



# American International University- Bangladesh

## Summer2019-2020

### Project Report

Group Name: MUSE

### Advance Database Management System

**Project Title:** Car Shop Management System

**Section:** A

Student Name	Student Id
Ahnaf Sayed	18-36920-1
Arshad Shahoriar	18-37104-1
Pronay Saha	18-36464-1
Kh. Reaz Faruk	18-37581-1

## Table Creation

### Owner:

```
CREATE TABLE owner(  
  o_id number(10),  
  o_name varchar(30),  
  o_password varchar(20),  
  constraint pk_owner primary key(o_id)  
)
```

### Sequence

```
CREATE SEQUENCE seq_o_id  
start with 1  
increment by 1  
nomaxvalue  
nominvalue  
Nocycle;
```

```
INSERT INTO owner VALUES (seq_o_id.nextval, 'Reaz', 'Reaz1234');
```

### Manager:

```
CREATE TABLE manager(  
  m_id number(10),  
  m_name varchar(30),  
  m_password varchar(20),  
  m_sal number(10),  
  m_comm number(10),  
  o_id number(10),  
  constraint pk_manager primary key(m_id),  
  constraint fk_man_own foreign key(o_id) references owner(o_id)  
)
```

## Sequence

```
CREATE SEQUENCE seq_m_id  
start with 1  
increment by 1  
nomaxvalue  
nominvalue  
Nocycle;
```

```
INSERT INTO manager VALUES (seq_m_id.nextval, ' Reaz ', ' Reaz1234', '45000', '5000', '1');
```

## Customer:

```
CREATE TABLE customer(  
u_id number(10),  
u_name varchar2(30),  
u_password varchar2(20),  
address varchar2(10),  
name varchar2(10),  
mobile_no number(10),  
constraint pk_customer primary key(u_id)  
)
```

## Sequence

```
CREATE SEQUENCE seq_u_id  
start with 4  
increment by 1  
nomaxvalue  
nominvalue  
Nocycle;
```

```
INSERT INTO customer VALUES (seq_u_id.nextval, ' Reaz ', ' Reaz1234', 'Khilgaon', 'Dhaka',  
'12345678');
```

## Car:

```
CREATE TABLE car(  
  c_id number(10),  
  c_name varchar(30),  
  price number(10),  
  model varchar2(20),  
  availability varchar2(20),  
  booked_id number(10),  
  m_id number(10),  
  o_id number(10),  
  u_id number(10),  
  constraint pk_car primary key(c_id),  
  constraint fk_car_man foreign key(m_id) references manager(m_id),  
  constraint fk_car_own foreign key(o_id) references owner(o_id),  
  constraint fk_car_cus foreign key(u_id) references customer(u_id)  
)
```

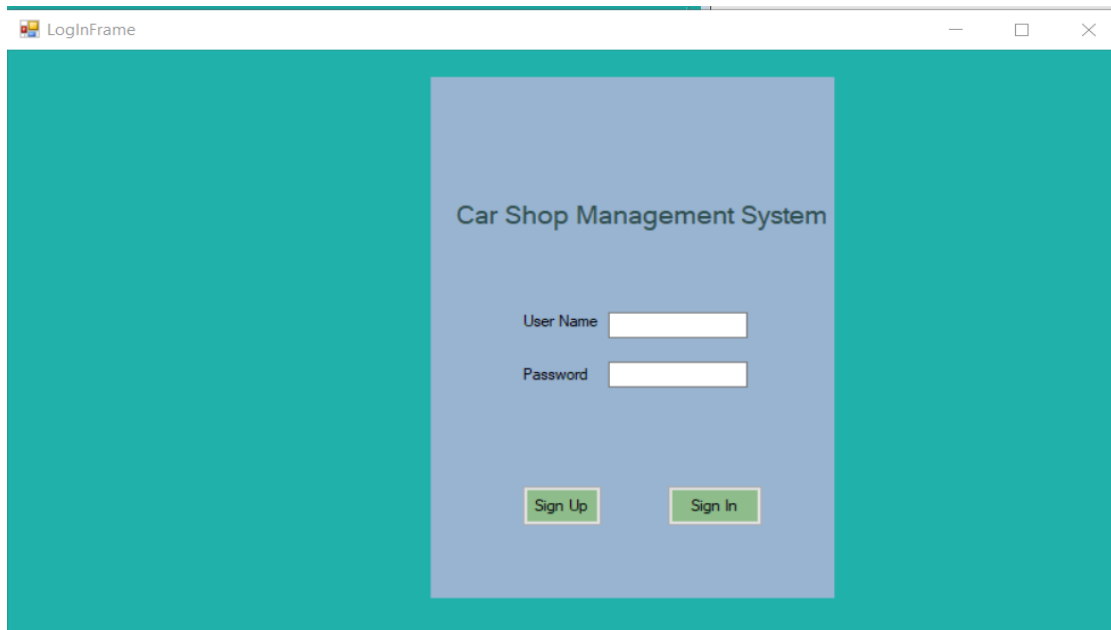
## Sequence

```
CREATE SEQUENCE seq_c_id  
start with 1  
increment by 1  
nomaxvalue  
nominvalue  
Nocycle;
```

```
INSERT INTO car VALUES (seq_c_id.nextval, 'Ferari', '5000000', 'r1', 'yes', '1', '1', '1', '6');
```

