

# DUTCHESS COUNTY BUS TRANSPORTATION SYSTEM

MSCS\_542L\_256\_23S

The Four-Ce



Marist College School of Computer Science and Mathematics

Submitted To: Dr. Reza Sadeghi

10-25-2023

# PROJECT REPORT OF DUTCHESS COUNTY BUS TRANSPORTATION SYSTEM

## Team Name

The Four-Ce

## Team Members

- |   |  |
|---|--|
| 1. Sindhuja Ravikanth                               | <a href="mailto:Sindhuja.Ravikanth1@marist.edu">Sindhuja.Ravikanth1@marist.edu</a> (Team Head)             |
| 2. Shanmukha Chowdary Nalla<br>(Teammember)         | <a href="mailto:ShanumukhaChowdary.Nalla1@marist.edu">ShanumukhaChowdary.Nalla1@marist.edu</a>             |
| 3. Gaurav Bapurao Sherla<br>member)                 | <a href="mailto:GauravBapurao.Sherla1@marist.edu">GauravBapurao.Sherla1@marist.edu</a> (Team               |
| 4. Katipally Chanakya Vardhan Reddy<br>(Teammember) | <a href="mailto:ChanakyaVardhanReddy.Katipally1@marist.edu">ChanakyaVardhanReddy.Katipally1@marist.edu</a> |

# TABLE OF CONTENTS

## Contents

1.INTRODUCTION .....	1
2.PROJECT OBJECTIVES .....	2
3.REVIEW .....	3
4.MERITS .....	4
5.GITHUB REPOSITORY .....	5
6.ENTITY RELATIONSHIP MODEL (ER MODEL) .....	6
7.ENHANCED ENTITY RELATIONSHIP MODEL(EER MODEL) .....	19
8.DATABASE DEVELOPMENT .....	22
9.REFERENCES.....	33

# TABLE OF FIGURES

**Figure 6. 1 Entity Relationship ..... 15**

**Figure 6. 2 External ER USER POV ..... 16**

**Figure 6. 3 External ER ADMIN POV ..... 17**

**Figure 6. 4 Relationship of Entity Relationship Model ..... 18**

**Figure 7. 1 Enhanced Entity Relationship ..... 21**

## DESCRIPTION OF TEAM MEMBERS

### 1. Sindhuja Ravikanth

I attained my bachelor's degree in computer science from Keshav Memorial Institute of Technology in 2022. I worked as a Software Developer in TCP Wave for 1 year 9 months. I have come to Marist College for my master's degree in computer science. I am proficient in Java, MySQL, and JavaScript. I am here to develop my skills and develop better products.

### 2. Shanmukha Chowdary Nalla

I graduated from the Indian Institute of Technology Design and Manufacturing, Jabalpur in the year 2022. I started working in the software field in the 7<sup>th</sup> semester at Merkle Company as an Analyst and then as a Salesforce Marketing Cloud Developer till June 2023. I am a Certified Salesforce Marketing Cloud Developer, Email Specialist, and Machine Learning Engineer. I chose this course because it helps me enhance my skillset in data management which is always handy.

### 3. Gaurav Bapurao Sherla

I am an aspiring AWS and ML developer. I want to make the world a better place by empowering technology with the help of AI, especially in the mining sector. Where many lives are lost while mining ore from hundreds of feet beneath the earth, I have worked as an ML developer for a startup where I gained some useful insights with the help of this course I will be able to get in-depth practical knowledge about DBMS and this would help me enhance my knowledge, which can help me boost my skills for my future growth.

### 4. Katipally Chanakya Vardhan Reddy

I am Katipally Chanakya Vardhan Reddy, a Computer Science graduate from Methodist College of Engineering and Technology. Currently, I'm pursuing my Master's in MSCS with a focus on cloud computing at Marist College. I excel in Python, C, and C++ and have a strong background in problem-solving.

## **1.INTRODUCTION**

In an era defined by rapid urbanization and increased reliance on public transportation, the Dutchess County Bus Transportation System (DCBTS) stands as a vital lifeline for the residents of Dutchess County, New York. The DCBTS is an essential part of the daily lives of countless commuters, offering a means to connect with their destinations efficiently and sustainably. However, in a world where technological advancements are reshaping how we interact with the world around us, it is imperative that our public transportation system evolves in tandem.

The objective of this project is to propel the DCBTS into the future by revolutionizing its digital presence and user experience. With the advent of the Information Age, accessibility to transportation information is crucial, and the current DCBTS app, while functional, requires a comprehensive overhaul to meet the needs and expectations of modern commuters.

The primary aim of this project is to develop a user-friendly DCBTS application that caters to a diverse range of users. This includes commuters seeking to efficiently plan their journeys, tourists exploring the county, and an administrator with specialized privileges to manage and optimize the system.

In this project, we are planning to simplify the process of planning daily commutes via DCPT buses for all individuals which would be user-friendly to the users of all the age groups above 10 years. With the help of our project model, we want to cater our services to our users by providing them with enough data in a simplified format. Our user interface will provide the users with a menu where they can put in their source and destination locations and all the bus routes and buses in respect to their search will be displayed. Which can help them keep track of their bus timings and changes of any kind in the bus schedule. The users can also mark their favorite routes and buses which they mostly use for their commute and can keep track of them without any inconvenience.

## **2.PROJECT OBJECTIVES**

**The following should be supported by the system:**

- The application should be able to prompt the commuters for the login details and allow them to change credentials with proper authentication.
- The application should allow the administrator to make changes in the user records.
- The application should provide the capability for the users or administrator to modify their contact information and profile changes with proper authentication.
- Users should be able to access information about their bus routes and buses at any point in time.
- The administrator should have access to the user's credentials through a valid process.
- The administrator should be able to make changes in the commuting-related data.
- The application should have the capability to access separate login databases like commuter login and administrator login. So that they can access their respective data and use the application based on their requirements.

### **3.REVIEW**

#### **3.1 Dutchess County Public Transportation (DCPT) [1]**

- This app is based on the whole public transportation system of Dutchess County.
- The app shows the live tracking of buses around Dutchess County.
- The buses are identified by their bus numbers with which people can identify which bus would be comfortable to board based on their schedule.
- The app also shows the user's live location, so that the user can identify which bus is the nearest to them.
- This app shows buses in a map form.

#### **3.2 NYC Transit: MTA Subway & Bus [2]**

- This app is based on the Subways and Buses of New York.
- This app shows the best ways to get from a source to a destination around New York.
- It saves different locations as favorites.

#### **3.3 Moovit [3]**

- This app is based on all types of transport in many countries worldwide.
- It shows directions from one place to another and suggests quicker ways to get from a source to a destination irrespective of the transit type.



## **4.MERITS**

- The DCBTS helps everyone to search for a route from source to destination and plan their day based on the time estimation of the buses in Dutchess County.
- This application shows all the stops between the source and destination which is not available in any of the reviewed applications.
- This application will keep track of days when the buses will not be available like public holidays which is not available in any of the reviewed applications.
- This application will support multiple users including an Administrator.
- This application will also show the bus fare by taking into consideration the user's age which is not available in any of the reviewed applications.

## **5.GITHUB REPOSITORY**

[https://github.com/SindhujaRavikanth2001/DBMS\\_Project](https://github.com/SindhujaRavikanth2001/DBMS_Project)

## 6.ENTITY RELATIONSHIP MODEL (ER MODEL)

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects, or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education, and research.

