**GssSport**

ระบบจัดการจองสนามกีฬา

A screenshot of a computer

Description automatically generated

**รายละเอียด**

ระบบที่ออกแบบมาเพื่อให้เจ้าของกิจการสามารถสำรวจข้อมูลและวิเคราะห์ผลได้อย่างสะดวกสบาย โดยสามารถค้นหาสถานที่ที่ผู้ใช้ชื่นชอบที่สุด รวมถึงการติดตามยอดขาย ยอดจอง และสถานะต่างๆ ได้อย่างแม่นยำ นอกจากนี้ยังมีฟังก์ชันการกรองข้อมูลตามความต้องการของผู้ใช้ เช่น การกรองสถานะ การจัดลำดับตามวันเวลา หรือยอดเงินชำระ นอกจากนี้ ผู้ใช้ยังสามารถค้นหาสนามกีฬาตามประเภทที่ต้องการ และตรวจสอบความคิดเห็นของผู้คนเกี่ยวกับสถานที่ต่างๆ ได้อีกด้วย

**โครงสร้าง Collections**

A group of white papers with text

Description automatically generated

**Facilities**

ใช้สำหรับเก็บข้อมูลของสถานที่ที่สามารถจองได้ โดยแต่ละเอกสารจะมี:

* **available\_times**: ช่วงเวลาที่สถานที่เปิดให้จองในรูปแบบอาเรย์
* **location**: ข้อมูลที่ตั้งของสถานที่ เช่น:
  + **city**: เมืองที่สถานที่ตั้งอยู่
  + **coordinates**: ละติจูดและลองจิจูดสำหรับการทำแผนที่
* **name**: ชื่อของสถานที่
* **price\_per\_hour**: โครงสร้างราคาต่อชั่วโมง แยกตามประเภทกีฬา (เช่น แบดมินตัน, ปิงปอง)
* **rating**: คะแนนเฉลี่ยของสถานที่
* **sports\_type**: ประเภทกีฬาที่สามารถเล่นได้ในสถานที่นั้น

**Accounts**  
ใช้สำหรับเก็บข้อมูลของผู้ใช้งานแอปพลิเคชัน โดยในแต่ละเอกสารจะมีข้อมูลพื้นฐานของผู้ใช้งานและประวัติการจองล่าสุด

* **email**: อีเมลของผู้ใช้งาน
* **is\_member**: บ่งบอกว่าผู้ใช้งานเป็นสมาชิกหรือไม่
* **name**: ชื่อเต็มของผู้ใช้งาน
* **phone**: เบอร์โทรศัพท์ของผู้ใช้งาน
* **recent\_bookings**: รายการจองล่าสุดในรูปแบบอาเรย์ โดยแต่ละรายการจะมี ID การจอง วันที่ ชื่อสถานที่ และช่วงเวลาที่จอง

**Bookings**  
ใช้สำหรับเก็บข้อมูลการจองแต่ละครั้ง โดยในแต่ละเอกสารจะบันทึกรายละเอียดของการจอง เช่น

* **date**: วันที่ทำการจอง
* **facility\_id**: รหัสเฉพาะของสถานที่ที่จอง
* **facility\_name**: ชื่อของสถานที่ที่จอง
* **price**: ราคาทั้งหมดของการจอง
* **sports\_type**: ประเภทของกีฬา (เช่น แบดมินตัน)
* **status**: สถานะของการจอง (เช่น ยืนยันแล้ว, รอดำเนินการ)
* **time\_slot**: ช่วงเวลาที่จอง
* **user\_name**: ชื่อของผู้ที่ทำการจอง

**Review (รีวิว)**  
ใช้สำหรับเก็บรีวิวของสถานที่จากผู้ใช้งาน โดยแต่ละเอกสารจะมี:

* **comment**: ความคิดเห็นหรือคำติชมเกี่ยวกับสถานที่
* **created\_at**: เวลาที่รีวิวถูกสร้างขึ้น
* **facility\_id**: รหัสเฉพาะของสถานที่ที่ถูกรรีวิว
* **facility\_name**: ชื่อของสถานที่ที่ถูกรรีวิว
* **rating**: คะแนนที่ผู้ใช้งานให้
* **user\_id**: รหัสเฉพาะของผู้ที่รีวิว
* **user\_name**: ชื่อของผู้ที่รีวิว

**ความสัมพันธ์ระหว่าง Collection**

* **Accounts → Bookings**: ผู้ใช้งานใน **Accounts** สามารถมีการจองหลายครั้งที่ถูกเก็บไว้ใน **Bookings** โดยเชื่อมต่อกันผ่าน user\_name
* **Bookings → Facilities**: การจองแต่ละครั้งจะอ้างอิงถึงสถานที่ใน **Facilities** ผ่าน facility\_id
* **Facilities → Review**: รีวิวจะเชื่อมโยงกับสถานที่ใน **Facilities** ผ่าน facility\_id เพื่อแสดงคำติชมและคะแนนสำหรับสถานที่นั้น

