# Phase V – Final Report

by

Pınar SAVAŞTÜRK

2015510052

December, 2018

İZMİR

Contents

[Phase V – Final Report 1](#_Toc533465131)

[Detail Problem Description 3](#_Toc533465132)

[Solution System and Operation List 3](#_Toc533465133)

[System Constraints 4](#_Toc533465134)

[ER Diagram 5](#_Toc533465135)

[The Database Schema 6](#_Toc533465136)

[Relational Algebra for 3 Queries 6](#_Toc533465137)

[SQL Statements 6](#_Toc533465138)

[Screen Shots 6](#_Toc533465139)

[Used Technology, Tools, and Challenges 7](#_Toc533465141)

## Detail Problem Description

The main problem in the bus tracking systems is renewing the page. These applications were written in ASP, ASP.NET or other old technologies. In this project, I intended to build a bus tracking system with new and popular technologies like Angular6 (ES2017) and Node.js.

In our city Buca, some places don’ t have 4.5G communication, so internet speed is low. In a normal ASP code the page will be loaded until all information is fetched, but in Angular I load all static HTML code and then the program starts waiting for data which will be fetched from the remote server.

With new techonology and techniques, asynchronous codes gain much more attention. Because, a developer can write code like synchronous but code will work asynchronous. This is very important, because cpu intensive or IO operations will not block thread or main thread in program. The developer can inrease the throughput.

Another advatage of Node.js has got cross-platform support, and it is open source project. Also npm packages can be downloaded from anywhere which has got internet connection, this is a pros for Node.js

## Solution System and Operation List

In the project all tables have got insert, delete, update and get operations. I used n-tier architecture in the project. I built an architecture and its can be seen in Fig.1.

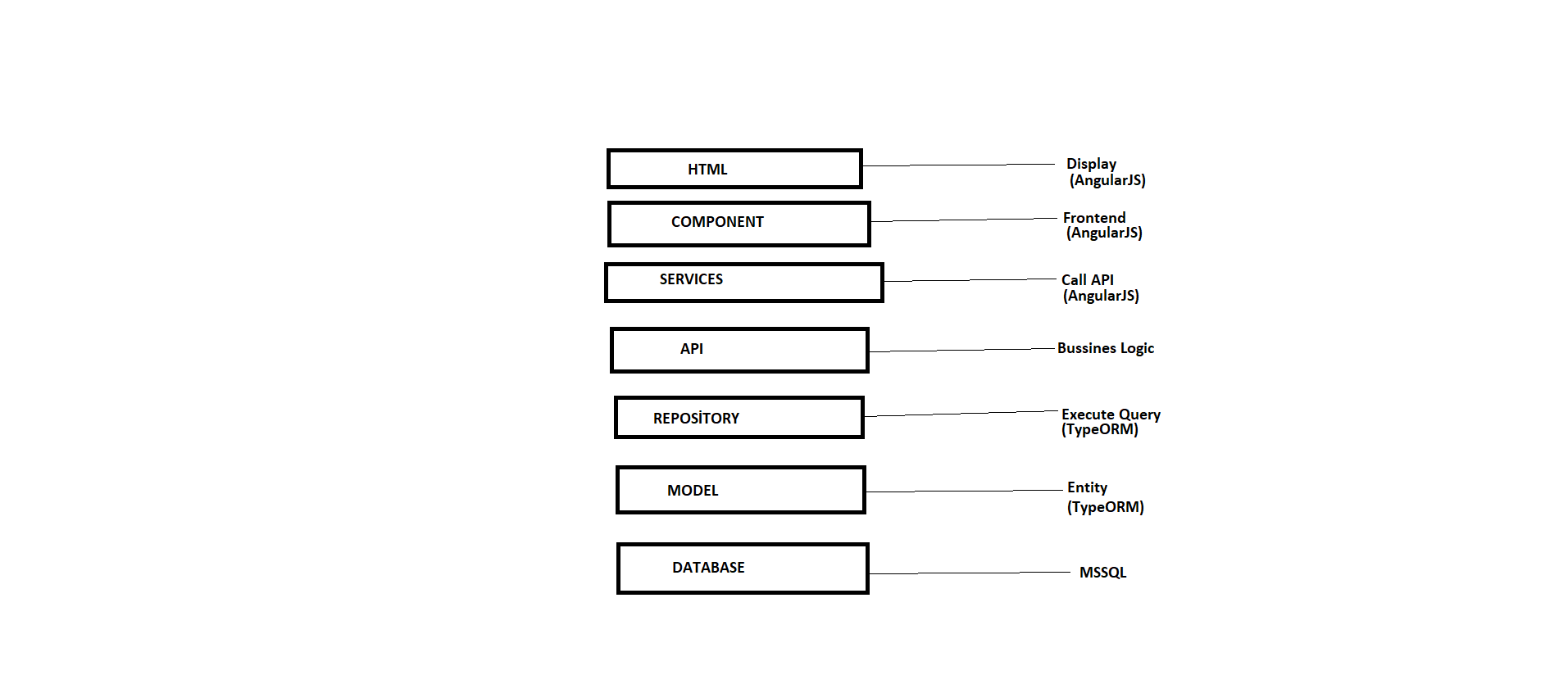


Figure 1: Layers of the Application

All database operations were placed in repository layer. It calls stored procedures which are defined in database. I used type-orm with Node.js. I have written interfaces(models) for modelling and passing data through layers. For both front-end and back-end I used typescript.

Another layer is api. In this layer, I placed business logic codes. For example, extreact data from post, delete, put or get method, adn then send it to repository for database process, then return a value or a message about the process.

The service layer is written for front-end. The logic is like subscribing a newspaper. Make a request and then subscribe the service, then wait for reply. This feature provides loading static HTML elements and if it si fetced load other elements.

The component layer was used for managing front-end operations, like button event handling, routing, calling services, drawing...

The HTML layer was used for presentation.

