก (Confirm Cancel Modal) ซึ่งเป็นหน้าต่างป๊อปอัป (Modal Dialog) ที่จะปรากฏขึ้นเมื่อผู้ใช้กดปุ่ม "Cancel Booking" ในหน้าประวัติการจอง หน้าต่างนี้มีวัตถุประสงค์เพื่อป้องกันความผิดพลาดจากการกดปุ่มโดยไม่ตั้งใจ โดยจะแสดงรายละเอียดสำคัญของการจองที่จะยกเลิก ได้แก่ วันที่, เวลา และหมายเลขโต๊ะ พร้อมข้อความสอบถามความแน่ใจ "Are you sure you want to cancel?" และมีปุ่มทางเลือก 2 ปุ่ม คือ "Yes, Cancel Booking" เพื่อยืนยันการลบรายการ หรือ "No, Keep Booking" เพื่อปิดหน้าต่างและกลับสู่หน้าเดิม

#### **ส่วนที่ 4: หน้าจอสำหรับผู้ดูแลระบบ (Admin Pages)**

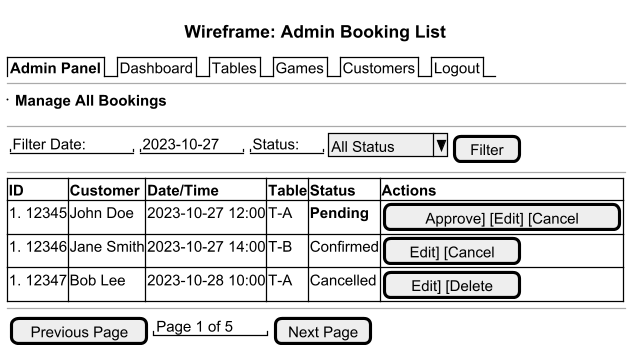
**10. แดชบอร์ดผู้ดูแลระบบ (Admin Dashboard)** หน้าแรกของแอดมิน แสดงภาพรวมสถิติ กราฟสรุปการจอง และลิงก์ไปยังเมนูจัดการต่างๆ



ภาพประกอบที่ 3.10 แสดง Wireframe ของ แดชบอร์ดผู้ดูแลระบบ (Admin Dashboard)

คำอธิบายภาพ: จากภาพที่ 3.10 แสดง Wireframe ของ แดชบอร์ดผู้ดูแลระบบ (Admin Dashboard) ซึ่งเป็นศูนย์กลางควบคุมข้อมูลของร้าน โดยส่วนบนแสดงการ์ดสถิติสรุปรายวัน (Statistic Cards) เช่น จำนวนการจองทั้งหมด, จำนวนที่รออนุมัติ และจำนวนที่ยกเลิก ส่วนกลางแสดงแผนภูมิ (Charts) ทั้งแบบกราฟเส้นแสดงแนวโน้มการจองรายสัปดาห์ และกราฟวงกลมแสดงสัดส่วนสถานะการจอง เพื่อช่วยให้ผู้ดูแลร้านวิเคราะห์ภาพรวมการใช้งานได้อย่างรวดเร็ว

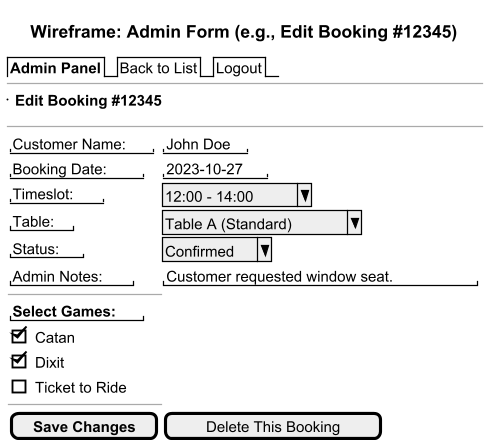
**11. หน้ารายการจองทั้งหมด (Admin Booking List)** แสดงตารางรายการจองทั้งหมดในระบบ สามารถกรองข้อมูลและกดเข้าไปจัดการ (แก้ไข/อนุมัติ/ยกเลิก) ได้



ภาพประกอบที่ 3.11 แสดง Wireframe ของ หน้ารายการจองทั้งหมด (Admin Booking List)

คำอธิบายภาพ: จากภาพที่ 3.11 แสดง Wireframe ของ หน้ารายการจองทั้งหมด (Admin Booking List) เป็นตารางข้อมูลหลักสำหรับผู้ดูแลระบบ ประกอบด้วยตัวกรอง (Filter) สำหรับค้นหาตามวันที่หรือสถานะ ส่วนเนื้อหาตารางแสดงรายละเอียดการจองครบถ้วน ทั้งชื่อลูกค้า วันเวลาที่จอง โต๊ะที่เลือก และสถานะปัจจุบัน ด้านขวาสุดของแต่ละแถวมีปุ่ม Action สำหรับกดเข้าไปแก้ไข (Edit), อนุมัติ (Approve) หรือยกเลิก (Cancel) รายการจองนั้นๆ ได้

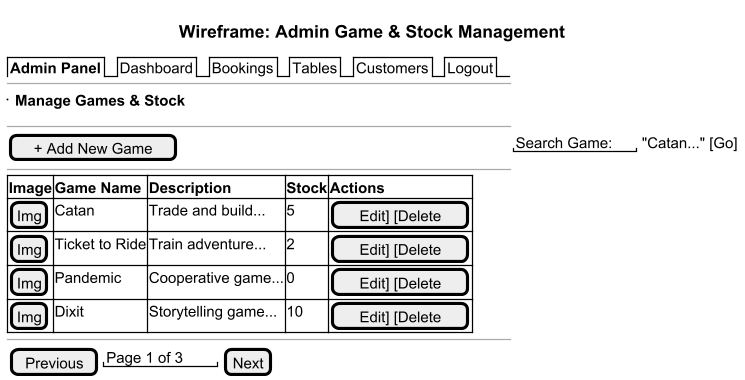
**12. หน้าฟอร์มจัดการข้อมูล (Admin Generic Form - ตัวอย่าง: แก้ไขการจอง)** แบบฟอร์มมาตรฐานสำหรับแอดมินใช้ในการ เพิ่ม หรือ แก้ไข ข้อมูลต่างๆ (เช่น การจอง, โต๊ะ, เกม)



ภาพประกอบที่ 3.12 แสดง Wireframe ของ หน้าฟอร์มแก้ไขข้อมูล (Admin Edit Form)

คำอธิบายภาพ: จากภาพที่ 3.12 แสดง Wireframe ของ หน้าฟอร์มแก้ไขข้อมูล (Admin Edit Form) ซึ่งเป็นหน้าจอมาตรฐานสำหรับผู้ดูแลระบบในการจัดการรายละเอียดข้อมูลต่างๆ (ในภาพตัวอย่างคือการแก้ไขข้อมูลการจอง) ประกอบด้วยฟิลด์ข้อมูลที่จำเป็น เช่น ชื่อลูกค้า, วันที่จอง, ช่วงเวลา, และสถานะการจอง (Status) ที่สามารถเลือกปรับเปลี่ยนได้จากเมนู Dropdown นอกจากนี้ยังมีส่วนสำหรับเลือกรายการเกม (Checkboxes) และช่องกรอกหมายเหตุ (Admin Notes) ด้านล่างสุดมีปุ่ม "Save Changes" เพื่อบันทึกข้อมูลลงฐานข้อมูล และปุ่ม "Delete" สีแดงสำหรับลบข้อมูลถาวร

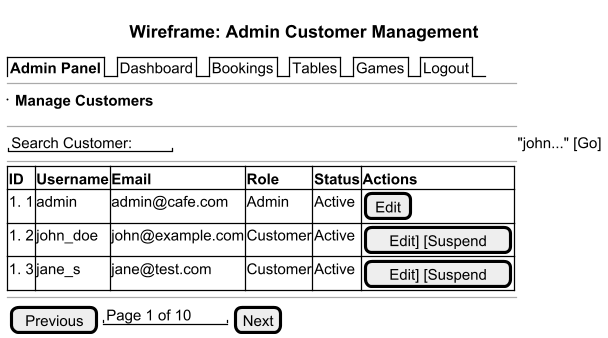
**13. หน้าจัดการสต็อกเกม (Admin Game Management)** แสดงรายการเกมทั้งหมด สามารถเพิ่มเกมใหม่ แก้ไขรายละเอียด หรือปรับปรุงจำนวนสต็อกได้



ภาพประกอบที่ 3.13 แสดง Wireframe ของ หน้าจัดการสต็อกเกม (Game Management)

คำอธิบายภาพ: จากภาพที่ 3.13 แสดง Wireframe ของ หน้าจัดการสต็อกเกม (Game Management) ใช้สำหรับบริหารจัดการทรัพยากรในร้าน แสดงรายการเกมทั้งหมดพร้อมรูปภาพตัวอย่างและจำนวนสต็อกปัจจุบัน ผู้ดูแลระบบสามารถกดปุ่ม "Add New Game" เพื่อเพิ่มเกมใหม่ หรือกดปุ่มเครื่องมือเพื่อแก้ไขรายละเอียดและปรับปรุงจำนวนสต็อกเกม ให้สอดคล้องกับความเป็นจริงในร้าน

**14. หน้าจัดการลูกค้า (Admin Customer List)** แสดงรายชื่อสมาชิกทั้งหมดในระบบ สามารถแก้ไขข้อมูล หรือระงับการใช้งานสมาชิกได้



ภาพประกอบที่ 3.14 แสดง Wireframe ของ หน้าจัดการข้อมูลลูกค้า (Customer Management)

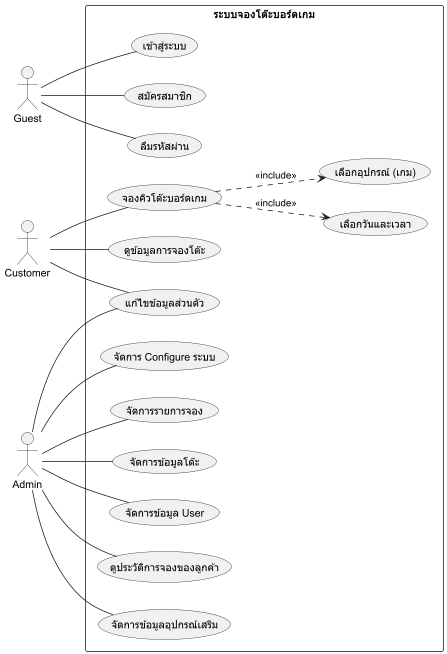
คำอธิบายภาพ: จากภาพที่ 3.14 แสดง Wireframe ของ หน้าจัดการข้อมูลลูกค้า (Customer Management) หน้าจอนี้แสดงรายชื่อสมาชิกทั้งหมดที่ลงทะเบียนในระบบในรูปแบบตาราง โดยแสดงข้อมูลสำคัญ ได้แก่ รหัสสมาชิก (ID), ชื่อผู้ใช้ (Username), อีเมล และสถานะบัญชี (Active/Suspended) ผู้ดูแลระบบสามารถใช้ช่องค้นหาด้านบนเพื่อหาชื่อลูกค้าที่ต้องการได้อย่างรวดเร็ว และในแต่ละแถวจะมีปุ่ม Action เพื่อเข้าไปแก้ไขข้อมูลส่วนตัว หรือกดปุ่ม "Suspend" เพื่อระงับการใช้งานชั่วคราวในกรณีที่สมาชิกทำผิดกฎของร้าน

## 3.4 Use case diagram

## Usecase Diagram เป็นแผนผังเพื่อแสดงฟังก์ชั่นการทำงานของระบบโดยรวมแสดงส่วนประกอบ ในระบบและกิจกรรม ที่เกิดขึ้นในระบบ สัญลักษณ์สำคัญต่างๆที่ใช้สำหรับการเขียน Usecase Diagram แสดงในตารางที่ 3.1

###### ตารางที่ 3.1 สัญลักษณ์ของ Use case Diagram

| **สัญลักษณ์** | **การใช้งาน** |
| --- | --- |
|  | Use case คือส่วนย่อยของระบบงาน แทนด้วยวงรีและชื่อของ Use case ภายในวงรี |
|  | Actor คือบุคคลหรือระบบงานอื่นที่ใช้งานระบบหรือได้รับประโยชน์จากระบบซึ่งอยู่ภายนอกระบบ แทนด้วยรูปคนและมีชื่อบทบาทการใช้งานระบบ |
|  | เส้นตรงที่แสดงถึงการใช้งาน Use case ของผู้กระทํา |
|  | กรอบสี่เหลี่ยม แสดงถึงขอบเขตของระบบโดย แสดงชื่อระบบภายในหรือด้านบนกรอกส่ีเหลี่ยม Use case อยู่ภายในกรอบสี่เหล่ียม และ actor อยู่ภายนอกกรอบสี่เหลี่ยม |
|  | ความสัมพันธ์แบบ «include» แสดงว่า Use case หนึ่งดําเนินการ ตามขั้นตอนของ Use case อื่น โดยแทนด้วยสัณลักษณ์ลูกศรเส้นประซึ่ง Use case ที่หางลูกศรเรียกใช้งาน Use case ที่หัวลูกศร ทุกครั้งที่มีการทํางาน |



ภาพประกอบที่ 3.15 Usecase Diagram ของระบบจองโต๊ะบอร์ดเกม

###### 

###### 

###### 

###### ตารางที่ 3.2 อธิบาย Use Case หน้าที่ของระบบในภาพประกอบที่ 3.

| Use Case ID | Use Case Name (ชื่อ) | Description (คำอธิบาย) |
| --- | --- | --- |
| UC-01 | เข้าสู่ระบบ | ผู้ใช้งานทำการยืนยันตัวตน (Username/Password) เพื่อเข้าใช้งานระบบตามสิทธิ์ (Customer/Admin) |
| UC-02 | สมัครสมาชิก | ผู้ใช้งานทั่วไป (Guest) ลงทะเบียนสร้างบัญชีใหม่เพื่อเริ่มใช้งานระบบจอง |
| UC-03 | ลืมรหัสผ่าน | ผู้ใช้งานขอรีเซ็ตรหัสผ่านใหม่ผ่านทางอีเมล กรณีจำรหัสผ่านเดิมไม่ได้ |
| UC-04 | แก้ไขข้อมูลส่วนตัว | สมาชิกสามารถแก้ไขข้อมูลส่วนตัว เช่น ชื่อ-สกุล เบอร์โทรศัพท์ หรือเปลี่ยนรหัสผ่านได้ |
| UC-05 | จองคิวโต๊ะบอร์ดเกม | ลูกค้าทำรายการจองโต๊ะ โดยระบุวันเวลา โต๊ะที่ต้องการ และเกมที่จะเล่น |
| UC-06 | จัดการ Configure ระบบ | ผู้ดูแลระบบตั้งค่าพื้นฐาน เช่น เวลาเปิด-ปิดร้าน หรือเงื่อนไขการจอง |
| UC-07 | จัดการรายการจอง | ผู้ดูแลระบบตรวจสอบ อนุมัติ หรือยกเลิกการจองของลูกค้า |
| UC-08 | จัดการข้อมูลโต๊ะ | ผู้ดูแลระบบ เพิ่ม/ลบ/แก้ไข ข้อมูลโต๊ะและสถานะความพร้อมใช้งาน |
| UC-09 | จัดการข้อมูล User | ผู้ดูแลระบบจัดการบัญชีผู้ใช้ ดูคะแนนความประพฤติ หรือระงับการใช้งาน |
| UC-10 | ดูประวัติการจองลูกค้า | ผู้ดูแลระบบเรียกดูประวัติการจองย้อนหลังของลูกค้าเพื่อตรวจสอบข้อมูล |
| UC-11 | จัดการข้อมูลอุปกรณ์ | ผู้ดูแลระบบ เพิ่ม/ลบ/แก้ไข รายชื่อบอร์ดเกมและจำนวนสต็อกในระบบ |
| UC-12 | ดูข้อมูลการจองโต๊ะ | ลูกค้าตรวจสอบสถานะการจองของตนเอง หรือดูประวัติการจองที่ผ่านมา |

###### ตารางที่ 3.3 Use Case สมัครสมาชิก

| Use Case Title : สมัครสมาชิก (Register) | Use Case ID : UC-02 |
| --- | --- |
| Primary Actor **:** Guest (ผู้เยี่ยมชม) | |
| Pre-conditions : ผู้ใช้งานยังไม่เคยมีบัญชีในระบบมาก่อน | |
| Main Flow : 1. ผู้ใช้งานเข้าสู่หน้าสมัครสมาชิก  2. กรอกข้อมูล: Username, ชื่อ (Name), Email, รหัสผ่าน, และยืนยันรหัสผ่าน  3. กดปุ่ม "Register"  4. ระบบตรวจสอบความถูกต้อง (เช่น รหัสผ่านตรงกันหรือไม่)  5. ระบบบันทึกข้อมูลสมาชิกใหม่และเปลี่ยนสถานะเป็น Logged In หรือพาไปหน้า Login | |
| Exceptional Flow 1 : หาก Username ซ้ำ ระบบแจ้งเตือนว่ามีผู้ใช้นี้แล้ว | |
| Exceptional Flow 2 : หากรหัสผ่านไม่ตรงกัน: ระบบแจ้งเตือนให้กรอกใหม่ | |

###### ตารางที่ 3.4 Use Case ลงชื่อเข้าสู่ระบบ

| Use Case Title : เข้าสู่ระบบ (Login) | Use Case ID : UC-01 |
| --- | --- |
| Primary Actor **:** Guest (ผู้เยี่ยมชม) | |
| Pre-conditions : ผู้ใช้งานต้องสมัครสมาชิกเรียบร้อยแล้ว (ผ่าน UC-02) | |
| Main Flow : 1. ผู้ใช้งานเลือกเมนู "เข้าสู่ระบบ"  2. ผู้ใช้งานกรอกชื่อผู้ใช้และรหัสผ่าน  3. กดปุ่ม "เข้าสู่ระบบ"  4. ระบบตรวจสอบข้อมูลเทียบกับฐานข้อมูล  5.หากถูกต้อง ระบบนำผู้ใช้งานไปยังหน้าแรกในสถานะสมาชิก | |
| Exceptional Flow 1 : หากชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง ระบบจะแจ้งเตือนและให้กรอกใหม่ | |

###### ตารางที่ 3.5 Use Case จองคิวโต๊ะบอร์ดเกม

| Use Case Title : จองคิวโต๊ะบอร์ดเกม (Book Table) | Use Case ID : UC-05 |
| --- | --- |
| Primary Actor **:** Customer (ลูกค้า) | |
| Pre-conditions : ผู้ใช้งานต้องเข้าสู่ระบบ (Login) แล้ว | |
| Main Flow : 1. ลูกค้าเลือกเมนู "จองโต๊ะ"  2. ลูกค้าเลือกวันที่และช่วงเวลาที่ต้องการ  3. ระบบแสดงรายการโต๊ะที่ว่างในช่วงเวลานั้น  4. ลูกค้าเลือกโต๊ะที่ต้องการ  5. ลูกค้าเลือกเกมที่ต้องการเล่น  6. ลูกค้ากดยืนยันการจอง  7. กดปุ่มยืนยัน ระบบบันทึกข้อมูลและแสดงหน้าจองสำเร็จ | |
| Exceptional Flow 1 : หากช่วงเวลาถูกจองตัดหน้า: ระบบแจ้งเตือนว่าเวลาไม่ว่าง | |
| Exceptional Flow 2 : หากเลือกเกมเกินจำนวนสต็อก: ระบบแจ้งเตือนให้ลดจำนวน | |

###### ตารางที่ 3.6 Use Case จัดการรายการจอง

| Use Case Title : จัดการรายการจอง (Manage Booking) | Use Case ID : UC-07 |
| --- | --- |
| Primary Actor **:** Admin (ผู้ดูแลระบบ) | |
| Pre-conditions : ผู้ดูแลระบบต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ผู้ดูแลระบบเข้าสู่หน้า "รายการจอง"  2. ระบบแสดงรายการจองทั้งหมดและสถานะ (รออนุมัติ, อนุมัติแล้ว, ยกเลิก)  3. ผู้ดูแลระบบเลือกรายการที่ต้องการจัดการ  4. ผู้ดูแลระบบกดเปลี่ยนสถานะ (เช่น กด "อนุมัติ" หรือ "ยกเลิก")  5. ระบบบันทึกสถานะใหม่และแจ้งเตือนผลไปยังลูกค้า | |
| Exceptional Flow 1 : หากรายการจองถูกยกเลิกไปแล้ว จะไม่สามารถกลับมาอนุมัติซ้ำได้ | |
| Exceptional Flow 2 : ข้อมูลการจองผิดพลาดหรือไม่พบข้อมูลในฐานข้อมูล | |

###### ตารางที่ 3.7 Use Case ลืมรหัสผ่าน

| Use Case Title : ลืมรหัสผ่าน (Forgot Password) | Use Case ID : UC-03 |
| --- | --- |
| Primary Actor **:** Guest (ผู้เยี่ยมชม) | |
| Pre-conditions : ผู้ใช้งานต้องเคยสมัครสมาชิกและมีอีเมลอยู่ในระบบ | |
| Main Flow : 1. ผู้ใช้งานเลือกเมนู "ลืมรหัสผ่าน" จากหน้าเข้าสู่ระบบ  2. ผู้ใช้งานกรอกอีเมลที่เคยลงทะเบียนไว้  3. ระบบตรวจสอบอีเมลและส่งลิงก์สำหรับตั้งรหัสผ่านใหม่ไปทางอีเมล  4. ผู้ใช้งานคลิกลิงก์จากอีเมลและเข้าสู่หน้าตั้งรหัสผ่านใหม่  5. ผู้ใช้งานกรอกรหัสผ่านใหม่และกดยืนยัน  6. ระบบบันทึกรหัสผ่านใหม่และแจ้งเตือนว่าสำเร็จ | |
| Exceptional Flow 1 : หากไม่พบอีเมลในระบบ ระบบจะแจ้งเตือนว่าไม่พบข้อมูลผู้ใช้ | |
| Exceptional Flow 2 : หากลิงก์รีเซ็ตรหัสผ่านหมดอายุ ผู้ใช้ต้องทำรายการใหม่อีกครั้ง | |

