Year 2 Semester 2 2025

IT Project

GYM Management System

(ZFit)

Information Technology Project

IT2080

Group ID:

ITP25_B2_W225

Group Details:

Campus : Malabe

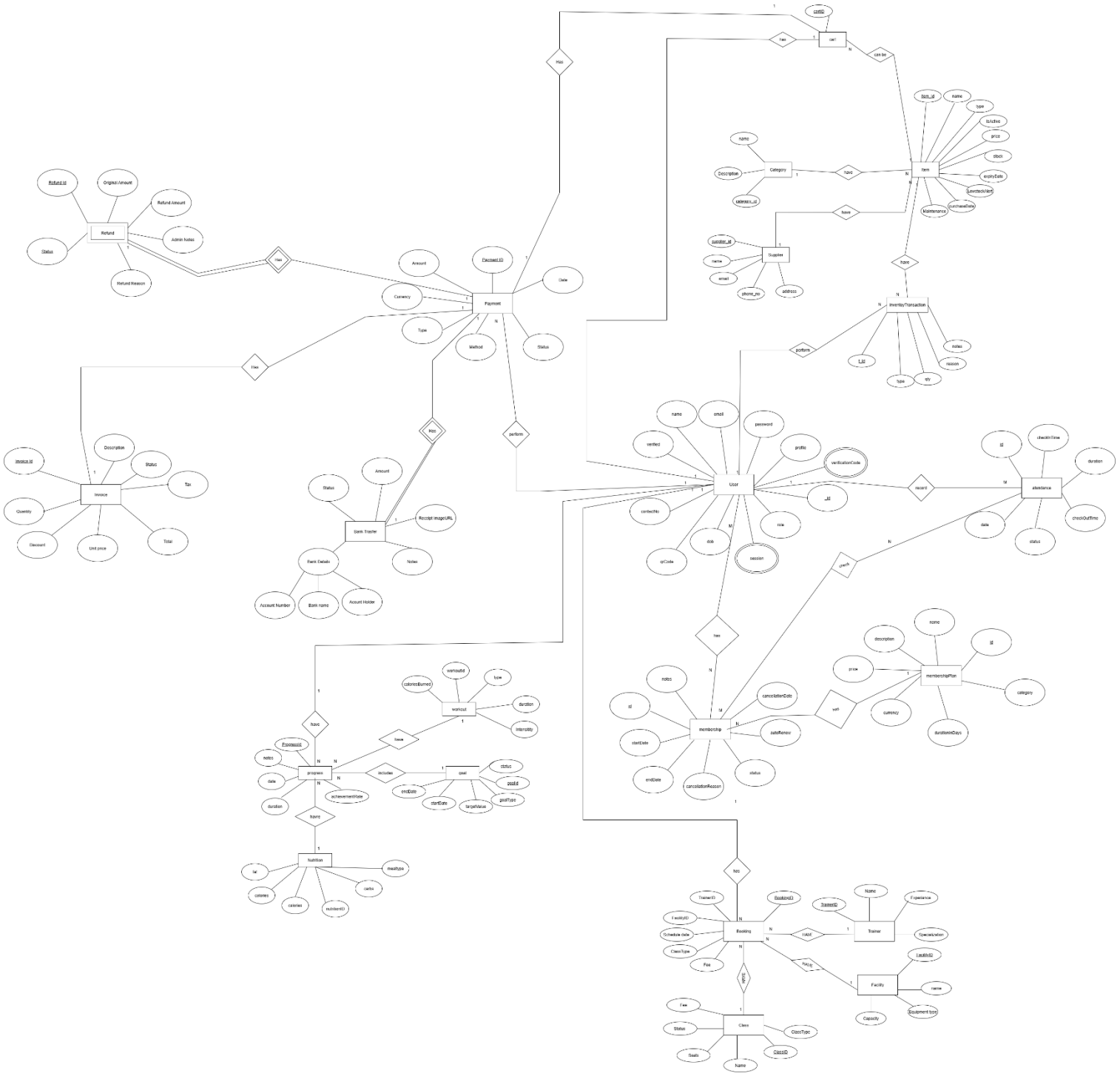
Group Number : ITP25_B2_W225

	IT Number	Student Name	Student E-mail Address	Contact Number
1	IT 23 5621 10	H R G N Karunathilaka	IT23562110@my.sliit.lk	0764891241
2	IT 23 5418 70	N M B M L B Nawaratne	IT23541870@my.sliit.lk	0763541666
3	IT 23 5804 80	K B P Kavisika	IT23580480@my.sliit.lk	0718827129
4	IT 23 7536 62	O A l De Silva	IT23753662@my.sliit.lk	0740357412
5	IT 23 7414 78	B Dhayabari	IT23741478@my.sliit.lk	0770772330