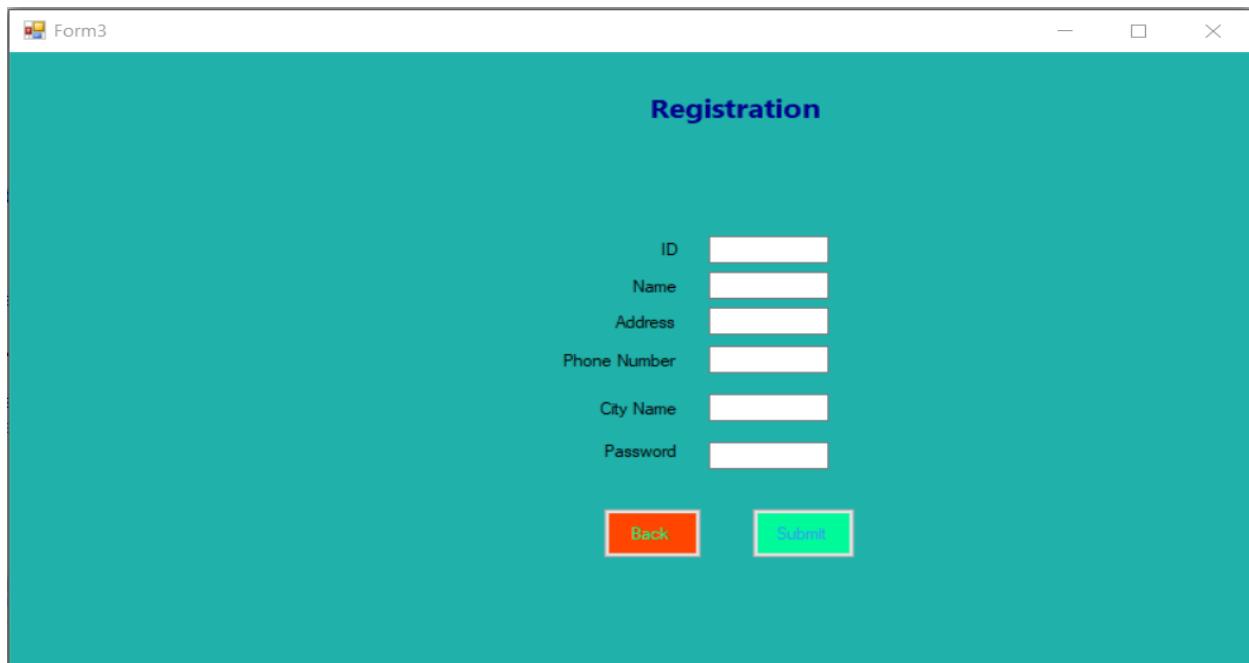
# User Interface

## Login Page



The screenshot shows a window titled "LogInFrame" with a teal background. In the center, there is a light blue rectangular box containing the text "Car Shop Management System". Below this text, there are two input fields: "User Name" and "Password". At the bottom of the box, there are two buttons: "Sign Up" and "Sign In".

## Registration Page



The screenshot shows a window titled "Form3" with a teal background. In the center, there is a registration form with the title "Registration" in bold blue text. The form contains six input fields: "ID", "Name", "Address", "Phone Number", "City Name", and "Password". At the bottom of the form, there are two buttons: "Back" (orange) and "Submit" (green).

