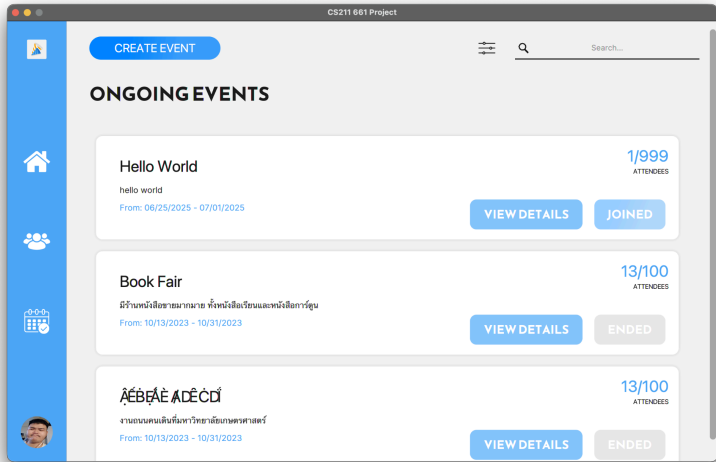


WK1 - Code Meets Design

ชื่อ-นามสกุล: ศิริสุข ทานธรรม รหัสสิต 6610402230 หมู่เรียน 1

หัวข้อ	คำอธิบาย
ชื่อระบบ (ไทย)	โปรแกรมการจัดการอีเวนต์
ชื่อระบบ (อังกฤษ)	EventFiesta
ชื่อผู้พัฒนา	1.นายณัฐภัทร ยัมละมัย 6510405504 2.นายธีรภัทร อนันต์ไพศาลสิน 6510405580 3.นายธีรวัฒน์ กิวยประเสริฐ 6510405598 4.นายณพภูฏ ฐานะปฏิพล 6510405610
ภาพหน้าจอ	 <p>The screenshot shows a web application interface for 'ONGOING EVENTS'. It features a sidebar with navigation icons (home, events, users, calendar) and a main content area with three event cards. The first card is 'Hello World' with 1/999 attendees, the second is 'Book Fair' with 13/100 attendees, and the third is 'ÀÉBÆÆ ADEÖÖŦ' with 13/100 attendees. Each card has 'VIEW DETAILS' and 'JOINED' or 'ENDED' buttons.</p>
วัตถุประสงค์	เพื่อให้การจัดการ event ให้ สามารถทำได้ผ่าน application อำนวยความสะดวกให้ผู้จัดกิจกรรม และผู้ลงทะเบียนกิจกรรมสามารถตรวจสอบรายละเอียดและกำหนดการของกิจกรรมที่ลงทะเบียน
บทบาทผู้ใช้งาน	ADMIN (ผู้ดูแลระบบ) NORMAL (account ของ user ปกติ)
ข้อกำหนด	ADMIN <ul style="list-style-type: none"> - ไม่มีการสร้างบัญชีผู้ดูแลระบบ และมีผู้ดูแลระบบเริ่มต้น - จัดการรหัสผ่านของผู้ดูแลระบบได้ - สามารถดูรายชื่อ account ที่ login ล่าสุดได้ NORMAL <ul style="list-style-type: none"> - สามารถลงทะเบียนและสมัครสมาชิกได้ - สามารถแก้ไขรหัสผ่านและรูปภาพโปรไฟล์ได้ - สามารถดูและค้นหารายการอีเวนต์ได้ - สามารถดูรายการอีเวนต์ที่เคยลงทะเบียน - สามารถเช็คข้อมูลตารางกิจกรรมรายอีเวนต์ได้ - สามารถสร้างและจัดการอีเวนต์ของตนเองได้ - สามารถเปิดรับสมัครหรือลงทะเบียนเป็นทีมผู้จัดการอีเวนต์ได้ - หากเป็นผู้ร่วมทีมจัดอีเวนต์สามารถแสดงความคิดเห็นภายในอีเวนต์ได้

ปัญหาที่พบ

Paht : src/main/java/cs211/project/controllers/MainPageController.java

ปัญหาที่ 1: duplicate code ที่มีการทำงานที่เหมือนกัน

ปัญหาที่ 2: มีการกำหนด path ไปยัง file ที่เป็น static

```
try {  
    FileInputStream imgStream = new FileInputStream(imagePath);  
    Image img = new Image(imgStream);  
    imgStream.close();  
    imgCircle.setFill(new ImagePattern(img));  
} catch (FileNotFoundException | InvalidPathException | NullPointerException e) {  
    Image img = new Image(getClass().getResourceAsStream("/cs211/project/views/img/default_user_profile.png"));  
    imgCircle.setFill(new ImagePattern(img));  
} catch (IllegalArgumentException e) {  
    Image img = new Image(getClass().getResourceAsStream("/cs211/project/views/img/default_user_profile.png"));  
    imgCircle.setFill(new ImagePattern(img));  
} catch (IOException e) {  
    throw new RuntimeException(e);  
}
```

ผลกระทบ

- ทำให้โครงสร้างการทำงานของ logic มีความซับซ้อนและไม่กระชับ
- หากต้องการแก้ไขโค้ดทำได้ยากกว่าเพราะมีความซ้ำซ้อนทำให้ต้องแก้หลายที่
- โค้ดที่มีความซ้ำซ้อนอาจทำให้เกิด bugs ได้เนื่องจากโปรแกรมเมอร์อาจลืมหรือแก้ไขให้ครบทุกจุด
- การกำหนด path แบบ static ซ้ำซ้อนอาจทำให้การแก้ไขมีความลำบากและกรณีที่ไม่สามารถค้นหาไฟล์ได้ก็จะเกิด FileNotFoundException