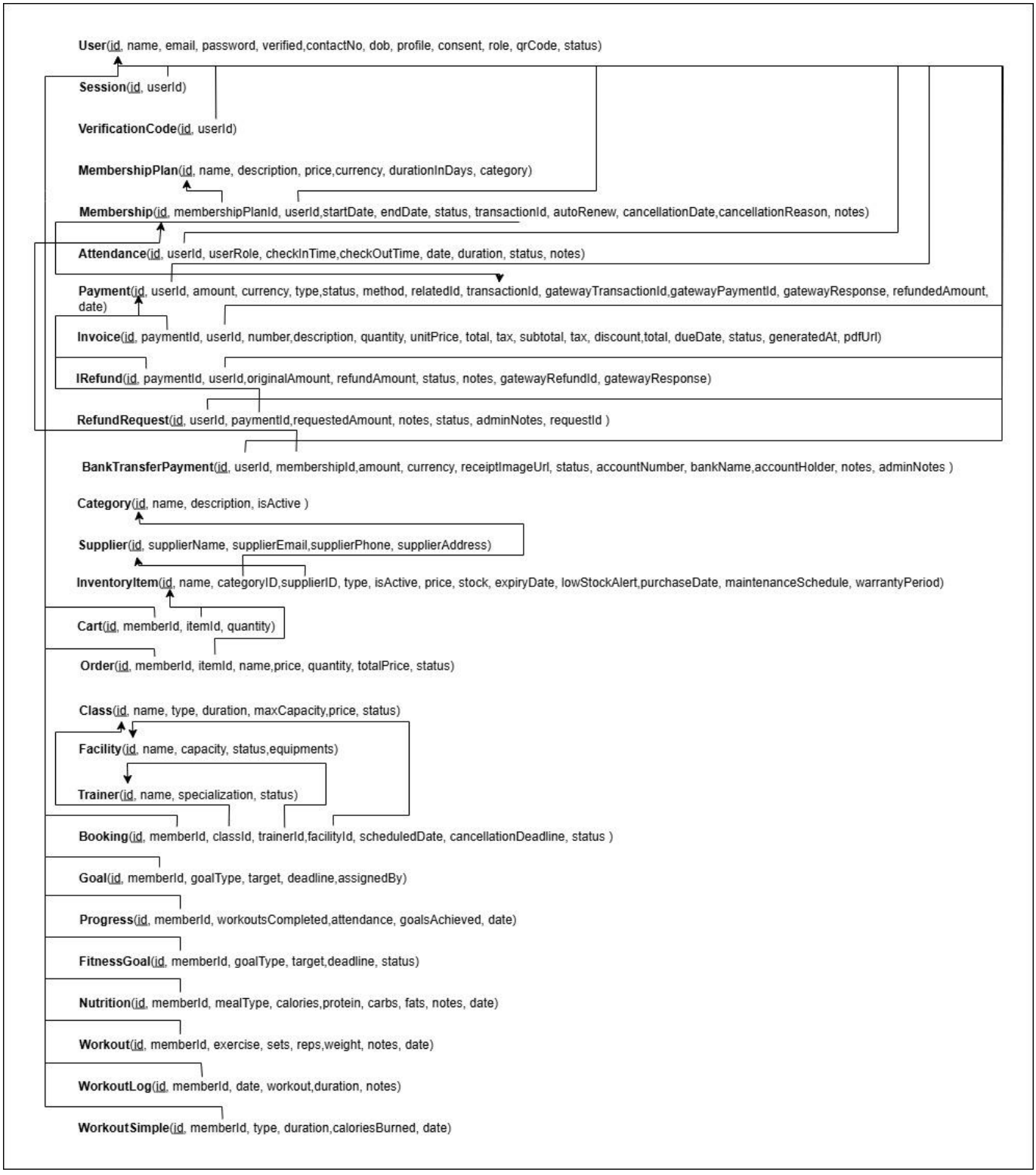
Contents

1. ER Diagram

Link - [Zfit-ER-Diagram]