We have used ERDPlus to create the ER diagram. ERDPlus is an open- source software modeling tool that supports the UML (Unified Modeling Language) framework for system and software modeling.

Entities	Attributes
<b>User</b>	UserID, FirstName, LastName, DOB, Email, PhoneNumber, Username, Password
<b>Admin</b>	AdminID, Username, Password, Email, PhoneNumber, EmployeeID
<b>Bus</b>	BusID, BusNumber, Capacity,LicensePlateNumber, BusType
<b>Ticket Type</b>	TicketTypeID, TypeName, Price, ValidityPeriod, Description, UserID
<b>Payment</b>	PaymentID, Amount, UserID, TicketTypeID, PaymentDate
<b>Reservation</b>	ReservationID, PaymentID, BusID, ReservationDate, NumberOfReservations
<b>Bus Route</b>	RouteID, StartLocation, EndLocation, BusID
<b>Route Stop Sequence</b>	StopSequenceID, RouteID, StopID,NumberofStops, ArrivalTime
<b>Employee</b>	EmployeeID, DepartmentID, FirstName,LastName, Position

<b>Department</b>	DepartmentID, DepartmentName, Location
<b>Schedule</b>	ScheduleID, BusID, RouteID, DayFlag, DepartureTime, ArrivalTime
<b>Bus Stop</b>	StopID, StopName, Location
<b>Notification</b>	NotificationID, Message, DateAndTime, UserID, AdminID, ReservationID
<b>Holidays</b>	HolidayID, HolidayDate, Description, NotificationID

## 6.1 IMPLEMENTATION OF ER DIAGRAM

### Entities and Attributes:

#### **Entity:** User

**Description:** A table to describe the passengers in the Dutchess County Bus Transportation System.

#### **Attributes:**

UserID – A unique identifier to identify each user. (Primary Key) FirstName – First name of the user.

LastName – Last name of the user. DOB – The date of birth of the user. Email – E-mail ID of the user.

PhoneNumber – The contact number of the user.

Username – The username of the user with which they want to login. Password – Password of the user.

#### **Entity:** Admin

**Description:** A table to describe the administrators who handle all the changes in the DCBTS.

#### **Attributes:**

AdminID – A unique identifier to identify each admin. (Primary Key)

Username – The username of the admin with which they want to login. Password – The password of the admin.

Email – E-mail ID of the admin.

PhoneNumber – The contact number of the admin. EmployeeID – The employee identifier of the admin.

#### **Entity:** Bus

**Description:** A table to describe all the properties of a bus.

#### **Attributes:**

BusID – A unique identifier to identify each bus. (Primary Key)

BusNumber – A designated bus number to identify the route of the bus. Capacity – The seating capacity of the bus.

LicensePlateNumber – The license plate number of the bus. BusType – The type of the bus. E.g.: Coach

**Entity:** Ticket Type

**Description:** A table to describe the type of each ticket.

**Attributes:**

TicketTypeID – A unique identifier to identify each ticket type.(Primary Key)

TypeName – Type name of the ticket. E.g.: Student ticket  
Price – Price of the ticket.

ValidityPeriod – Validity period of the ticket.

Description – A description to explain which ticket applies to what age group.

UserID – (Foreign Key) to relate ticket type and user.

**Entity:** Payment

**Description:** A table to describe the payments for tickets.

**Attributes:**

PaymentID – A unique identifier for each payment. (Primary Key)  
Amount – Amount paid in the payment.

UserID – Identifier of the user who made the payment. (Foreign Key)

TicketTypeID – Identifier of the ticket type for which the payment has been made. (Foreign Key)

PaymentDate – The date on which the payment is made.

**Entity:** Reservation

**Description:** A table to describe all the seating reservations on the bus.

**Attributes:**

ReservationID – A unique identifier for each reservation. (Primary Key)

PaymentID – Payment identifier with which the seating reservation was made. (Foreign Key)

BusID – Identifier of the bus on which the reservation is made. (Foreign Key)

ReservationDate – The date for which the reservation is made.

NumberOfReservations – No. of reservations that have been made.

**Entity:** Bus Route

**Description:** A table to describe all the bus routes in the Dutchess County Bus Transportation System.

**Attributes:**

RouteID – A unique identifier to identify each route. (Primary Key)

StartLocation – Start location of the route.

EndLocation – End location of the route.

BusID – Identifier of the bus that will go on that route. (Foreign Key)

**Entity:** Route Stop Sequence

**Description:** A table to describe the route stop sequence from a source to a destination.

**Attributes:**

StopSequenceID – A unique identifier to identify a route stop sequence. (Primary Key)

RouteID – Route identifier to which the stop sequence is associated. (ForeignKey)

StopID – Identifier of every stop in the stop sequence. (Foreign Key)

NumberOfStops – No. of stops in the route sequence.

ArrivalTime – The time at which the destination will arrive.

**Entity:** Employee

**Description:** A table to describe all the employees employed in the DCBTS.

**Attributes:**

EmployeeID – A unique identifier to identify each employee. (Primary Key)

DepartmentID – Identifier of the department in which the employee is employed. (Foreign Key)

FirstName – First name of the employee. LastName – Last name of the employee. Position – Designation of the employee.

**Entity:** Department

**Description:** A table to describe all the departments in the DCBTS.

**Attributes:**

DepartmentID – A unique identifier to identify each department. (PrimaryKey)

DepartmentName – Name of the department. Location – Location of the department.

**Entity:** Schedule

**Description:** A table to describe the bus schedule.

**Attributes:**

ScheduleID – An identifier to identify a schedule. (Primary Key) BusID –

Identifier of the bus. (Foreign Key)

RouteID – Identifier of the routes. (Foreign Key) DayFlag – Flag to identify the day of week.

DepartureTime – Time when the bus departs. ArrivalTime – Time when the bus arrives.

**Entity:** Bus Stop

**Description:** A table to describe all the bus stops in DCBTS.

**Attributes:**

StopID – A unique identifier to identify each stop. (Primary Key) StopName –

The name of the stop.

Location – The location of the stop.

**Entity:** Notification

**Description:** A table to describe all the notifications that address festivals and other holidays.

**Attributes:**

NotificationID – A unique identifier to identify each notification. (PrimaryKey)

HolidayID – Identifier of the holiday to which the notification is associated. (Foreign Key)

ReservationID – Identifier of the reservation which will trigger a notification. (Foreign Key)

Message – Message content of the notification.

DateAndTime – Date and Time of when the notification will be displayed.



UserID – The identifier of the user who would see the notification. (ForeignKey)

AdminID – The identifier of the admin who would create the notification.  
(Foreign Key)

**Entity:** Holidays

**Description:** A table to describe the holidays during which the buses will not be available.

**Attributes:**

HolidayID – A unique identifier to identify each holiday. (Primary Key)

HolidayDate – The date on which the holiday falls.

Description – Description of the holiday.

NotificationID – Identifier of the notification that will display the about this holiday. (Foreign Key)

**Multivalued Attributes:**

- In the Ticket Type table, Price and ValidityPeriod can have multiple attributes.
- In the Route Stop Sequence table, the NumberOfStops attribute is multi-valued as it indicates the count of stops on a route.
- In the Reservation table, the NumberOfReservations attribute is multi-valued.

**Composite Attributes:**

- In the User and Employee table, we have User's name and Employee's name which will be a combination of FirstName and LastName.

