# **Software Design and Development**

## **ENG501 Assessment 3 Report**

### **Model Mapping Documentation**

## **Table of Contents**

1. Introduction

2. Part A: Domain Class Model to ER Model Mapping

3. Part B: ER Model to Domain Class Model Mapping

4. Assumptions and Design Decisions

5. Conclusion

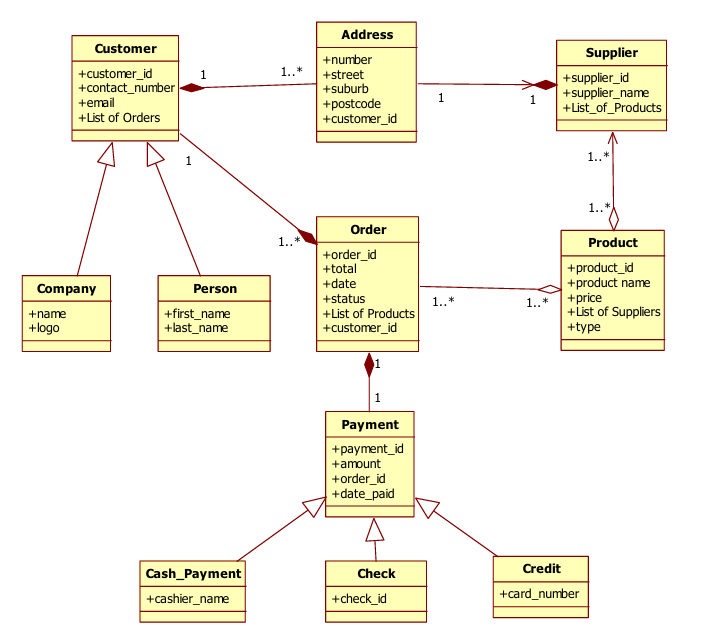
## **1. Introduction**

This report documents the mapping process between Domain Class Models and Entity-Relationship Models (ERM) for two distinct systems: - A Rent-A-Car system (Domain Class to ERM mapping) - A Student Enrollment system (ERM to Domain Class mapping)

The document includes the mapping processes, design decisions, assumptions made, and detailed explanations of the relationships and structures implemented.

## **2. Part A: Domain Class Model to ER Model Mapping**

### **2.1 Source Domain Class Model**



*Model for Rent-A-Car System*

### **2.2 Resulting ER Mode**

### 

( Detailed modell attached separately in the Zip)

*Figure 2: Mapped ER Model for Rent-A-Car System*

### **2.3 Mapping Details**

#### **2.3.1 Entity Mapping**

1. **Customer Entity**

– Primary entity with customer\_id as PK

– Attributes: contact\_number, email

– Serves as parent entity for Person and Company

2. **Person Entity**

– Inherits from Customer via shared PK

– Attributes: first\_name, last\_name

– customer\_id serves as both PK and FK

3. **Company Entity**

– Inherits from Customer via shared PK

– Attributes: name, logo

– customer\_id serves as both PK and FK

4. **Order Entity**

– Independent entity with order\_id as PK

– Attributes: total, date, status

– FK: customer\_id referencing Customer

5. **Product Entity**

– Independent entity with product\_id as PK

– Attributes: product\_name, price, type

6. **Payment Entity**

– Parent entity with payment\_id as PK

– Attributes: amount, date\_paid

– FK: order\_id referencing Order

#### **2.3.2 Relationship Mapping**

1. **Customer-Order Relationship**

– One-to-many (1..\*)

– Implemented via FK in Order table

– Mandatory participation from Order side

2. **Order-Product Relationship**

– Many-to-many (*..*)

– Implemented via junction table ORDER\_PRODUCT

– Composite PK: order\_id + product\_id

3. **Product-Supplier Relationship**

– Many-to-many (*..*)

– Implemented via junction table PRODUCT\_SUPPLIER

– Composite PK: product\_id + supplier\_id

4. **Payment Inheritance**

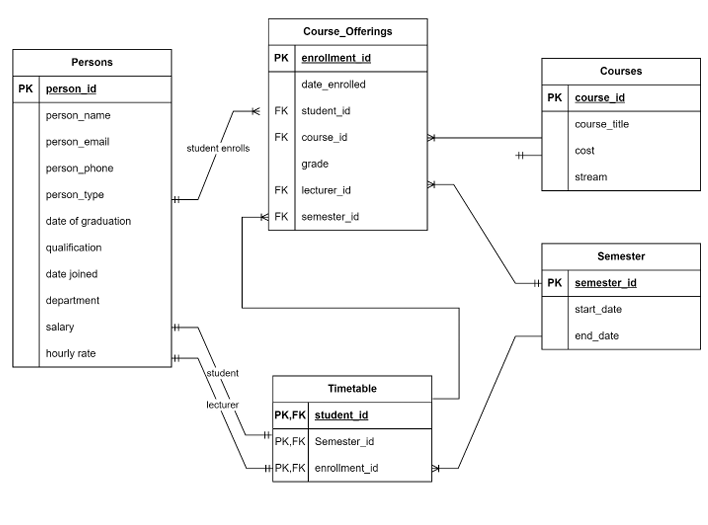
– Three child entities: Cash\_Payment, Check, Credit