###### ตารางที่ 3.8 Use Case แก้ไขข้อมูลส่วนตัว

| Use Case Title : แก้ไขข้อมูลส่วนตัว (Edit Profile) | Use Case ID : UC-04 |
| --- | --- |
| Primary Actor **:** Authenticated User (ลูกค้า / ผู้ดูแลระบบ) | |
| Pre-conditions : ผู้ใช้งานต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ผู้ใช้งานเลือกเมนู "ข้อมูลส่วนตัว"  2. ระบบแสดงข้อมูลปัจจุบันของผู้ใช้  3. ผู้ใช้งานกด "แก้ไข" และทำการแก้ไขข้อมูลที่ต้องการ (เช่น เบอร์โทร, รูปโปรไฟล์)  4. ผู้ใช้งานกดยืนยันการบันทึก  5. ระบบบันทึกข้อมูลและแสดงผลข้อมูลที่อัปเดตแล้ว | |
| Exceptional Flow : - | |

###### ตารางที่ 3.9 Use Case ดูข้อมูลการจองโต๊ะ

| Use Case Title : ดูข้อมูลการจองโต๊ะ (View My Bookings) | Use Case ID : UC-12 |
| --- | --- |
| Primary Actor **:** Customer (ลูกค้า) | |
| Pre-conditions : ผู้ใช้งานต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ลูกค้าเลือกเมนู "ประวัติการจอง"  2. ระบบแสดงรายการจองทั้งหมดของลูกค้า (ทั้งที่รออนุมัติ, อนุมัติแล้ว, และประวัติเก่า)  3. ลูกค้ากดดูรายละเอียดของแต่ละรายการเพื่อดูข้อมูลโต๊ะ รอบเวลา และเกมที่จองไว้ | |
| Exceptional Flow 1 : หากไม่พบประวัติการจอง ระบบจะแสดงข้อความว่าไม่มีรายการจอง | |

###### ตารางที่ 3.10 Use Case จัดการ Configure ระบบ

| Use Case Title : จัดการ Configure ระบบ (System Configuration) | Use Case ID : UC-06 |
| --- | --- |
| Primary Actor **:** Admin (ผู้ดูแลระบบ) | |
| Pre-conditions : ผู้ดูแลระบบต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ผู้ดูแลระบบเลือกเมนู "ตั้งค่าระบบ"  2. ระบบแสดงค่าปัจจุบัน (เช่น เวลาเปิด-ปิดร้าน, ระยะเวลาจองขั้นต่ำ)  3. ผู้ดูแลระบบแก้ไขค่าที่ต้องการ  4. ผู้ดูแลระบบกดยืนยันการบันทึก  5. ระบบอัปเดตค่า Config เพื่อนำไปใช้ในการคำนวณรอบเวลาจอง | |
| Exceptional Flow 1 : หากกรอกค่าผิดพลาด (เช่น เวลาปิดร้านเร็วกว่าเวลาเปิด) ระบบจะแจ้งเตือนและไม่บันทึกข้อมูล | |

###### ตารางที่ 3.11 Use Case จัดการข้อมูลโต๊ะ

| Use Case Title : จัดการข้อมูลโต๊ะ (Manage Tables) | Use Case ID : UC-08 |
| --- | --- |
| Primary Actor **:** Admin (ผู้ดูแลระบบ) | |
| Pre-conditions : ผู้ดูแลระบบต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ผู้ดูแลระบบเลือกเมนู "จัดการโต๊ะ"  2. ระบบแสดงรายการโต๊ะทั้งหมดและสถานะ  3. ผู้ดูแลระบบเลือก "เพิ่มโต๊ะใหม่" หรือ "แก้ไข" โต๊ะเดิม  4. ระบุข้อมูลโต๊ะ (หมายเลขโต๊ะ, จำนวนที่นั่ง, สถานะใช้งาน)  5. กดยืนยันเพื่อบันทึก  6. ระบบอัปเดตข้อมูลโต๊ะในฐานข้อมูล | |
| Exceptional Flow 1 : หากระบุหมายเลขโต๊ะซ้ำกับที่มีอยู่ ระบบจะแจ้งเตือน | |
| Exceptional Flow 2 : หากลบโต๊ะที่มีการจองค้างอยู่ ระบบจะไม่อนุญาตให้ลบหรือแจ้งเตือนยืนยัน | |

###### ตารางที่ 3.12 Use Case จัดการข้อมูลโต๊ะ

| Use Case Title : จัดการข้อมูล User (Manage Users) | Use Case ID : UC-09 |
| --- | --- |
| Primary Actor **:** Admin (ผู้ดูแลระบบ) | |
| Pre-conditions : ผู้ดูแลระบบต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ผู้ดูแลระบบเลือกเมนู "จัดการผู้ใช้งาน"  2. ระบบแสดงรายชื่อสมาชิกทั้งหมด  3. ผู้ดูแลระบบสามารถค้นหาชื่อสมาชิกและกดดูรายละเอียด  4. ผู้ดูแลระบบสามารถแก้ไขสิทธิ์ หรือระงับการใช้งาน (Ban) สมาชิกที่มีพฤติกรรมไม่เหมาะสม  5. ระบบบันทึกสถานะล่าสุดของสมาชิกนั้น | |
| Exceptional Flow 1 : ไม่สามารถระงับสิทธิ์ของผู้ดูแลระบบด้วยกันเองได้ | |

###### ตารางที่ 3.13 Use Case ดูประวัติการจองของลูกค้า

| Use Case Title : ดูประวัติการจองของลูกค้า (View Customer History) | Use Case ID : UC-10 |
| --- | --- |
| Primary Actor **:** Admin (ผู้ดูแลระบบ) | |
| Pre-conditions : ผู้ดูแลระบบต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ผู้ดูแลระบบเลือกผู้ใช้งานจากหน้ารายชื่อ (ต่อเนื่องจาก UC-09) หรือค้นหาจากรหัสการจอง  2. ระบบแสดงประวัติการจองทั้งหมดของลูกค้ารายนั้น  3. ผู้ดูแลระบบตรวจสอบความถี่ในการจอง หรือประวัติการผิดนัด (No-show) เพื่อประกอบการ  ตัดสินใจให้คะแนนความประพฤติ | |
| Exceptional Flow 1 : - | |

###### ตารางที่ 3.14 Use Case จัดการข้อมูลอุปกรณ์เสริม

| Use Case Title : จัดการข้อมูลอุปกรณ์เสริม/เกม (Manage Games) | Use Case ID : UC-11 |
| --- | --- |
| Primary Actor **:** Admin (ผู้ดูแลระบบ) | |
| Pre-conditions : ผู้ดูแลระบบต้องเข้าสู่ระบบแล้ว | |
| Main Flow : 1. ผู้ดูแลระบบเลือกเมนู "จัดการเกม/อุปกรณ์"  2. ระบบแสดงรายการเกมที่มีในร้าน  3. ผู้ดูแลระบบเลือกเพิ่มเกมใหม่ หรือแก้ไขจำนวนสต็อกเกมเดิม  4. ระบุรายละเอียด (ชื่อเกม, ประเภท, จำนวน, รูปภาพ)  5. กดยืนยันการบันทึก  6. ระบบอัปเดตข้อมูลสินค้าในคลัง | |
| Exceptional Flow 1 : หากลบเกมที่มีลูกค้าจองล่วงหน้าไว้ ระบบแจ้งเตือนผลกระทบต่อการจองนั้น | |

## 

## 

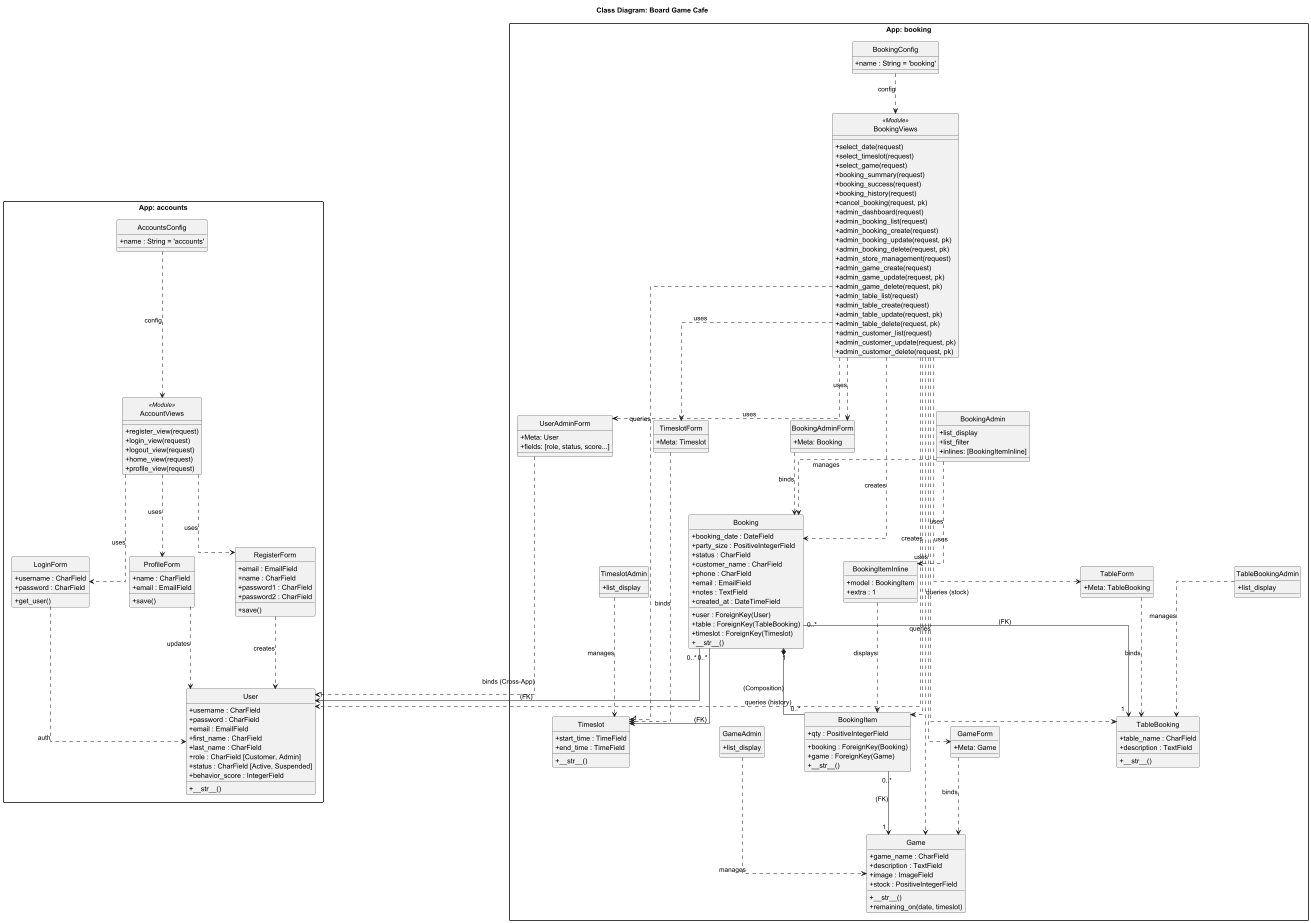
## 

## 3.5 Class diagram

แผนภาพคลาสแสดงโครงสร้างของคลาสในระบบที่พัฒนาด้วย Django Framework โดยแบ่งตาม Application ได้ดังนี้

ตารางที่ 3.15 สัญลักษณ์ที่ใช้ในแผนภาพคลาส

| สัญลักษณ์ (Symbol) | ชื่อเรียก (Name) | ความหมาย (Description) |
| --- | --- | --- |
|  | Class | กล่องแสดงคลาส ประกอบด้วย 3 ส่วน:   1. ชื่อคลาส (Class Name) 2. แอตทริบิวต์ (Attributes) 3. เมธอด (Methods) |
|  | Composition | องค์ประกอบ (Composition): หัวเพชรทึบ  ใช้แสดงความสัมพันธ์ที่ "ขาดจากกันไม่ได้" (เช่น Tournament หาย Match ก็ต้องหายด้วย) |
|  | Association | ความสัมพันธ์ทั่วไป Association): เส้นทึบ  ใช้แสดงการเชื่อมโยงระหว่างคลาสที่ทำงานร่วมกัน (เช่น User เชื่อมกับ Profile) |
|  | Dependency | การพึ่งพา (Dependency): เส้นประหัวลูกศร  ใช้แสดงว่าคลาสหนึ่งเรียกใช้อีกคลาสหนึ่งชั่วคราว (เช่น View เรียกใช้ Form) |



###### 

###### ภาพประกอบที่ 3.16 Class Diagram ของระบบจองร้านบอร์ดเกม

ตารางที่ 3.16 รายละเอียดคลาสในส่วนจัดการผู้ใช้งาน (Accounts Module)

| **ชื่อคลาส (Class Name)** | **ประเภท (Type)** | **รายละเอียด (Description)** |
| --- | --- | --- |
| User | Model | คลาสสำหรับเก็บข้อมูลผู้ใช้งานในระบบ ซึ่งสืบทอดมาจาก AbstractUser ของ Django |
| Attributes |  |  |
| - username | CharField | ชื่อผู้เข้าใช้ระบบ |
| - password | CharField | รหัสผ่าน (เข้ารหัส) |
| - email | EmailField | ที่อยู่อีเมล |
| - role | CharField | สิทธิ์การใช้งาน (Customer, Admin) |
| - status | CharField | สถานะบัญชี (Active, Suspended) |
| - behavior\_score | IntegerField | คะแนนพฤติกรรมลูกค้า |
| Methods |  |  |
| - \_\_str\_\_() | String | คืนค่าชื่อผู้ใช้ (username) เมื่อถูกเรียกแสดงผล |
| RegisterForm | Form | คลาสสำหรับจัดการฟอร์มสมัครสมาชิก |
| LoginForm | Form | คลาสสำหรับจัดการฟอร์มเข้าสู่ระบบ |
| ProfileForm | Form | คลาสสำหรับจัดการฟอร์มแก้ไขข้อมูลส่วนตัว |

ตารางที่ 3.17 รายละเอียดคลาสในส่วนการจอง (Booking Module)

| **ชื่อคลาส (Class Name)** | **ประเภท (Type)** | **รายละเอียด (Description)** |
| --- | --- | --- |
| Booking | Model | คลาสหลักสำหรับเก็บข้อมูลการจอง เชื่อมโยงระหว่าง ผู้ใช้, โต๊ะ และช่วงเวลา |
| Attributes |  |  |
| - user | ForeignKey | อ้างอิงไปยังคลาส User (ผู้จอง) |
| - table | ForeignKey | อ้างอิงไปยังคลาส TableBooking (โต๊ะที่จอง) |
| - timeslot | ForeignKey | อ้างอิงไปยังคลาส Timeslot (เวลาที่จอง) |
| - status | CharField | สถานะการจอง (Pending, Confirmed, Cancelled) |
| - booking\_date | DateField | วันที่เข้าใช้บริการ |
| - party\_size | Integer | จำนวนคนที่มาใช้บริการ |
| Methods |  |  |
| - \_\_str\_\_() | String | คืนค่าข้อมูลสรุปการจอง (User - Table - Date) |
| BookingItem | Model | คลาสเก็บรายละเอียดเกมที่ถูกเลือกในการจองแต่ละครั้ง |
| Attributes |  |  |
| - booking | ForeignKey | อ้างอิงไปยังคลาส Booking |
| - game | ForeignKey | อ้างอิงไปยังคลาส Game |
| - qty | Integer | จำนวนชุดของเกมที่จอง |
| Game | Model | คลาสเก็บข้อมูลบอร์ดเกมและสต็อก |
| Attributes |  |  |
| - game\_name | CharField | ชื่อเกม |
| - stock | Integer | จำนวนสต็อกทั้งหมดที่มีในร้าน |
| Methods |  |  |
| - remaining\_on() | Integer | คำนวณจำนวนสต็อกคงเหลือ ณ วันและเวลาที่กำหนด |
| TableBooking | Model | คลาสเก็บข้อมูลโต๊ะภายในร้าน |
| Timeslot | Model | คลาสเก็บข้อมูลรอบเวลาที่เปิดให้บริการ (เช่น 10:00-12:00) |

ตารางที่ 3.18 รายละเอียดคลาสในส่วนควบคุมการทำงาน (Views & Controllers)

| **ชื่อคลาส (Class Name)** | **ประเภท (Type)** | **รายละเอียด (Description)** |
| --- | --- | --- |
| AccountViews | Module | กลุ่มฟังก์ชันสำหรับจัดการบัญชีผู้ใช้ |
| - register\_view | Function | ประมวลผลการสมัครสมาชิกและบันทึกข้อมูล User |
| - login\_view | Function | ตรวจสอบข้อมูลล็อกอินและสร้าง Session |
| - profile\_view | Function | แสดงและบันทึกการแก้ไขข้อมูลส่วนตัว |
| BookingViews | Module | กลุ่มฟังก์ชันสำหรับจัดการระบบจอง |
| - select\_date | Function | รับค่าวันที่และส่งต่อไปยังขั้นตอนเลือกเวลา |
| - select\_timeslot | Function | ดึงข้อมูล Table และ Timeslot เพื่อแสดงสถานะว่าง/ไม่ว่าง |
| - select\_game | Function | แสดงรายการเกมพร้อมจำนวนคงเหลือ (Stock) ณ เวลานั้น |
| - booking\_success | Function | บันทึกข้อมูล Booking และ BookingItem ลงฐานข้อมูล (Transaction) |
| - admin\_dashboard | Function | แสดงภาพรวมสถิติและกราฟสำหรับผู้ดูแลระบบ |
| - admin\_booking\_\* | Function | กลุ่มฟังก์ชันสำหรับ Admin (ดู, สร้าง, แก้ไข, ลบ การจอง) |

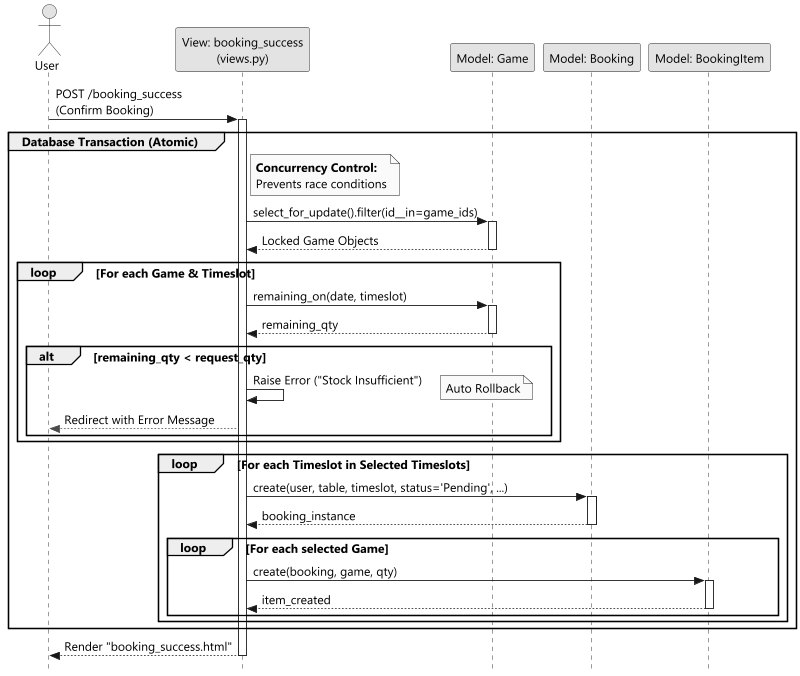
## 3.6 Sequence diagram

## สัญลักษณ์ในแผนภาพลำดับ (Sequence Diagram)

ตารางที่ 3.19 สัญลักษณ์ที่ที่ใช้ใน Sequence Diagram

| **สัญลักษณ์ (Symbol)** | **ชื่อเรียก (Name)** | **ความหมาย (Description)** |
| --- | --- | --- |
|  | Actor | **ผู้กระทำ** ผู้ใช้งานหรือระบบภายนอกที่มีปฏิสัมพันธ์กับระบบ (เช่น ผู้เยี่ยมชม, สมาชิก, ผู้ดูแลระบบ) |
| │  (เส้นตรงแนวตั้ง) | Lifeline | **เส้นชีวิต** เส้นประแนวตั้งแสดงถึงช่วงเวลาที่ออบเจกต์หรือคลาสมีตัวตนอยู่ในกระบวนการ |
| **(Activation bar)** | Activation Bar | **แถบการทำงาน** แท่งสี่เหลี่ยมบน Lifeline แสดงช่วงเวลาที่ออบเจกต์กำลังประมวลผล (Active) |
| **──➤**  (หัวลูกศรทึบ) | Synchronous Message | **ข้อความแบบซิงโครนัส** การส่งคำสั่งแบบรอการตอบกลับ (เส้นทึบหัวลูกศรทึบ) เช่น การเรียกใช้ View หรือ API |
| **<--**  (เส้นประ) | Return Message | **ข้อความตอบกลับ** การส่งผลลัพธ์หรือข้อมูลกลับไปยังผู้เรียก (เส้นประหัวลูกศรเปิด) เช่น การส่ง Template กลับไปแสดงผล |
| **(Alt / else)** | Alternative Fragment | **ทางเลือก (Alternative)** กรอบแสดงเงื่อนไขการทำงาน (If-Else) เช่น การตรวจสอบว่า Login สำเร็จหรือไม่ |
| **(Loop)** | Loop Fragment | **การวนซ้ำ (Loop)** กรอบแสดงการทำงานซ้ำ เช่น การวนลูปดึงข้อมูลผู้เข้าแข่งขันมาแสดงในหน้าเว็บ |

