# Enum Casing Fix - Documentation

## Problem Statement

The application was experiencing 500 errors when fetching invoices due to enum casing mismatches between:
- Database values (lowercase: `custom`, `gross`, `payroll`, `fixed`, etc.)
- Prisma schema enum definitions (uppercase with @map: `CUSTOM`, `GROSS`, `PAYROLL`, `FIXED`)
- Code references (mixed uppercase and string literals)

**Error Example:**

```
Invalid prisma.invoice.findFirst() invocation: Value 'CUSTOM' not found in enum 'MarginType'
```

## Root Cause

1. Database contained lowercase enum values from existing data
2. Prisma schema used uppercase enum values with `@map()` directives
3. Code mixed enum references (e.g., `MarginType.FIXED`) with string literals (e.g., `'CUSTOM'`)
4. `@map()` directive wasn't being applied consistently, causing runtime mismatches

## Solution Implemented

### 1. Updated Prisma Schema Enums to Lowercase

Changed all enum definitions to use lowercase values directly, matching the database:

```
// Before
enum PaymentModel {
  GROSS          @map("gross")
  PAYROLL        @map("payroll")
  PAYROLL_WE_PAY @map("payroll_we_pay")
  SPLIT          @map("split")
}

// After
enum PaymentModel {
  gross
  payroll
  payroll_we_pay
  split
}
```

**Affected Enums:**
- `PaymentModel` (gross, payroll, payroll_we_pay, split)
- `MarginType` (fixed, variable, custom, percentage)
- `BankAccountUsage` (salary, gross, expenses, other)
- `PaymentMethodType` (bank_account, credit_card, debit_card, paypal, stripe, wise, revolut, other)

- `PaymentType` (received, sent)
- `RecipientType` (admin, contractor, payroll, client, agency, other)
- `RemittanceStatus` (pending, completed, failed)

### 2. Updated All Code References

Replaced all uppercase enum references with lowercase throughout the codebase:

```
// Before
if (contract.paymentModel === PaymentModel.GROSS) { ... }
marginType: MarginType.FIXED

// After
if (contract.paymentModel === PaymentModel.gross) { ... }
marginType: MarginType.fixed
```

**Files Updated:** 360+ TypeScript/TSX files

### 3. Fixed String Literal Issues

Replaced hardcoded string literals with proper enum values:

```
// Before
updateData.marginType = 'CUSTOM' as MarginType

// After
updateData.marginType = MarginType.custom
```

### 4. Added Case-Insensitive Helpers

Created `lib/utils/enum-helpers.ts` with normalization functions for backward compatibility:

```
// Normalize any string (regardless of case) to proper enum value
const marginType = normalizeMarginType('CUSTOM') // Returns MarginType.custom
const marginType = normalizeMarginType('custom') // Returns MarginType.custom
```

### 5. Updated Enum Utility Functions

Updated helper functions in:
- `lib/constants/payment-models.ts` - PaymentModel utilities
- `lib/constants/bank-usage.ts` - BankAccountUsage utilities
- `lib/services/MarginService.ts` - MarginType normalization

## Files Modified

### Core Schema

- `prisma/schema.prisma` - Enum definitions updated to lowercase

### Services

- `lib/services/MarginService.ts` - Enum references and normalization
- `lib/services/MarginCalculationService.ts` - Enum usage
- `lib/services/PaymentWorkflowService.ts` - Enum usage
- `lib/services/RemittanceService.ts` - Enum usage

### Constants & Utilities

- `lib/constants/payment-models.ts` - PaymentModel helpers
- `lib/constants/bank-usage.ts` - BankAccountUsage helpers
- `lib/utils/enum-helpers.ts` - NEW: Case-insensitive enum utilities

### API Routers

- `server/api/routers/invoice.ts` - Enum usage
- `server/api/routers/contract.ts` - Enum usage
- `server/api/routers/timesheet.ts` - Enum usage
- `server/api/routers/bank.ts` - Enum usage
- `server/api/routers/payment.ts` - Enum usage

### Components

- `components/invoices/detail/InvoiceWorkflowActions.tsx` - Enum comparisons
- `components/invoices/InvoiceReviewModal.tsx` - Enum usage
- `components/invoices/MarginConfirmationCard.tsx` - Enum display
- `components/timesheets/TimesheetSubmissionForm.tsx` - Enum usage
- `components/profile/banks/BankAccountForm.tsx` - Enum usage
- Plus 30+ other component files

## Testing Checklist

After regenerating Prisma client ( `npx prisma generate` ), verify:

1. **Invoice Operations**
   - ✅ `invoice.getById()` works without 500 errors
   - ✅ Invoice list loads correctly
   - ✅ Invoice creation with margins works
   - ✅ Margin calculations are accurate

2. **Contract Operations**
   - ✅ Contract creation with payment models
   - ✅ Contract margin type selection
   - ✅ Contract salaryType display

3. **Payment Workflow**
   - ✅ Gross payment flow (self-invoicing)
   - ✅ Payroll payment flow
   - ✅ Split payment flow

4. **Timesheet Operations**
   - ✅ Timesheet submission
   - ✅ Timesheet approval with margins

## Migration Notes

**NO DATABASE MIGRATION REQUIRED** - This is a code-only fix that aligns the code with existing database values.

### Steps to Deploy

1. Pull the latest code with enum fixes
2. Run `npx prisma generate` to regenerate Prisma client
3. Restart the application
4. Test invoice operations

### Rollback Plan

If issues occur, the fix can be reverted by:
1. Restoring previous Prisma schema with @map directives
2. Reverting code changes
3. Regenerating Prisma client

## Benefits

1. **No Data Migration:** Works with existing lowercase database values
2. **Type Safety:** Maintains TypeScript enum type safety
3. **Consistency:** Single source of truth for enum values
4. **Backward Compatible:** Case-insensitive helpers handle legacy data
5. **Future Proof:** Prevents similar issues with new enums

## Additional Notes

- The `percentage` value in MarginType is normalized to `variable` for backward compatibility
- All enum comparisons now work case-insensitively through helper functions
- String literals have been eliminated in favor of enum values
- Added comprehensive logging for unknown enum values to aid debugging

## Related Issues

- Invoice 500 errors on `getById` operation
- Margin type validation failures
- Payment model comparison issues
- Bank account usage type mismatches

---

**Last Updated:** December 22, 2024
**Version:** 1.0