**Derived Attributes:**

- In the Payment table, the Amount attribute can be derived from the Price in the Ticket Type.

**Weak entity:**

- Notification: The "Notification" entity is a weak entity because it relies on either "Admin" or "User" for its existence.

**Strong entity:**

- Bus, Bus Route, Ticket Type, User, Bus Stop, Schedule, Holidays, Route stop sequence, Payment, Reservation, Employee, and Department are strong entities that do not depend on any other entities for their existence or identity.

**Participations:**

**Total Participation:**

- Each payment must be associated with a user, indicating total participation.
- Each department should be associated with Employee, indicating total participation.
- Each route stop sequence should be associated with bus stop, indicating total participation.
- Each bus can be associated with many routes in different schedules same vice versa, which indicates Bus, Route, Schedule a total participation.

**Partial Participation:**

- Users may or may not have associated records in other tables.
- Not all administrators need to have associated employee records.
- Not all holidays may have associated notification records.
- Bus may or may not be associated with reservation.
- Not all notifications need to be associated with User and Admin.
- Not all reservations may be associated with notification records.

## Cardinality Ratios:

### 1-N Cardinality Ratio:

- User: One user can make multiple payments (1-N cardinality).
- Notification: One admin or user can have multiple notifications (1-N cardinality).
- Payment: One user can make multiple payments (1-N cardinality).
- Reservation: One user can make multiple reservations (1-N cardinality).
- Employee: One department can have multiple employees (1-N cardinality).
- BusStop: Each bus stop can be involved with multiple route stop sequence (1-N cardinality).

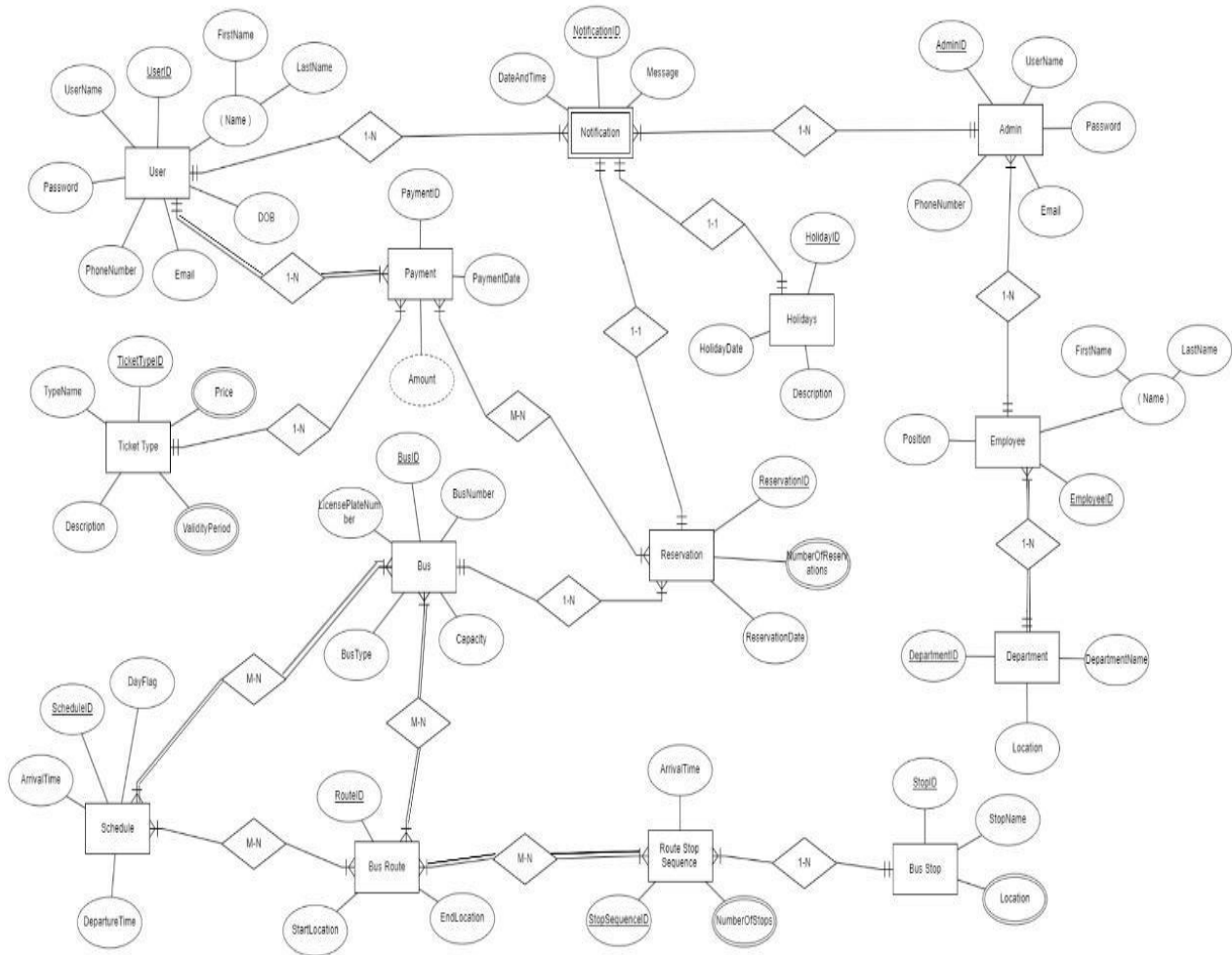
### 1-1 Cardinality Ratio:

- Reservation: Each reservation is associated with one notification (1-1 cardinality).
- Holidays: Each holiday is associated with one notification (1-1 cardinality).

### M-N Cardinality Ratio:

- Route stop sequence: Many routes can have many stops.
- Bus: Multiple buses can be associated with multiple routes.
- Payment: multiple payments lead to multiple reservations.
- Schedule: Different schedules linked to available buses.