**Design Pattern**

**ใน app นี้มีการใช้ pattern ที่ยืดหยุ่นแล้วก็หลากหลาย ได้แก่**

Polymorphic Pattern และ Attribute Pattern การเก็บข้อมูลหลายรูปแบบไว้ใน Collection เดียว & ใช้ Attribute Pattern เพื่อจัดเก็บประเภทกีฬาและราคาต่อชั่วโมง :

**A screenshot of a computer

Description automatically generated**

* **Facilities เก็บข้อมูลที่แตกต่างกันในฟิลด์ เช่น price\_per\_hour ซึ่งเป็น Map ของประเภทกีฬาและราคา**
* **price\_per\_hour ใช้ Attribute Pattern เพื่อจัดเก็บประเภทกีฬาและราคาต่อชั่วโมง**

Outlier Pattern การย้ายฟิลด์ขนาดใหญ่หรือฟิลด์ที่ไม่พบบ่อยออกไปยัง Collection แยก

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

มีการเก็บฟิลด์ที่เกี่ยวกับรีวิว (Review) ใน Collection แยกจาก Facilities

Computed Pattern การเก็บข้อมูลที่คำนวณหรือประมวลผลแล้ว

**A screenshot of a computer

Description automatically generated**

ฟิลด์ rating ใน Facilities เป็นค่าเฉลี่ยของคะแนนที่คำนวณมาจากข้อมูลรีวิว

Subset Pattern : การเก็บเฉพาะส่วนของข้อมูลในเอกสารหลักเพื่อเพิ่มประสิทธิภาพในการอ่าน

A screenshot of a computer

Description automatically generated

ฟิลด์ recent\_bookings ใน Accounts เก็บเฉพาะข้อมูลการจองล่าสุด แทนที่จะเก็บข้อมูลการจองทั้งหมด

Extended Reference Pattern: การฝังข้อมูลบางส่วนของเอกสารที่อ้างอิงในเอกสารหลักA screenshot of a black screen

Description automatically generated

ฟิลด์ facility\_name และ facility\_id ใน Bookings เป็น Extended Reference จาก Facilities

Queries ต่อไปนี้มีการใช้ในหลายส่วน เนื่องจากผมพยายามจะ implement firebase ให้เหมือนเขียน application จริงๆ โดยมาที่หน้าแรก

A screenshot of a computer

Description automatically generated

1.Aggregate stats ใช้กับไว้นับจำนวนหายอด status รอดำเนินการ ยอดจองทั้งหมดและผลรวมของยอดขายทั้งหมด

ดังนี้

Query:

async function getStats() {  
 const bookingsRef = await collection(*db*, "bookings");  
 statPending.value = (  
 await getCountFromServer(  
 query(bookingsRef, where("status", "==", "รอดำเนินการ")),  
 )  
 ).data().count;  
  
 const aggBooking = (  
 await getAggregateFromServer(bookingsRef, {  
 statSale: sum("price"),  
 statsBooking: count(),  
 })  
 ).data();  
  
 statBooking.value = aggBooking.statsBooking;  
 statSale.value = aggBooking.statSale;  
}

result:

A screenshot of a phone

Description automatically generated

2. query range today

ใช้สำหรับหารายการที่มีการจองในวันนี้โดยใช้ timestamp กำหนดช่วงระยะเวลาภายในวันปัจจุบัน

Query:

async function getBookingToday() {  
 const bookingsRef = collection(*db*, "bookings");  
 const startOfDay = new *Date*();  
 startOfDay.setHours(0, 0, 0, 0);  
 const endOfDay = new *Date*();  
 endOfDay.setHours(23, 59, 59, 999);  
  
 const qry = query(  
 bookingsRef,  
 where("date", ">=", startOfDay),  
 where("date", "<=", endOfDay),  
 );  
  
 const querySnapshot = await getDocs(qry);  
 bookingToday.value = querySnapshot.docs.map((doc) => ({  
 id: doc.id,  
 ...doc.data(),  
 }));  
}

result:

A screenshot of a computer

Description automatically generated

3.Top 5 Popular Facilities หา 5 อันดับแรกของสถานที่ที่ยอดนิยมที่สุด

Query:

async function getTop5Facilities() {  
 const facilitiesRef = await collection(*db*, "facilities");  
 const qry = query(facilitiesRef, orderBy("rating", "desc"), limit(5));  
 const querySnapshot = await getDocs(qry);  
  
 top5Facilities.value = querySnapshot.docs.map((doc) => ({  
 id: doc.id,  
 ...doc.data(),  
 }));  
}

result:

A screenshot of a sports schedule

Description automatically generated

4.Sort and filter bookings and facilities ค้นหา status และ และเรียงลำดับตามตัวเลือกที่ผู้ชายต้องการ โดยใช้การ query แบบ dynamic ด้วยการ spread array list queries และ ใช้ค่าร่วมกับ pathquery การบันทึก