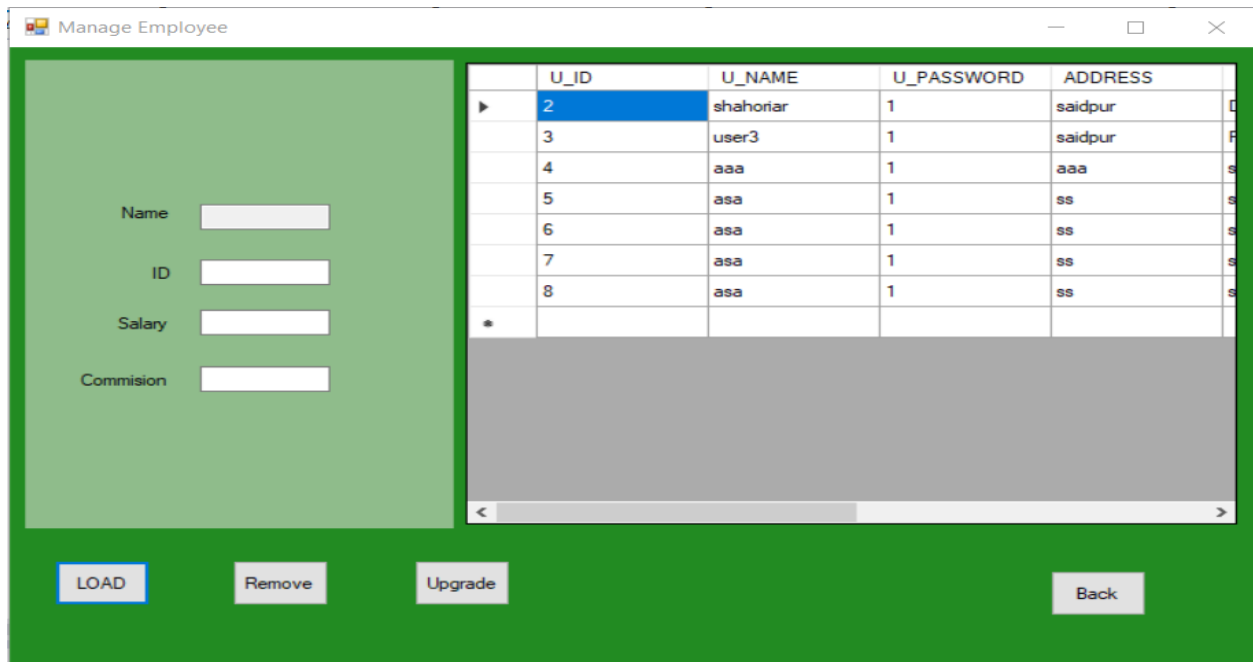
## Owner Dashboard and User Dashboard

The screenshot shows a web application window titled "Form4". The dashboard has a teal background. At the top right, there is a "logOut" link. Below it, a navigation bar contains "Home", "Profile", and "Booked Car". The main heading is "Welcome to Car Shop Mangement System". On the left side, there is a vertical menu with the following items: "Manage Employee", "Manage car details", "All User", and "Manage Car". In the center, there is a large grey rectangular placeholder. On the right side, there is a "Search Cars" input field with a "Search" button below it. Further down on the right, there are links for "Show All Cars", "Purchase History", and "See Balance".

## Update info

The screenshot shows a web application window with a light blue background. In the top left corner, there is a red button labeled "Home". The main content is a form with a tan background. The form contains the following fields and labels: "UserID:" with the value "1", "UserName" with the value "arshad", "Address" with the value "saidpur", "City" with the value "Nilphamary", "Mobile NO" with the value "1950977110", and "PASSWORD" with a masked value "\*". At the bottom of the form, there are two buttons: "Load" and "Update".