การแก้ไข

- รวม logic ของโปรแกรมที่มีความซ้ำซ้อนเข้าด้วยกัน
- static path แก้ไขด้วยการใช้ class ทำทำงานด้าน DataSource เพื่อตรวจไฟล์
- และใช้งาน Dependices Injection design pattern เพื่อให้โค้ดกระชับและมีความ Dynamic
- สามารถแก้ไขการ handle error ให้มีความชัดเจนมากขึ้นได้ด้วยการสร้าง class เพื่อ handle error ที่ต้องการแสดงผลบน application
- และ class ที่เกี่ยวข้องกับการทำงานส่วนอื่นก็จัดการ handle error เฉพาะส่วนนั้นๆ

Paht : src/main/java/cs211/project/controllers/CreateEventController.java

ปัญหาที่ 3: มีการกำหนดค่าตัวแปร local ที่ไม่มีการเปลี่ยนแปลงและมีข้อมูลแบบ static

```
ObservableList<String> eventCategories = FXCollections.observableArrayList(  
    "Professional Events",  
    "Social Events",
```

	<div>"Cultural Events", "Entertainment Events", "Educational Events", "Fundraising Events", "Sports and Recreational Events", "Special Events", "Community Events", "Expo and Convention Events", "Arts and Creativity Events", "Food and Beverage Events", "Technology Events", "Wellness Events", "Civic and Government Events", "Performance Arts Events", "Religious and Spiritual Events", "Awards and Recognition Events", "Tourism and Promotion Events", "Others"</div> <div>);</div> <div>ผลกระทบ</div> <div><ul style="list-style-type: none">- ทำให้เกิดความซับซ้อนเกินความจำเป็นและไม่คล่องตัวต่อการอ่าน- หาก class มีการสร้างหลาย object code ส่วนนี้จะกิน memory เมื่อเรียกใช้หลายๆ object พร้อมๆกัน- ยากต่อการเปลี่ยนแปลงแก้ไขเพราะเป็นการ hardcode โปรแกรมเมอร์จะพลาดในการแก้ไขได้</div> <div>การแก้ไข</div> <div><ul style="list-style-type: none">- เปลี่ยนเป็น static variable เพื่อให้มีการ allocation เพียงครั้งเดียวต่อเมื่อสร้าง object หลายๆ object ก็ยังคงใช้งาน variable ตัวเดียวกันทำให้ประหยัดพื้นที่- เนื่องจากไม่ได้มีการเปลี่ยนแปลงสามารถใช้งาน const เพื่อกำหนดได้- แก้ไขการทำ hardcode ด้วยการสร้าง datasource เพื่อใช้ read ได้ผ่าน file แล้ว assign ตัวแปรด้วย static {} scope เพื่อให้มีการทำงานเพียงครั้งเดียวเมื่อ load class</div>
	<div>Paht :</div> <div><ul style="list-style-type: none">- src/main/java/cs211/project/models/Event.java- src/main/java/cs211/project/models/EventSchedule.java- src/main/java/cs211/project/models/Team.java- src/main/java/cs211/project/models/TeamSchedule.java</div> <div>ปัญหาที่ 4: class ที่ไม่มีการทำงานนอกเหนือจาก constructor</div> <div>ปัญหาที่ 5: การทำงานของ EventSchedule และ TeamSchedule มีความคล้ายกัน</div>

	<pre> public class EventSchedule { private String time ; private String nameAct; private String detail; private long idRecord; private static long nextId = 1; private String eventId ; public EventSchedule(String time,String nameAct, String detail,String eventId){ this.time = time; this.nameAct = nameAct; this.detail = detail; this.idRecord = nextId; this.eventId = eventId; nextId++; } } getter setter . . . public class TeamSchedule { private String nameAct_teamSchedule ; private String detail; private String status; private String idRecord; public static long nextTeamScheduleId; private String eventId ; private String teamId; public TeamSchedule(String idRecord, String nameAct_teamSchedule, String detail, String status, String eventId, String teamId) { this.nameAct_teamSchedule = nameAct_teamSchedule; this.detail = detail; this.status = status; this.idRecord = idRecord; this.eventId = eventId; this.teamId = teamId; nextTeamScheduleId++; } } getter setter . . . </pre> <p>ผลกระทบ</p> <ul style="list-style-type: none"> - ทำให้มีการสร้าง class โดยไม่จำเป็นและยากต่อการจัดการเพราะ class ไม่มี implementation - คลาสที่ไม่มี implementation เช่น EventSchedule และ TeamSchedule มีคลาสหลักอยู่แล้วคือ Event และ Team ทำให้เวลาต้องการเรียกใช้ทำได้ซับซ้อนเพราะหากต้องการใช้สองคลาสต้องเสียเวลามาสร้างอีกคลาส
--	---

	<ul style="list-style-type: none">- สองคลาสที่ีการทำงานคล้ายกันทำให้หาความต้องการเรียกใช้งาน class ที่ต้องการใช้งานมีความซับซ้อนเพราะต้อง handle ทั้งสองคลาสนี้ <p>การแก้ไข</p> <ul style="list-style-type: none">- เนื่องจาก EventSchedule และ TeamSchedule มีคลาสหลักอยู่แล้วคือ Event และ Team และทั้งสองคลาสนี้ไม่ได้มี implementation อะไรจึงควรนำไป implement ร่วมกันในคลาส Event และ Team เพื่อไม่ให้เกิดความซ้ำซ้อน- internal logic ของ EventSchedule และ TeamSchedule มีความคล้ายกันสามารถใช้งาน Interface Schedulable เพื่อให้สามารถนำไปใช้งานได้ง่ายและมีจุดร่วมกัน สร้างคลาสเพื่อ handle logic ได้ง่ายขึ้นด้วยการทำ dependencies injection และจัดการ logic ภายในได้ใน method เดียว
--	---