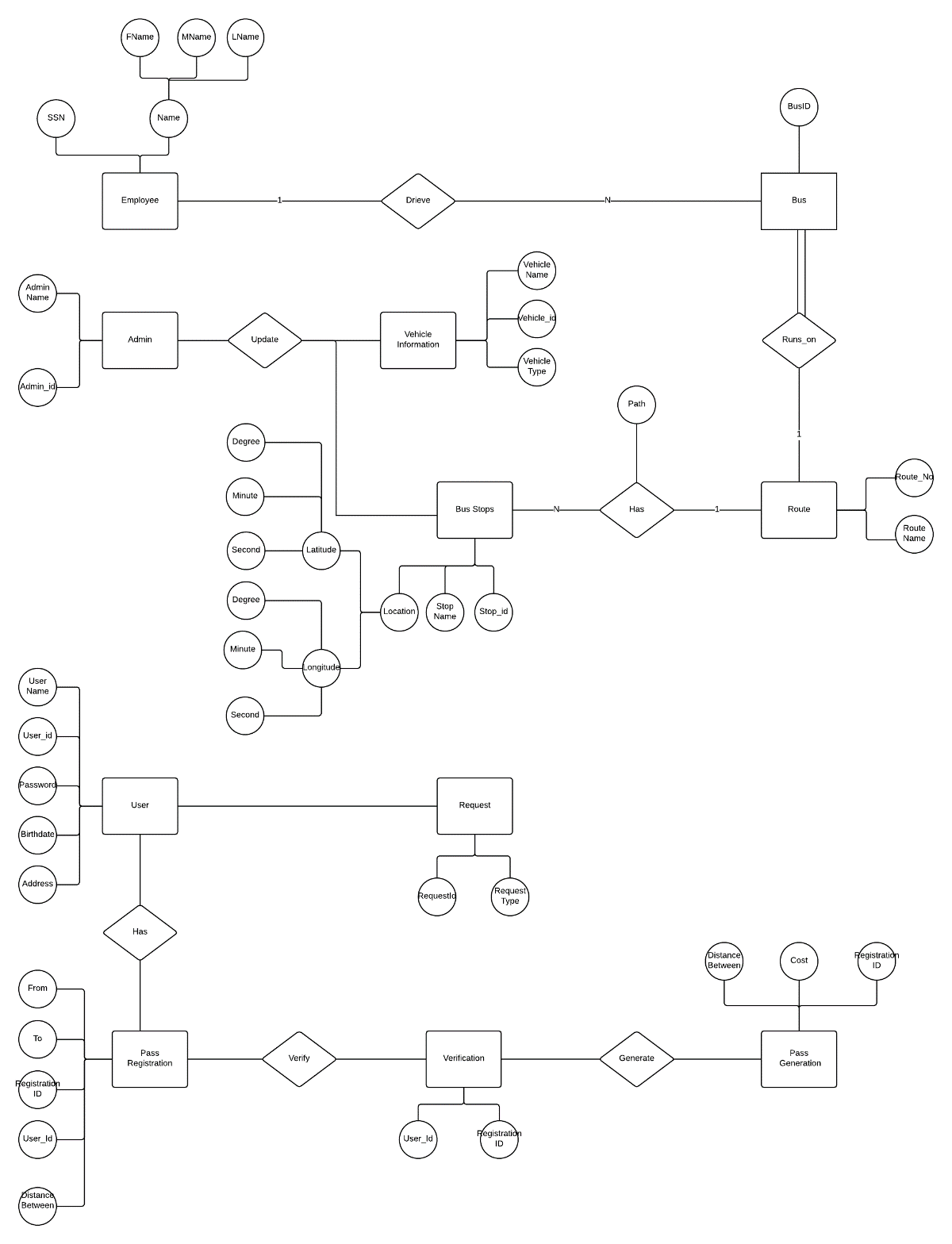
All layers can be seen in the prject navigation-bar. All layers are seperated clearly. For example api layer can be changed without changing other layers.

## System Constraints

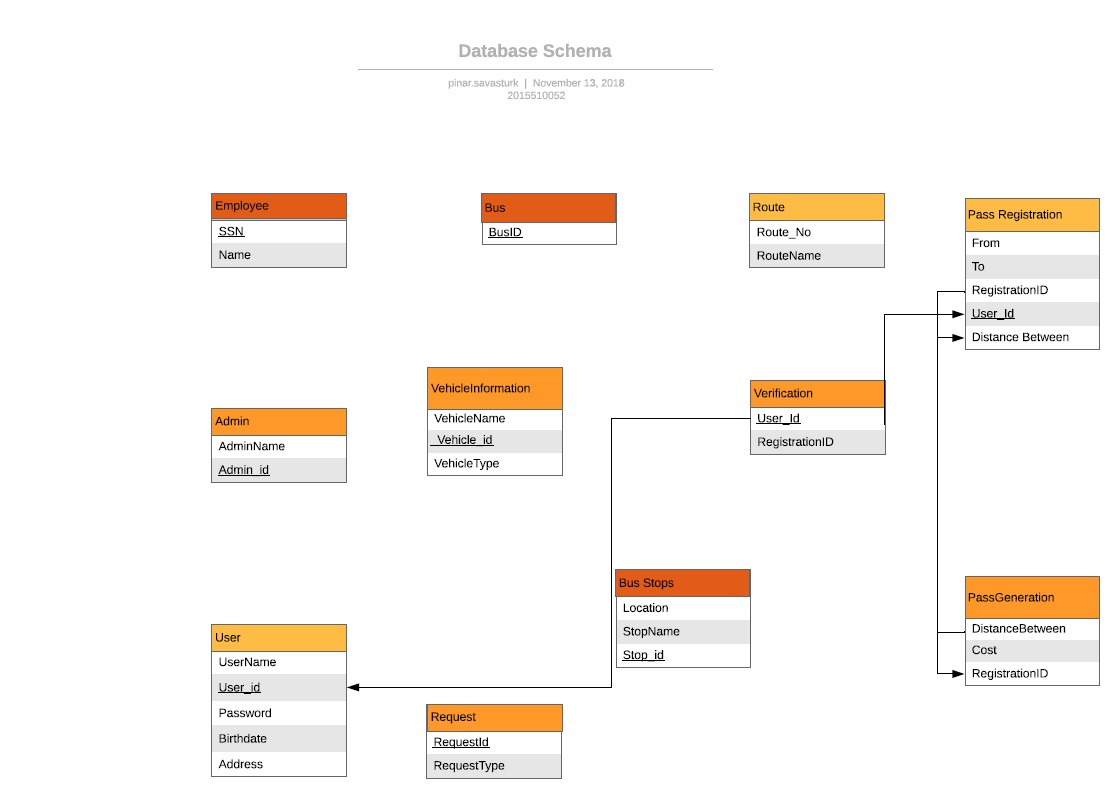
In the front-end, css was not used. In the database, the main constraint is update operations. In the project I maintain an interface, so sometimes it causes some problems like data inconsistency. Maybe the data in the view is not updated or the main table is not updated.