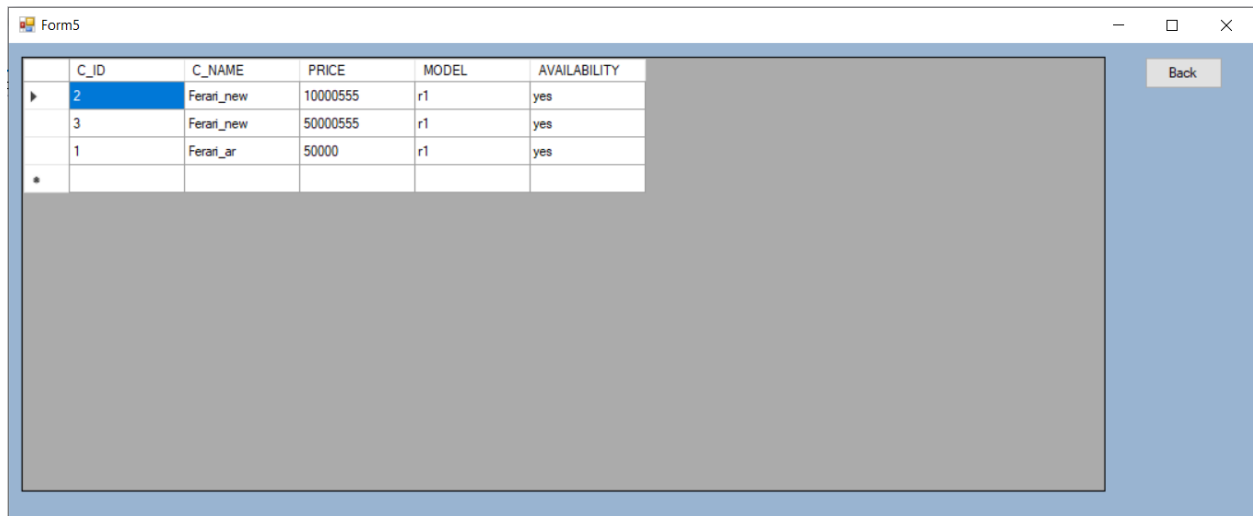
## Manage Employee



The 'Manage Employee' window features a green background. On the left, there is a form with labels 'Name', 'ID', 'Salary', and 'Commision' (sic), each followed by a text input field. On the right, a table displays employee data. The table has columns for U\_ID, U\_NAME, U\_PASSWORD, and ADDRESS. Row 2 is highlighted in blue. Below the table is a large grey rectangular area. At the bottom, there are four buttons: 'LOAD', 'Remove', 'Upgrade', and 'Back'.

	U_ID	U_NAME	U_PASSWORD	ADDRESS
▶	2	shahoriar	1	saidpur
	3	user3	1	saidpur
	4	aaa	1	aaa
	5	asa	1	ss
	6	asa	1	ss
	7	asa	1	ss
	8	asa	1	ss
*				

## Show All Cars



The 'Form5' window has a light blue background. It contains a table with columns C\_ID, C\_NAME, PRICE, MODEL, and AVAILABILITY. Row 2 is highlighted in blue. Below the table is a large grey rectangular area. A 'Back' button is located on the right side of the window.

	C_ID	C_NAME	PRICE	MODEL	AVAILABILITY
▶	2	Ferari_new	10000555	r1	yes
	3	Ferari_new	50000555	r1	yes
	1	Ferari_ar	50000	r1	yes
*					

## Manage Car Details

Back

Manage Cars

Car Name

Car ID

Availability

PRICE

Model

BookID

OwnerID

M\_ID

U\_ID

	C_ID	C_NAME	PRICE	MODEL	AVAILABILITY	
▶	2	Ferari_new	10000555	r1	yes	2
	3	Ferari_new	50000555	r1	yes	2
	1	Ferari_ar	50000	r1	yes	1
*						

<

>

Update

Reload

Insert

Deleted

## Book Car

The screenshot shows a Java Swing window titled "Form7". The window's title bar is light blue and contains standard Windows window controls (minimize, maximize, close). The main content area has a teal background. On the left side, there is a large, empty gray rectangle with a dashed border. On the right side, there are five text input fields arranged vertically, each with a label to its left: "ID", "CarID", "Car Name", "PRICE", and "Available/Not". Above the "ID" field is a button labeled "Back". Below the "Available/Not" field are two buttons: "BUY CAR" and "Booked Car".



## View:

1. Create a view to display customer name and their id.

```
create view view_customer as select u_name, u_id from customer;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

```
View created.
```

0.45 seconds

2. Create a view to display car name and their id.

```
create view view_car as select c_name, c_id from car;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

```
View created.
```

0.08 seconds

3. Create a view to display manager and their id.

```
create view view_manager as select m_name, m_id from manager;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

```
View created.
```

0.05 seconds

4. Create a view to display owner and their id.

create view view\_owner as select o\_name, o\_id from owner;

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

View created.

0.08 seconds

### Procedure:

1. Create a procedure to update customer.

```
CREATE OR REPLACE PROCEDURE updateCustomer
(id IN NUMBER,
username IN VARCHAR2,
password IN VARCHAR2,
address IN VARCHAR2,
state IN VARCHAR2,
phone IN NUMBER)
IS
BEGIN
UPDATE customer SET u_name=username, u_password=password,
address=address, name=state, mobile_no=phone WHERE u_id=id;
END;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Procedure created.

