

Blue Bird

Order Management System

Black Sabbath

Amadeus Darren Leander - 2101628915

Bernard Wijaya - 2101636910

Dave Zachary - 2101671195

Ryan Adipradana - 2101626903

Benny Wiryawan - 1901497546

Computer Science Global Class – LA02

**Preface**

In this project, we created an object-oriented database system to simulate Blue Bird’s taxi ordering service. The form-based application is created using C# Windows Form Application and runs on the Windows operating system. The application involves users, drivers and also the admin who manages the database.

**Table of Contents**

[Preface](#_airrw93fl7rx) 2

[Table of Contents](#_8jhbxdijtjpb) 2

[Requirement Analysis](#_tkshfgyva3fb) 3

[Conceptual Design](#_nshonlepoqy5) 3

[Logical Design](#_842kty6s8qv0) 4

[Object Definition Language](#_dzl35wwlhzac) 5

[Report Generation and Object Manipulation Language](#_wxtsbnlx9552) 7

[Conclusion](#_1070u0aggnbr) 10

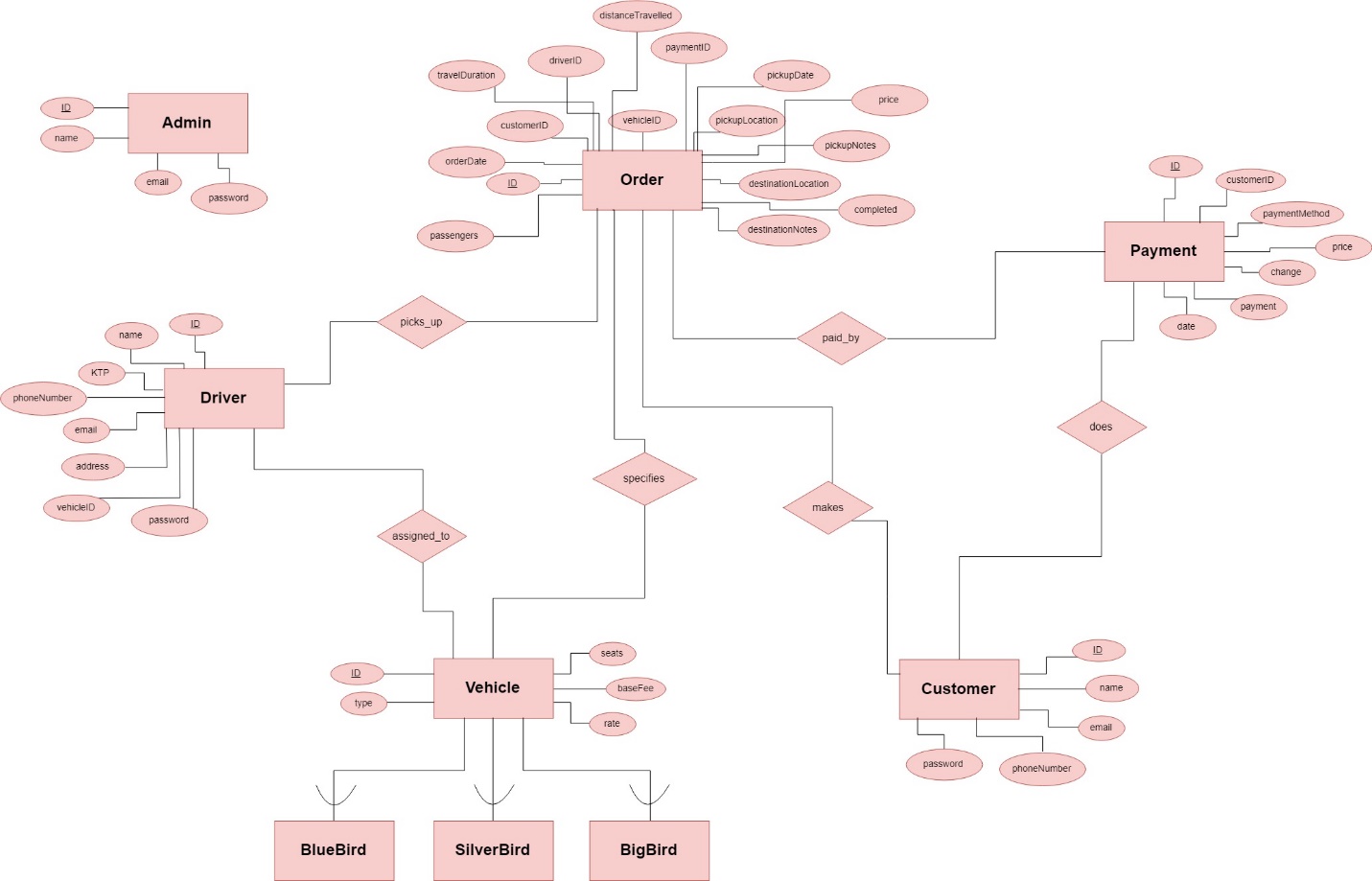
[References](#_3qyo0n6fkuwj) 10

**Requirement Analysis**

To simulate the Blue Bird’s taxi ordering process, first we must understand the individuals and organizations involved in the process. By observing the real-world domain, we concluded that the platform consists of interactions between users, drivers and system admins.   
The entire process flow starts with the user, whose role is to create an order which they can also cancel. Drivers must then accept the order and after picking up and delivering the user to their destination, they must confirm that the ride has been completed. The admin holds the most tasks in the system because they must manage the users, drivers, vehicles and transactions.

**Conceptual Design**

In this process, we model the real-world domain by creating the data model required to fulfill the business requirements. To model it we created the Entity Relationship Diagram below.



**Logical Design**

The logical design is represented by a class diagram below.



**Object Definition Language**

**class** Order (**extent** Order) {

**attribute** string ID;

**attribute** DateTime orderDate;

**attribute** string customerID;

**attribute** string driverID;

**attribute** string vehicleID;

**attribute** string paymentID;

**attribute** string pickupLocation;

**attribute** string pickupNotes;

**attribute** string destinationLocation;

**attribute** string destinationNotes;

**attribute** DateTime pickupDate;

**attribute** int