4.1. bookings

A screenshot of a computer

Description automatically generated

A grey rectangular screen with white text

Description automatically generated

Query:

watch(() => {  
 if (filterStatus.value) {  
 router.push({  
 name: "booking",  
 query: { ...route.query, status: filterStatus.value },  
 });  
 } else {  
 router.push({  
 name: "booking",  
 query: { ...deleteField(route.query, "status") },  
 });  
 }  
 getBookings();  
}, [filterStatus.value]);  
  
watch(() => {  
 if (sortBy.value !== "date-desc") {  
 router.push({  
 name: "booking",  
 query: { ...route.query, sortBy: sortBy.value },  
 });  
 } else {  
 router.push({  
 name: "booking",  
 query: { ...deleteField(route.query, "sortBy") },  
 });  
 }  
  
 getBookings();  
}, [sortBy.value]);

async function getBookings() {  
 const bookingRef = await collection(*db*, "bookings");  
 const pathQuery = route.query;  
 const listQuery = [];  
 if (pathQuery.status) listQuery.push(where("status", "==", pathQuery.status));  
 if (pathQuery.sortBy) {  
 const [sort, destination] = pathQuery.sortBy.split("-");  
 listQuery.push(orderBy(sort, destination));  
 } else listQuery.push(orderBy("date", "desc"));  
 const qry = query(bookingRef, ...listQuery);  
 const allBookings = (  
 await getCountFromServer(query(bookingRef, ...listQuery))  
 ).data().count;  
 pageSize.value = *Math*.ceil(allBookings / PER\_PAGES);  
 if (currentPage.value > pageSize.value - 1)  
 currentPage.value = pageSize.value - 1;  
 else if (currentPage.value < 0) currentPage.value = 0;  
  
 if (currentPage.value === 0) {  
 router.push({  
 name: "booking",  
 query: { ...deleteField(route.query, "page") },  
 });  
 } else {  
 router.push({  
 name: "booking",  
 query: { ...route.query, page: currentPage.value + 1 },  
 });  
 }  
  
 bookings.value = (await getDocs(qry)).docs.map((doc) => ({  
 id: doc.id,  
 ...doc.data(),  
 }));  
}

result

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

4.2. facilities

A screenshot of a computer

Description automatically generated

Query :

async function getFacilities() {  
 const facilityRef = await collection(*db*, "facilities");  
 const pathQuery = route.query;  
 const listQuery = [];  
 if (pathQuery.sportType)  
 listQuery.push(where("sports\_type", "array-contains", pathQuery.sportType));  
 const qry = query(facilityRef, ...listQuery);  
 facilities.value = (await getDocs(qry)).docs.map((doc) => ({  
 id: doc.id,  
 ...doc.data(),  
 }));  
  
 pageSize.value = *Math*.ceil(facilities.value.length / PER\_PAGES);  
 if (currentPage.value > pageSize.value - 1)  
 currentPage.value = pageSize.value - 1;  
 else if (currentPage.value < 0) currentPage.value = 0;  
}  
  
watchEffect(() => {  
 if (filterSportType.value) {  
 router.push({  
 name: "facility",  
 query: { ...route.query, sportType: filterSportType.value },  
 });  
 } else {  
 router.push({  
 name: "facility",  
 query: { ...deleteField(route.query, "sportType") },  
 });  
 }  
 getFacilities();  
}, [filterSportType.value]);

result:

A screenshot of a computer

Description automatically generated

5.Facility Detail สำหรับดูข้อมูล Facility และ review ของแต่ละ facility detail

Query:

async function getFacility() {  
 const facilityRef = await doc(*db*, "facilities", facilityId);  
 const querySnap = await getDoc(facilityRef);  
 facilityDetail.value = { id: querySnap.id, ...querySnap.data() };  
}  
  
async function getReviewsByFacilityId() {  
 const reviewsRef = await collection(*db*, "reviews");  
 const qry = query(reviewsRef, where("facility\_id", "==", facilityId));  
 const querySnap = await getDocs(qry);  
 reviews.value = querySnap.docs.map((review) => ({  
 id: review.id,  
 ...review.data(),  
 }));  
}

result:

A screenshot of a computer

Description automatically generated

6.Booking Detail สำหรับดูข้อมูล booking และ ข้อมูลของผู้จอง โดยค้นหาตาม user\_name ในแต่ละ booking

Query:

async function getBooking() {  
 const bookingRef = await doc(*db*, "bookings", bookingId);  
 const querySnap = await getDoc(bookingRef);  
 bookingDetail.value = { id: querySnap.id, ...querySnap.data() };  
}  
  
async function getUserByName() {  
 const userRef = await collection(*db*, "account");  
 const qry = query(  
 userRef,  
 where("name", "==", bookingDetail.value.user\_name),  
 );  
 const querySnap = (await getDocs(qry)).docs[0];  
 userDetail.value = { id: querySnap.id, ...querySnap.data() };  
}

Result:

A screenshot of a computer

Description automatically generated