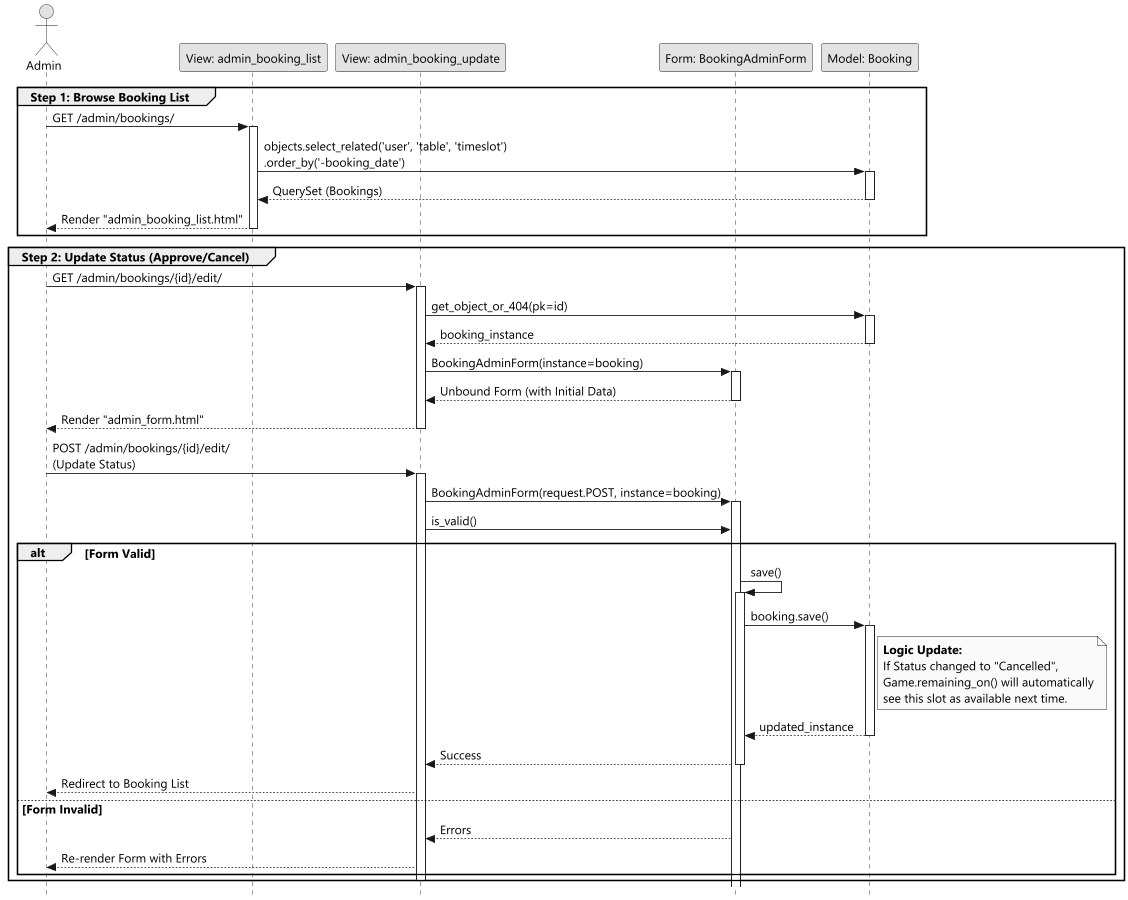
### 3.6.1 Sequence Diagram ของการจอง

ภาพประกอบที่ 3.17 Sequence Diagram ของการจอง

คำอธิบาย: แผนภาพนี้แสดงลำดับการทำงานของกระบวนการจองที่มีการจัดการความถูกต้องของข้อมูล (Data Integrity) สูงสุด โดยมีขั้นตอนดังนี้:

1. การร้องขอการจอง: ผู้ใช้งาน (Customer) ทำการเลือกวัน เวลา โต๊ะ และรายการเกมที่ต้องการเล่น จากนั้นกดยืนยันการจองผ่านหน้าเว็บไซต์ ระบบจะส่งข้อมูลทั้งหมด (FormData) ไปยังเซิร์ฟเวอร์
2. เริ่มทำธุรกรรม (Start Transaction): ระบบจะเริ่มกระบวนการ Atomic Transaction เพื่อให้มั่นใจว่าทุกขั้นตอนต้องสำเร็จพร้อมกัน หากมีขั้นตอนใดผิดพลาด ระบบจะยกเลิกการทำงานทั้งหมด (Rollback) ทันที
3. การล็อคข้อมูล (Row Locking): ระบบทำการดึงข้อมูลเกมที่ถูกเลือกและทำการล็อคแถวข้อมูล (Row Lock) ด้วยคำสั่ง select\_for\_update เพื่อป้องกันไม่ให้ผู้ใช้งานคนอื่นทำการจองเกมเดียวกันในช่วงเวลาเสี้ยววินาทีเดียวกัน (Prevent Race Condition)
4. ตรวจสอบสต็อก (Stock Validation): ระบบวนลูปตรวจสอบจำนวนสต็อกคงเหลือ (remaining\_on) ของแต่ละเกมในช่วงเวลาที่เลือก
   * กรณีสินค้าไม่พอ: ระบบจะยกเลิก Transaction ทั้งหมดและส่งข้อความแจ้งเตือน "สินค้าหมด" (Out of Stock) กลับไปยังผู้ใช้งานทันที
   * กรณีสินค้าพอ: ระบบจะดำเนินการสร้างข้อมูลลงในตาราง Booking และ BookingItem
5. ยืนยันการจอง: เมื่อบันทึกข้อมูลสำเร็จ ระบบจะทำการ Commit Transaction และส่งผลลัพธ์หน้า "จองสำเร็จ" กลับไปแสดงผลให้ผู้ใช้งานทราบ

### 3.6.2 Sequence Diagram ของผู้ดูแลระบบจัดการการจอง



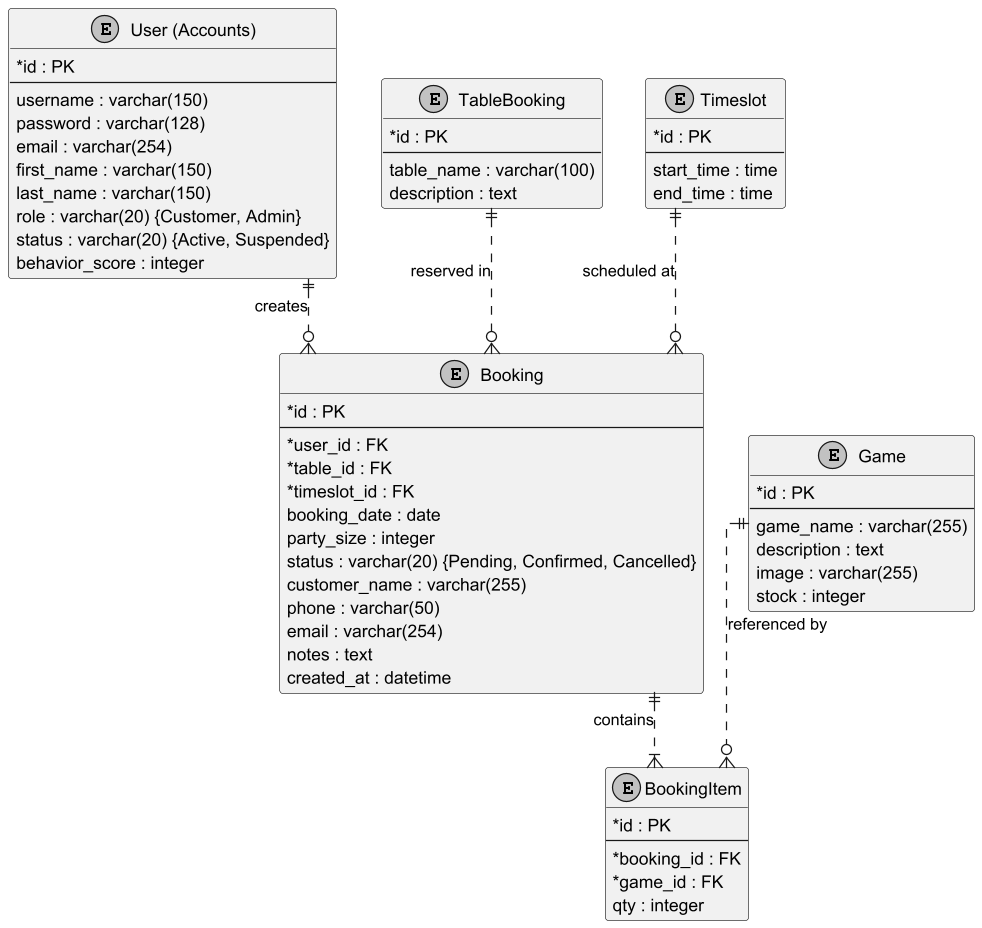
ภาพประกอบที่ 3.18 Sequence Diagram ของผู้ดูแลระบบจัดการการจอง

คำอธิบาย: แผนภาพนี้แสดงลำดับการทำงานเมื่อผู้ดูแลระบบต้องการตรวจสอบและแก้ไขสถานะการจอง (เช่น อนุมัติ หรือ ยกเลิก) โดยมีขั้นตอนดังนี้:

1. เรียกดูข้อมูล: ผู้ดูแลระบบ (Admin) เข้าสู่หน้ารายละเอียดการจอง ระบบจะทำการดึงข้อมูล (Query) จากฐานข้อมูลมาแสดงผลบนหน้าจอ
2. ส่งคำสั่งแก้ไข: ผู้ดูแลระบบทำการเปลี่ยนแปลงสถานะการจอง (เช่น เปลี่ยนจาก Pending เป็น Confirmed) และกดบันทึก
3. บันทึกข้อมูล: ระบบทำการตรวจสอบความถูกต้องของข้อมูล (Validation) และทำการอัปเดตสถานะใหม่ลงในฐานข้อมูล
4. ตอบกลับ: เมื่อฐานข้อมูลอัปเดตเสร็จสมบูรณ์ ระบบจะส่งข้อความแจ้งเตือน "บันทึกสำเร็จ" กลับมายังผู้ดูแลระบบ พร้อมแสดงข้อมูลที่เป็นปัจจุบัน

## 3.7 Entity Relationship Diagram

แผนภาพความสัมพันธ์ระหว่างข้อมูล (ERD) แสดงโครงสร้างการจัดเก็บข้อมูลของระบบจองร้านบอร์ดเกม โดยประกอบด้วยเอนทิตี (Entity) หลักจำนวน 6 เอนทิตี และมีความสัมพันธ์ (Relationship) ดังนี้



ภาพประกอบที่ 3.19 แผนภาพ Entity Relationship Diagram (ERD)

คำอธิบายแผนภาพ: จากภาพที่ 3.xx แสดงโครงสร้างความสัมพันธ์ของตารางในฐานข้อมูล (Entity Relationship Diagram) ของระบบจองร้านบอร์ดเกม ซึ่งมีการจัดเก็บข้อมูลแบบเชื่อมโยงกันเพื่อประสิทธิภาพในการเรียกใช้และรักษาความถูกต้องของข้อมูล โดยมีรายละเอียดดังนี้

* User & Accounts: ตาราง User (Accounts) เป็นตารางหลักที่เก็บข้อมูลสมาชิกและผู้ดูแลระบบ โดยเชื่อมโยงกับรายการจองเพื่อระบุตัวตนของผู้ทำรายการ
* Booking System (ระบบการจอง):
  + Booking ทำหน้าที่เป็นศูนย์กลางของข้อมูลการจอง เชื่อมโยงกับ User (ผู้จอง), TableBooking (โต๊ะที่เลือก) และ Timeslot (ช่วงเวลาที่จอง) เพื่อระบุรายละเอียดการจองในแต่ละครั้ง
  + BookingItem เก็บรายละเอียดของบอร์ดเกมที่ถูกเลือกในการจองนั้น ๆ โดยมีความสัมพันธ์แบบ One-to-Many กับตาราง Booking (การจอง 1 ครั้งมีได้หลายรายการเกม)
* Store Resources (ทรัพยากรในร้าน):
  + TableBooking เก็บข้อมูลโต๊ะภายในร้าน เพื่อใช้ตรวจสอบสถานะการจองในแต่ละช่วงเวลา
  + Timeslot เก็บข้อมูลรอบเวลาที่เปิดให้บริการ เพื่อกำหนดขอบเขตเวลาในการจองให้เป็นมาตรฐาน
  + Game เก็บข้อมูลรายละเอียดของบอร์ดเกมและจำนวนสต็อกรวมทั้งหมด โดยจะถูกอ้างอิงผ่าน BookingItem เพื่อใช้ในการคำนวณจำนวนสต็อกคงเหลือจริง ณ ช่วงเวลาที่ลูกค้าเลือก

### 3.7.1 พจนานุกรมข้อมูล (Data Dictionary)

พจนานุกรมข้อมูลเป็นการอธิบายรายละเอียดของโครงสร้างตารางในฐานข้อมูล เพื่อให้ทราบถึงคุณลักษณะของข้อมูลในแต่ละฟิลด์ที่ใช้ในระบบ โดยมีรายละเอียดดังนี้

###### ตารางที่ 3.20 พจนานุกรมตาราง ผู้ใช้งาน (Accounts\_User)

| ชื่อฟิลด์ (Attribute) | ชนิดข้อมูล (Data Type) | Key | รายละเอียด |
| --- | --- | --- | --- |
| id | int | PK | รหัสประจำตัวผู้ใช้ (Auto Increment) |
| username | string |  | ชื่อบัญชีผู้ใช้สำหรับการเข้าสู่ระบบ |
| password | string |  | รหัสผ่านที่ผ่านการเข้ารหัส (PBKDF2 Hash) |
| email | string |  | อีเมลสำหรับติดต่อและแจ้งเตือน |
| first\_name | string |  | ชื่อจริงของผู้ใช้งาน |
| last\_name | string |  | นามสกุลของผู้ใช้งาน |
| role | string |  | บทบาท (Customer = ลูกค้า, Admin = ผู้ดูแล) |
| status | string |  | สถานะบัญชี (Active = ปกติ, Suspended = ระงับการใช้งาน) |
| behavior\_score | int |  | คะแนนพฤติกรรม (สำหรับประเมินความน่าเชื่อถือ) |
| last\_login | datetime |  | วันที่และเวลาที่เข้าใช้งานล่าสุด |

###### ตารางที่ 3.21 พจนานุกรมตาราง การจอง (Booking\_Booking)

| ชื่อฟิลด์ (Attribute) | ชนิดข้อมูล (Data Type) | Key | รายละเอียด |
| --- | --- | --- | --- |
| id | int | PK | รหัสรายการจอง (Auto Increment) |
| user\_id | int | FK | รหัสผู้ใช้งานที่ทำการจอง (เชื่อมกับตาราง Accounts\_User) |
| table\_id | int | FK | รหัสโต๊ะที่เลือกจอง (เชื่อมกับตาราง Booking\_TableBooking) |
| timeslot\_id | int | FK | รหัสช่วงเวลาที่จอง (เชื่อมกับตาราง Booking\_Timeslot) |
| booking\_date | date |  | วันที่ลูกค้าเลือกเข้าใช้บริการ |
| status | string |  | สถานะรายการ (Pending, Confirmed, Cancelled) |
| customer\_name | string |  | ชื่อ-นามสกุล ของลูกค้าที่ระบุในรายการจอง |
| phone | string |  | เบอร์โทรศัพท์สำหรับติดต่อ |
| email | string |  | อีเมลสำหรับส่งรายละเอียดการจอง |
| created\_at | datetime |  | วันที่และเวลาที่ทำรายการจองในระบบ |

###### ตารางที่ 3.22 พจนานุกรมตาราง รายการเกมที่จอง (Booking\_BookingItem)

###### 

| ชื่อฟิลด์ (Attribute) | ชนิดข้อมูล (Data Type) | Key | รายละเอียด |
| --- | --- | --- | --- |
| id | int | PK | รหัสรายการย่อย (Auto Increment) |
| booking\_id | int | FK | รหัสการจองหลัก (เชื่อมกับตาราง Booking\_Booking) |
| game\_id | int | FK | รหัสบอร์ดเกมที่เลือก (เชื่อมกับตาราง Booking\_Game) |
| qty | int |  | จำนวนชุดของเกมที่จอง |

###### 

###### ตารางที่ 3.23 พจนานุกรมตาราง บอร์ดเกม (Booking\_Game)

###### 

| ชื่อฟิลด์ (Attribute) | ชนิดข้อมูล (Data Type) | Key | รายละเอียด |
| --- | --- | --- | --- |
| id | int | PK | รหัสบอร์ดเกม (Auto Increment) |
| game\_name | string |  | ชื่อของบอร์ดเกม |
| description | text |  | คำอธิบายเกม กติกาเบื้องต้น หรือประเภทของเกม |
| path\_image | string |  | ตำแหน่งเก็บไฟล์รูปภาพปกเกมใน Server |
| stock | int |  | จำนวนสต็อกรวมทั้งหมดที่มีให้บริการในร้าน |

###### 

###### ตารางที่ 3.24 พจนานุกรมตาราง โต๊ะบอร์ดเกม (Booking\_TableBooking)

###### 

| ชื่อฟิลด์ (Attribute) | ชนิดข้อมูล (Data Type) | Key | รายละเอียด |
| --- | --- | --- | --- |
| id | int | PK | รหัสโต๊ะ (Auto Increment) |
| table\_name | string |  | ชื่อหรือหมายเลขโต๊ะ (เช่น Table 01, Zone A) |
| description | text |  | รายละเอียดของโต๊ะ เช่น จำนวนที่นั่งสูงสุด |

###### 

###### ตารางที่ 3.25 พจนานุกรมตาราง ช่วงเวลา (Booking\_Timeslot)

###### 

| ชื่อฟิลด์ (Attribute) | ชนิดข้อมูล (Data Type) | Key | รายละเอียด |
| --- | --- | --- | --- |
| id | int | PK | รหัสช่วงเวลา (Auto Increment) |
| start\_time | time |  | เวลาเริ่มต้นรอบการให้บริการ |
| end\_time | time |  | เวลาสิ้นสุดรอบการให้บริการ |

###### 

###### 

###### 

# บทที่ 4 การพัฒนาระบบ

ในการพัฒนาระบบจองบอร์ดเกม (Board Game Booking System) ผู้จัดทำได้เลือกใช้เครื่องมือและเทคโนโลยีในการพัฒนา Web Application โดยใช้ภาษา Python ร่วมกับ Django Framework และใช้ฐานข้อมูล PostgreSQL โดยมีรายละเอียดการพัฒนาและการตั้งค่าระบบดังนี้

## 4.1 โครงสร้างแอปพลิเคชันและการตั้งค่า (Project Configuration)

### 4.1.1 การจัดการ Settings.py สำหรับการเชื่อมต่อฐานข้อมูลและ Static Files

ไฟล์ settings.py เป็นส่วนหัวใจหลักในการกำหนดค่าต่าง ๆ ของระบบ โดยในโปรเจกต์นี้ได้มีการตั้งค่าที่สำคัญดังนี้

1. INSTALLED\_APPS: การลงทะเบียนแอปพลิเคชันที่ใช้งาน ซึ่งประกอบด้วยแอปพลิเคชันหลักคือ accounts (จัดการผู้ใช้) และ booking (จัดการการจอง) รวมถึงไลบรารีเสริมอย่าง tailwind สำหรับจัดการ CSS

  
INSTALLED\_APPS = [

# ... Django Default Apps ...

"tailwind",

"theme",

'django\_browser\_reload',

'accounts',

'booking',

]



1. DATABASES: การตั้งค่าการเชื่อมต่อฐานข้อมูล PostgreSQL โดยระบุ Engine, ชื่อฐานข้อมูล, ผู้ใช้งาน และรหัสผ่าน

  
DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'boardgame\_db',

'USER': 'postgres',

'PASSWORD': 'xxx', # รหัสผ่านถูกซ่อนเพื่อความปลอดภัย

'HOST': 'localhost',

'PORT': '5432',

}

}



1. Authentication Model: การกำหนดให้ระบบใช้ Model ผู้ใช้งานที่สร้างขึ้นเอง (Custom User Model) แทนค่าเริ่มต้นของ Django

  
AUTH\_USER\_MODEL = 'accounts.User'



### 4.1.2 การออกแบบระบบ URL Routing (Urls.py) แบบแยกแอปพลิเคชัน

ระบบมีการออกแบบ URL Routing โดยแบ่งแยกหน้าที่ชัดเจนระหว่างไฟล์ urls.py ระดับโปรเจกต์และระดับแอปพลิเคชัน เพื่อให้ง่ายต่อการบำรุงรักษา

* ระดับโปรเจกต์ (config/urls.py): ทำหน้าที่เป็น Gateway หลักในการกระจาย Request ไปยังแอปพลิเคชันต่างๆ  
  Python

  
urlpatterns = [

path('admin/', admin.site.urls),

path('', home\_view, name='home'),

path('login/', login\_view, name='login'),

path('register/', register\_view, name='register'),

path('booking/', include('booking.urls')), # เชื่อมโยงไปยัง URLs ของแอป booking

]

## 4.2 โครงสร้างไฟล์และโฟลเดอร์ของระบบ (Django Directory Structure)

### 4.2.1 โครงสร้างระดับโครงการ (Project Level) และหน้าที่ของไฟล์ WSGI/ASGI

โครงสร้างหลักของโปรเจกต์ถูกจัดเก็บภายใต้โฟลเดอร์ config (หรือชื่อโปรเจกต์หลัก) โดยมีไฟล์สำคัญคือ:

* manage.py: ยูทิลิตี้สำหรับสั่งการระบบ เช่น การรันเซิร์ฟเวอร์ (runserver) หรือการไมเกรตฐานข้อมูล (migrate)
* wsgi.py (Web Server Gateway Interface): จุดเชื่อมต่อสำหรับ Web Server แบบดั้งเดิมเพื่อนำแอปพลิเคชันขึ้นใช้งานจริง (Deployment)
* asgi.py (Asynchronous Server Gateway Interface): จุดเชื่อมต่อสำหรับรองรับการทำงานแบบ Asynchronous

### 4.2.2 โครงสร้างระดับแอปพลิเคชัน (App Level)

ระบบแบ่งออกเป็น 2 แอปพลิเคชันหลักตามขอบเขตหน้าที่การทำงาน ได้แก่:

1. accounts: รับผิดชอบเรื่องการยืนยันตัวตน (Authentication), การจัดการข้อมูลผู้ใช้ และโปรไฟล์
2. booking: รับผิดชอบฟังก์ชันหลักทางธุรกิจ คือ การจองโต๊ะ, การเลือกบอร์ดเกม, การจัดการสต็อก และระบบหลังบ้านของผู้ดูแลระบบ (Admin Dashboard)

### 4.2.3 หน้าที่ของไฟล์สำคัญในแต่ละแอปพลิเคชัน

ในแต่ละแอปพลิเคชัน (accounts, booking) จะประกอบด้วยไฟล์มาตรฐาน MVC (Model-View-Controller) ของ Django ดังนี้:

* models.py: กำหนดโครงสร้างข้อมูล (Schema) และความสัมพันธ์ระหว่างตารางในฐานข้อมูล
* views.py: เขียน Logic การทำงาน ประมวลผลข้อมูลจาก Model และส่งผลลัพธ์ไปยัง Template
* forms.py: จัดการการรับค่าจากผู้ใช้ (Input Validation) และสร้าง HTML Form อัตโนมัติ
* urls.py: กำหนดเส้นทาง URL ภายในแอปพลิเคชันนั้นๆ
* admin.py: ลงทะเบียน Model เพื่อให้สามารถจัดการข้อมูลผ่านหน้า Django Admin Site ได้

### 4.2.4 การจัดหมวดหมู่ไฟล์เทมเพลต (Templates) และไฟล์สถิต (Static Files)

* Templates: ไฟล์ HTML จะถูกเก็บไว้ในโฟลเดอร์ templates/ โดยมีการแยกโฟลเดอร์ย่อยตามชื่อแอปพลิเคชัน เช่น templates/accounts/ และ templates/booking/
* Static Files: ไฟล์ CSS, JavaScript และรูปภาพ จะถูกจัดการผ่านแอปพลิเคชัน theme (Tailwind CSS) และเก็บรวบรวมไว้สำหรับการแสดงผล

## 4.3 การพัฒนาฟังก์ชันการทำงานด้วย Django (Core Implementation)

### 4.3.1 การขยายความสามารถของ User Model (Custom User Model)

ระบบได้ทำการขยาย User Model ของ Django (AbstractUser) ในไฟล์ accounts/models.py เพื่อเพิ่มฟิลด์ที่จำเป็นต่อระบบสมาชิกและระบบแบนผู้ใช้งาน

class User(AbstractUser):

ROLE\_CHOICES = [

('customer', 'Customer'),

('admin', 'Admin'),

]

STATUS\_CHOICES = [

('active', 'Active'),

('suspended', 'Suspended'),

]

name = models.CharField(max\_length=255, blank=True)

role = models.CharField(max\_length=20, choices=ROLE\_CHOICES, default='customer')

status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='active')

behavior\_score = models.IntegerField(default=0)

### **4.3.2 การเขียน Logic ใน Views.py (Business Logic Implementation)**

ในส่วนนี้จะอธิบายการทำงานของฟังก์ชันหลักใน views.py โดยแบ่งตาม Use Case Diagram ดังนี้

#### **4.3.2.1 การสมัครสมาชิก (Register)**

ฟังก์ชัน register\_view ใน accounts/views.py ทำหน้าที่รับข้อมูลจากฟอร์ม ตรวจสอบความถูกต้อง และสร้างบัญชีผู้ใช้ใหม่ พร้อมทั้งทำการ Login ให้อัตโนมัติเมื่อสมัครสำเร็จ

def register\_view(request):

if request.method == "POST":

form = RegisterForm(request.POST)

if form.is\_valid():

user = form.save()

login(request, user)

return redirect("home")

else:

form = RegisterForm()

return render(request, "accounts/register.html", {"form": form})

#### **4.3.2.2 การจอง - เลือกวันที่ (Select Date)**

ฟังก์ชัน select\_date ใน booking/views.py เป็นขั้นตอนแรกของการจอง ทำหน้าที่รับวันที่ที่ลูกค้าต้องการเข้าใช้บริการ

@login\_required

def select\_date(request):

if request.method == "POST":

date\_str = request.POST.get("booking\_date")

if not date\_str:

messages.error(request, "กรุณาเลือกวันที่")

else:

# เก็บวันที่ลง Session เพื่อใช้ในขั้นตอนถัดไป

request.session["booking\_date"] = date\_str

return redirect("select\_timeslot")

return render(request, "booking/select\_date.html")

### **4.3.2.3 การจอง - เลือกเกมและสรุปรายการ (Booking Summary)**

ฟังก์ชัน booking\_summary ทำหน้าที่รวบรวมข้อมูลทั้งหมดจาก Session (วันที่, รอบเวลา, เกม) เพื่อสร้างรายการ Booking และ BookingItem ลงในฐานข้อมูลจริง

@login\_required

def booking\_summary(request):

# ... (code ดึงข้อมูลจาก session) ...

if request.method == "POST":

with transaction.atomic():

booking = Booking.objects.create(

user=request.user,

table=table,

timeslot=timeslot,

booking\_date=booking\_date,

# ...

)

for game\_id, qty in cart.items():

BookingItem.objects.create(booking=booking, game\_id=game\_id, qty=qty)

# ...

return redirect("booking\_success")

# ...

**4.3.2.4 ผู้ดูแลระบบ - แดชบอร์ด (Admin Dashboard)** ฟังก์ชัน admin\_dashboard ใช้สำหรับแสดงภาพรวมของระบบ เช่น สถิติการจองรายวัน

@admin\_required

def admin\_dashboard(request):

today = timezone.now().date()

# ดึงข้อมูลการจองแยกตามวัน

daily\_stats = (

Booking.objects.annotate(date=TruncDate('booking\_date'))

.values('date')

.annotate(count=Count('id'))

.order\_by('-date')

)

return render(request, "admin/dashboard.html", {

"daily\_stats": daily\_stats,

# ...

})

### 4.3.3 การสร้างและจัดการฟอร์มด้วย Django Forms (Forms.py)

การจัดการ Input ของผู้ใช้ทำผ่าน Django Forms เพื่อความปลอดภัยและความสะดวกในการตรวจสอบข้อมูล (Validation)

* UserAdminForm: ฟอร์มสำหรับผู้ดูแลระบบเพื่อแก้ไขข้อมูลสมาชิก รวมถึงการปรับสถานะ (Status) และคะแนนความพฤติกรรม (Behavior Score)

  
class UserAdminForm(forms.ModelForm):

class Meta:

model = get\_user\_model()

fields = ["username", "email", "name", "role", "status", "behavior\_score", ...]

widgets = {

"role": forms.Select(attrs={"class": BASE\_INPUT\_CLASS}),

# ...

}



* GameForm: ฟอร์มสำหรับจัดการข้อมูลบอร์ดเกม (เพิ่ม/ลบ/แก้ไข)

  
class GameForm(forms.ModelForm):

class Meta:

model = Game

fields = ["game\_name", "description", "image", "stock"]



## 4.4 การจัดการรูปแบบและการออกแบบ (Theming)

### 4.4.1 การบูรณาการ Tailwind CSS เข้ากับ Django Templates

ระบบใช้ Django-Tailwind ในการจัดการ CSS Framework เพื่อความสวยงามและความรวดเร็วในการจัดหน้าเว็บ โดยมีการติดตั้งแอปพลิเคชันชื่อ theme ซึ่งจะทำการ Compile ไฟล์ CSS จาก Tailwind ให้เป็น Static files ที่ Django สามารถเรียกใช้งานได้

### 4.4.2 การใช้ Template Inheritance และการสร้าง Reusable Components

เพื่อลดความซ้ำซ้อนของโค้ด HTML ระบบใช้ฟีเจอร์ Template Inheritance ของ Django:

* Base Template (base.html): เป็นโครงสร้างหลักของทุกหน้า ประกอบด้วยส่วน Header (Navbar), Sidebar, และ Footer รวมถึงการโหลด Script และ CSS หลัก
* Child Templates: หน้าจออื่นๆ เช่น login.html หรือ booking\_list.html จะทำการสืบทอด ({% extends 'base.html' %}) และเขียนเฉพาะเนื้อหาในส่วน {% block content %} เท่านั้น

# บทที่ 5 การทดสอบระบบ

ขั้นตอนการทดสอบระบบเป็นกระบวนการสำคัญเพื่อตรวจสอบความถูกต้องของการทำงานในแต่ละฟังก์ชัน (Functionality) และความเชื่อถือได้ของระบบ (Reliability) โดยมีวัตถุประสงค์เพื่อหาข้อผิดพลาด (Bugs) และแก้ไขก่อนนำไปใช้งานจริง เพื่อให้การพัฒนาโปรแกรมเป็นไปอย่างมีประสิทธิภาพสูงสุด

## 5.1 Unit Testing

การทดสอบหน่วยย่อย (Unit Testing) เป็นการทดสอบการทำงานของโมดูลหรือฟังก์ชันที่เล็กที่สุดในระบบ เพื่อให้มั่นใจว่าตรรกะ (Logic) ภายในทำงานถูกต้อง โดยในระบบจองบอร์ดเกมได้มีการทดสอบส่วนสำคัญดังนี้

### 5.1.1 การทดสอบฟังก์ชันคำนวณสต็อกและตรวจสอบข้อมูล

1. การทดสอบเมธอด remaining\_on ใน Model Game:
   * วัตถุประสงค์: ตรวจสอบว่าระบบคำนวณจำนวนบอร์ดเกมคงเหลือในแต่ละช่วงเวลาได้ถูกต้องหรือไม่ โดยหักลบจำนวนที่ถูกจอง (Pending/Confirmed) ออกจากจำนวนทั้งหมด
   * วิธีการ: จำลองข้อมูลการจองในฐานข้อมูล และเรียกใช้ฟังก์ชัน game.remaining\_on(date, timeslot) เพื่อเปรียบเทียบค่าที่ได้กับค่าที่คาดหวัง
2. การทดสอบความถูกต้องของแบบฟอร์ม (Form Validation):
   * วัตถุประสงค์: ตรวจสอบว่า RegisterForm และ BookingAdminForm สามารถดักจับข้อมูลที่ไม่ถูกต้องได้ เช่น การใส่อีเมลผิดรูปแบบ หรือการจองโต๊ะซ้ำในช่วงเวลาเดียวกัน
   * วิธีการ: ส่งข้อมูล Input ที่ผิดพลาดผ่านฟอร์มและตรวจสอบ Error Message ที่ระบบตอบกลับมา

## 5.2 การทดสอบการทำงานร่วมกัน (Integration Testing)

การทดสอบการทำงานร่วมกันเป็นการตรวจสอบการส่งต่อข้อมูลระหว่างโมดูลต่าง ๆ ว่าสามารถทำงานประสานกันได้ถูกต้องหรือไม่

### 5.2.1 การทดสอบกระบวนการตั้งแต่จองนัดหมายจนถึงการบันทึกการรักษา (Booking Flow)

กระบวนการจองบอร์ดเกมมีการส่งผ่านข้อมูลหลายขั้นตอนผ่านตัวแปร Session (request.session) การทดสอบครอบคลุมลำดับการทำงานดังนี้:

1. Select Date: ผู้ใช้เลือกวันที่ -> ระบบบันทึกวันที่ลง Session.
2. Select Timeslot: ผู้ใช้เลือกเวลา -> ระบบตรวจสอบโต๊ะว่างและบันทึกเวลาลง Session.
3. Select Game: ผู้ใช้เลือกเกม -> ระบบคำนวณสต็อกและบันทึกรายการเกมลง Session.
4. Booking Summary: ระบบดึงข้อมูลทั้งหมดจาก Session มาแสดงผลเพื่อยืนยัน.
5. Save to Database: เมื่อกดยืนยัน ระบบบันทึกข้อมูลลงตาราง Booking และ BookingItem พร้อมตัดสต็อกและเคลียร์ค่า Session.

## 5.3 ผลการทดสอบและกรณีทดสอบ (Test Cases)

การทดสอบระบบได้ดำเนินการตามกรณีทดสอบ (Test Scenarios) ที่ครอบคลุมทุก Use Case ทั้งฝั่งผู้ใช้งานทั่วไป (Customer) และผู้ดูแลระบบ (Admin) โดยมีรายละเอียดดังนี้

#### ตารางที่ 5.1 ผลการทดสอบระบบส่วนการจัดการผู้ใช้งาน (User Management)

| รหัส (ID) | กรณีทดสอบ (Test Scenario) | ข้อมูลนำเข้า (Input Data) | ผลลัพธ์ที่คาดหวัง (Expected Result) | ผลลัพธ์จริง | สถานะ |
| --- | --- | --- | --- | --- | --- |
| TC-01 | สมัครสมาชิก (Register) - สำเร็จ | Username, Email, Password ที่ถูกต้อง | ระบบสร้างบัญชีผู้ใช้ใหม่และเข้าสู่ระบบให้อัตโนมัติ (Redirect ไปหน้า Home) | ตามคาดหวัง | Pass |
| TC-02 | สมัครสมาชิก - ข้อมูลซ้ำ | Username ที่มีอยู่แล้วในระบบ | ระบบแจ้งเตือนว่าชื่อผู้ใช้นี้ถูกใช้งานแล้ว | ตามคาดหวัง | Pass |
| TC-03 | เข้าสู่ระบบ (Login) - สำเร็จ | Username, Password ที่ถูกต้อง | ระบบเปลี่ยนสถานะเป็น Logged in และพาไปหน้า Home | ตามคาดหวัง | Pass |
| TC-04 | เข้าสู่ระบบ - รหัสผิด | Username ถูกต้อง, Password ผิด | ระบบแจ้งเตือน "ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง" | ตามคาดหวัง | Pass |
| TC-05 | แก้ไขโปรไฟล์ (Edit Profile) | ชื่อแสดงผล (Name), Email ใหม่ | ข้อมูลในฐานข้อมูลถูกอัปเดต และแสดงข้อความ Success Message | ตามคาดหวัง | Pass |

ตารางที่ 5.2 ผลการทดสอบระบบการจอง (Booking System)

| รหัส (ID) | กรณีทดสอบ (Test Scenario) | ข้อมูลนำเข้า (Input Data) | ผลลัพธ์ที่คาดหวัง (Expected Result) | ผลลัพธ์จริง | สถานะ |
| --- | --- | --- | --- | --- | --- |
| TC-06 | ค้นหาโต๊ะว่าง (Select Date/Time) | เลือกวันที่และช่วงเวลาที่ต้องการ | ระบบแสดงรายการช่วงเวลาและโต๊ะที่ว่างให้เลือก | ตามคาดหวัง | Pass |
| TC-07 | เลือกเกม (Select Game) | ระบุจำนวนเกมที่ต้องการจอง | ระบบแสดงเฉพาะเกมที่มีสต็อกเหลือเพียงพอในช่วงเวลานั้น | ตามคาดหวัง | Pass |
| TC-08 | ยืนยันการจอง (Confirm Booking) | ตรวจสอบข้อมูลและกด "ยืนยันการจอง" | ระบบบันทึกข้อมูลลงตาราง Booking, BookingItem และแสดงหน้า Success | ตามคาดหวัง | Pass |
| TC-09 | ดูประวัติการจอง (View History) | เข้าเมนู "ประวัติการจอง" | แสดงรายการจองทั้งหมดของผู้ใช้ เรียงจากล่าสุดไปเก่า | ตามคาดหวัง | Pass |
| TC-10 | ยกเลิกการจอง (Cancel Booking) | กดปุ่ม "ยกเลิก" ในรายการจอง | สถานะการจองเปลี่ยนเป็น Cancelled และคืนสต็อกเกม/โต๊ะ | ตามคาดหวัง | Pass |

ตารางที่ 5.3 ผลการทดสอบระบบผู้ดูแลระบบ (Admin Management)

| รหัส (ID) | กรณีทดสอบ (Test Scenario) | ข้อมูลนำเข้า (Input Data) | ผลลัพธ์ที่คาดหวัง (Expected Result) | ผลลัพธ์จริง | สถานะ |
| --- | --- | --- | --- | --- | --- |
| TC-11 | ดูภาพรวมระบบ (Dashboard) | เข้าเมนู Dashboard | แสดงกราฟหรือตารางสรุปยอดการจองรายวันได้ถูกต้อง | ตามคาดหวัง | Pass |
| TC-12 | เพิ่มบอร์ดเกมใหม่ (Add Game) | ชื่อเกม, รายละเอียด, รูปภาพ, จำนวนสต็อก | เกมใหม่ถูกเพิ่มลงในฐานข้อมูลและแสดงในหน้ารายการเกม | ตามคาดหวัง | Pass |
| TC-13 | แก้ไขข้อมูลการจอง (Edit Booking) | เปลี่ยนสถานะจาก Pending เป็น Confirmed | สถานะการจองเปลี่ยนแปลง และมีผลต่อการตัดสต็อก | ตามคาดหวัง | Pass |
| TC-14 | จัดการโต๊ะ (Manage Table) | เพิ่ม/ลบ/แก้ไข ข้อมูลโต๊ะ | ข้อมูลโต๊ะเปลี่ยนแปลง และมีผลต่อการเลือกจองของลูกค้า | ตามคาดหวัง | Pass |
| TC-15 | จัดการผู้ใช้งาน (Manage Customer) | เปลี่ยนสถานะผู้ใช้เป็น Suspended (ระงับ) | ผู้ใช้ไม่สามารถล็อกอินหรือทำรายการจองได้ | ตามคาดหวัง | Pass |

## 5.4 การประเมินความพึงพอใจและประสิทธิภาพของระบบ

หลังจากเสร็จสิ้นการทดสอบฟังก์ชันการทำงาน (Functional Test) ผู้จัดทำได้ทำการประเมินประสิทธิภาพและความพึงพอใจของผู้ใช้งาน (User Acceptance Test) โดยใช้แบบสอบถามกับกลุ่มตัวอย่าง จำนวน [ใส่จำนวน] คน โดยแบ่งหัวข้อการประเมินและได้ผลลัพธ์ดังนี้

เกณฑ์การให้คะแนน: 5 = มากที่สุด, 4 = มาก, 3 = ปานกลาง, 2 = น้อย, 1 = น้อยที่สุด

#### ตารางที่ 5.4 ผลการประเมินความพึงพอใจของผู้ใช้งาน

| หัวข้อการประเมิน | ค่าเฉลี่ย (xˉ) | ส่วนเบี่ยงเบนมาตรฐาน (S.D.) | ระดับความพึงพอใจ |
| --- | --- | --- | --- |
| 1. ด้านความสามารถในการทำงาน (Functional Requirement) |  |  |  |
| 1.1 ระบบสามารถทำรายการจองและบันทึกข้อมูลได้ถูกต้อง | 4.20 | 0.65 | มาก |
| 1.2 ความถูกต้องของการตรวจสอบสถานะโต๊ะว่างและเกม | 4.15 | 0.58 | มาก |
| 1.3 ระบบมีการแจ้งเตือนข้อมูลครบถ้วน | 3.45 | 0.70 | ปานกลาง |
| 2. ด้านความง่ายในการใช้งาน (Usability) |  |  |  |
| 2.1 ความสวยงามและความเหมาะสมของการออกแบบหน้าจอ | 3.80 | 0.62 | มาก |
| 2.2 ความง่ายในการเรียนรู้การใช้งานระบบในครั้งแรก | 3.50 | 0.75 | ปานกลาง |
| 2.3 ความสะดวกในการเข้าถึงเมนูต่างๆ | 4.05 | 0.55 | มาก |
| 3. ด้านประสิทธิภาพระบบ (Performance) |  |  |  |
| 3.1 ความรวดเร็วในการประมวลผลและการเปลี่ยนหน้าจอ | 3.40 | 0.68 | ปานกลาง |
| 3.2 การแสดงผลบนอุปกรณ์ที่แตกต่างกัน (Responsive) | 3.90 | 0.60 | มาก |
| สรุปผลรวมเฉลี่ย | 3.80 | 0.64 | มาก |

# บทที่ 6 สรุปและข้อเสนอแนะ

จากการดำเนินโครงงานพัฒนาระบบจองบอร์ดเกม (Board Game Booking System) ด้วย Web Application ผู้จัดทำได้ทำการวิเคราะห์ ออกแบบ และพัฒนาระบบจนเสร็จสมบูรณ์ โดยสามารถสรุปผลการดำเนินงาน ปัญหาที่พบ และข้อเสนอแนะเพื่อการพัฒนาต่อยอดได้ดังนี้

## 6.1 สรุปความสามารถของระบบ

ระบบจองบอร์ดเกมถูกพัฒนาขึ้นเพื่อแก้ปัญหาการจัดการการจองที่ซ้ำซ้อนและการตรวจสอบสต็อกเกมที่ยุ่งยาก โดยระบบสามารถแบ่งการทำงานออกเป็น 2 ส่วนหลัก ตามสิทธิ์การใช้งาน ดังนี้

1. สำหรับผู้ใช้งานทั่วไป (Customer)

* ระบบสมาชิก: สามารถสมัครสมาชิก (Register), เข้าสู่ระบบ (Login) และแก้ไขข้อมูลส่วนตัว (Profile) ได้
* ระบบการจอง (Booking Flow): ผู้ใช้สามารถทำรายการจองผ่านขั้นตอนที่ชัดเจน (Step-by-step) ตั้งแต่การเลือกวันที่, เลือกช่วงเวลา (Timeslot), เลือกโต๊ะที่นั่ง และเลือกบอร์ดเกมที่ต้องการเล่น
* การตรวจสอบสถานะเรียลไทม์: ระบบมีการคำนวณจำนวนสต็อกเกมและโต๊ะว่างในช่วงเวลาที่เลือกอย่างแม่นยำ เพื่อป้องกันการจองซ้ำ (Overbooking)
* ประวัติการจอง: สามารถดูรายการจองย้อนหลังและสถานะการจองปัจจุบัน รวมถึงสามารถยกเลิกการจองได้ด้วยตนเองหากสถานะยังรอดำเนินการ

2. สำหรับผู้ดูแลระบบ (Admin)

* แดชบอร์ด (Dashboard): แสดงภาพรวมสถิติการจองรายวัน เพื่อช่วยในการวางแผนเตรียมร้าน
* การจัดการข้อมูลหลัก (Master Data Management): สามารถเพิ่ม ลบ แก้ไข ข้อมูลบอร์ดเกม (รวมถึงรูปภาพ), ข้อมูลโต๊ะที่นั่ง และช่วงเวลาให้บริการ
* การจัดการการจอง: สามารถตรวจสอบรายการจองของลูกค้า ปรับสถานะ (Pending/Confirmed/Cancelled) หรือลบรายการจองที่มีปัญหาได้
* การจัดการผู้ใช้งาน: สามารถดูรายชื่อสมาชิก และทำการระงับบัญชี (Suspend) หากพบผู้ใช้งานที่มีคะแนนความประพฤติ (Behavior Score) ต่ำกว่าเกณฑ์

## 6.2 ปัญหาและอุปสรรคในการพัฒนา

ในการพัฒนาระบบจัดการการจองร้านบอร์ดเกม ผู้พัฒนาได้พบปัญหาและอุปสรรคในหลายด้าน ทั้งด้านเทคนิค การออกแบบระบบ และการทดสอบการทำงาน โดยสามารถสรุปได้ดังนี้