passengers;

**attribute** TimeSpan travelDuration;

**attribute** int distanceTravelled;

**attribute** int price;

**attribute** bool completed;

**relationship** Customer made\_by **inverse** Customer::makes;

**relationship** Driver picked \_by **inverse** Driver::picks\_up;

**relationship** Vehicle specifies **inverse** Vehicle::specified\_by;

**relationship** Payment paid\_by **inverse** Payment::pays;

}

**class** Driver (**extent** Driver) {

**attribute** string ID;

**attribute** string name;

**attribute** string KTP;

**attribute** string phoneNumber;

**attribute** string email;

**attribute** string password;

**attribute** string vehicleID;

**relationship** **list**<Order> picks\_up **inverse** Order::picked\_by;

**relationship** Vehicle assigned\_to **inverse** Vehicle::assigned\_to;

}

**class** Vehicle (**extent** Vehicle) {

**attribute** string ID;

**attribute** string type;

**attribute** int seats;

**attribute** int baseFee;

**attribute** int rate;

**relationship** Driver assigned\_to **inverse** Driver::assigned\_to;

**relationship** Order specified\_by **inverse** Order::specifies;

}

**class** BlueBird **extends** Vehicle (**extent** BlueBird) {

}

**class** SilverBird **extends** Vehicle (**extent** SilverBird) {

}

**class** BigBird **extends** Vehicle (**extent** BigBird) {

}

**class** Customer (**extent** Customer) {

**attribute** string ID;

**attribute** string name;

**attribute** string email;

**attribute** string phoneNumber;

**attribute** string password;

**relationship** **list**<Order> makes **inverse** Order::made\_by;

**relationship** **list**<Payment> does **inverse** Payment::done\_by;

}

**class** Payment (**extent** Payment) {

**attribute** string ID;

**attribute** string customerID;

**attribute** string paymentMethod;

**attribute** int price;

**attribute** int payment;

**attribute** int change;

**attribute** DateTime date;

**relationship** Order pays **inverse** Order::paid\_by;

**relationship** Customer done\_by **inverse** Customer::does;

}

**class** Admin (**extent** Admin) {

**attribute** string ID;

**attribute** string name;

**attribute** string email;

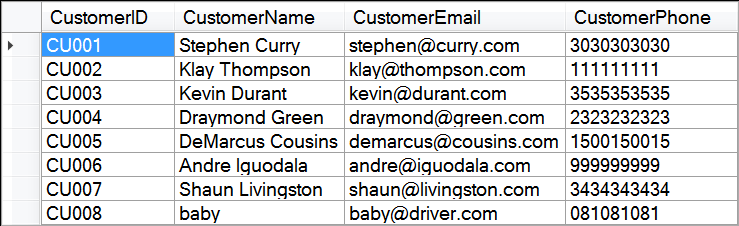
**attribute** string password;

}

**Report Generation and Object Manipulation Language**

Our database system can generate 5 different reports which are: customer report, driver report, vehicle report, order report and admin report.