2. Normalized Schema



3. Models

```
// User Schema
const userSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    verified: { type: Boolean, default: false },
    contactNo: { type: String },
    dob: { type: Date },
    profile: { type: String },
    consent: { type: Boolean, default: false },
    role: { type: String, enum: ['member', 'staff', 'admin'], default: 'member' },
    qrCode: { type: String },
    status: { type: String, enum: ['active', 'inactive'], default: 'active' }
}, { timestamps: true });
```

```
// Session Schema
const sessionSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }
}, { timestamps: true });
```

```
// VerificationCode Schema
const verificationCodeSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }
}, { timestamps: true });
```

```
// MembershipPlan Schema
const membershipPlanSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    name: { type: String, required: true },
    description: { type: String },
    price: { type: Number, required: true },
    currency: { type: String, default: 'LKR' },
    durationInDays: { type: Number, required: true },
    category: { type: String }
}, { timestamps: true });
```

```
// Membership Schema
const membershipSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    membershipPlanId: { type: mongoose.Schema.Types.ObjectId, ref: 'MembershipPlan', required: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    startDate: { type: Date, required: true },
    endDate: { type: Date, required: true },
    status: { type: String, enum: ['active', 'expired', 'cancelled'], default: 'active' },
    transactionId: { type: String },
    autoRenew: { type: Boolean, default: false },
    cancellationDate: { type: Date },
    cancellationReason: { type: String },
    notes: { type: String }
}, { timestamps: true });
```

```
// Supplier Schema
const supplierSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    supplierName: { type: String, required: true },
    supplierEmail: { type: String },
    supplierPhone: { type: String },
    supplierAddress: { type: String }
}, { timestamps: true });
```

```
// InventoryItem Schema
const inventoryItemSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    name: { type: String, required: true },
    categoryId: { type: mongoose.Schema.Types.ObjectId, ref: 'Category', required: true },
    supplierId: { type: mongoose.Schema.Types.ObjectId, ref: 'Supplier', required: true },
    type: { type: String, required: true },
    isActive: { type: Boolean, default: true },
    price: { type: Number, required: true },
    stock: { type: Number, required: true },
    expiryDate: { type: Date },
    lowStockAlert: { type: Number, default: 10 },
    purchaseDate: { type: Date },
    maintenanceSchedule: { type: String },
    warrantyPeriod: { type: Number }
}, { timestamps: true });
```

```
// Workout Schema
const workoutSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    exercise: { type: String, required: true },
    sets: { type: Number, required: true },
    reps: { type: Number, required: true },
    notes: { type: String },
    date: { type: Date, default: Date.now }
}, { timestamps: true });

// WorkoutLog Schema
const workoutLogSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    date: { type: Date, required: true },
    workout: { type: String, required: true },
    duration: { type: Number, required: true },
    notes: { type: String }
}, { timestamps: true });

// WorkoutSample Schema
const workoutSampleSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    type: { type: String, required: true },
    duration: { type: Number, required: true },
    caloriesBurned: { type: Number },
    date: { type: Date, default: Date.now }
}, { timestamps: true });
```