0.09 seconds

2. Create a procedure to print car details.

```
CREATE OR REPLACE PROCEDURE print_car
(id IN NUMBER)
IS
car_details car%ROWTYPE;
BEGIN
    SELECT * INTO car_details
    FROM car WHERE c_id=id;

    dbms_output.put_line(car_details.c_name || ' price is: ' || car_details.price || '
and Model is: ' || car_details.model);

END;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Procedure created.

0.05 seconds

3. Create a procedure to update owner.

```
CREATE OR REPLACE PROCEDURE updateOwner
(id IN NUMBER,
username IN VARCHAR2,
password IN VARCHAR2)
IS
BEGIN
UPDATE owner SET o_name=username, o_password=password WHERE o_id=id;
END;
```

**Results** Explain Describe Saved SQL History

Procedure created.

0.01 seconds

## Function:

1. Create a function to display data from manager table.

```
CREATE OR REPLACE FUNCTION selectMsg(p_name IN VARCHAR2)
RETURN VARCHAR2
IS
BEGIN
RETURN (p_name);
END;
```

**Results** Explain Describe Saved SQL History

Function created.

0.05 seconds

```
SELECT selectMsg(m_name), selectMsg(m_id) FROM manager WHERE m_id=1;
```

**Results** Explain Describe Saved SQL History

SELECTMSG(M_NAME)	SELECTMSG(M_ID)
-------------------	-----------------

Reaz	1
------	---

1 rows returned in 0.01 seconds

[CSV Export](#)

2. Create/replace a function to show the number of total customers.

```
CREATE OR REPLACE FUNCTION totalCustomer
RETURN NUMBER
IS
total NUMBER := 0;
BEGIN
    SELECT count(*) INTO total
    FROM customer;
    RETURN total;
END;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Function created.

0.08 seconds

```
DECLARE
customer NUMBER(10);
BEGIN
customer := totalCustomer();
dbms_output.put_line('Customer: ' || customer);
END;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Customer: 6

Statement processed.

0.00 seconds

3. Create a function to display total number of cars.

```
CREATE OR REPLACE FUNCTION totalCar
return number as
    totalnum number(10);
BEGIN
    select count(c_id) into totalnum from car;
    return totalnum;
end;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Function created.

0.00 seconds

```
DECLARE
    totalnum number(10);
BEGIN
    totalnum:=totalCar();
    dbms_output.put_line('Total Cars: ' || totalnum);
end;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Total Cars: 6

Statement processed.

0.00 seconds

## Trigger:

1. Create a trigger for any new customer inserted into the customer table.

```
create trigger customer_t
after insert on customer
for each row
when (new.u_id > 0)
begin
    dbms_output.put_line('New Customer Added');
end;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Trigger created.

0.20 seconds

2. Create a trigger for any new manager inserted into the manager table.

```
create trigger manager_t
after insert on manager
for each row
when (new.m_id > 0)
begin
    dbms_output.put_line('New Manager Added');
end;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Trigger created.

0.12 seconds

3. Create a trigger for any new car inserted into the car table.

```
create trigger car_t
after insert on car
for each row
when (new.m_id > 0)
begin
    dbms_output.put_line('New Car Added');
end;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Trigger created.

0.03 seconds



## Package:

1. Create a package which has a procedure to update car model.

```
CREATE OR REPLACE PACKAGE pkg_car IS  
  PROCEDURE update_model(model in varchar);  
END pkg_car;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Package created.

0.00 seconds

```
CREATE OR REPLACE PACKAGE BODY pkg_car IS  
  PROCEDURE update_model(model in varchar)  
  as  
  BEGIN  
    update car set model=model;  
    dbms_output.put_line(model);  
  end;  
END pkg_car;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Package Body created.

0.00 seconds

2. Create a package with a procedure to update salary of the managers.

```
CREATE OR REPLACE PACKAGE pkg_manager IS  
  PROCEDURE update_salary(salary in number);  
  
END pkg_manager;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Package created.

0.00 seconds

```
CREATE OR REPLACE PACKAGE BODY pkg_manager IS  
  PROCEDURE update_salary(salary in number)  
  as  
    BEGIN  
      update manager set m_sal=salary;  
      dbms_output.put_line(salary);  
    end;  
END pkg_manager;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Package Body created.

0.00 seconds