All data must be send in JSON format. Because I used typescript in the project. TypeScipt was designed for web developing, so all interfaces and objects can be represented like a JSON object.

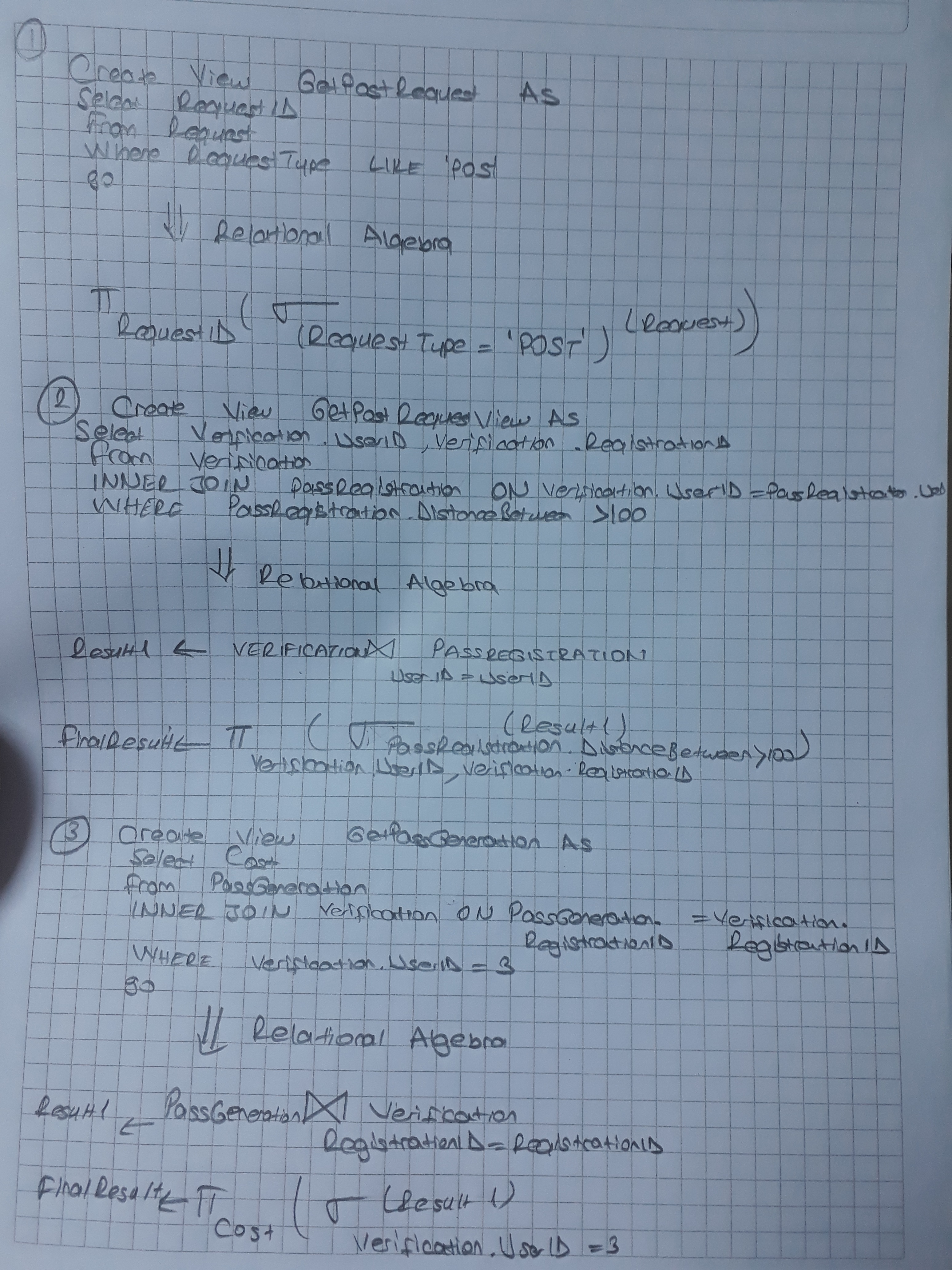
## ER Diagram



## The Database Schema



## Relational Algebra for 3 Queries



## SQL Statements

CREATE DATABASE BusTrackingSystemDatabase;

CREATE TABLE Employee(

FName varchar(50),

MName varchar(50),

LName varchar(50),

Salary int,

SSN int IDENTITY(1,1) PRIMARY KEY

);

go

CREATE procedure sp\_getEmpbySSN(@SSN int)

as

begin

select \* from Employee where SSN = @SSN

end

exec sp\_getEmpbySSN @SSN = 1

go

CREATE procedure sp\_getEmp

as

begin

select \* from Employee

end

go

CREATE procedure sp\_insertEmp(@FName varchar(50), @MName varchar(50), @LName varchar(50),@Salary int, @SSN int)

as

begin

insert into Employee values (@FName, @MName, @LName,@Salary, @SSN)

end

go

CREATE procedure sp\_updateEmp(@FName varchar(50), @MName varchar(50), @LName varchar(50),@Salary int, @SSN int)

as

begin

update Employee set FName = @FName, MName = @MName, LName = @LName,Salary = @Salary, SSN = @SSN where SSN = @SSN

end

go

go

CREATE procedure sp\_deleteEmp(@SSN int)

as

begin

delete from Employee where SSN = @SSN

end

CREATE VIEW GetSalary AS

SELECT Salary

FROM Employee

WHERE Salary > (SELECT AVG(Salary) FROM Employee)

go

--trigger code

/\* After Insert trigger on employee table \*/

INSERT INTO Employee(FName, MName, LName, Salary) VALUES ('Abuzer', 'Yandan', 'Bakan',1000);

INSERT INTO Employee(FName, MName, LName, Salary) VALUES ('şemsettin', 'Yandann', 'Bakann',2000) ;

INSERT INTO Employee(FName, MName, LName, Salary) VALUES ('kukulet', 'Yandannn', 'Bakannn',300) ;

INSERT INTO Employee(FName, MName, LName, Salary) VALUES ('şehriban', 'Yandannnn', 'Bakannnn',1000);

INSERT INTO Employee(FName, MName, LName, Salary) VALUES ('ornitorenk', 'Yandannnnn', 'Bakannnnn',2000) ;

