# Coding Review of SCHEDOOL

**Code Review : reservationServices.ts**

```
/*
     File: reservationServices.js
     Author: Poon - Nunthatinn Veerapaiboon (nveerap@cmkl.ac.th)
     Description: Main reservation logics
  - Conflicts check
  - Recursively upsert reservation
     Date: 2025-10-07

*/
```

**There is some minor issue with names it's "reservationServices.ts" not "reservationServices.js"**

```
/*
     File: reservationServices.ts
     Written by: Nunthatinn Veerapaiboon (nveerap@cmkl.ac.th)
     Modified by: [Nunthatinn Veerapaiboon], [2025-10-09]
     Description: Handles reservation logic including conflict
detection and weekly repetition.
     Last Modified: 2025-10-07
*/
```

---

**Function argument comment & Error handling**

```ts
// Check conflicts
export async function checkConflicts({
  reservationId = null,
  roomId,
  timeStart,
  timeEnd,
  rep = 1
}: CheckConflictArgs): Promise<boolean> {
  const slots = generateWeeklySlots(timeStart, timeEnd, rep);

  const reservationsOnSlotDates = await prisma.reservation.findMany({
    where: {
      roomId: roomId,
      OR: slots.map(slot => {
        const dayStart = new Date(slot.date + 'T00:00:00.000Z');
        const dayEnd   = new Date(slot.date + 'T23:59:59.999Z');
        return {
          timeStart: { lt: dayEnd },
          timeEnd:   { gt: dayStart },
        };
      }),
    },
    orderBy: { timeStart: 'asc' },
  });

  for (const slot of slots) {
    const slotStart = new Date(slot.timeStart);
    const slotEnd = new Date(slot.timeEnd);

    for (const res of reservationsOnSlotDates) {
      // Skip itself if editing an existing reservation
      if (reservationId && res.reservationId === reservationId) continue;

      const resStart = res.timeStart;
      const resEnd = res.timeEnd;
```

```
      // If the reservation ends before the slot starts OR
      // starts after the slot ends → no overlap, skip
      if (resEnd <= slotStart || resStart >= slotEnd) continue;

      // Check for real overlap using helper function
      if (isOverlap(slotStart, slotEnd, resStart, resEnd)) {
        throw new Error(
          `Conflict detected: Room ${roomId} already booked from ${res.timeStart}
to ${res.timeEnd}`
        );
      }
    }
  }

  return true; // no conflicts
}
```

**As you can see there is no comment about how function works, input, outputs I suggest slapping this comment on to this.**

```
/**
 * Checks for overlapping reservations for a given room and timeslot.
 * @param {CheckConflictArgs} args - Reservation parameters.
 * @returns {Promise<boolean>} - Returns true if no conflicts found.
 * @throws Error if conflict detected.
 */
```

**About error handling : when you use an async function it's a good practice to try/catch to handle errors and also use print/log for clear error messages.**

```
export async function checkConflicts(args: CheckConflictArgs):
Promise<boolean> {
      try {
            // ...existing logic...
            return true;
      } catch (error) {
            console.error(`[checkConflicts] Error: ${error.message}`);
```

```
            throw error;
        }
}
```

**Code Review : [index.ts]**

```
/*
        File: index.js
        Author: Win - Thanawin Pattanaphol (tpattan@cmkl.ac.th)
        Description: Main Project File
        Date: 2025-10-07

        License: GNU General Public License Version 3.0
*/
```

**There is some minor name error. Here is a suggestion**

```
/*
        File: index.ts
        Written by: Thanawin Pattanaphol (tpattan@cmkl.ac.th)
        Modified by: [Your Name], [Date]
        Description: Main entry point demonstrating Prisma ORM CRUD
operations.
        Last Modified: 2025-10-07
        License: GNU General Public License Version 3.0
*/
```

**Function Comments**

**There’s no comment about how these function works**

```
async function main() {
    /*
        These are demonstrations of the CRUD operations in Prisma
ORM.
    */

    // CREATE
```

```
    const new_reservation = await prisma.reservation.create({
        data: {
            reservationId: 1,
            roomId: 'thisisanexampleid',
            timeStart: "something something something",
            timeEnd: "something something something"
        }
    })
    console.log(new_reservation);

    // READ

    const rooms = await prisma.room.findMany()
    console.log(rooms);

    // UPDATE

    const update_reservation = await prisma.reservation.update({
        where: {
            reservationId: 1
        },
        data: {
            timeEnd: "thisisanexampletimestamp"
        }
    })

    // DELETE

    const delete_reservation = await prisma.reservation.delete({
        where: {
            reservationId: 1,
        },
    })

}

main()
    .then(async () => {
        await prisma.$disconnect()
```

```
    })
    .catch(async (e) => {
        console.error(e)
        await prisma.$disconnect()

    })
```

**So we just need to put**

```
/**
 * Demonstrates basic CRUD operations using Prisma ORM.
 * Creates, reads, updates, and deletes a sample reservation record.
 */
```