### 1) ปัญหาด้านการออกแบบตรรกะการจอง (Booking Logic)

ระบบจำเป็นต้องป้องกันปัญหาการจองซ้ำ (Double Booking) โดยเฉพาะในกรณีที่มีผู้ใช้งานหลายคนทำรายการจองในช่วงเวลาเดียวกัน ทำให้ต้องออกแบบกระบวนการทำงานของระบบให้รองรับการทำธุรกรรมฐานข้อมูลแบบ Atomic Transaction ผ่าน Django ORM และ PostgreSQL เพื่อให้ข้อมูลมีความถูกต้อง (Data Integrity) และไม่เกิดการตัดสต็อกเกมเกินจำนวนที่มีอยู่จริง

### 2) ปัญหาด้านการจัดการฐานข้อมูล (Database Management)

การออกแบบโครงสร้างฐานข้อมูล (Database Schema) ให้รองรับความสัมพันธ์ระหว่างตาราง เช่น Users, Tables, Games และ Bookings จำเป็นต้องวิเคราะห์ความสัมพันธ์ (Relationship) อย่างรอบคอบ หากออกแบบไม่ดีอาจทำให้เกิดข้อมูลซ้ำซ้อน (Redundancy) หรือการ Query ที่ซับซ้อนและช้า

### 3) ปัญหาด้านประสิทธิภาพของระบบ (Performance)

เมื่อมีข้อมูลการจองจำนวนมาก การดึงข้อมูลเพื่อแสดงผลในหน้า Dashboard หรือหน้าประวัติการจองอาจใช้เวลานาน ผู้พัฒนาต้องปรับปรุง Query ให้มีประสิทธิภาพ และใช้เทคนิคการเลือกข้อมูลเฉพาะส่วนที่จำเป็น (Query Optimization) เพื่อลดภาระของเซิร์ฟเวอร์

### 4) ปัญหาด้านการออกแบบส่วนติดต่อผู้ใช้ (UI/UX)

การออกแบบหน้าเว็บไซต์ให้ใช้งานง่าย (User-Friendly) และรองรับการแสดงผลแบบ Responsive บนอุปกรณ์หลายขนาด เป็นความท้าทายสำคัญ โดยเฉพาะหน้าจอเลือกช่วงเวลา (Timeslot) และหน้าเลือกเกม ซึ่งต้องแสดงสถานะว่าง/ไม่ว่าง และจำนวนสต็อกแบบชัดเจน เพื่อลดความสับสนของผู้ใช้งาน

### 5) ปัญหาด้านความปลอดภัยของระบบ (Security)

ระบบต้องมีการป้องกันช่องโหว่ เช่น

* การโจมตีแบบ CSRF (Cross-Site Request Forgery)
* การโจมตีแบบ SQL Injection
* การจัดการรหัสผ่านอย่างปลอดภัย (Password Hashing)

แม้ว่า Django จะมีระบบความปลอดภัยพื้นฐานมาให้แล้ว แต่ผู้พัฒนาจำเป็นต้องกำหนดสิทธิ์การเข้าถึงข้อมูล (Authorization) อย่างเหมาะสม เพื่อแยกบทบาทระหว่างผู้ใช้ทั่วไปและผู้ดูแลระบบ

### 6) ปัญหาด้านการติดตั้งและสภาพแวดล้อมการพัฒนา (Environment Setup)

ในระหว่างการพัฒนา พบปัญหาความแตกต่างของสภาพแวดล้อมระหว่างเครื่องพัฒนา (Development) และเครื่องเซิร์ฟเวอร์ (Production) เช่น เวอร์ชันของ Python, Django หรือฐานข้อมูล ทำให้ต้องใช้ Docker เพื่อจำลองสภาพแวดล้อมให้เหมือนกัน ลดปัญหา “ทำงานได้บนเครื่องเรา แต่รันบนเซิร์ฟเวอร์ไม่ได้”

## 6.3 แนวทางในการพัฒนาต่อ

เพื่อให้ระบบมีความสมบูรณ์และตอบโจทย์ทางธุรกิจมากยิ่งขึ้น ผู้จัดทำมีข้อเสนอแนะในการพัฒนาระบบต่อยอดในอนาคต ดังนี้

1. ระบบชำระเงินออนไลน์ (Payment Gateway): เชื่อมต่อกับผู้ให้บริการชำระเงิน (เช่น Stripe, Omise หรือ QR PromptPay) เพื่อให้ลูกค้าสามารถชำระค่ามัดจำหรือค่าบริการล่วงหน้าได้ทันที เพื่อลดปัญหาการจองแล้วไม่มาใช้บริการ (No-show)
2. ระบบแจ้งเตือนผ่านช่องทางอื่น (Notification System): เพิ่มการแจ้งเตือนสถานะการจองผ่าน LINE Notify หรือ SMS นอกเหนือจากการตรวจสอบผ่านหน้าเว็บ เพื่อให้ลูกค้าไม่พลาดข้อมูลสำคัญ
3. ระบบรีวิวและให้คะแนน (Review & Rating): เปิดโอกาสให้สมาชิกสามารถรีวิวบอร์ดเกมหรือการให้บริการของร้านได้ เพื่อเป็นข้อมูลประกอบการตัดสินใจให้ลูกค้ารายอื่น
4. การออกรายงานขั้นสูง (Advanced Reporting): พัฒนาระบบออกรายงานสรุปรายได้ และรายงานความนิยมของบอร์ดเกมแต่ละประเภท ในรูปแบบไฟล์ Excel หรือ PDF เพื่อให้ผู้ดูแลระบบนำข้อมูลไปวิเคราะห์ทางการตลาดได้ง่ายขึ้น

# บรรณานุกรม

**1) เอกสารคู่มือและเอกสารอ้างอิงของเทคโนโลยี (Official Documentation)**

Django Software Foundation. (2024). *Django documentation (Version 5.2)*. สืบค้นเมื่อ 15 สิงหาคม 2025, จาก<https://docs.djangoproject.com/>

Python Software Foundation. (2024). *Python 3 documentation*. สืบค้นเมื่อ 12 สิงหาคม 2025, จาก<https://docs.python.org/3/>

PostgreSQL Global Development Group. (2024). *PostgreSQL documentation*. สืบค้นเมื่อ 3 กันยายน 2025, จาก<https://www.postgresql.org/docs/>

Tailwind Labs. (2024). *Tailwind CSS documentation*. สืบค้นเมื่อ 19 กันยายน 2025, จาก<https://tailwindcss.com/docs>

Chart.js Contributors. (2024). *Chart.js documentation*. สืบค้นเมื่อ 2 ตุลาคม 2025, จาก<https://www.chartjs.org/docs/>

Docker Inc. (2024). *Docker documentation*. สืบค้นเมื่อ 8 กันยายน 2025, จาก<https://docs.docker.com/>

Git Project. (2024). *Git reference manual*. สืบค้นเมื่อ 24 สิงหาคม 2025, จาก<https://git-scm.com/docs>

ภาคผนวก

# ภาคผนวก ก การติดตั้งเครื่องมือที่ใช้พัฒนาโปรแกรม

การติดตั้งเครื่องมือที่ใช้ในการพัฒนาเว็บแอพพลิเคชันระบบจองคิวซ่อมรถจักรยานยนต์และ AI Chatbot ให้คำปรึกษา (THE\_ONE) มีโปรแกรมและเครื่องมือที่จำเป็นในการพัฒนาระบบ ดังต่อไปนี้

การติดตั้ง Visual Studio Code

การติดตั้ง Python

การติดตั้ง Docker

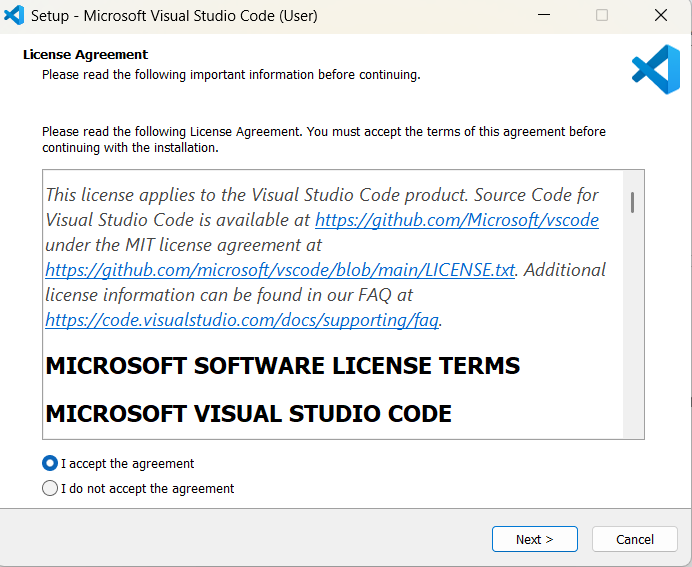
## ก.1 ขั้นตอนการติดตั้งโปรแกรม Visual Studio Code หรือ VS Code

1. สามารถดาวน์โหลด Visual Studio Code ได้ที่ <https://code.visualstudio.com/download> โปรแกรม VSCode โดยเลือกให้ตรงกับ OS ของเครื่องคอมพิวเตอร์



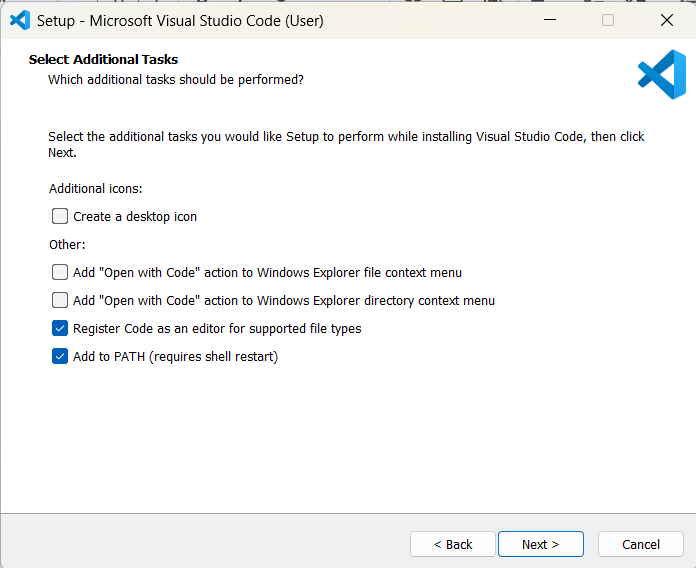
###### ภาพที่ ก.1 หน้าเว็บดาวน์โหลด Visual Studio Code

1. หลังดาวน์โหลดโปรแกรมเสร็จดับเบิ้ลคลิก หรือคลิกขวาและกด “Open” โปรแกรมที่ดาวน์โหลดมา ระบบจะแสดงหน้าต่างดังรูปที่ ก.2



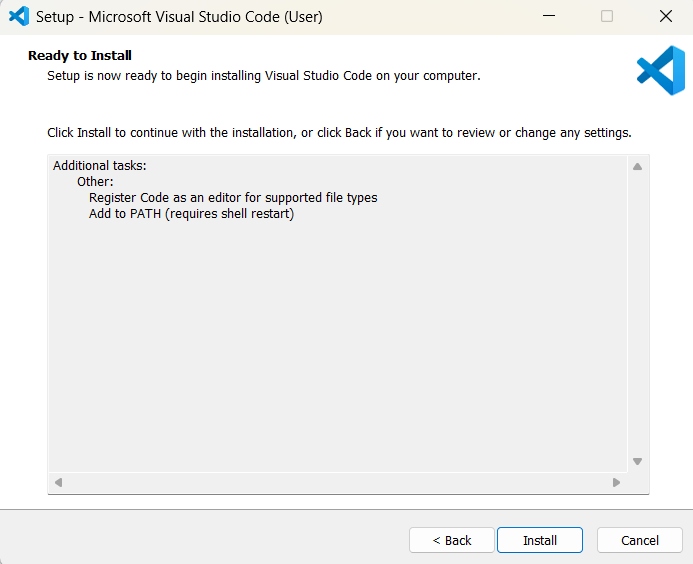
###### ภาพที่ ก.2 หน้าต่าง License Agreement

1. เลือก I accept the agreement แล้วกดปุ่ม Next > ระบบจะแสดงดังรูปที่ ก.3



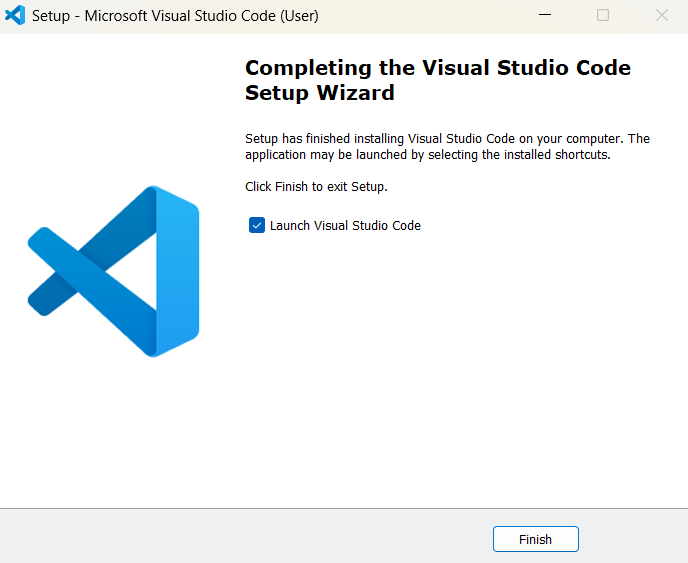
###### ภาพที่ ก.3 หน้าต่าง Select Additional Tasks

1. เลือก Add to PATH (requires shell restart) แล้วกด Next > ระบบจะแสดงหน้าต่างการ Install ดังรูปภาพที่ ก.4



###### ภาพที่ ก.4 หน้าต่างReady To Install

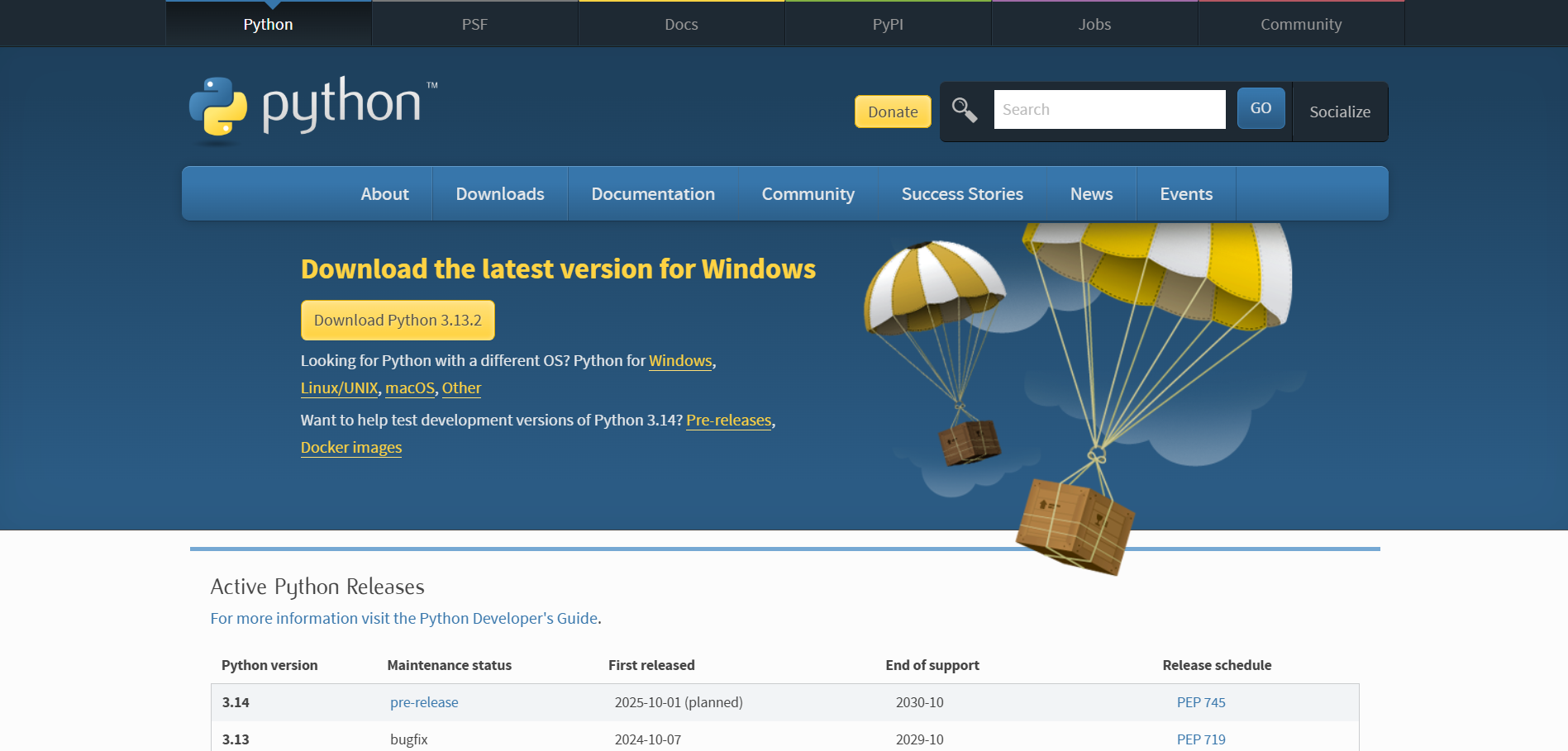
1. กด Install ในรูปภาพที่ ก.4 แล้วรอระบบติดตั้ง เมื่อเสร็จสิ้นระบบจะแสดงหน้าต่างดังภาพที่ ก.5



###### ภาพที่ ก.5 หน้าต่างการติดตั้งเสร็จสิ้น

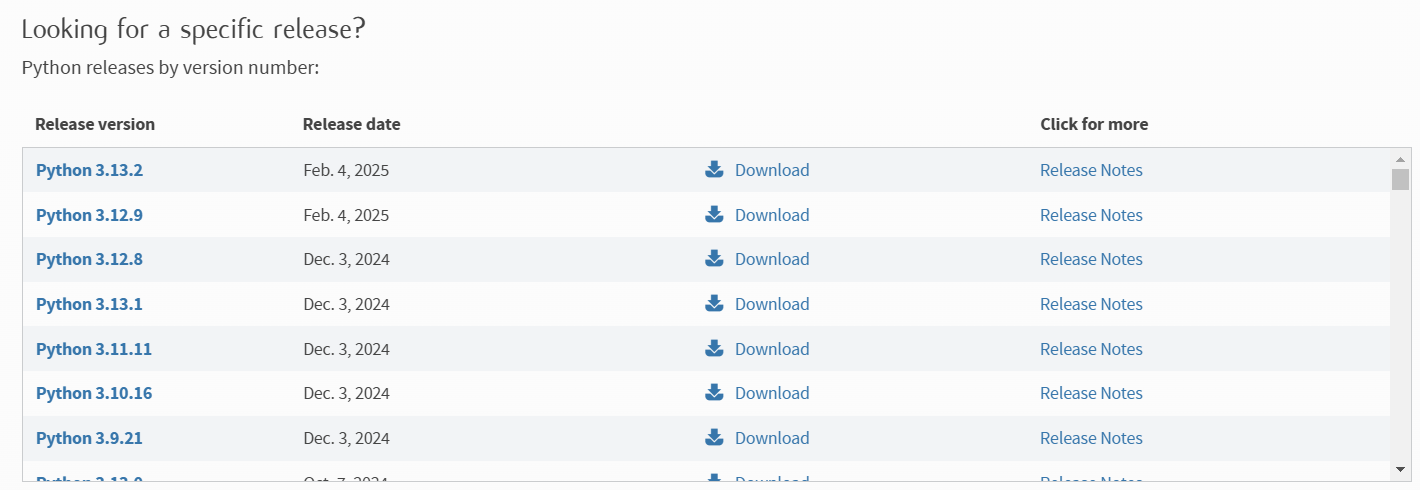
## ก.2 ขั้นตอนการติดตั้ง Python

1. สามารถดาวน์โหลด Python ได้ที่ https://www.python.org/downloads/ ดังแสดงในภาพที่ ก.6

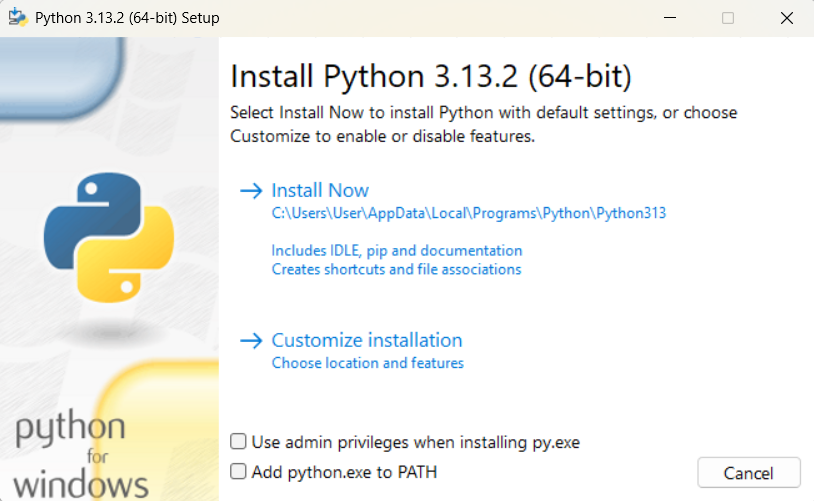


###### ภาพที่ ก.6 หน้าเว็บดาวน์โหลด Python

1. เลือกดาวน์โหลดเวอร์ชั้น 3.8 ขึ้นไป หลังจากกดดาวน์โหลดเลื่อนลงมาล่างสุดจะเป็นดังภาพที่ ก.7