## ENTITY RELATIONSHIP DIAGRAM



**Figure 6.1 Entity Relationship**

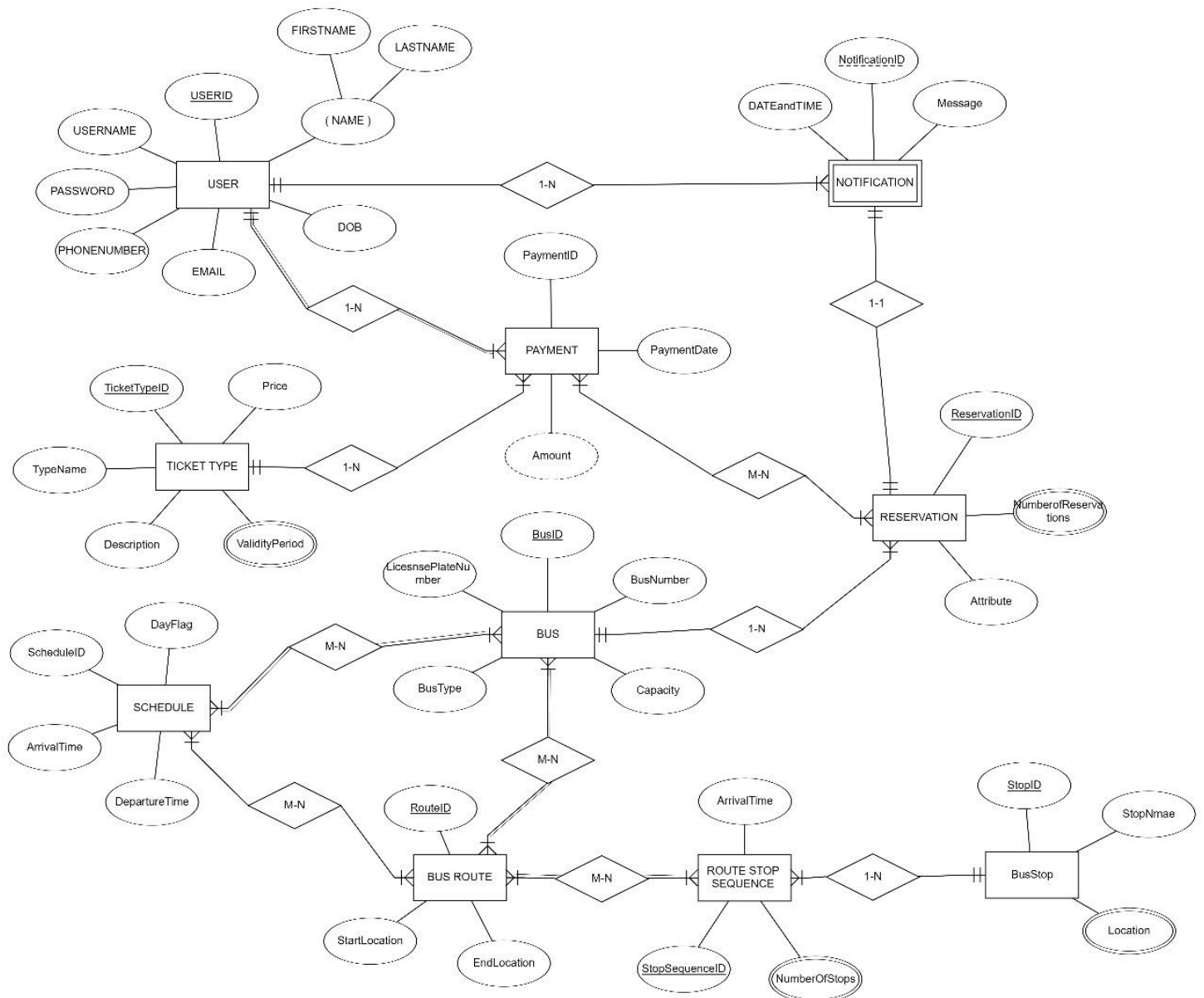


Figure 6. 2 External ER USER POV

As an external ER user, I can see that the system is designed to help me manage my bus reservations. I can use the system to view available bus routes and schedules, make reservations, and pay for my tickets. I can also use the system to track my reservations and receive notifications about changes to my schedule.

The system is easy to use and navigate. I can quickly find the information I need and make reservations with just a few clicks. I appreciate that the system provides me with real-time information about bus availability and schedules. This helps me to plan my trips and make sure that I am on time for my reservations.

I am also impressed with the system's security features. I can feel confident that my personal information is safe and secure when I make a reservation. Overall, I am very satisfied with the external ER user experience of this system.

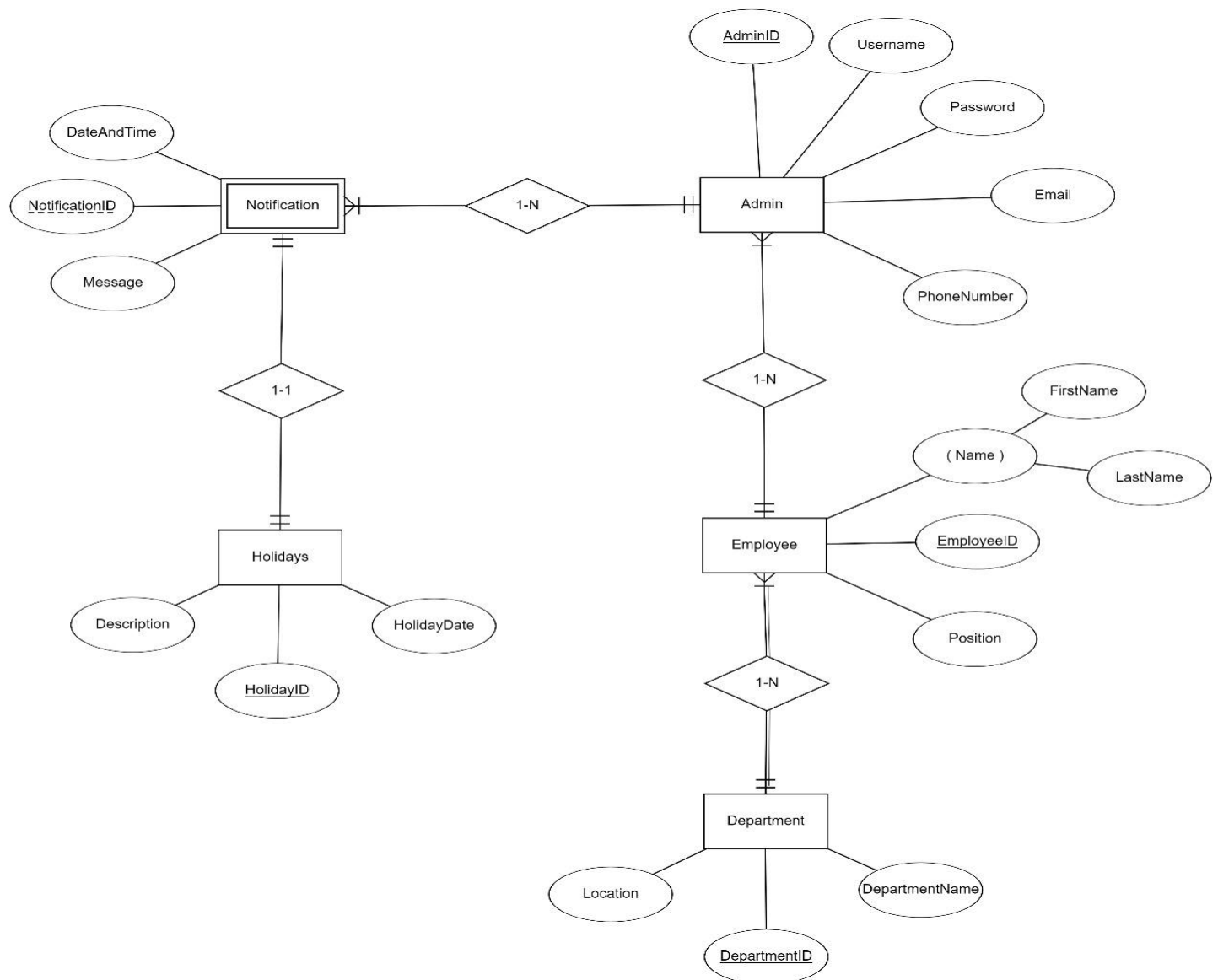


Figure 6. 3 External ER ADMIN POV

As an external ER admin, I am responsible for managing the system that allows employees to communicate with each other. This system is critical to our organization, as it allows employees to stay connected and collaborate on projects.

The system is easy to use and navigate. Employees can quickly find the information they need and communicate with each other with just a few clicks. I appreciate that the system provides me with real-time information about employee activity. This helps me to identify and address any potential issues early on.

