Bug Fix: Invoice Line Generation from Timesheet Dates

Issue Description

When creating a timesheet and entering work dates, the invoice line generation was inconsistent. For example:
- If 2 working days were entered, sometimes incorrect numbers of days were created
- The bug was caused by improper date handling and timezone issues

Root Cause Analysis

The bug was located in `/server/api/routers/timesheet.ts` in the `createRange` mutation (lines 218-295).

Three Main Issues:

1. **Lack of Date Normalization**
   - Dates from the frontend were not normalized to UTC midnight
   - This caused timezone-related inconsistencies
   - When dates crossed timezone boundaries, calculations could be off by a day

2. **Potential Date Object Reference Issues**
   - The code used `date: cursor` directly, storing a reference to the cursor object
   - While the code created new Date objects with `new Date(cursor.getTime() + ...)`, this wasn't the safest approach
   - The fix explicitly creates new Date objects for each entry

3. **Daylight Saving Time (DST) Issues**
   - The original code used millisecond addition: `cursor.getTime() + 24 * 60 * 60 * 1000`
   - This breaks during DST transitions where days can be 23 or 25 hours long
   - Using UTC date methods (`setUTCDate`) avoids this problem

The Fix

Changes Made to `server/api/routers/timesheet.ts`:

1. Date Normalization (Lines 218-224)

```
// BEFORE:
const start = new Date(input.startDate);
const end = new Date(input.endDate);

// AFTER:
// 🔥 FIX: Parse and normalize dates to UTC midnight to avoid timezone issues
const start = new Date(input.startDate);
const end = new Date(input.endDate);

// Normalize to UTC midnight to ensure consistent date handling
start.setUTCHours(0, 0, 0, 0);
end.setUTCHours(0, 0, 0, 0);
```

Why this helps:

- Ensures all dates are at midnight UTC
- Eliminates timezone-related edge cases
- Makes date comparisons consistent

2. Proper Date Object Creation (Lines 286-302)

```
// BEFORE:
const entries = [];
let cursor = new Date(start);

while (cursor <= end) {
  entries.push({
    timesheetId: ts.id,
    date: cursor,  // ❌ Potential reference issue
    hours: new Prisma.Decimal(hoursPerDay),
    amount: null,
  });

  cursor = new Date(cursor.getTime() + 24 * 60 * 60 * 1000);  // ❌ DST issues
}

// AFTER:
// 🔥 FIX: Generate daily entries with proper date handling
// Create a new Date object for each entry to avoid reference issues
// Use UTC date manipulation to avoid DST and timezone issues
const entries = [];
const cursor = new Date(start);

while (cursor <= end) {
  entries.push({
    timesheetId: ts.id,
    date: new Date(cursor),  // ✅ Create NEW Date object for each entry
    hours: new Prisma.Decimal(hoursPerDay),
    amount: null,
  });

  // 🔥 FIX: Use UTC date manipulation to avoid DST issues
  cursor.setUTCDate(cursor.getUTCDate() + 1);  // ✅ UTC date arithmetic
}
```

Why this helps:
- `new Date(cursor)` creates a distinct Date object for each entry
- Prevents any potential reference-sharing issues
- `setUTCDate(cursor.getUTCDate() + 1)` properly handles DST transitions
- Works correctly across month boundaries (e.g., Jan 31 → Feb 1)

Test Cases Verified

All test cases pass with the fix:

1. ✅ **Two working days** (2024-01-01 to 2024-01-02) → 2 entries
2. ✅ **Same day** (2024-01-01 to 2024-01-01) → 1 entry
3. ✅ **Five working days** (2024-01-01 to 2024-01-05) → 5 entries
4. ✅ **DST transition period** (2024-03-10 to 2024-03-14) → 5 entries
5. ✅ **Cross month boundary** (2024-01-30 to 2024-02-02) → 4 entries
6. ✅ **Dates with time components** → Normalized correctly
7. ✅ **Date object independence** → No shared references

Invoice Line Generation Impact

The fix in timesheet entry creation directly fixes the invoice line generation because:

1. **Invoice lines are created from timesheet entries** (lines 676-684)
2. **One invoice line per timesheet entry** (correct behavior)
3. With the fixed entry generation, invoice lines will now be accurate

```
// Invoice line creation (already correct, just needed correct entries)
for (const entry of timesheet.entries) {
  lineItems.push({
    description: `Work on ${new Date(entry.date).toISOString().slice(0, 10)} (${entry.
hours}h)`,
    quantity: new Prisma.Decimal(1), // 1 day per entry
    unitPrice: rate,
    amount: rate,
  });
}
```

Commit Information

Branch: `expenses-structure`
Commit Message: `fix: correct invoice line generation from timesheet dates to prevent duplicate/incorrect days`
Files Changed: `server/api/routers/timesheet.ts`

Prevention

To prevent similar issues in the future:

1. Always normalize dates to UTC midnight when dealing with date-only values
2. Always create new Date objects instead of reusing references
3. Use UTC date methods for date arithmetic to avoid DST issues

4. Add validation tests for edge cases (DST, month boundaries, leap years)

Related Code

- **Timesheet Entry Creation:** `/server/api/routers/timesheet.ts` (createRange mutation)
- **Invoice Line Generation:** `/server/api/routers/timesheet.ts` (sendToAgency mutation, lines 676-684)
- **Frontend Date Input:** `/components/timesheets/TimesheetSubmissionForm.tsx`