– Implemented using shared primary key strategy

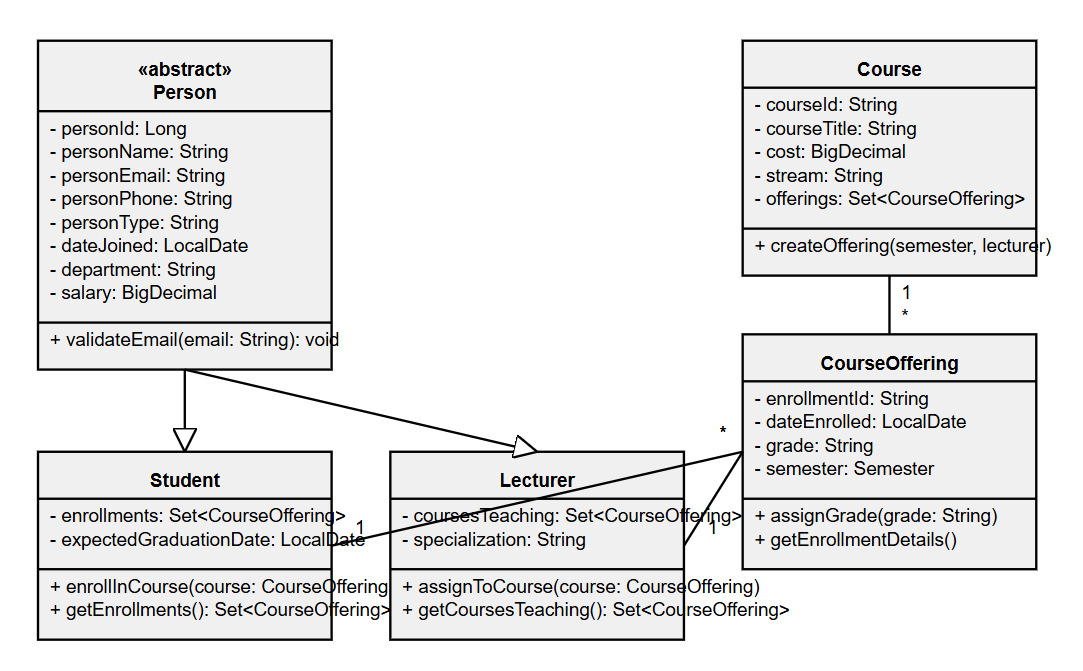
– Each child table has payment\_id as both PK and FK

## **3. Part B: ER Model to Domain Class Model Mapping**

### **3.1 Source ER Model**



### **3.2 Resulting Domain Class Model**

\

( More detailed model is attached in the Zip )

*Figure 4: Mapped Domain Class Model for Student Enrollment System*

### **3.3 Mapping Details**

#### **3.3.1 Class Mapping**

1. **Person Class**

class Person {  
 private int personId;  
 private String personName;  
 private String personEmail;  
 private String personPhone;  
 private String personType;  
 private Date dateOfGraduation;  
 private String qualification;  
 private Date dateJoined;  
 private String department;  
 private double salary;  
 private double hourlyRate;  
 }

2. **Course Class**

class Course {  
 private String courseId;  
 private String courseTitle;  
 private double cost;  
 private String stream;  
 }

3. **CourseOffering Class**

class CourseOffering {  
 private String enrollmentId;  
 private Date dateEnrolled;  
 private String grade;  
 private String studentId;  
 private String courseId;  
 private String lecturerId;  
 private String semesterId;  
 }

#### **3.3.2 Relationship Mapping**

1. **Person Inheritance**

– Abstract Person class with Student and Lecturer subclasses

– Inheritance based on personType attribute

– All common attributes in parent class

2. **Course-CourseOffering**

– One-to-many relationship

– Course (1) — (\*) CourseOffering

– Navigable from both ends

3. **Semester-CourseOffering**

– One-to-many relationship

– Semester (1) — (\*) CourseOffering

– Bidirectional navigation

## **4. Assumptions and Design Decisions**

### **4.1 General Assumptions**

1. **Primary Keys**

– All IDs are assumed to be system-generated

– String IDs represent business-meaningful codes

– Integer IDs represent system-generated sequences

2. **Null Values**

– Required fields implemented as NOT NULL

– Optional relationships allow NULL foreign keys

### **4.2 Part A Specific Assumptions**

1. **Inheritance Implementation**

– Used shared primary key strategy for all inheritance

– Assumes single inheritance (no multiple inheritance)

2. **List Attributes**

– Implemented via junction tables

– No array or collection storage in base tables

### **4.3 Part B Specific Assumptions**

1. **Person Types**

– personType field determines Student/Lecturer classification

– All person attributes relevant to both types

2. **Timetable Implementation**

– Serves as an association class

– Maintains scheduling information

– Links CourseOffering with Semester

## **5. Conclusion**

The mapping process successfully transformed: - Domain Class Model to ER Model for the Rent-A-Car system - ER Model to Domain Class Model for the Student Enrollment system

Both mappings preserve: - Data integrity - Relationship constraints - Business rules - System functionality

The resulting models are ready for: - Database implementation (ER Model) - Object-oriented development (Domain Class Model)