I am also impressed with the system's security features. I can feel confident that our employees' communications are safe and secure. Overall, I am very satisfied with the external ER admin experience of this system.

The system is well-designed and meets the needs of our organization. I am confident that it will continue to be a valuable tool for our employees in the future.










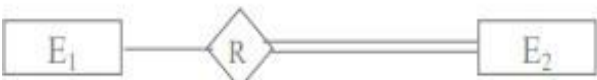

<u>Symbol</u>	<u>Meaning</u>
	ENTITY TYPE
	WEAK ENTITY TYPE
	RELATIONSHIP TYPE
	IDENTIFYING RELATIONSHIP TYPE
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF E <sub>2</sub> IN R
	CARDINALITY RATIO 1:N FOR E <sub>1</sub> :E <sub>2</sub> IN R

Figure 6. 4 Relationship of Entity Relationship Model

## **7.ENHANCED ENTITY RELATIONSHIP MODEL(EER MODEL)**

EER Diagram, also abbreviated as Enhanced Entity-relationship diagram, helps us create and maintain detailed databases through high-level models and tools. In addition, they are developed on the basic ER diagrams and are its extended version.

EER Diagrams basically help in creating and maintaining excellent databases with the help of smart and efficient techniques. It is a visual representation of the plan or the overall outlook of the database you intend to create.

We have used MySQL Workbench to create the EER diagram. Enhanced Entity-Relationship (EER) diagrams are an essential part of the modeling interface in MySQL Workbench. EER diagrams provide a visual representation of the relationships among the tables in our model.

When the application became complex, the traditional ER model was not enough to draw a sophisticated diagram. Therefore, the ER model was developed further. It is known as the Enhanced ER diagram. There are three concepts added to the existing ER model in the Enhanced ER diagram (EER). Those are generalization, specialization, and aggregation. In generalization, the lower-level entities can be combined to produce a higher-level entity. Specialization is the opposite of generalization. In specialization, the high-level entities can be divided into lower-level entities. Aggregation is a process when the relation between two entities is treated as a single entity.

Additionally, it includes the concepts of a subclass and superclass (Is-a). Furthermore, it introduces the concept of a union type or category, which represents a collection of objects that is the union of objects of different entity types. The EER model also includes EER diagrams which are conceptual models that accurately represent the requirements of complex databases.



## 7.1 IMPLEMENTATION OF EER

This Enhanced Entity Relationship (EER) diagram offers a comprehensive model for a complex system comprising essential entities, including payment, employee, admin, ticket type, time, date, last name, department, name, and Sunday Bus schedules. Each entity plays a distinctive role within the database structure. Payment, for instance, is intricately connected to a specific ticket type, employee, and date, ensuring precise payment records. Meanwhile, employees and admins are associated with departments and identified by their First name and Last name, forming the backbone of personnel management.

Ticket types are uniquely defined by their names and prices, preventing any duplications, while time and date entities offer precise temporal and date-based representations. Names act as versatile identifiers across various domains, connecting entities within the system. Additionally, Sunday schedules are distinctly linked to a department, date, and time for efficient planning.

The relationships between these entities are systematically outlined, ensuring data integrity and retrieval efficiency. For example, strict associations like payments being tied to one ticket type, employee, and date eliminate any chances of duplicate payments. This design not only safeguards data but also streamlines data retrieval, making it easier to extract meaningful information from the database.

In conclusion, this well-designed EER diagram offers an efficient model for the system's organization. It ensures data integrity, enhances data retrieval performance, and provides clarity, making it a valuable tool for managing and organizing complex data structures within the described system.

## ENHANCED ENTITY RELATIONSHIP DIAGRAM

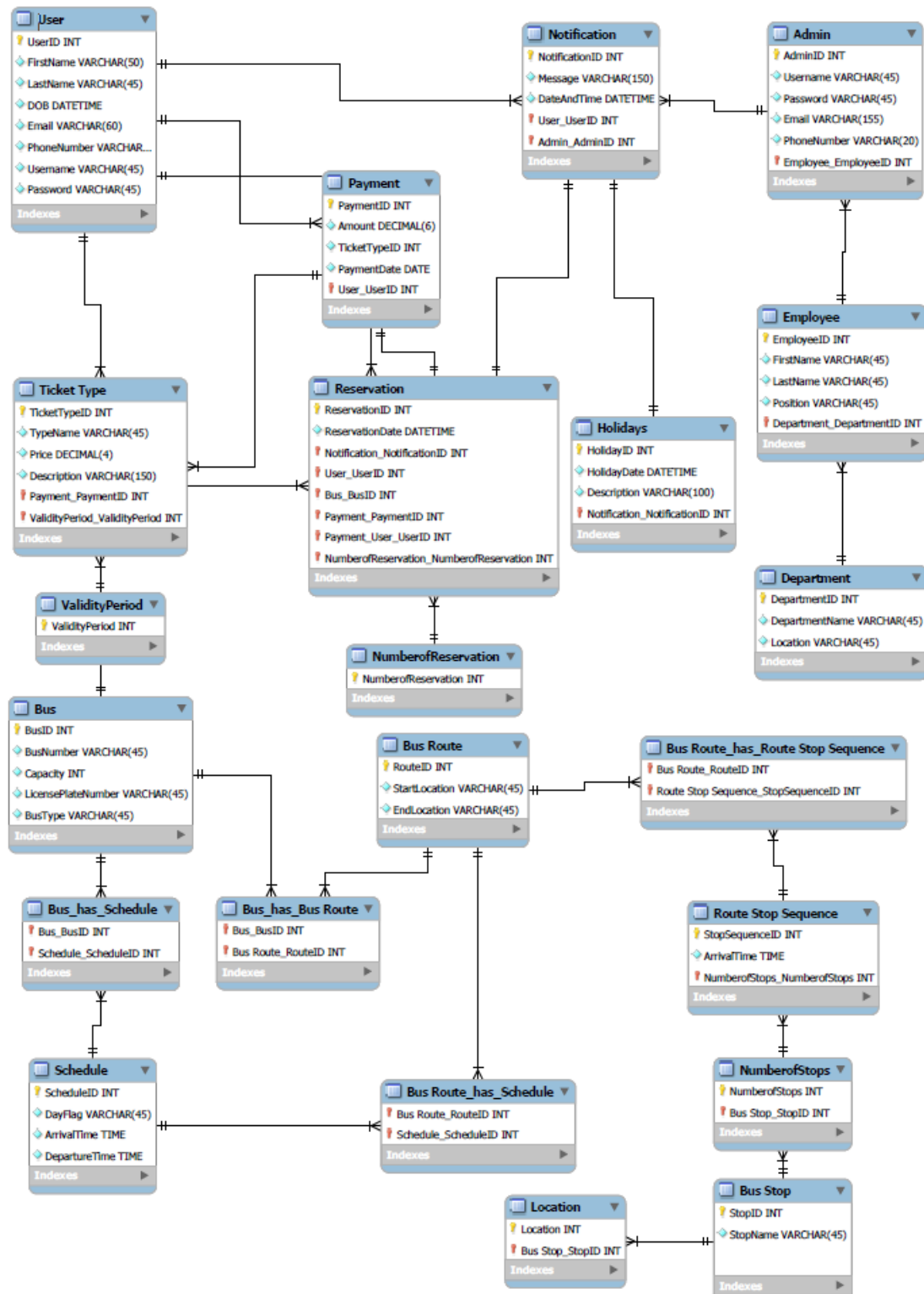


Figure 7.1 Enhanced Entity Relationship

## 8.DATABASE DEVELOPMENT

### 8.1 CREATE STATEMENTS

-- MySQL Workbench Forward Engineering

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

-----

-- Schema Dutchess\_county\_bus\_transportation\_DBMS\_project

-----

-----

-- Schema Dutchess\_county\_bus\_transportation\_DBMS\_project

-----

```
CREATE SCHEMA IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`
DEFAULT CHARACTER SET utf8 ;
```