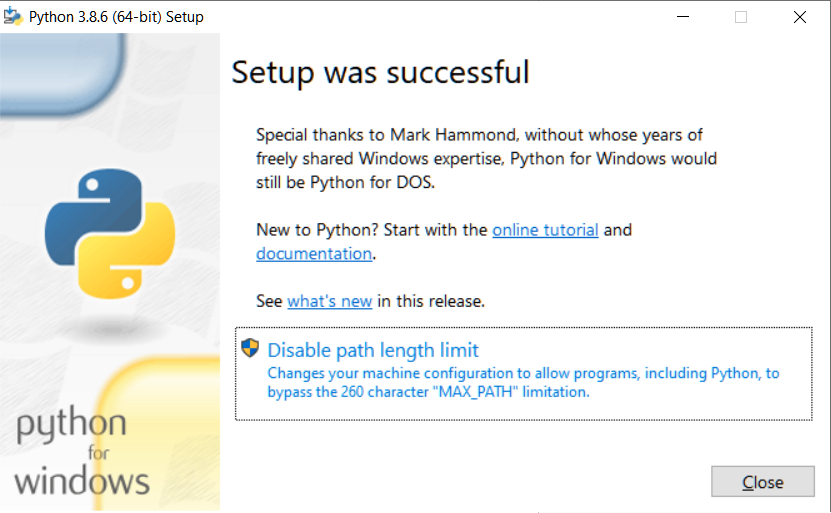


###### ภาพที่ ก.7 ภาพตัวเลือกที่ต้องการดาวน์โหลด

1. หลังจากกดดาวน์โหลดและทำการติดตั้ง ให้ติ้กถูกที่ Add python.exe to PATHจะแสดงดังภาพที่ ก.8

###### ภาพที่ ก.8 ภาพเมื่อทำการติดตั้ง Python

1. หลังจากทำการ Install Now จะแสดงดังภาพที่ ก.9



#### ภาพที่ ก.9 ภาพ ติดตั้งpython สำเร็จ

#### 

**ก.3 การติดตั้ง Docker Desktop**

Docker Desktop ใช้สำหรับจำลองสภาพแวดล้อม Server ในรูปแบบ Container โปรเจกต์ THE\_ONE ใช้ Docker Compose ในการรัน Service ต่อไปนี้:

- PostgreSQL (ankane/pgvector:latest) - ฐานข้อมูลหลักของระบบ รองรับ Vector Search

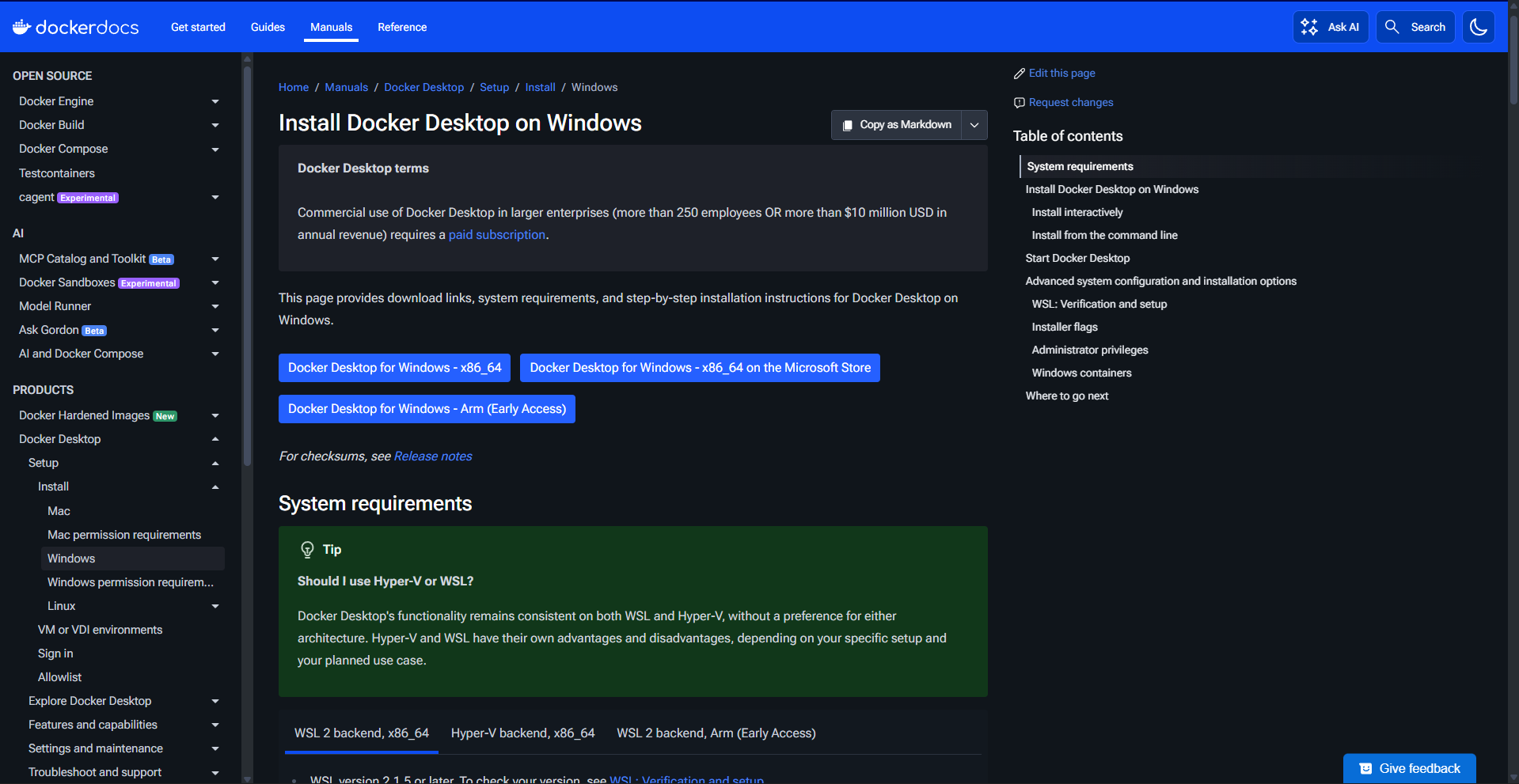
- Redis (redis:alpine) - ใช้สำหรับ Caching

- n8n (n8nio/n8n:latest) - Workflow Automation สำหรับประมวลผล RAG Chatbot

- Adminer (adminer:latest) - เครื่องมือจัดการฐานข้อมูลผ่าน Web UI

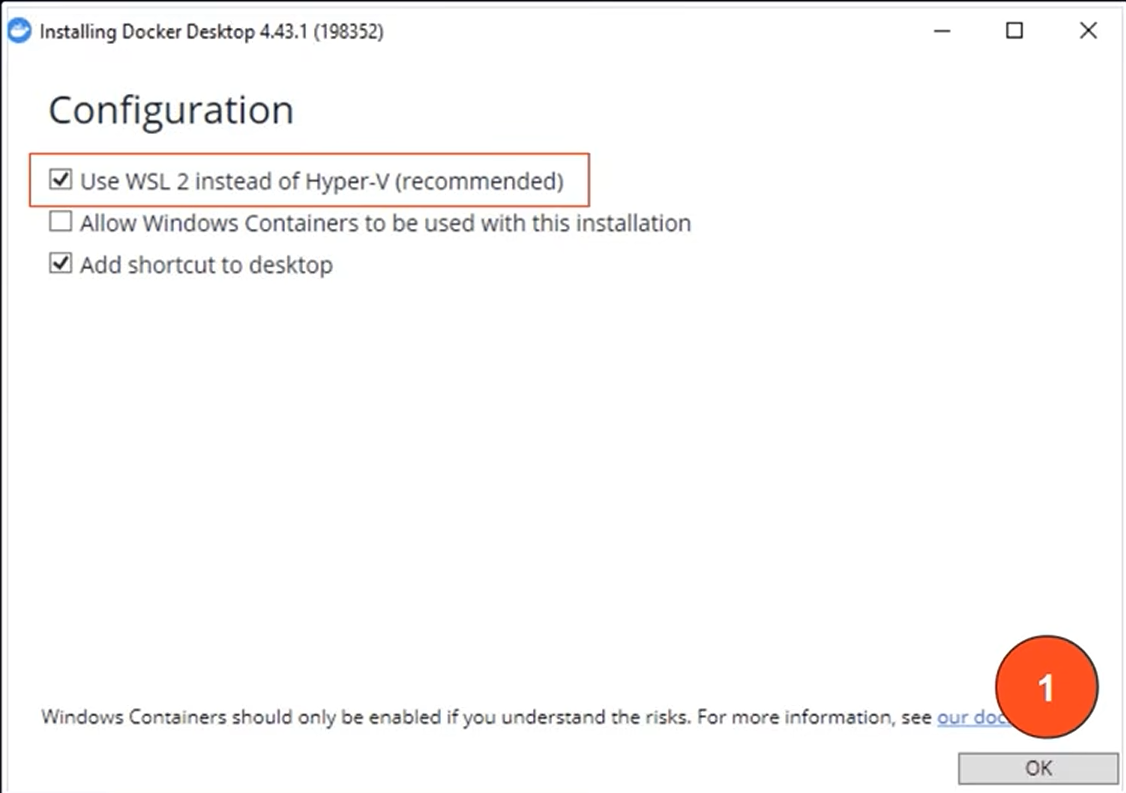
- ngrok (ngrok/ngrok:latest) - Tunnel สำหรับเปิดให้เข้าถึง Django จากภายนอก

ขั้นตอนการติดตั้ง:

1. เข้าไปที่ <https://docs.docker.com/desktop/setup/install/windows-install/> เพื่อทำการดาวน์โหลด Docker Desktop โดยเลือกให้ตรงกับ OS ของเครื่องคอมพิวเตอร์ 

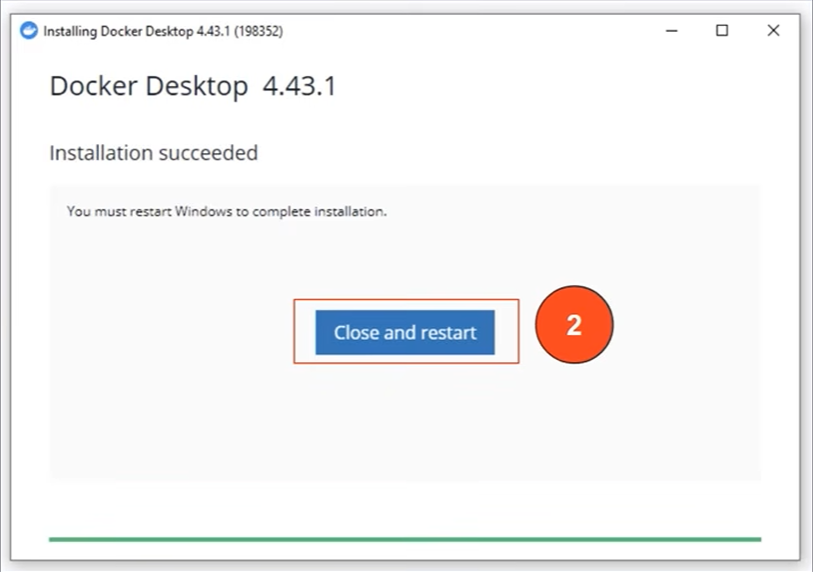
ภาพที่ ก.10 ภาพหน้าเว็บดาวน์โหลด Docker Desktop

2. หลังดาวน์โหลดโปรแกรมเสร็จดับเบิ้ลคลิก หรือคลิกขวาและกด “Open” โปรแกรมที่ดาวน์โหลดมา ระบบจะแสดงหน้าต่างดังรูปที่ ก.11



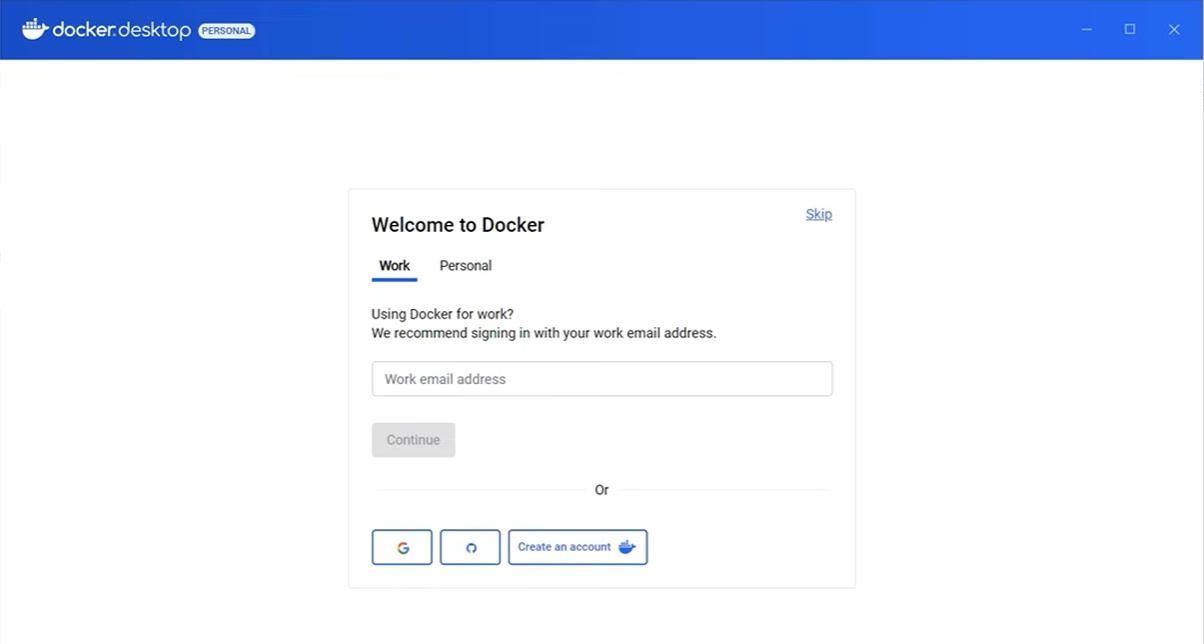
ภาพที่ ก.11 หน้าต่างติดตั้ง Docker

3. ให้ทำการกด ok ดังรูปภาพที่ ก.11 จากนั้น docker จะทำการติดตั้งให้รอจนกว่าจะเสร็จแล้ว docker จะแสดงหน้าต่างถัดไปดังรูปที่ ก.12



ภาพที่ ก.12 หน้าต่างติดตั้ง Docker สำเร็จ

4. หลังจากทำการติดตั้งเสร็จแล้วให้กดปุ่ม restart จากนั้นรอแล้วทำการเปิด docker ขึ้นมาเป็นอันเสร็จสิ้นดังภาพที่ ก.13



ภาพที่ ก.13 การติดตั้ง docker เสร็จสิ้น

# ภาคผนวก ข คู่มือการติดตั้งระบบ

## คู่มือการติดตั้งระบบ

คู่มือฉบับนี้จัดทำขึ้นเพื่ออธิบายขั้นตอนการติดตั้งระบบจองบอร์ดเกม (Board Game Booking System) สำหรับสภาพแวดล้อมการพัฒนา (Development Environment) โดยครอบคลุมตั้งแต่การเตรียมความพร้อมของเครื่องคอมพิวเตอร์ การติดตั้งโปรแกรมที่จำเป็น และการตั้งค่าเริ่มต้นระบบ

## ข.1 ความต้องการของระบบ (System Requirements)

ก่อนทำการติดตั้งระบบ เครื่องคอมพิวเตอร์แม่ข่ายหรือเครื่องสำหรับนักพัฒนาควรมีคุณสมบัติดังนี้

1. ระบบปฏิบัติการ (OS): Windows 10/11, macOS หรือ Linux (Ubuntu 20.04+)
2. ภาษาและเครื่องมือที่ต้องติดตั้ง (Prerequisites):
   * Python: เวอร์ชัน 3.10 ขึ้นไป
   * PostgreSQL: เวอร์ชัน 13 ขึ้นไป (สำหรับฐานข้อมูล)
   * Node.js: เวอร์ชัน 16 ขึ้นไป (สำหรับจัดการ Tailwind CSS)
   * Git: สำหรับดึงซอร์สโค้ด (Optional)

## ข.2 ขั้นตอนการติดตั้ง (Installation Steps)

### 1. การเตรียมฐานข้อมูล (Database Setup)

เนื่องจากระบบใช้ PostgreSQL ในการจัดเก็บข้อมูล จำเป็นต้องสร้างฐานข้อมูลเปล่าก่อนเริ่มใช้งาน

1. เปิดโปรแกรม pgAdmin 4 หรือ Command Line ของ PostgreSQL
2. สร้างฐานข้อมูลใหม่โดยใช้คำสั่ง SQL หรือผ่านหน้าจอ GUI ดังนี้:

**  
CREATE DATABASE boardgame\_db;**

****(หมายเหตุ: ชื่อฐานข้อมูลต้องตรงกับที่ระบุในไฟล์ settings.py)

### 2. การติดตั้งและตั้งค่าโปรเจกต์ (Project Setup)

1. ดาวน์โหลดซอร์สโค้ด: ทำการแตกไฟล์โปรเจกต์ (Unzip) หรือ Clone จาก Git Repository ไปยังโฟลเดอร์ที่ต้องการ
2. เปิด Terminal/Command Prompt: เข้าไปยังโฟลเดอร์ของโปรเจกต์

**  
cd path/to/your/project**

****

1. สร้างจำลองสภาพแวดล้อม (Virtual Environment): เพื่อแยกไลบรารีของโปรเจกต์ออกจากระบบหลัก
   * สำหรับ Windows:

**  
python -m venv venv**

**venv\Scripts\activate**

****

* + สำหรับ macOS/Linux:

**  
python3 -m venv venv**

**source venv/bin/activate**

* + ****
  + สังเกต: เมื่อคำสั่งสำเร็จ จะมีคำว่า (venv) ปรากฏหน้าบรรทัดคำสั่ง

### 3. การติดตั้งไลบรารีที่จำเป็น (Install Dependencies)

ติดตั้ง Django และไลบรารีเสริมต่างๆ ผ่านคำสั่ง pip:

**pip install django psycopg2-binary django-tailwind django-browser-reload**

****(หากมีไฟล์ requirements.txt สามารถใช้คำสั่ง pip install -r requirements.txt)

### 4. การติดตั้ง Tailwind CSS Dependencies

ระบบมีการใช้ Tailwind CSS ในการตกแต่งหน้าตาเว็บไซต์ จำเป็นต้องติดตั้ง Node Modules ผ่านแอปพลิเคชัน theme ที่ตั้งค่าไว้

**python manage.py tailwind install**

### ****5. การตั้งค่าการเชื่อมต่อฐานข้อมูล

เปิดไฟล์ settings.py และตรวจสอบส่วน DATABASES ให้ตรงกับเครื่องที่ติดตั้ง:

**DATABASES = {**

**'default': {**

**'ENGINE': 'django.db.backends.postgresql',**

**'NAME': 'boardgame\_db', # ชื่อฐานข้อมูลที่สร้างในข้อ 1**

**'USER': 'postgres', # ชื่อผู้ใช้ PostgreSQL ของคุณ**

**'PASSWORD': 'your\_password', # รหัสผ่าน PostgreSQL ของคุณ**

**'HOST': 'localhost',**

**'PORT': '5432',**

**}**

**}**

### ****6. การสร้างตารางในฐานข้อมูล (Migration)

ทำการแปลงโครงสร้าง Model (เช่น User, Booking, Game) ให้เป็นตารางจริงในฐานข้อมูล

**python manage.py makemigrations**

**python manage.py migrate**

### ****7. การสร้างบัญชีผู้ดูแลระบบ (Create Superuser)

สร้างบัญชีสำหรับเข้าใช้งานส่วนจัดการระบบ (Admin Dashboard)

**python manage.py createsuperuser**

****ระบบจะให้กรอก Username, Email, และ Password ตามลำดับ

## ข.3 การเริ่มใช้งานระบบ (Running the System)

ในการรันระบบเพื่อใช้งาน จะต้องเปิด Terminal 2 หน้าต่าง เพื่อรัน Django Server และ Tailwind Process ควบคู่กัน

หน้าต่างที่ 1: รัน Django Server

**python manage.py runserver**

****หน้าต่างที่ 2: รัน Tailwind (สำหรับ Compile CSS แบบ Real-time) (จำเป็นเฉพาะกรณีที่มีการแก้ไขไฟล์ HTML/CSS หากใช้งานทั่วไปขั้นตอนนี้อาจข้ามได้ถ้าไฟล์ CSS ถูก build ไว้แล้ว)

**python manage.py tailwind start**

****การเข้าใช้งาน:

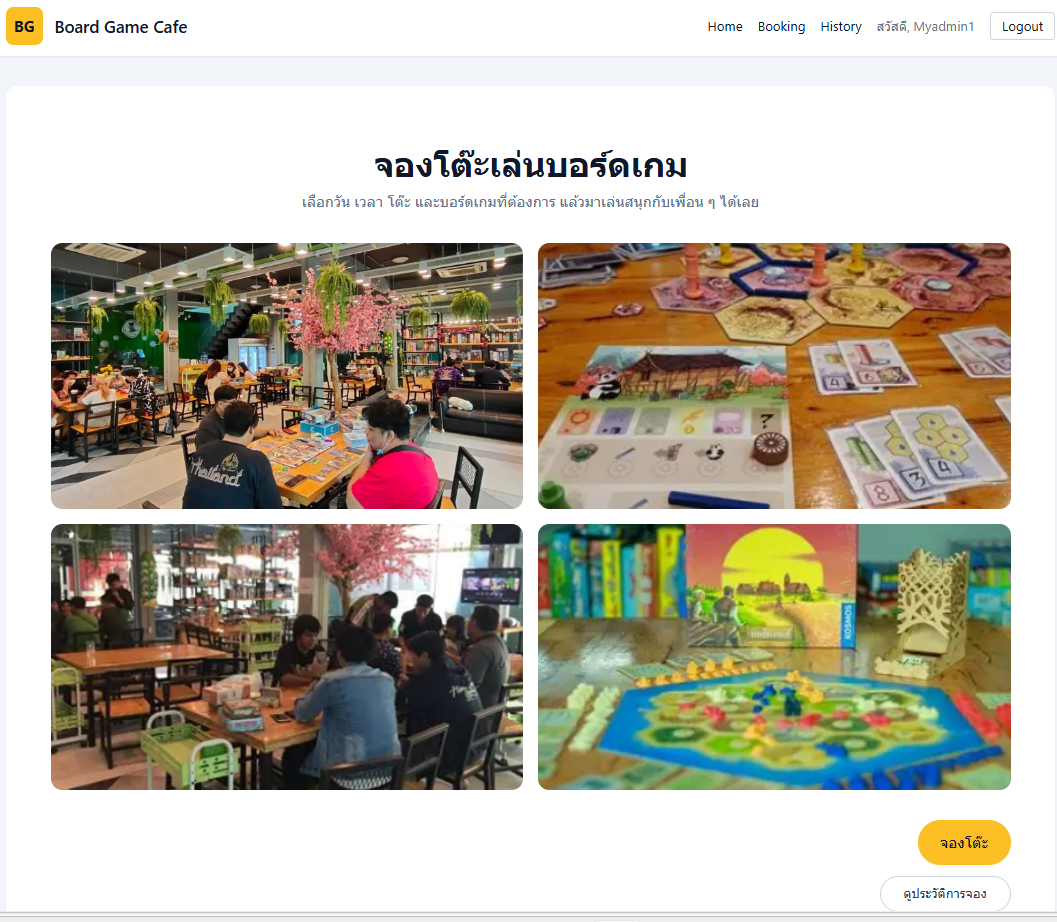
* เปิดเว็บเบราว์เซอร์แล้วไปที่: http://127.0.0.1:8000
* หน้าผู้ดูแลระบบ: http://127.0.0.1:8000/admin