**Customer Report:**



from Customer x in DB.conn

orderby x.ID ascending

select new

{

CustomerID = x.ID,

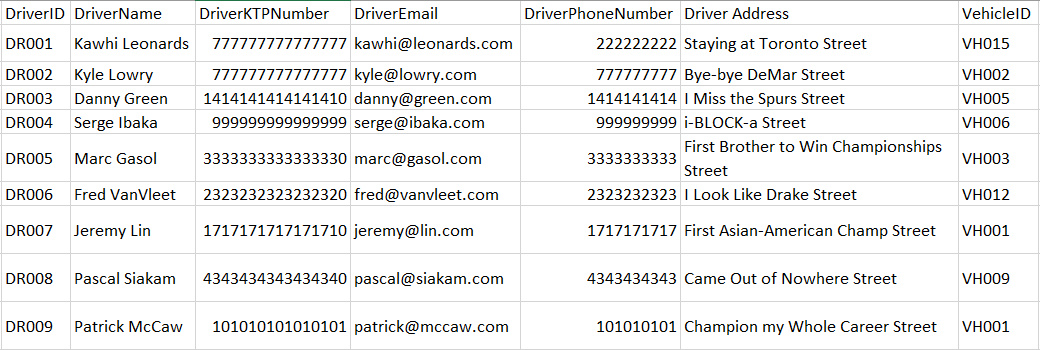
CustomerName = x.name,

CustomerEmail = x.email,

CustomerPhone = x.phoneNumber,

}

**Driver Report:**



from Driver x in DB.conn

orderby x.ID ascending

select new

{

DriverID = x.ID,

DriverName = x.name,

DriverKTPNumber = x.KTP,

DriverEmail = x.email,

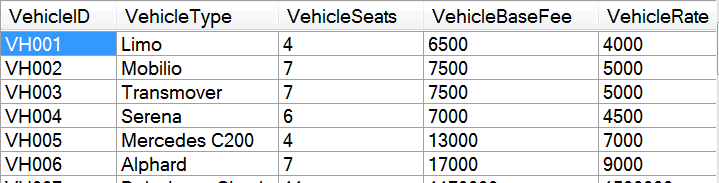
DriverPhoneNumber = x.phoneNumber,

DriverAddress = x.address,

VehicleID = x.vehicleID

}

**Vehicle Report:**



from Vehicle x in DB.conn

orderby x.ID ascending

select new

{

VehicleID = x.ID,

VehicleType = x.type,

VehicleSeats = x.seats,

VehicleBaseFee = x.baseFee,

VehicleRate = x.rate

}

**Order Report:**



from Order x in DB.conn

orderby x.completed ascending, x.pickupDate ascending

select new

{

OrderID = x.ID,

PickupLocation = x.pickupLocation,

Destination = x.destinationLocation,

OrderDate = x.orderDate,

PickupDate = x.pickupDate,

Completed = x.completed

}

**Admin Report:**



from Admin x in DB.conn

orderby x.ID ascending

select new

{

AdminID = x.ID,

AdminName = x.name,

AdminEmail = x.email,

}

**Conclusion**

The conclusion of this project is to simulate and make a prototype of BlueBird taxi ordering system which has its own database. The prototype has several services and features such as:

* Customer features
  + Making orders
  + Finishing orders
* Driver features
  + Pick up orders
* Admin features
  + Manage Customers
  + Manage Drivers
  + Manage Vehicles
  + View Orders
  + Manage Admins
* Report generation
  + Total number of users per service
  + Top 5 most used vehicles
  + Total income earned per month
  + History of completed trips
  + List of ongoing trips

All users must either login or register first before using the application. Due to security reasons, registering through the application is only available to customers.

This Object-Oriented Database System’s project uses C# Windows Form Application as the main programming language and uses db4o, especially LINQ, to create the ODLs and OQLs which are the main components to access and create databases based on our C# code.

**References**

* https://www.bluebirdgroup.com/
* https://www.cnnindonesia.com/ekonomi/20190422203127-92-388640/tarif-taksi-blue-bird-listrik-sama-dengan-konvensional
* https://www.cnbcindonesia.com/news/20181114100359-4-42047/mana-lebih-mahal-tarif-baru-taksi-online-atau-blue-bird-cs
* https://www.bluebirdgroup.com/wp-content/uploads/2019/01/About-Us-BB-Group.jpg
* https://bluebirdgroup.com/wp-content/uploads/2018/11/bluebird-logo-new.png