-----

-- Schema SQL

-----

```
USE `Dutchess_county_bus_transportation_DBMS_project` ;
```

-----

-- Table `Dutchess\_county\_bus\_transportation\_DBMS\_project`.`User`

-----

```
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`User` (
  `UserID` INT NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(50) NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `DOB` DATETIME NOT NULL,
  `Email` VARCHAR(60) NOT NULL,
  `PhoneNumber` BIGINT(14) NOT NULL,
  `Username` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UserID`),
  UNIQUE INDEX `Username_UNIQUE` (`Username` ASC) VISIBLE,
  UNIQUE INDEX `UserID_UNIQUE` (`UserID` ASC) VISIBLE)
ENGINE = InnoDB;
```

-----

-- Table `Dutchess\_county\_bus\_transportation\_DBMS\_project`.`Bus`

-----

```
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Bus` (
  `BusID` INT NOT NULL AUTO_INCREMENT,
  `BusNumber` VARCHAR(45) NOT NULL,
  `Capacity` INT NOT NULL,
  `LicensePlateNumber` VARCHAR(45) NOT NULL,
```

```
`BusType` VARCHAR(45) NOT NULL,
PRIMARY KEY (`BusID`),
UNIQUE INDEX `BusID_UNIQUE` (`BusID` ASC) VISIBLE,
UNIQUE INDEX `BusNumber_UNIQUE` (`BusNumber` ASC) VISIBLE,
UNIQUE INDEX `LicensePlateNumber_UNIQUE` (`LicensePlateNumber` ASC) VISIBLE)
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Bus Route`
-----
```

```
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Bus
Route` (
  `RouteID` INT NOT NULL AUTO_INCREMENT,
  `StartLocation` VARCHAR(45) NOT NULL,
  `EndLocation` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`RouteID`))
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Ticket Type`
-----
```

```
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Ticket
Type` (
  `TicketTypeID` INT NOT NULL,
  `TypeName` VARCHAR(45) NOT NULL,
  `Price` DECIMAL(4) NOT NULL,
  `ValidityPeriod` INT NOT NULL,
  `Description` VARCHAR(150) NOT NULL,
  PRIMARY KEY (`TicketTypeID`),
  UNIQUE INDEX `TicketTypeID_UNIQUE` (`TicketTypeID` ASC) VISIBLE,
  UNIQUE INDEX `TypeName_UNIQUE` (`TypeName` ASC) VISIBLE)
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Department`
-----
```

```
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Department` (
  `DepartmentID` INT NOT NULL AUTO_INCREMENT,
  `DepartmentName` VARCHAR(45) NOT NULL,
  `Location` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`DepartmentID`),
  UNIQUE INDEX `DepartmentID_UNIQUE` (`DepartmentID` ASC) VISIBLE)
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Employee`
-----
```

```
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Employee` (
```

```
`EmployeeID` INT NOT NULL AUTO_INCREMENT,
`FirstName` VARCHAR(45) NOT NULL,
`LastName` VARCHAR(45) NOT NULL,
`Position` VARCHAR(45) NOT NULL,
`Department_DepartmentID` INT NOT NULL,
PRIMARY KEY (`EmployeeID`, `Department_DepartmentID`),
UNIQUE INDEX `EmployeeID_UNIQUE` (`EmployeeID` ASC) VISIBLE,
INDEX `fk_Employee_Department1_idx` (`Department_DepartmentID` ASC) VISIBLE,
CONSTRAINT `fk_Employee_Department1`
  FOREIGN KEY (`Department_DepartmentID`)
    REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Department`
      (`DepartmentID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Admin`
-- -----
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Admin`
(
  `AdminID` INT NOT NULL AUTO_INCREMENT,
  `Username` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(155) NOT NULL,
  `PhoneNumber` BIGINT(14) NOT NULL,
  `Employee_EmployeeID` INT NOT NULL,
  PRIMARY KEY (`AdminID`, `Employee_EmployeeID`),
  UNIQUE INDEX `AdminID_UNIQUE` (`AdminID` ASC) VISIBLE,
  UNIQUE INDEX `Email_UNIQUE` (`Email` ASC) VISIBLE,
  UNIQUE INDEX `PhoneNumber_UNIQUE` (`PhoneNumber` ASC) VISIBLE,
  INDEX `fk_Admin_Employee1_idx` (`Employee_EmployeeID` ASC) VISIBLE,
  CONSTRAINT `fk_Admin_Employee1`
    FOREIGN KEY (`Employee_EmployeeID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Employee`
        (`EmployeeID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Bus Stop`
-- -----
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Bus
Stop` (
  `StopID` INT NOT NULL AUTO_INCREMENT,
  `StopName` VARCHAR(45) NOT NULL,
  `Location` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`StopID`),
  UNIQUE INDEX `StopID_UNIQUE` (`StopID` ASC) VISIBLE)
ENGINE = InnoDB;
```

```

-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Notification`
-----
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Notification` (
  `NotificationID` INT NOT NULL AUTO_INCREMENT,
  `Message` VARCHAR(150) NOT NULL,
  `DateAndTime` DATETIME NOT NULL,
  `User_UserID` INT NOT NULL,
  `Admin_AdminID` INT NOT NULL,
  PRIMARY KEY (`NotificationID`, `User_UserID`, `Admin_AdminID`),
  UNIQUE INDEX `NotificationID_UNIQUE` (`NotificationID` ASC) VISIBLE,
  INDEX `fk_Notification_User1_idx` (`User_UserID` ASC) VISIBLE,
  INDEX `fk_Notification_Admin1_idx` (`Admin_AdminID` ASC) VISIBLE,
  CONSTRAINT `fk_Notification_User1`
    FOREIGN KEY (`User_UserID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`User` (`UserID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_Notification_Admin1`
    FOREIGN KEY (`Admin_AdminID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Admin` (`AdminID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Holidays`
-----
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Holidays` (
  `HolidayID` INT NOT NULL AUTO_INCREMENT,
  `HolidayDate` DATETIME NOT NULL,
  `Description` VARCHAR(100) NOT NULL,
  `Notification_NotificationID` INT NOT NULL,
  PRIMARY KEY (`HolidayID`, `Notification_NotificationID`),
  INDEX `fk_Holidays_Notification1_idx` (`Notification_NotificationID` ASC) VISIBLE,
  CONSTRAINT `fk_Holidays_Notification1`
    FOREIGN KEY (`Notification_NotificationID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Notification`
        (`NotificationID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Route Stop Sequence`
-----
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Route
Stop Sequence` (
  `StopSequenceID` INT NOT NULL AUTO_INCREMENT,
  `NumberOfStops` INT NOT NULL,

```

```
`ArrivalTime` DATETIME NOT NULL,
`Bus Stop_StopID` INT NOT NULL,
PRIMARY KEY (`StopSequenceID`, `Bus Stop_StopID`),
UNIQUE INDEX `StopSequenceID_UNIQUE` (`StopSequenceID` ASC) VISIBLE,
INDEX `fk_Route Stop Sequence_Bus Stop1_idx` (`Bus Stop_StopID` ASC) VISIBLE,
CONSTRAINT `fk_Route Stop Sequence_Bus Stop1`
  FOREIGN KEY (`Bus Stop_StopID`)
    REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Bus Stop` (`StopID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Payment`
-----
```

```
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Payment` (
  `PaymentID` INT NOT NULL,
  `Amount` DECIMAL(6) NOT NULL,
  `PaymentDate` DATETIME NOT NULL,
  `User_UserID` INT NOT NULL,
  `Ticket Type_TicketTypeID` INT NOT NULL,
  PRIMARY KEY (`PaymentID`, `User_UserID`, `Ticket Type_TicketTypeID`),
  INDEX `fk_Payment_User1_idx` (`User_UserID` ASC) VISIBLE,
  INDEX `fk_Payment_Ticket Type1_idx` (`Ticket Type_TicketTypeID` ASC) VISIBLE,
  CONSTRAINT `fk_Payment_User1`
    FOREIGN KEY (`User_UserID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`User` (`UserID`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Payment_Ticket Type1`
    FOREIGN KEY (`Ticket Type_TicketTypeID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Ticket Type`
      (`TicketTypeID`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Reservation`
-----
```

```
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Reservation` (
  `ReservationID` INT NOT NULL AUTO_INCREMENT,
  `ReservationDate` DATETIME NOT NULL,
  `NumberOfReservations` INT NOT NULL,
  `Notification_NotificationID` INT NOT NULL,
  `Bus_BusID` INT NOT NULL,
  PRIMARY KEY (`ReservationID`, `Notification_NotificationID`, `Bus_BusID`),
  INDEX `fk_Reservation_Notification1_idx` (`Notification_NotificationID` ASC) VISIBLE,
  INDEX `fk_Reservation_Bus1_idx` (`Bus_BusID` ASC) VISIBLE,
  CONSTRAINT `fk_Reservation_Notification1`
    FOREIGN KEY (`Notification_NotificationID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Notification` (`NotificationID`)
```



```
REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Notification`
(`NotificationID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Reservation_Bus1`
FOREIGN KEY (`Bus_BusID`)
REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Bus` (`BusID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Bus_has_Bus Route`
-----
```

```
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Bus_has_Bus Route` (
  `Bus_BusID` INT NOT NULL,
  `Bus Route_RouteID` INT NOT NULL,
  PRIMARY KEY (`Bus_BusID`, `Bus Route_RouteID`),
  INDEX `fk_Bus_has_Bus Route_Bus Route1_idx` (`Bus Route_RouteID` ASC) VISIBLE,
  INDEX `fk_Bus_has_Bus Route_Bus1_idx` (`Bus_BusID` ASC) VISIBLE,
  CONSTRAINT `fk_Bus_has_Bus Route_Bus1`
    FOREIGN KEY (`Bus_BusID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Bus` (`BusID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bus_has_Bus Route_Bus Route1`
    FOREIGN KEY (`Bus Route_RouteID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Bus Route` (`RouteID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Bus Route_has_Route Stop
Sequence`
-----
```

```
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Bus
Route_has_Route Stop Sequence` (
  `Bus Route_RouteID` INT NOT NULL,
  `Route Stop Sequence_StopSequenceID` INT NOT NULL,
  PRIMARY KEY (`Bus Route_RouteID`, `Route Stop Sequence_StopSequenceID`),
  INDEX `fk_Bus Route_has_Route Stop Sequence_Route Stop Sequence1_idx` (`Route Stop
Sequence_StopSequenceID` ASC) VISIBLE,
  INDEX `fk_Bus Route_has_Route Stop Sequence_Bus Route1_idx` (`Bus Route_RouteID` ASC)
VISIBLE,
  CONSTRAINT `fk_Bus Route_has_Route Stop Sequence_Bus Route1`
    FOREIGN KEY (`Bus Route_RouteID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Bus Route` (`RouteID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bus Route_has_Route Stop Sequence_Route Stop Sequence1`
    FOREIGN KEY (`Route Stop Sequence_StopSequenceID`)
```



```
REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Route Stop Sequence`
(`StopSequenceID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Schedule`
-----
```

```
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Schedule` (
  `ScheduleID` INT NOT NULL AUTO_INCREMENT,
  `DayFlag` VARCHAR(45) NOT NULL,
  `ArrivalTime` DATETIME NOT NULL,
  `DepartureTime` DATETIME NOT NULL,
  PRIMARY KEY (`ScheduleID`),
  UNIQUE INDEX `ScheduleID_UNIQUE` (`ScheduleID` ASC) VISIBLE)
ENGINE = InnoDB;
```

```
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Bus_has_Schedule`
-----
```

```
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Bus_has_Schedule` (
  `Bus_BusID` INT NOT NULL,
  `Schedule_ScheduleID` INT NOT NULL,
  PRIMARY KEY (`Bus_BusID`, `Schedule_ScheduleID`),
  INDEX `fk_Bus_has_Schedule_Schedule1_idx` (`Schedule_ScheduleID` ASC) VISIBLE,
  INDEX `fk_Bus_has_Schedule_Bus1_idx` (`Bus_BusID` ASC) VISIBLE,
  CONSTRAINT `fk_Bus_has_Schedule_Bus1`
    FOREIGN KEY (`Bus_BusID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Bus` (`BusID`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Bus_has_Schedule_Schedule1`
    FOREIGN KEY (`Schedule_ScheduleID`)
      REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Schedule` (`ScheduleID`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Bus Route_has_Schedule`
-----
```

```
CREATE TABLE IF NOT EXISTS `Dutchess_county_bus_transportation_DBMS_project`.`Bus
Route_has_Schedule` (
  `Bus Route_RouteID` INT NOT NULL,
  `Schedule_ScheduleID` INT NOT NULL,
  PRIMARY KEY (`Bus Route_RouteID`, `Schedule_ScheduleID`),
  INDEX `fk_Bus Route_has_Schedule_Schedule1_idx` (`Schedule_ScheduleID` ASC) VISIBLE,
  INDEX `fk_Bus Route_has_Schedule_Bus Route1_idx` (`Bus Route_RouteID` ASC) VISIBLE,
  CONSTRAINT `fk_Bus Route_has_Schedule_Bus Route1`
```

```

FOREIGN KEY (`Bus Route_RouteID`)
REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Bus Route` (`RouteID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Bus Route_has_Schedule_Schedule1`
FOREIGN KEY (`Schedule_ScheduleID`)
REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Schedule` (`ScheduleID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `Dutchess_county_bus_transportation_DBMS_project`.`Payment_has_Reservation`
-----
CREATE TABLE IF NOT EXISTS
`Dutchess_county_bus_transportation_DBMS_project`.`Payment_has_Reservation` (
  `Payment_PaymentID` INT NOT NULL,
  `Payment_User_UserID` INT NOT NULL,
  `Payment_Ticket Type_TicketTypeID` INT NOT NULL,
  `Reservation_ReservationID` INT NOT NULL,
  `Reservation_Notification_NotificationID` INT NOT NULL,
  `Reservation_Bus_BusID` INT NOT NULL,
  PRIMARY KEY (`Payment_PaymentID`, `Payment_User_UserID`, `Payment_Ticket
Type_TicketTypeID`, `Reservation_ReservationID`, `Reservation_Notification_NotificationID`,
`Reservation_Bus_BusID`),
  INDEX `fk_Payment_has_Reservation_Reservation1_idx` (`Reservation_ReservationID` ASC,
`Reservation_Notification_NotificationID` ASC, `Reservation_Bus_BusID` ASC) VISIBLE,
  INDEX `fk_Payment_has_Reservation_Payment1_idx` (`Payment_PaymentID` ASC,
`Payment_User_UserID` ASC, `Payment_Ticket Type_TicketTypeID` ASC) VISIBLE,
  CONSTRAINT `fk_Payment_has_Reservation_Payment1`
FOREIGN KEY (`Payment_PaymentID`, `Payment_User_UserID`, `Payment_Ticket
Type_TicketTypeID`)
REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Payment` (`PaymentID`,
`User_UserID`, `Ticket Type_TicketTypeID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Payment_has_Reservation_Reservation1`
FOREIGN KEY (`Reservation_ReservationID`, `Reservation_Notification_NotificationID`,
`Reservation_Bus_BusID`)
REFERENCES `Dutchess_county_bus_transportation_DBMS_project`.`Reservation`
(`ReservationID`, `Notification_NotificationID`, `Bus_BusID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

## 8.2 INSERT STATEMENTS

```

use dutchess_county_bus_transportation_dbms_project;
show tables;
-- Insert schedules
INSERT INTO Schedule (DayFlag, ArrivalTime, DepartureTime)
VALUES
    ('Monday', '06:00:00', '06:05:00'),
    ('Monday', '06:10:00', '06:15:00'),
    ('Tuesday', '07:00:00', '07:05:00'),
    ('Wednesday', '08:30:00', '08:35:00'),
    ('Thursday', '15:45:00', '15:50:00'),
    ('Friday', '13:10:00', '13:15:00'),
    ('Saturday', '11:15:00', '11:20:00'),
    ('Sunday', '09:30:00', '09:35:00');

-- Insert bus routes
INSERT INTO BusRoute (StartLocation, EndLocation)
VALUES
    ('Poughkeepsie', 'Fishkill'),
    ('Poughkeepsie', 'Beacon'),
    ('Poughkeepsie', 'Tivoli'),
    ('Poughkeepsie', 'Wassaic'),
    ('Poughkeepsie', 'Pawling'),
    ('Beacon', 'Hopewell Junction');
select *from busroute;

-- Insert bus stops
INSERT INTO BusStop (StopName, Location)
VALUES
    ('Poughkeepsie Station', '1 Station Plaza, Poughkeepsie, NY'),
    ('Beacon Station', '223 Fishkill Ave, Beacon NY'),
    ('Hopewell Junction', '123 Main St, Hopewell Junction, NY'),
    ('Tivoli', '456 Broadway, Tivoli, NY'),
    ('Fishkill', '789 Main St, Fishkill, NY'),
    ('Wassaic', '234 Rail Rd, Wassaic, NY');

-- Insert bus stop arrival times
INSERT INTO RouteStopSequence (NumberofStops, ArrivalTime, BusStop_StopID)
VALUES
    (1, '06:00:00', 1),
    (2, '06:15:00', 2),
    (3, '06:30:00', 3),
    (4, '06:45:00', 4),
    (5, '07:00:00', 5),
    (6, '07:15:00', 6);

-- Assign buses to schedules
INSERT INTO Bus_has_Schedule (Bus_BusID, Schedule_ScheduleID)
VALUES
    (1, 1),
    (2, 2),

```

```
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8);
```

-- Assign buses to routes

```
INSERT INTO Bus_has_BusRoute (Bus_BusID, BusRoute_RouteID)
```

```
VALUES
```

```
(1, 1),
(2, 2),
(17, 3),
(18, 4),
(19, 5),
(20, 6);
```

```
DESCRIBE Bus_has_BusRoute;
```

-- Insert ticket types

```
INSERT INTO TicketType (TicketTypeID, TypeName, Price, ValidityPeriod, Description)
```

```
VALUES
```

```
(1, 'Day Pass', 5.00, 1, 'Valid for one day'),
(2, '7-day Pass', 20.00, 7, 'Valid for 7 consecutive days'),
(3, '31-day Pass', 70.00, 31, 'Valid for 31 consecutive days');
```

-- Insert data into User table

```
INSERT INTO User (FirstName, LastName, DOB, Email, PhoneNumber, Username, Password)
```

```
VALUES
```

```
('John', 'Doe', '1990-01-01', 'john@example.com', '123-456-7890', 'johndoe', 'password123'),
('Jane', 'Smith', '1995-05-15', 'jane@example.com', '987-654-3210', 'janesmith', 'password456');
```

-- Insert data into Bus table

```
INSERT INTO Bus (BusNumber, Capacity, LicensePlateNumber, BusType)
```

```
VALUES
```

```
('A2', 50, 'ABC126', 'Coach'),
('C', 60, 'GHI789', 'School Bus'),
('D', 40, 'JKL012', 'Shuttle'),
('E', 45, 'MNO345', 'Shuttle'),
('F', 35, 'PQR678', 'Mini Bus'),
('H', 30, 'STU901', 'Mini Bus');
```

-- Insert data into Department table

```
INSERT INTO Department (DepartmentName, Location)
```

```
VALUES
```

```
('Operations', '123 Main St, Poughkeepsie, NY'),
('Maintenance', '456 Park Rd, Beacon, NY');
```

-- Insert data into Employee table

```
INSERT INTO Employee (FirstName, LastName, Position, Department_DepartmentID)
```

```
VALUES
```

```
('Sarah', 'Jones', 'Driver', 1),
('Mark', 'Lee', 'Mechanic', 2);
```

-- Insert data into Admin table

```
INSERT INTO Admin (Username, Password, Email, PhoneNumber, Employee_EmployeeID)
```

```
VALUES
```

## MSCS\_542L\_256\_23S\_ProjectProgress Report\_phase04\_The Four-Ce

```
('sarahj','password123', 'sarah@example.com','123-555-1234',1),  
('markl','password456', 'mark@example.com','987-555-4321',2);
```

```
-- Insert data into Payment table
```

```
INSERT INTO Payment (PaymentID, Amount, PaymentDate, User_UserID,  
TicketType_TicketTypeID)
```

```
VALUES
```

```
(1, 5.00, '2023-01-01', 1, 1),  
(2, 20.00, '2023-01-10', 2, 2);
```

```
-- Verify all data insertion
```

```
SELECT * FROM User;
```

```
SELECT * FROM Bus;
```

```
SELECT * FROM Department;
```

```
SELECT * FROM Employee;
```

```
SELECT * FROM Admin;
```

```
SELECT * FROM Payment;
```

```
SELECT * FROM Schedule;
```

```
SELECT * FROM BusRoute;
```

```
SELECT * FROM BusStop;
```

```
SELECT * FROM RouteStopSequence;
```

```
SELECT * FROM Bus_has_Schedule;
```

```
SELECT * FROM Bus_has_BusRoute;
```

```
SELECT * FROM TicketType;
```

## **9.REFERENCES**

1. Dutchess County Public Transportation Application.
2. NYC Transit: MTA Subway and Bus.
3. Moovit.