# ภาคผนวก ค คู่มือการใช้งานของระบบ

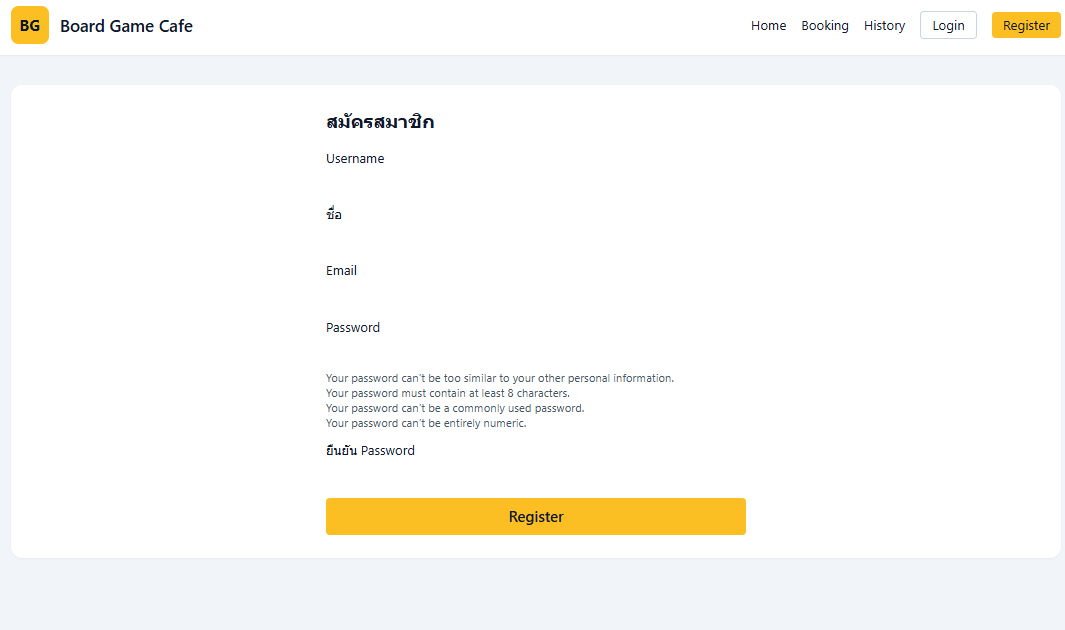
คู่มือการใช้งานระบบจองบอร์ดเกม (Board Game Booking System) แบ่งออกเป็น 2 ส่วนหลัก ได้แก่ ส่วนสำหรับผู้ใช้งานทั่วไป (Customer) และส่วนสำหรับผู้ดูแลระบบ (Administrator) โดยมีรายละเอียดขั้นตอนการใช้งานดังนี้

## ค.1 ส่วนของผู้ใช้งานทั่วไป (Customer)

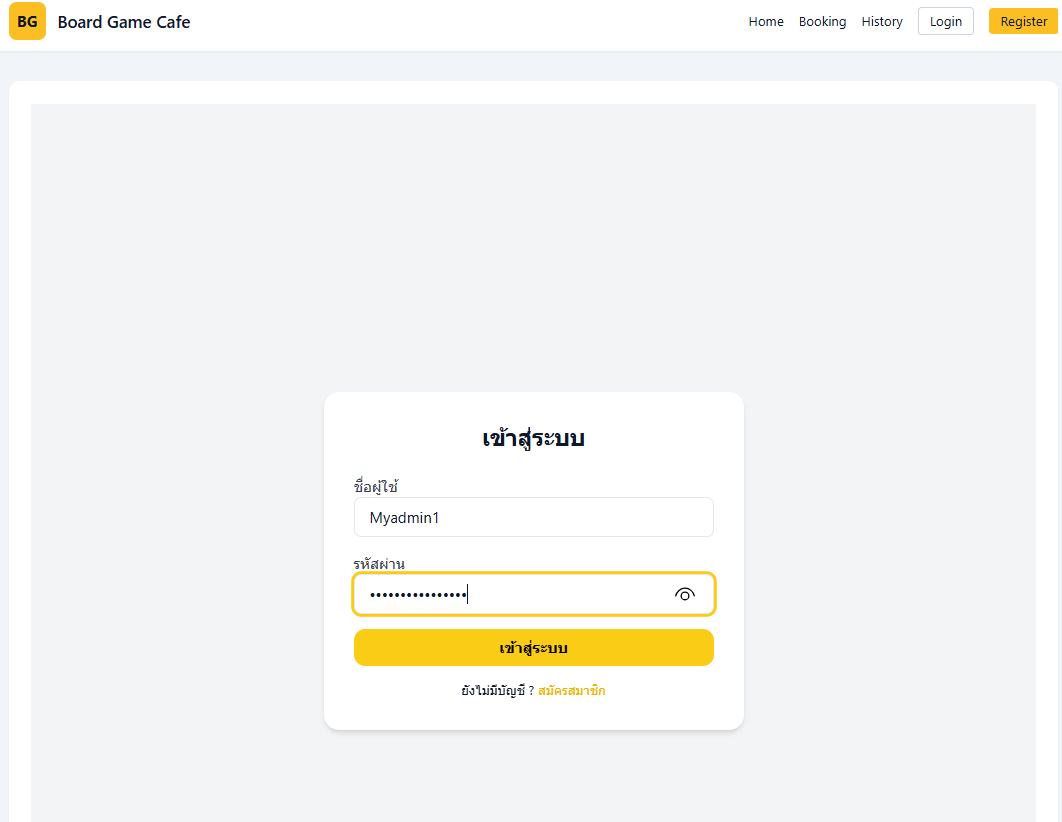
ผู้ใช้งานทั่วไปสามารถสมัครสมาชิก เข้าสู่ระบบ จองโต๊ะและบอร์ดเกม รวมถึงดูประวัติการจองได้

**1. การเข้าสู่ระบบและสมัครสมาชิก** เมื่อเข้าสู่เว็บไซต์ ผู้ใช้งานจะพบกับหน้าแรก หากยังไม่มีบัญชีให้ทำการสมัครสมาชิกก่อน

**ภาพที่ ค.1** หน้าจอหลัก (Home Page) และเมนูเข้าสู่ระบบ



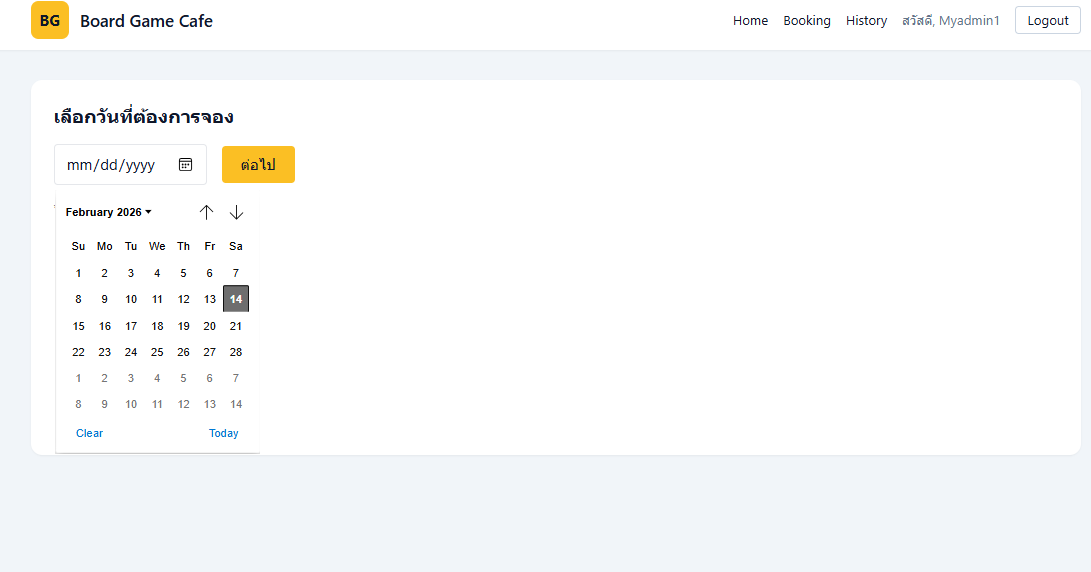
**ภาพที่ ค.2** หน้าจอสมัครสมาชิก (Register)

****

**ภาพที่ ค.3** หน้าจอเข้าสู่ระบบ (Login)

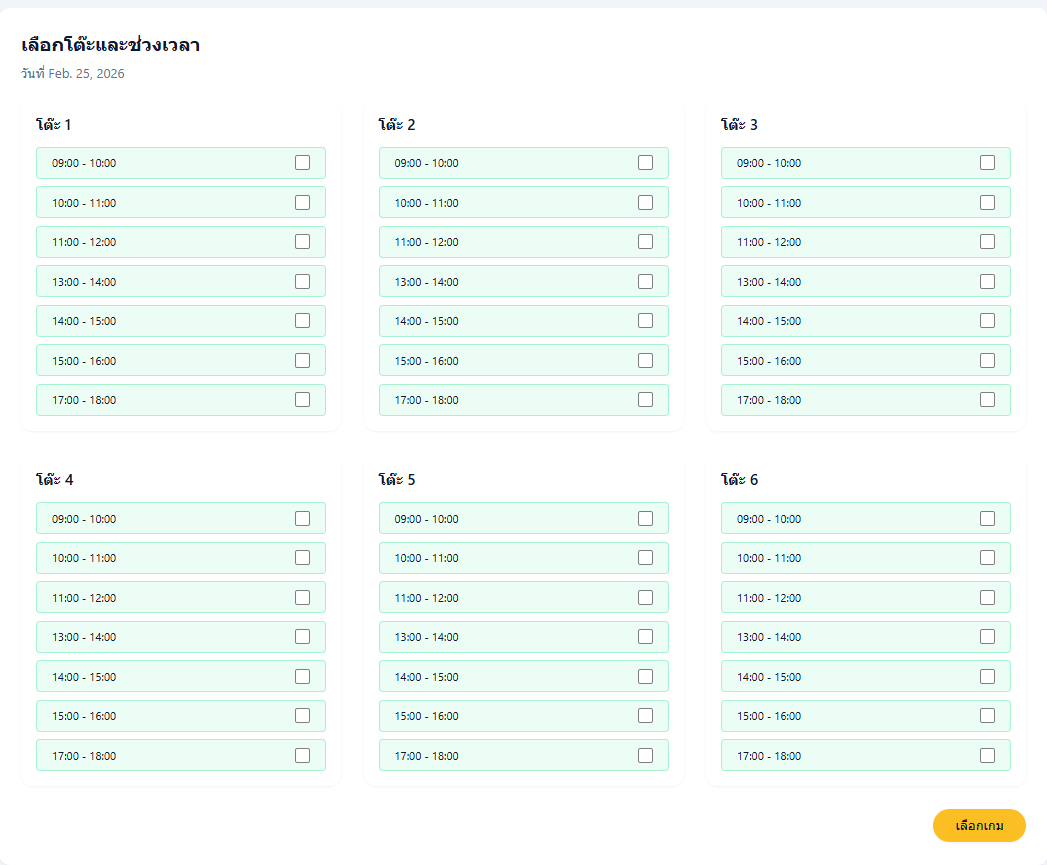
**2. ขั้นตอนการจอง (Booking Process)**

ระบบใช้ขั้นตอนการจองแบบ Step-by-step เพื่อความสะดวกของผู้ใช้งาน



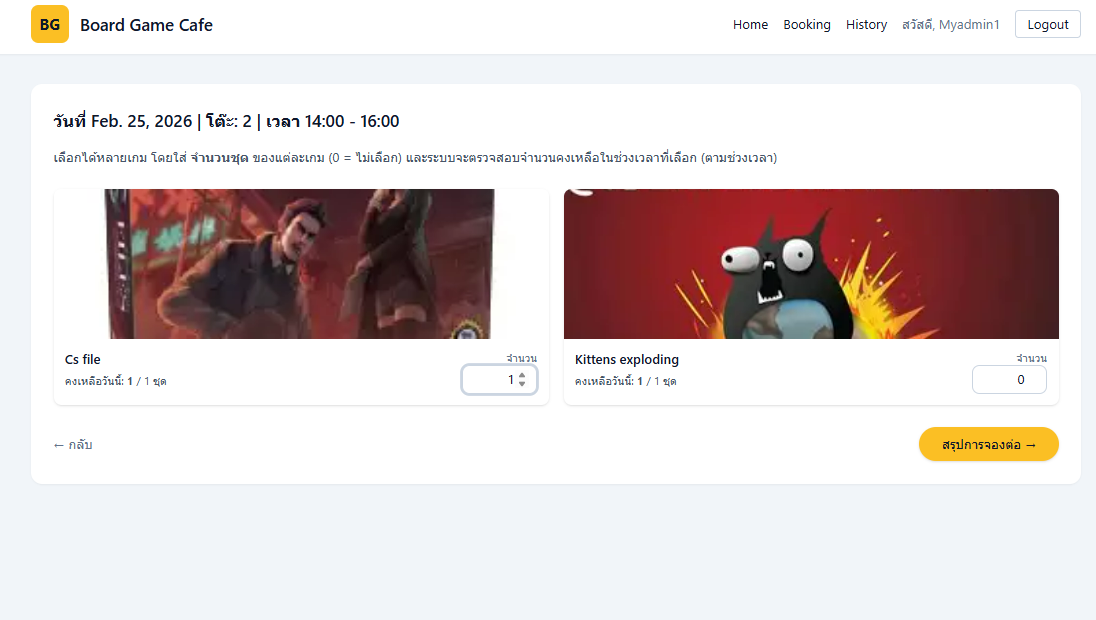
**ภาพที่ ค.4** ขั้นตอนที่ 1: การเลือกวันที่ (Select Date)

คำอธิบาย: แสดงหน้าจอที่มีช่องปฏิทินให้เลือกวันที่ต้องการใช้บริการ



**ภาพที่ ค.5** ขั้นตอนที่ 2: การเลือกช่วงเวลา (Select Timeslot)

คำอธิบาย: แสดงรายการช่วงเวลาที่มีให้บริการ และสถานะของโต๊ะว่าง



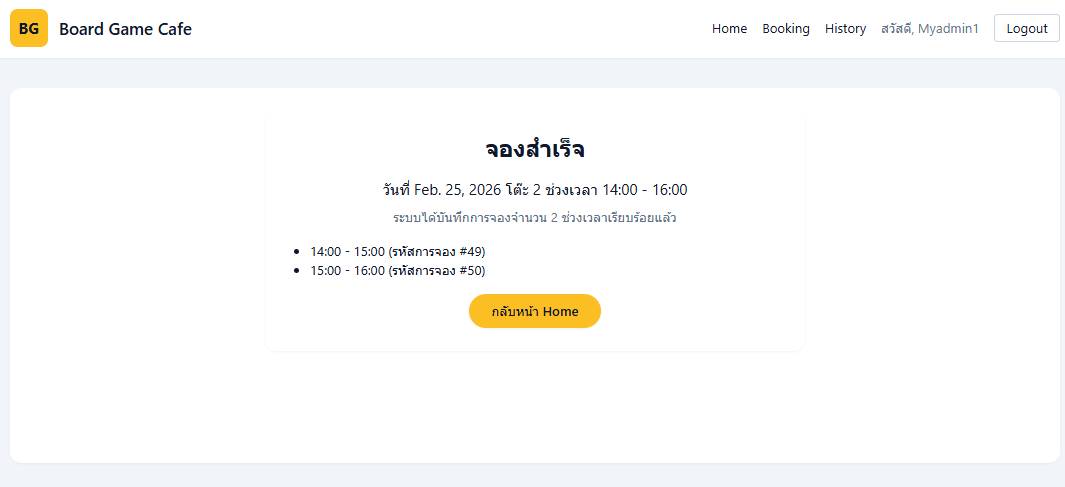
**ภาพที่ ค.6** ขั้นตอนที่ 3: การเลือกบอร์ดเกม (Select Game)

คำอธิบาย: แสดงรายการบอร์ดเกมพร้อมรูปภาพ และจำนวนสต็อกที่เหลือให้เลือก



**ภาพที่ ค.7** ขั้นตอนที่ 4: สรุปรายการจอง (Booking Summary)

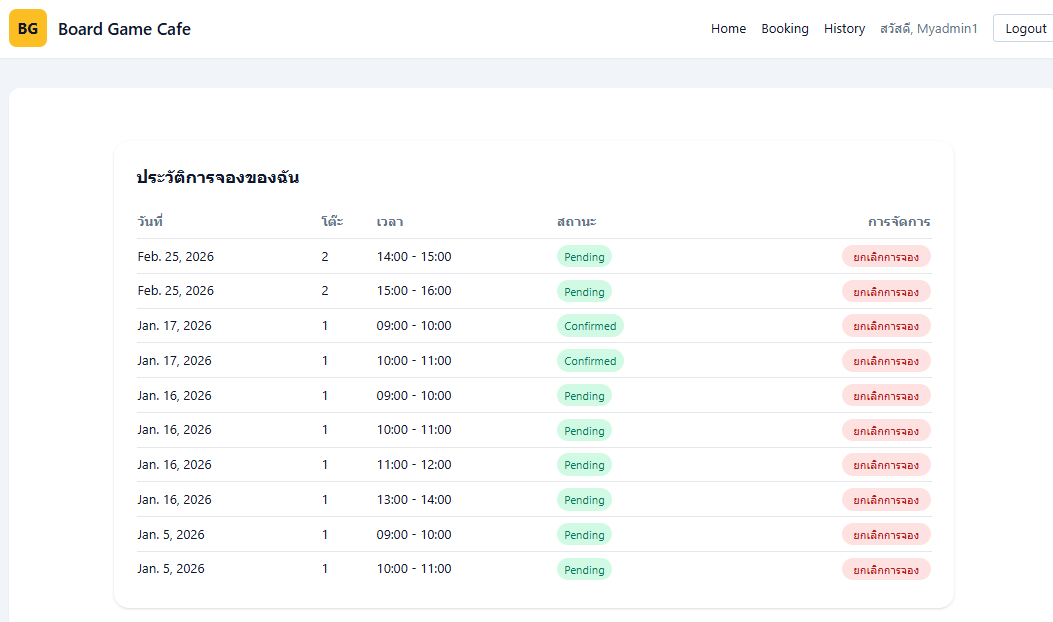
คำอธิบาย: แสดงรายละเอียดทั้งหมด (วันที่, เวลา, โต๊ะ, เกม) ให้ตรวจสอบก่อนกดยืนยัน



**ภาพที่ ค.8** หน้าจอแจ้งผลการจองสำเร็จ (Booking Success)

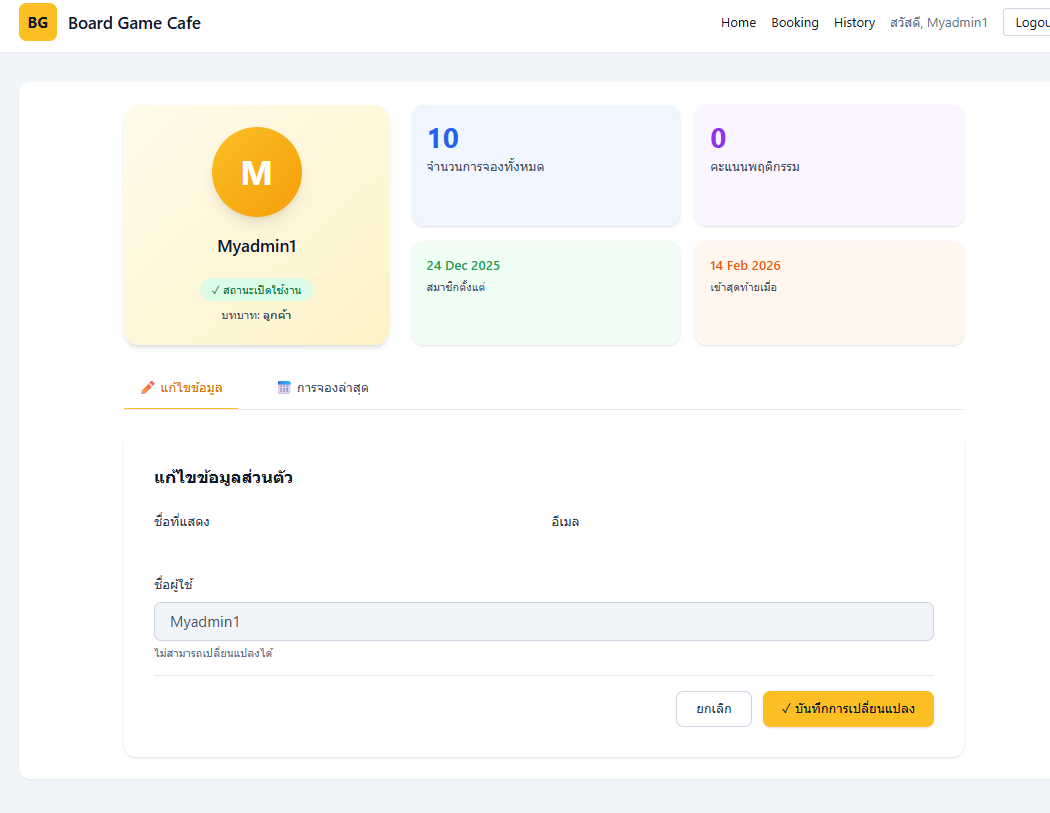
คำอธิบาย: หน้าจอแสดงข้อความขอบคุณหรือยืนยันว่าการจองเสร็จสมบูรณ์