(property) notes?: string | null | undefined

```
// Progress Schema
const progressSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    workoutsCompleted: { type: Number, default: 0 },
    attendance: { type: Number, default: 0 },
    goalsAchieved: { type: Number, default: 0 },
    date: { type: Date, default: Date.now }
}, { timestamps: true });

// FitnessGoal Schema
const fitnessGoalSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    goalType: { type: String, required: true },
    target: { type: Number, required: true },
    deadline: { type: Date, required: true },
    status: { type: String, enum: ['active', 'completed', 'paused'], default: 'active' }
}, { timestamps: true });

// Nutrition Schema
const nutritionSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    mealType: { type: String, required: true },
    calories: { type: Number, required: true },
    protein: { type: Number, required: true },
    carbs: { type: Number, required: true },
    fats: { type: Number, required: true },
    notes: { type: String },
    date: { type: Date, default: Date.now }
}, { timestamps: true });
```

```
// Cart Schema
const cartSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    itemId: { type: mongoose.Schema.Types.ObjectId, ref: 'InventoryItem', required: true },
    quantity: { type: Number, required: true, min: 1 }
}, { timestamps: true });
```

```
// Class Schema
const classSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    name: { type: String, required: true },
    type: { type: String, required: true },
    duration: { type: Number, required: true },
    maxCapacity: { type: Number, required: true },
    price: { type: Number, required: true },
    status: { type: String, enum: ['active', 'inactive'], default: 'active' }
}, { timestamps: true });
```

```
// Order Schema
const orderSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    itemId: { type: mongoose.Schema.Types.ObjectId, ref: 'InventoryItem', required: true },
    name: { type: String, required: true },
    price: { type: Number, required: true },
    quantity: { type: Number, required: true },
    totalPrice: { type: Number, required: true },
    status: { type: String, enum: ['pending', 'completed', 'cancelled'], default: 'pending' }
}, { timestamps: true });
```

```
// Session Schema
const sessionSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }
}, { timestamps: true });
```

```
// Facility Schema
const facilitySchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    name: { type: String, required: true },
    capacity: { type: Number, required: true },
    status: { type: String, enum: ['available', 'occupied', 'maintenance'], default: 'available' },
    equipment: [{ type: String }]
}, { timestamps: true });
```

```
// Trainer Schema
const trainerSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    name: { type: String, required: true },
    specialization: { type: String },
    status: { type: String, enum: ['available', 'busy'], default: 'available' }
}, { timestamps: true });
```

```
// Goal Schema
const goalSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    goalType: { type: String, required: true },
    target: { type: Number, required: true },
    deadline: { type: Date, required: true },
    assignedBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User' }
}, { timestamps: true });
```

```
// Booking Schema
const bookingSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    memberId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    classId: { type: mongoose.Schema.Types.ObjectId, ref: 'Class', required: true },
    trainerId: { type: mongoose.Schema.Types.ObjectId, ref: 'Trainer', required: true },
    facilityId: { type: mongoose.Schema.Types.ObjectId, ref: 'Facility', required: true },
    scheduledDate: { type: Date, required: true },
    cancellationDeadline: { type: Date },
    status: { type: String, enum: ['confirmed', 'cancelled', 'completed'], default: 'confirmed' }
}, { timestamps: true });
```

```
// BankTransferPayment Schema
const bankTransferPaymentSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    membershipId: { type: mongoose.Schema.Types.ObjectId, ref: 'Membership' },
    amount: { type: Number, required: true },
    currency: { type: String, default: 'LKR' },
    receiptImageUrl: { type: String, required: true },
    status: { type: String, enum: ['pending', 'approved', 'rejected'], default: 'pending' },
    accountNumber: { type: String },
    bankName: { type: String },
    accountHolder: { type: String },
    notes: { type: String },
    adminNotes: { type: String }
}, { timestamps: true });
```

```
// RefundRequest Schema
const refundRequestSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    paymentId: { type: mongoose.Schema.Types.ObjectId, ref: 'Payment', required: true },
    requestedAmount: { type: Number, required: true },
    notes: { type: String },
    status: { type: String, enum: ['pending', 'approved', 'rejected'], default: 'pending' },
    adminNotes: { type: String },
    requestId: { type: String, required: true, unique: true }
}, { timestamps: true });
```

```
// IRefund Schema
const iRefundSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    paymentId: { type: mongoose.Schema.Types.ObjectId, ref: 'Payment', required: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    originalAmount: { type: Number, required: true },
    refundAmount: { type: Number, required: true },
    status: { type: String, enum: ['pending', 'approved', 'rejected', 'processed'], default: 'pending' },
    notes: { type: String },
    gatewayRefundId: { type: String },
    gatewayResponse: { type: mongoose.Schema.Types.Mixed }
}, { timestamps: true });
```

```
// Invoice Schema
const invoiceSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    paymentId: { type: mongoose.Schema.Types.ObjectId, ref: 'Payment', required: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    number: { type: String, required: true, unique: true },
    description: { type: String },
    quantity: { type: Number, default: 1 },
    unitPrice: { type: Number, required: true },
    total: { type: Number, required: true },
    tax: { type: Number, default: 0 },
    subtotal: { type: Number, required: true },
    discountTotal: { type: Number, default: 0 },
    dueDate: { type: Date },
    status: { type: String, enum: ['draft', 'sent', 'paid', 'overdue'], default: 'draft' },
    generatedAt: { type: Date, default: Date.now },
    pdfUrl: { type: String }
}, { timestamps: true });
```

```
// Payment Schema
const paymentSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    amount: { type: Number, required: true },
    currency: { type: String, default: 'LKR' },
    type: { type: String, enum: ['membership', 'inventory', 'booking', 'other'], required: true },
    status: { type: String, enum: ['pending', 'completed', 'failed', 'refunded'], default: 'pending' },
    method: { type: String, enum: ['card', 'bank_transfer', 'cash'], required: true },
    relatedId: { type: String },
    transactionId: { type: String, unique: true },
    gatewayTransactionId: { type: String },
    gatewayPaymentId: { type: String },
    gatewayResponse: { type: mongoose.Schema.Types.Mixed },
    refundedAmount: { type: Number, default: 0 },
    notes: { type: String }
}, { timestamps: true });
```

```
// Attendance Schema
const attendanceSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    userRole: { type: String, enum: ['member', 'staff', 'admin'], required: true },
    checkInTime: { type: Date, required: true },
    checkOutTime: { type: Date },
    date: { type: Date, required: true },
    duration: { type: Number },
    status: { type: String, enum: ['checked-in', 'checked-out'], default: 'checked-in' },
    notes: { type: String }
}, { timestamps: true });
```