INSERT INTO Employee(FName, MName, LName, Salary) VALUES ('ceviz', 'Yandannnnnn', 'Bakannnnnn',1500) ;

INSERT INTO Employee(FName, MName, LName, Salary) VALUES ('şemsi', 'Yandannnnnnn', 'Bakannnnnn',1750) ;

IF OBJECT\_ID('TRG\_InsertSyncEmp') IS NOT NULL

DROP TRIGGER TRG\_InsertSyncEmp

GO

CREATE TRIGGER TRG\_InsertSyncEmp

ON dbo.Employee

AFTER INSERT AS

BEGIN

INSERT INTO Employee

SELECT \* FROM INSERTED

END

GO

IF OBJECT\_ID('TRG\_UpdateSyncEmp') IS NOT NULL

DROP TRIGGER TRG\_UpdateSyncEmp

GO

CREATE TRIGGER TRG\_UpdateSyncEmp

ON dbo.Employee

AFTER UPDATE AS

BEGIN

UPDATE Employee

SET Salary = Salary + 1000

WHERE Salary < 1000

END

GO

CREATE TABLE AdminInformation(

AdminName varchar(50),

AdminID int IDENTITY(1,1) PRIMARY KEY

);

INSERT INTO AdminInformation(AdminName) VALUES ('Abuzittin')

INSERT INTO AdminInformation(AdminName) VALUES ('Kemalettin')

INSERT INTO AdminInformation(AdminName) VALUES ('Şemsettin')

INSERT INTO AdminInformation(AdminName) VALUES ('Yandanbittin')

INSERT INTO AdminInformation(AdminName) VALUES ('Üsttenbitmedin')

go

CREATE procedure sp\_getAdminInformationbyAdminID(@AdminID int)

as

begin

select \* from AdminInformation where AdminID = @AdminID

end

go

CREATE procedure sp\_getAdminInformation

as

begin

select \* from AdminInformation

end

go

CREATE procedure sp\_insertAdminInformation(@AdminName varchar(50))

as

begin

insert into AdminInformation values (@AdminName)

end

go

CREATE procedure sp\_updateAdminInformation(@AdminName varchar(50), @AdminID int)

as

begin

update AdminInformation set AdminName = @AdminName where AdminID = @AdminID

end

go

go

CREATE procedure sp\_deleteAdminInformation(@AdminID int)

as

begin

delete from AdminInformation where AdminID = @AdminID

end

IF OBJECT\_ID('TRG\_InsertSyncAdmin') IS NOT NULL

DROP TRIGGER TRG\_InsertSyncAdmin

GO

CREATE TRIGGER TRG\_InsertSyncAdmin

ON dbo.AdminInformation

AFTER INSERT AS

BEGIN

INSERT INTO AdminInformation

SELECT AdminName FROM INSERTED

END

GO

SELECT \* from AdminInformation

GO

CREATE VIEW GetAdmin AS

SELECT AdminID

FROM AdminInformation

WHERE AdminName LIKE '%a%'

go

CREATE TABLE Bus(

BusID int NOT NULL PRIMARY KEY

);

INSERT INTO Bus(BusID) VALUES (1)

INSERT INTO Bus(BusID) VALUES (2)

INSERT INTO Bus(BusID) VALUES (3)

INSERT INTO Bus(BusID) VALUES (4)

INSERT INTO Bus(BusID) VALUES (5)

INSERT INTO Bus(BusID) VALUES (6)

INSERT INTO Bus(BusID) VALUES (7)

go

CREATE procedure sp\_getBus

as

begin

select \* from Bus

end

go

CREATE procedure sp\_insertBus(@BusID int)

as

begin

insert into Bus values (@BusID)

end

go

CREATE procedure sp\_updateBus(@BusID int)

as

begin

update Bus set BusID = @BusID where BusID = @BusID

end

go

go

CREATE procedure sp\_deleteBus(@BusID int)

as

begin

delete from Bus where BusID = @BusID

end

IF OBJECT\_ID('TRG\_InsertSyncBus') IS NOT NULL

DROP TRIGGER TRG\_InsertSyncBus

GO

CREATE TRIGGER TRG\_InsertSyncBus

ON dbo.Bus

AFTER INSERT AS

BEGIN

INSERT INTO Bus

SELECT \* FROM INSERTED

END

GO

SELECT \* from Bus

GO

CREATE VIEW GetBusLessThan5 AS

SELECT BusID

FROM Bus

WHERE BusID < 5

go

CREATE TABLE VehicleInformation(

VehicleName varchar(50),

VehicleID int IDENTITY(1,1) PRIMARY KEY,

VehicleType varchar(50)

);

INSERT INTO VehicleInformation(VehicleName, VehicleType ) VALUES ('berlinetta','ferrari')

INSERT INTO VehicleInformation(VehicleName, VehicleType ) VALUES ('berlinetta','f12')

INSERT INTO VehicleInformation(VehicleName, VehicleType ) VALUES ('şahin','beyaz kuş')

INSERT INTO VehicleInformation(VehicleName, VehicleType ) VALUES ('toyota','auris')

INSERT INTO VehicleInformation(VehicleName, VehicleType ) VALUES ('toyota','verso')

go

CREATE procedure sp\_getVehicleInformation

as

begin

select \* from VehicleInformation

end

go

CREATE procedure sp\_insertVehicleInformation(@VehicleName varchar(50), @VehicleType varchar(50))

as

begin

insert into VehicleInformation values (@VehicleName, @VehicleType )

end

go

CREATE procedure sp\_updateVehicleInformation(@VehicleName varchar(50),@VehicleID int, @VehicleType varchar(50))

as

begin

update VehicleInformation set VehicleName = @VehicleName, VehicleID = @VehicleID, VehicleType = @VehicleType where VehicleID = @VehicleID

end

go

go

CREATE procedure sp\_deleteVehicleInformation(@VehicleID int)

as

begin

delete from VehicleInformation where VehicleID = @VehicleID

end

IF OBJECT\_ID('TRG\_DeleteFerrariBus') IS NOT NULL

DROP TRIGGER TRG\_DeleteFerrariBus

GO

CREATE TRIGGER TRG\_DeleteFerrariBus

ON dbo.VehicleInformation

AFTER DELETE AS

BEGIN

DELETE

FROM VehicleInformation

WHERE VehicleInformation.VehicleType LIKE 'ferrari'

END

GO

CREATE VIEW GetBerlinetta AS

SELECT VehicleID, VehicleType

FROM VehicleInformation

WHERE VehicleName LIKE 'berlinetta'

go

CREATE TABLE BusStops(

LatitudeDegree int,

LatitudeMinute int,

LatitudeSecond int,

LongitudeDegree int,

LongitudeMinute int,

LongitudeSecond int,

StopName varchar(50),

StopID int IDENTITY(1,1) PRIMARY KEY

);

INSERT INTO BusStops(LatitudeDegree,LatitudeMinute,LatitudeSecond,LongitudeDegree,LongitudeMinute,LongitudeSecond,StopName

) VALUES (27,12,17,28,12,18,'ada kuşu')

INSERT INTO BusStops(LatitudeDegree,LatitudeMinute,LatitudeSecond,LongitudeDegree,LongitudeMinute,LongitudeSecond,StopName

) VALUES (12,12,15,25,13,11,'ada ördeği')

INSERT INTO BusStops(LatitudeDegree,LatitudeMinute,LatitudeSecond,LongitudeDegree,LongitudeMinute,LongitudeSecond,StopName

) VALUES (28,15,19,22,10,8,'ada böceği')

go

CREATE procedure sp\_insertBusStops(@LatitudeDegree int,@LatitudeMinute int, @LatitudeSecond int,@LongitudeDegree int, @LongitudeMinute int, @LongitudeSecond int ,@StopName varchar(50))

as

begin

insert into BusStops values (@LatitudeDegree, @LatitudeMinute, @LatitudeSecond ,@LongitudeDegree,@LongitudeMinute ,@LongitudeSecond, @StopName )

end

go

CREATE procedure sp\_getBusStops

as

begin

select \* from BusStops

end

go

CREATE procedure sp\_deleteBusStops(@StopID int)

as

begin

delete from BusStops where StopID = @StopID

end

go

CREATE procedure sp\_updateBusStops(@LatitudeDegree int,@LatitudeMinute int, @LatitudeSecond int,@LongitudeDegree int, @LongitudeMinute int, @LongitudeSecond int ,@StopName varchar(50),@StopID int)

as

begin

update BusStops set LatitudeDegree = @LatitudeDegree, LatitudeMinute = @LatitudeMinute, LatitudeSecond = @LatitudeSecond, LongitudeDegree = @LongitudeDegree,LongitudeMinute = @LongitudeMinute, LongitudeSecond = @LongitudeSecond, StopName =@StopName, StopID = @StopID where StopID = @StopID

end

go

IF OBJECT\_ID('TRG\_InsertSyncBusStop') IS NOT NULL

DROP TRIGGER TRG\_InsertSyncBusStop

GO

CREATE TRIGGER TRG\_InsertSyncBusStop

ON dbo.BusStops

AFTER INSERT AS

BEGIN

INSERT INTO BusStops

SELECT \* FROM INSERTED

END

GO

GO

CREATE VIEW GetStopDegree AS

SELECT LatitudeDegree, LongitudeDegree

FROM BusStops

WHERE LatitudeMinute > 10 AND LongitudeMinute < 13

go

CREATE TABLE RouteInformation(

RouteNo int NOT NULL PRIMARY KEY,

RouteName varchar(50)

);

INSERT INTO RouteInformation(RouteNo,RouteName )VALUES(1,'beyaz şahin')

INSERT INTO RouteInformation(RouteNo,RouteName )VALUES(2,'siyah ördek')

INSERT INTO RouteInformation(RouteNo,RouteName )VALUES(3,'kara balık')

CREATE procedure sp\_insertRouteInformation(@RouteNo int,@RouteName varchar(50))

as

begin

insert into RouteInformation values (@RouteNo, @RouteName)

end

go

CREATE procedure sp\_getRouteInformation

as

begin

select \* from RouteInformation

end

go

CREATE procedure sp\_deleteRouteInformation(@RouteNo int)

as

begin

delete from RouteInformation where RouteNo = @RouteNo

end

go

CREATE procedure sp\_updateRouteInformation(@RouteNo int,@RouteName varchar(50))

as

begin

update RouteInformation set RouteNo = @RouteNo, RouteName = @RouteName where RouteNo = @RouteNo

end

go

IF OBJECT\_ID('TRG\_DeleteRoute') IS NOT NULL

DROP TRIGGER TRG\_DeleteRoute

GO

CREATE TRIGGER TRG\_DeleteRoute

ON dbo.RouteInformation

AFTER DELETE AS

BEGIN

DELETE

FROM RouteInformation

WHERE RouteName LIKE '[!bsp]%'

END

GO

CREATE VIEW GetRoute AS

SELECT RouteNo

FROM RouteInformation

WHERE RouteName LIKE '%e%'

go

CREATE TABLE UserInformation(

UserName varchar(50),

UserID int NOT NULL PRIMARY KEY,

Password varchar(50),

Birthdate date,

Address varchar(250)

);

INSERT INTO UserInformation(UserName,UserID, Password,Birthdate,Address)VALUES('abuzittiin',1,'987654','2015-08-05','kuzey batı marmara')

INSERT INTO UserInformation(UserName,UserID, Password,Birthdate,Address)VALUES('kalemettin',2,'652314','1997-07-18','kuzey batı doğu')

INSERT INTO UserInformation(UserName,UserID, Password,Birthdate,Address)VALUES('silgilittin',3,'12345','2006-02-03','kuzey batı ege')

CREATE procedure sp\_insertUserInformation(@UserName varchar(50),@UserID int,@Password varchar(50),@Birthdate date,@Address varchar(50))

as

begin

insert into UserInformation values (@UserName, @UserID,@Password,@Birthdate,@Address )

end

go

CREATE procedure sp\_getUserInformation

as

begin

select \* from UserInformation

end

go

CREATE procedure sp\_deleteUserInformation(@UserID int)

as

begin

delete from UserInformation where UserID = @UserID

end

go

CREATE procedure sp\_updateUserInformation(@UserName varchar(50),@UserID int,@Password varchar(50),@Birthdate date,@Address varchar(50))

as

begin

update UserInformation set UserName = @UserName, UserID = @UserID, Password = @Password, Birthdate = @Birthdate, Address = @Address where UserID = @UserID

end

go

CREATE TABLE Request(

RequestID int IDENTITY(1,1) PRIMARY KEY,

RequestType varchar(50)

);

INSERT INTO Request( RequestType) VALUES ( 'GET')

INSERT INTO Request( RequestType) VALUES ( 'POST')

INSERT INTO Request( RequestType) VALUES ( 'GET')

INSERT INTO Request( RequestType) VALUES ( 'POST')

INSERT INTO Request( RequestType) VALUES ( 'GET')

CREATE procedure sp\_insertRequest(@RequestType varchar(50))

as

begin

insert into Request values (@RequestType)

end

go

CREATE procedure sp\_getRequest

as

begin

select \* from Request

end

go

CREATE procedure sp\_deleteRequest(@RequestID int)

as

begin

delete from Request where RequestID = @RequestID

end

go

CREATE procedure sp\_updateRequest(@RequestID int, @RequestType varchar(50))

as

begin

update Request set RequestID = @RequestID, RequestType = @RequestType where RequestID = @RequestID

end

go

IF OBJECT\_ID('TRG\_InsertSyncRequest') IS NOT NULL

DROP TRIGGER TRG\_InsertSyncRequest

GO

CREATE TRIGGER TRG\_InsertSyncRequest

ON dbo.Request

AFTER INSERT AS

BEGIN

INSERT INTO Request

SELECT \* FROM INSERTED

END

GO

SELECT \* from Request

GO

CREATE VIEW GetPostRequest AS

SELECT RequestID

FROM Request

WHERE RequestType LIKE 'POST'

go

CREATE TABLE PassRegistration(

FromRegistration varchar(50),

ToRegistration varchar(50),

RegistrationID int NOT NULL PRIMARY KEY,

UserID int NOT NULL ,

DistanceBetween int,

FOREIGN KEY (UserID) REFERENCES UserInformation(UserID)

);

INSERT INTO PassRegistration(FromRegistration, ToRegistration,RegistrationID,UserID,DistanceBetween) VALUES ('hay','huy',1,1,100)

INSERT INTO PassRegistration(FromRegistration, ToRegistration,RegistrationID,UserID,DistanceBetween) VALUES ('kuzey','güney',2,2,200)

INSERT INTO PassRegistration(FromRegistration, ToRegistration,RegistrationID,UserID,DistanceBetween) VALUES ('doğu','batı',3,3,-3)

INSERT INTO PassRegistration(FromRegistration, ToRegistration,RegistrationID,UserID,DistanceBetween) VALUES ('huy','hay',4,4,400)

IF OBJECT\_ID('TRG\_DeletePassRegistration') IS NOT NULL

DROP TRIGGER TRG\_DeletePassRegistration

GO

go

CREATE TRIGGER TRG\_DeletePassRegistration

ON dbo.PassRegistration

AFTER DELETE AS

BEGIN

DELETE

FROM RouteInformation

WHERE RouteName LIKE '[!bsp]%'

END

GO

CREATE procedure sp\_insertPassRegistration(@FromRegistration varchar(50),@ToRegistration varchar(50),@RegistrationID int,@UserID int,@DistanceBetween int)

as

begin

insert into PassRegistration values (@FromRegistration, @ToRegistration, @RegistrationID, @UserID, @DistanceBetween )

end

go

CREATE procedure sp\_getPassRegistration

as

begin

select \* from PassRegistration

end

go

CREATE procedure sp\_deletePassRegistration(@RegistrationID int )

as

begin

delete from PassRegistration where RegistrationID = @RegistrationID

end

IF OBJECT\_ID('TRG\_DeletePassRegistrationTable') IS NOT NULL

DROP TRIGGER TRG\_DeletePassRegistrationTable

GO

CREATE TRIGGER TRG\_DeletePassRegistrationTable

ON dbo.PassRegistration

AFTER DELETE AS

BEGIN

DELETE

FROM UserInformation

WHERE UserID = UserInformation.UserID

END

GO

go

CREATE procedure sp\_updatePassRegistration(@FromRegistration varchar(50), @ToRegistration varchar(50), @RegistrationID int ,@UserID int, @DistanceBetween int)

as

begin

update PassRegistration set FromRegistration = @FromRegistration, ToRegistration = @ToRegistration, RegistrationID = @RegistrationID, UserID = @UserID, DistanceBetween = @DistanceBetween where RegistrationID = @RegistrationID

end

go

IF OBJECT\_ID('TRG\_SetWrongDistance') IS NOT NULL

DROP TRIGGER TRG\_SetWrongDistance

GO

CREATE TRIGGER TRG\_SetWrongDistance

ON dbo.PassRegistration

AFTER UPDATE AS

BEGIN

UPDATE PassRegistration

SET PassRegistration.DistanceBetween = 0

FROM PassRegistration

INNER JOIN UserInformation ON PassRegistration.UserID = UserInformation.UserID

WHERE PassRegistration.DistanceBetween < 0

END

GO

IF OBJECT\_ID('TRG\_SetPassword') IS NOT NULL

DROP TRIGGER TRG\_SetPassword

GO

CREATE TRIGGER TRG\_SetPassword

ON dbo.UserInformation

AFTER UPDATE AS

BEGIN

UPDATE UserInformation

SET UserInformation.Password = '12345'

FROM UserInformation

INNER JOIN PassRegistration ON UserInformation.UserID = PassRegistration.UserID

END

GO

CREATE VIEW GetUserName AS

SELECT UserName, UserID

FROM UserInformation

WHERE Birthdate > '2008-11-11'

go

CREATE TABLE Verification(

UserID int NOT NULL,

RegistrationID int NOT NULL,

FOREIGN KEY (UserID) REFERENCES UserInformation(UserID),

FOREIGN KEY (RegistrationID) REFERENCES PassRegistration(RegistrationID)

);

INSERT INTO Verification(UserID, RegistrationID) VALUES (1, 1)

INSERT INTO Verification(UserID, RegistrationID) VALUES (2, 2)

INSERT INTO Verification(UserID, RegistrationID) VALUES (3, 3)

INSERT INTO Verification(UserID, RegistrationID) VALUES (4, 4)

CREATE procedure sp\_insertVerification(@UserID int,@RegistrationID int)

as

begin

insert into Verification values (@UserID, @RegistrationID)

end

go

CREATE procedure sp\_getVerification

as

begin

select \* from Verification

end

go

CREATE procedure sp\_deleteEmp(@SSN int)

as

begin

delete from Employee where SSN = @SSN

end

go

CREATE procedure sp\_updateEmp(@FName varchar(50), @MName varchar(50), @LName varchar(50),@Salary int, @SSN int)

as

begin

update Employee set FName = @FName, MName = @MName, LName = @LName,Salary = @Salary, SSN = @SSN where SSN = @SSN

end

go

IF OBJECT\_ID('TRG\_InsertSyncVerification') IS NOT NULL

DROP TRIGGER TRG\_InsertSyncVerification

GO

CREATE TRIGGER TRG\_InsertSyncVerification

ON dbo.Verification

AFTER INSERT AS

BEGIN

INSERT INTO Verification

SELECT \* FROM INSERTED

END

GO

SELECT \* from Verification

GO

CREATE VIEW GetPostRequestView AS

SELECT Verification.UserID, Verification.RegistrationID

FROM Verification

INNER JOIN PassRegistration ON Verification.UserID = PassRegistration.UserID

WHERE PassRegistration.DistanceBetween > 100

go

CREATE TABLE PassGeneration(

DistanceBetween int,

Cost int,

RegistrationID int NOT NULL

FOREIGN KEY (RegistrationID) REFERENCES PassRegistration(RegistrationID)

);

INSERT INTO PassGeneration(DistanceBetween,Cost,RegistrationID)VALUES(100,75,1)

INSERT INTO PassGeneration(DistanceBetween,Cost,RegistrationID)VALUES(175,125,2)

INSERT INTO PassGeneration(DistanceBetween,Cost,RegistrationID)VALUES(250,50,3)

INSERT INTO PassGeneration(DistanceBetween,Cost,RegistrationID)VALUES(500,150,4)

INSERT INTO PassGeneration(DistanceBetween,Cost,RegistrationID)VALUES(375,100,5)

CREATE procedure sp\_insertPassGeneration(@DistanceBetween int,@Cost int, @RegistrationID int)

as

begin

insert into PassGeneration values (@DistanceBetween, @Cost, @RegistrationID)

end

go

CREATE procedure sp\_getPassGeneration

as

begin

select \* from PassGeneration

end

go

CREATE procedure sp\_deleteEmp(@SSN int)

as

begin

delete from Employee where SSN = @SSN

end

go

CREATE procedure sp\_updateEmp(@FName varchar(50), @MName varchar(50), @LName varchar(50),@Salary int, @SSN int)

as

begin

update Employee set FName = @FName, MName = @MName, LName = @LName,Salary = @Salary, SSN = @SSN where SSN = @SSN

end

go

IF OBJECT\_ID('TRG\_UpdateSyncPassDistanceBetween') IS NOT NULL

DROP TRIGGER TRG\_UpdateSyncPassDistanceBetween

GO

CREATE TRIGGER TRG\_UpdateSyncPassDistanceBetween

ON dbo.PassGeneration

AFTER UPDATE AS

BEGIN

UPDATE PassGeneration

SET DistanceBetween = DistanceBetween + 100

WHERE Cost > (SELECT AVG(Cost) FROM PassRegistration);

END

GO

CREATE VIEW GetPassGeneration AS

SELECT Cost

FROM PassGeneration

INNER JOIN Verification ON PassGeneration.RegistrationID = Verification.RegistrationID

WHERE Verification.UserID = 3

go

CREATE VIEW getRegistrationView AS

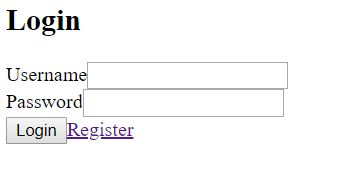
SELECT RegistrationID, FromRegistration,ToRegistration

FROM PassRegistration

WHERE PassRegistration.DistanceBetween > (SELECT AVG(DistanceBetween) FROM PassRegistration);

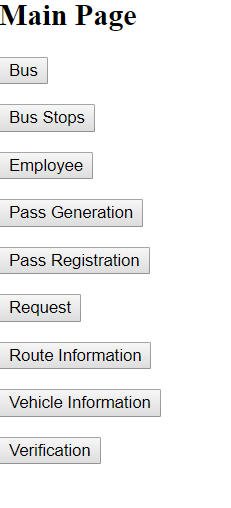
Go

## Screen Shots

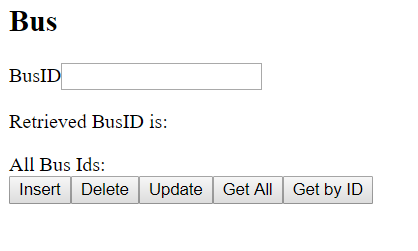


This is Login page. If user want to enter the system .(S)He must register the system.

This page take username and password.



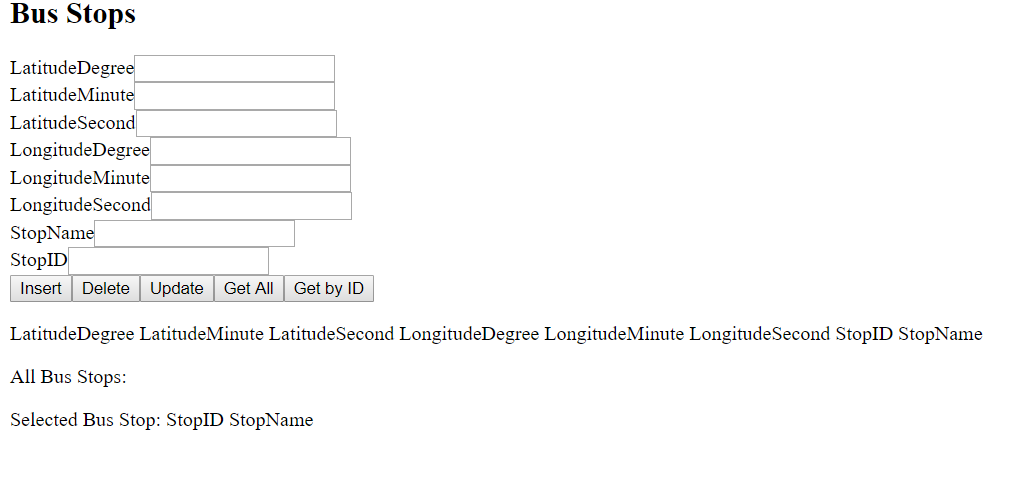
This is main page. When user login the system then the system direct to this page.User can acces this pages with main page.



This is Bus page. If user enter the bus click in main page, system direct the user this page.

User can apply insert, delete, update, get all busses and get bus by id methods in this page.

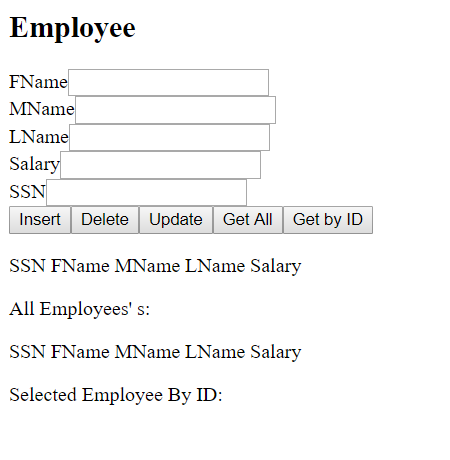
User must enter BusID parameter.



This is Bus Stops page. If user enter the bus stops click in main page, system direct the user this page.

User can apply insert, delete, update, get all bus stops and get bus stop by id methods in this page.

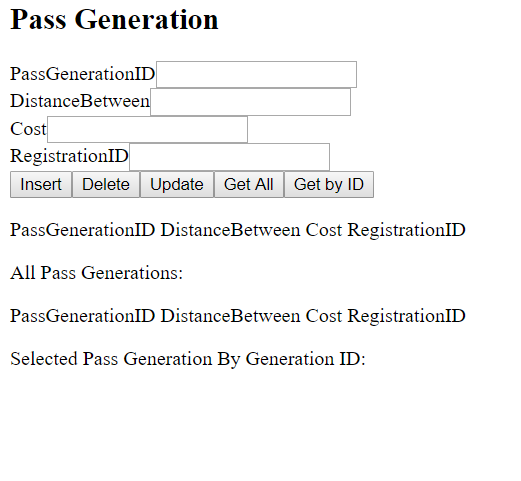
User must enter these attributes.



This is Employee page. If user enter the employee click in main page, system direct the user this page.

User can apply insert, delete, update, get all employees and get employee by SSN methods in this page.

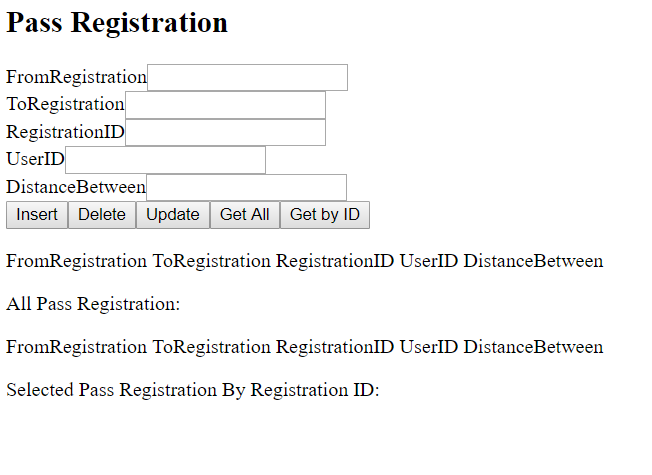
User must enter these attributes.



This is Pass Generation page. If user enter the pass generation click in main page, system direct the user this page.

User can apply insert, delete, update, get all pass generations and get pass generation by ID methods in this page.

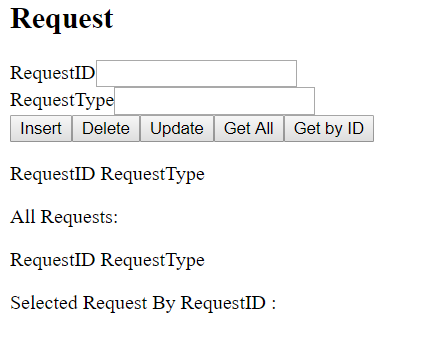
User must enter these attributes.



This is Pass Registration page. If user enter the pass registration click in main page, system direct the user this page.

User can apply insert, delete, update, get all pass registration and get pass registration by ID methods in this page.

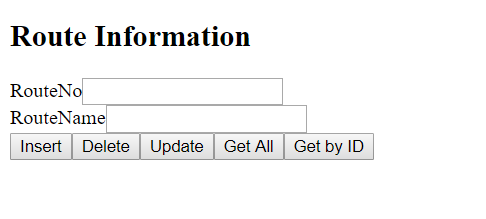
User must enter these attributes.



This is Request page. If user enter the request click in main page, system direct the user this page.

User can apply insert, delete, update, get all requests and get request by ID methods in this page.

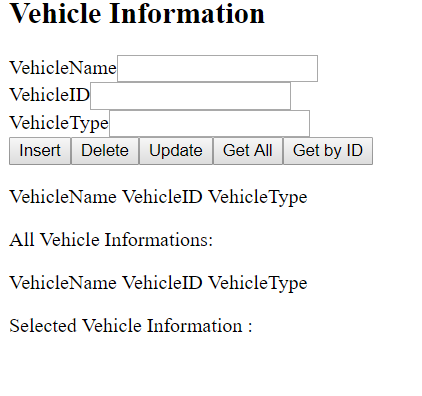
User must enter these attributes.



This is Route Information page. If user enter the route information click in main page, system direct the user this page.

User can apply insert, delete, update, get all route informations and get route information by RouteNo methods in this page.

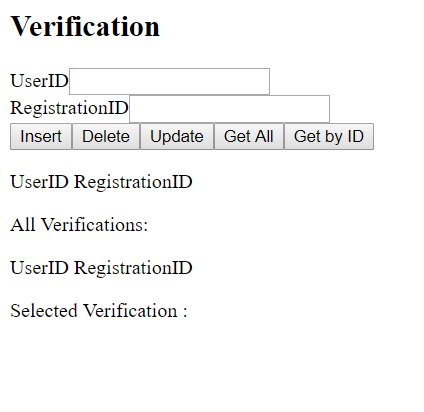
User must enter these attributes.



This is Vehicle Information page. If user enter the vehicle information click in main page, system direct the user this page.

User can apply insert, delete, update, get all vehicle informations and get vehicle information by ID methods in this page.

User must enter these attributes.



This is Verification page. If user enter the verification click in main page, system direct the user this page.

User can apply insert, delete, update, get all verifications and get verification by ID methods in this page.

User must enter these attributes.

## Used Technology, Tools, and Challenges

I used Node.js and Angular. For writing code I used typescirpt foth front-end and back-end. In IDE side I used Visual Studio Code. It has not got amny compiler or interpreter default. So I used gulp for building typescript code and it maps typescript to pure js(javaScript) code. For running code I used user defined tasks in the visual studio code.

The main challenge in the project is using a language which was not told in the class and not cougered to use it.