**3. การจัดการข้อมูลส่วนตัวและประวัติ**

****

**ภาพที่ ค.9** หน้าจอประวัติการจอง (Booking History)

คำอธิบาย: แสดงตารางรายการที่เคยจอง สถานะ (Pending/Confirmed) และปุ่มยกเลิก



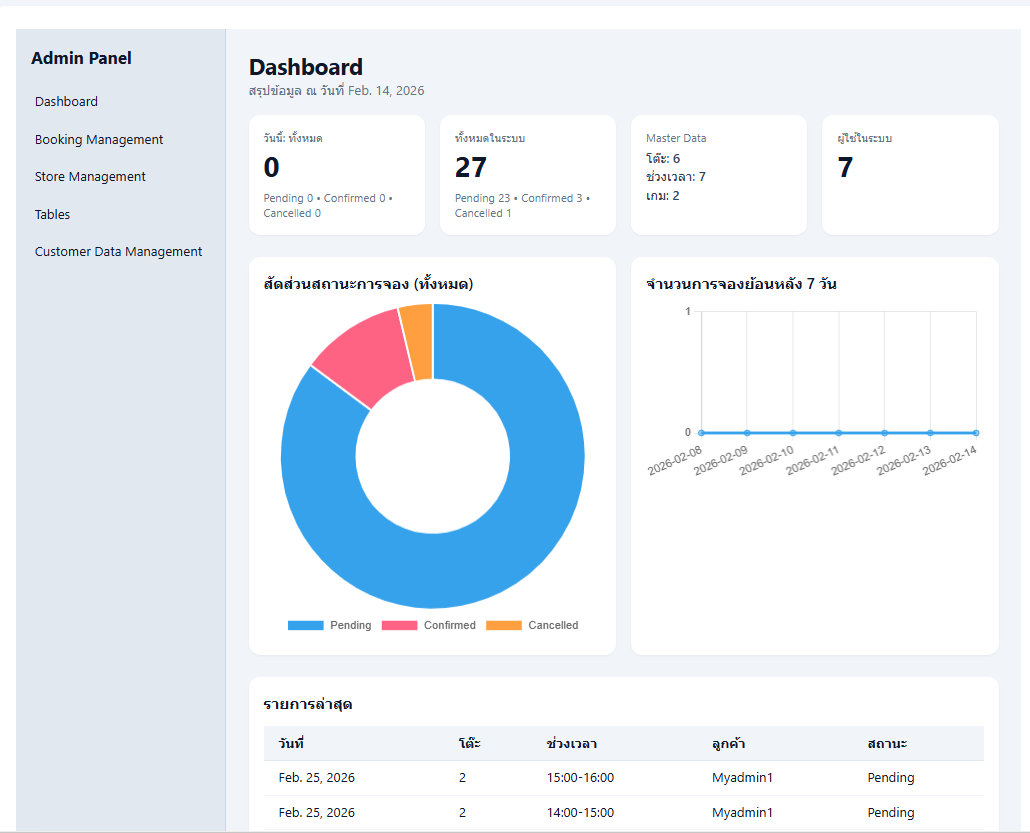
**ภาพที่ ค.10** หน้าจอแก้ไขโปรไฟล์ (User Profile)

คำอธิบาย: แสดงฟอร์มแก้ไขชื่อ, อีเมล และแสดงสถิติการจองของตนเอง

## ค.2 ส่วนของผู้ดูแลระบบ (Administrator)

ผู้ดูแลระบบสามารถเข้าถึง Dashboard และจัดการข้อมูลต่าง ๆ ผ่านเมนู Admin ที่กำหนดไว้

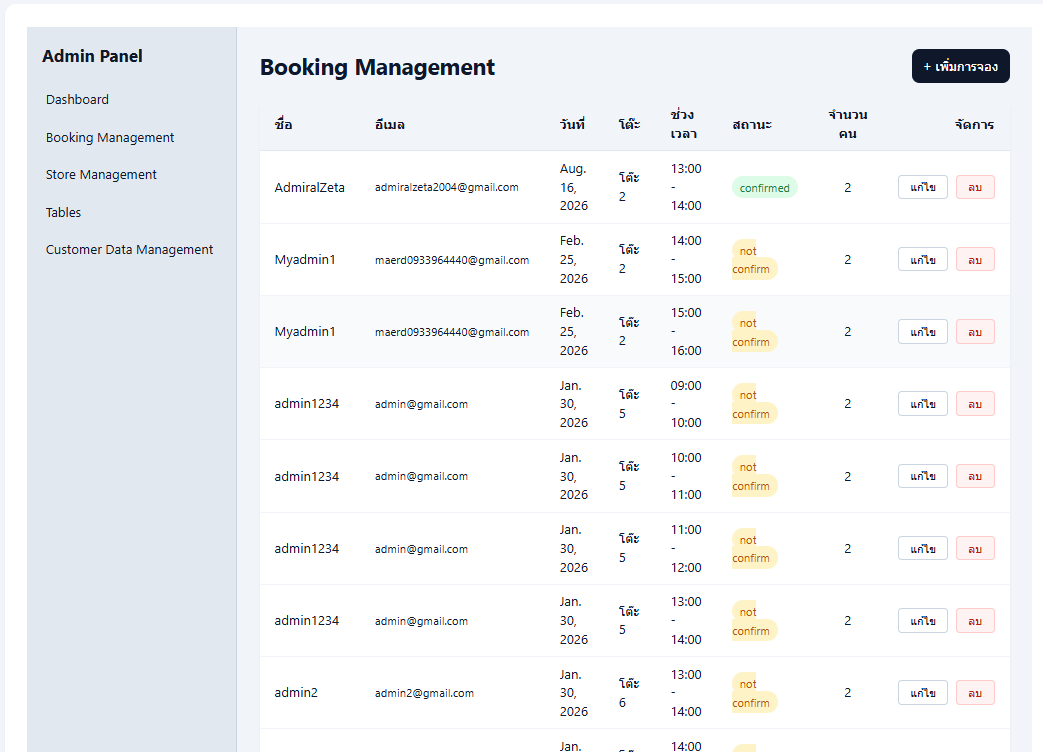
**1. ภาพรวมระบบ (Dashboard)**

****

**ภาพที่ ค.11** หน้าจอแดชบอร์ด (Admin Dashboard)

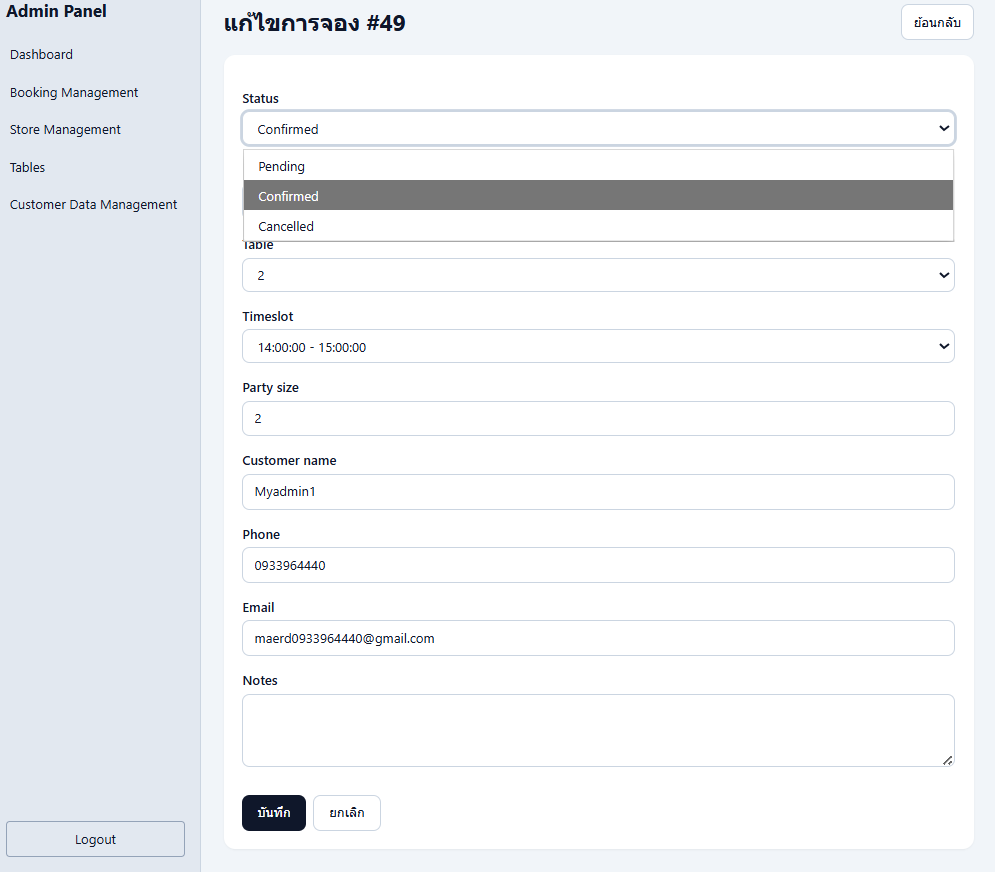
คำอธิบาย: แสดงกราฟหรือตารางสรุปสถิติยอดการจองรายวัน

**2. การจัดการการจอง (Booking Management)**

****

**ภาพที่ ค.12** หน้าจอรายการการจองทั้งหมด (Booking List)

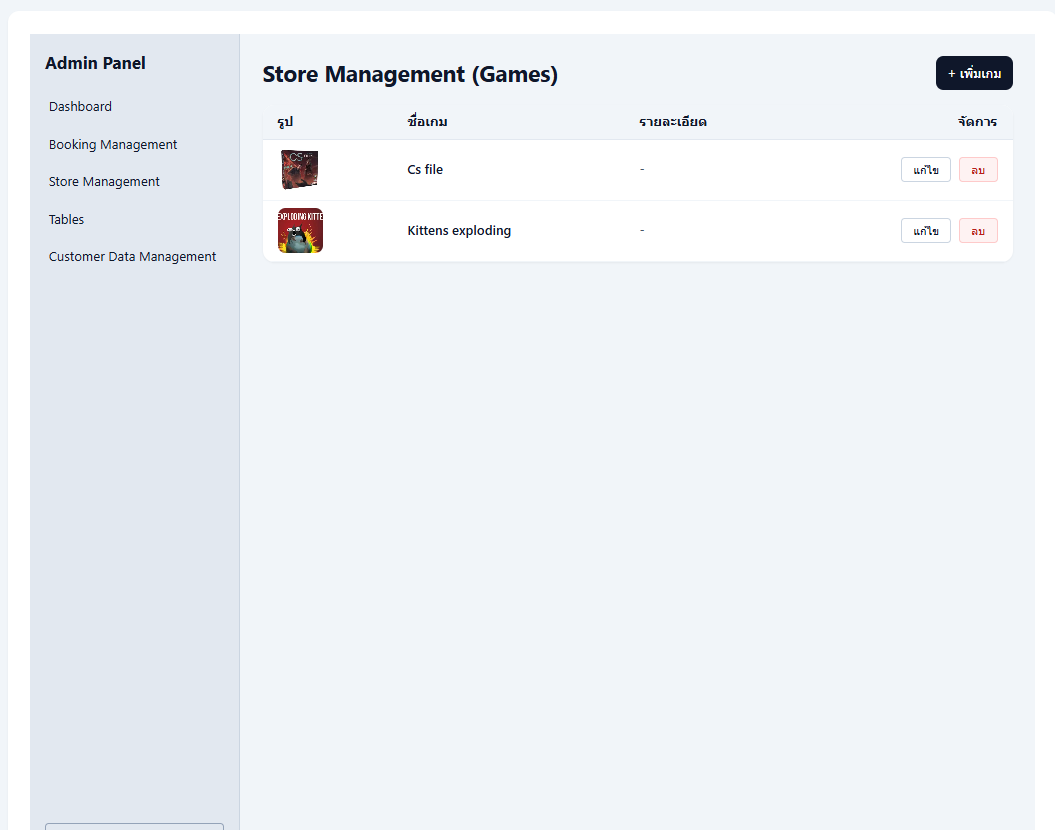
คำอธิบาย: ตารางแสดงรายชื่อลูกค้าที่จองเข้ามา พร้อมสถานะ



**ภาพที่ ค.13** หน้าจอแก้ไข/เปลี่ยนสถานะการจอง

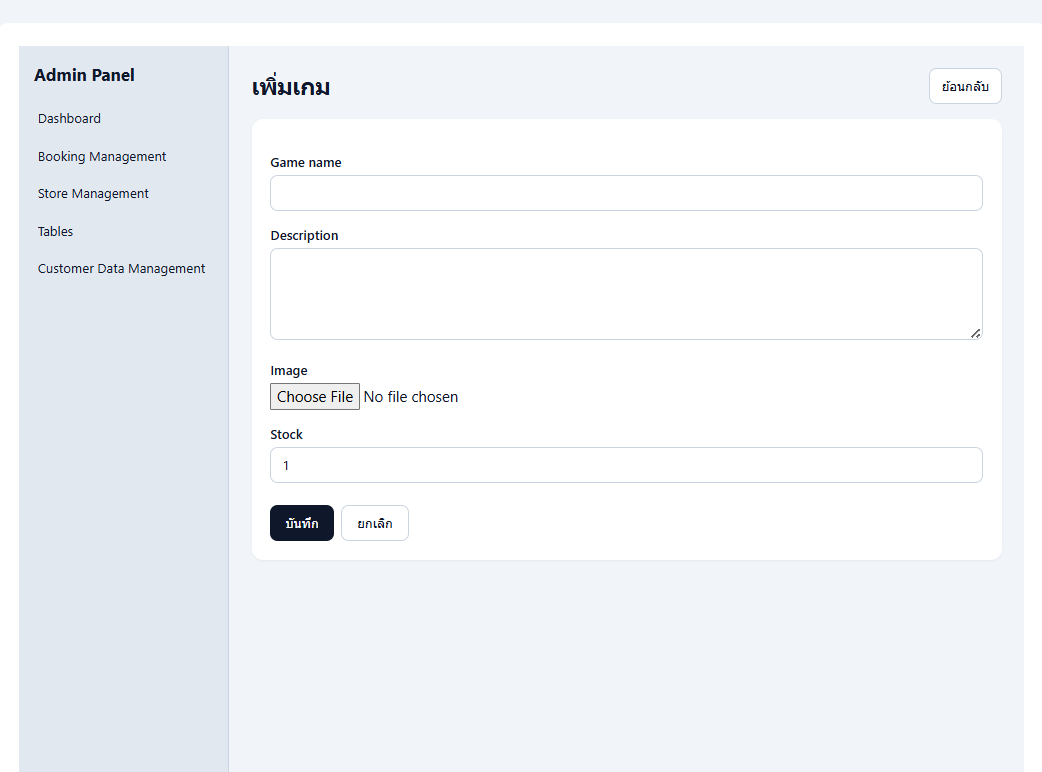
คำอธิบาย: ฟอร์มสำหรับแอดมินเพื่อเปลี่ยนสถานะเป็น Confirmed หรือ Cancelled

**3. การจัดการทรัพยากร (Resource Management)**

****

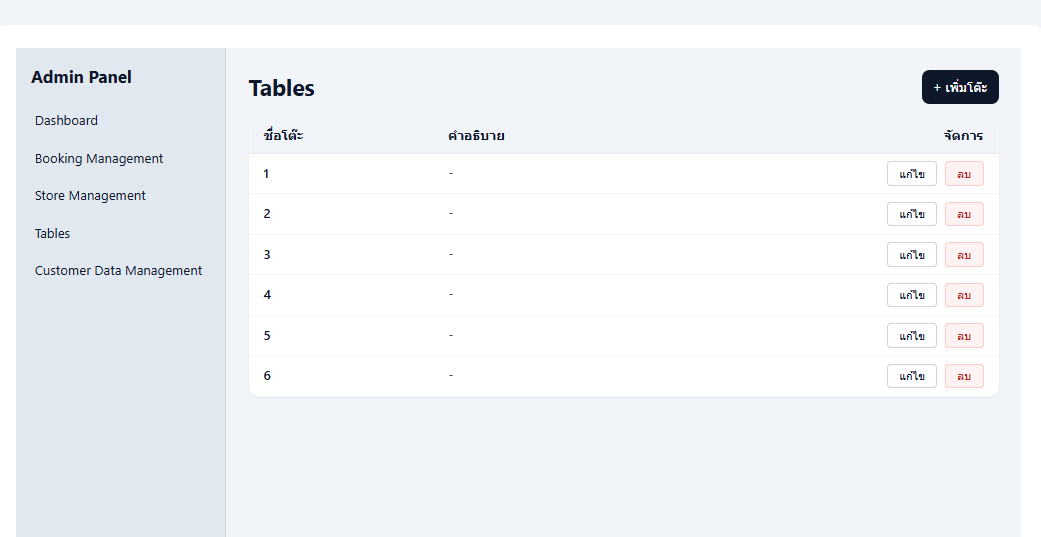
**ภาพที่ ค.14** หน้าจอจัดการคลังเกม (Game Management)

คำอธิบาย: รายชื่อเกมทั้งหมด มีปุ่มเพิ่ม/ลบ/แก้ไข



**ภาพที่ ค.15** หน้าจอเพิ่มบอร์ดเกมใหม่ (Add Game Form)

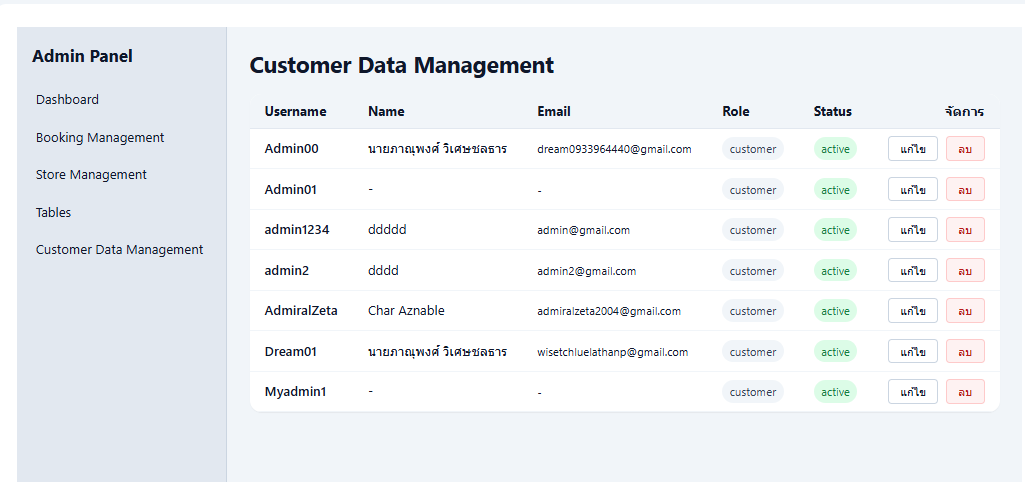
คำอธิบาย: ฟอร์มกรอกชื่อเกม รายละเอียด อัปโหลดรูป และใส่จำนวนสต็อก



**ภาพที่ ค.16** หน้าจอจัดการข้อมูลโต๊ะ (Table )

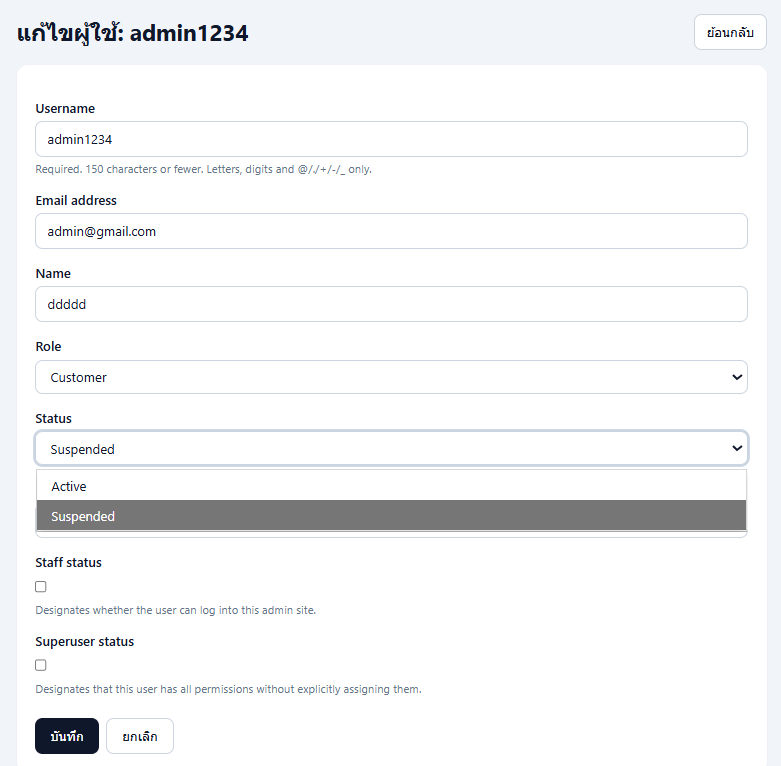
คำอธิบาย: หน้าจอสำหรับตั้งค่าว่าร้านมีกี่โต๊ะ และเปิดรอบเวลากี่โมงบ้าง

**4. การจัดการผู้ใช้งาน (User Management)**

****

**ภาพที่ ค.17** หน้าจอรายชื่อสมาชิก (Customer List)

คำอธิบาย: แสดงรายชื่อสมาชิกทั้งหมด



**ภาพที่ ค.18** หน้าจอปรับสถานะผู้ใช้งาน

คำอธิบาย: การกดระงับการใช้งานสมาชิกที่ทำผิดกฎ

# 

# ประวัติผู้พัฒนา

ชื่อ-นามสกุล : นายภาณุพงศ์ วิเศษชลธาร

รหัสประจำตัวนักศึกษา : 66114540520

วันเกิด : 24 มีนาคม 2548

ที่อยู่ที่สามารถติดต่อได้ : บ้านเลขที่ 111/38 หมู่ที่ 5 หมู่บ้าน,มงคลสมบัติ ตำบลหนองครก อำเภอเมืองศรีสะเกษ จังหวัดศรีสะเกษ -33000

เบอร์โทรศัพท์ : 093-396-44440

อีเมล : phanuphong.wi.66@ubu.ac.th

ระดับมัธยมต้น : โรงเรียนศรีสะเกษวิทยาลัย จังหวัดศรีสะเกษ

ระดับมัธยมปลาย : โรงเรียนศรีสะเกษวิทยาลัย จังหวัดศรีสะเกษ

ระดับอุดมศึกษา : ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ สาขาวิทยาการข้อมูลและนวัตกรรมซอฟต์แวร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี