CS 4400
Introduction to Database Systems
GTCR Project - Phase 2
GROUP 36

- Wenlei He:
    - Section C
    - Email address: wenlei.he@gatech.edu
    - T-square username: he30

- Zhen Guo:
    - Section C
    - Email address: zhen.guo@gatech.edu
    - T-square username: zguo44

- Minwei Gu:
    - Section C
    - Email address: minwei_gu@gatech.edu
    - T-square username: mgu7

- Yuanhao Wang:
    - Section C
    - Email address: ywang678@mail.gatech.edu
    - T-square username: ywang678

CS 4400 GTCR Project - Phase 1

EER Diagram

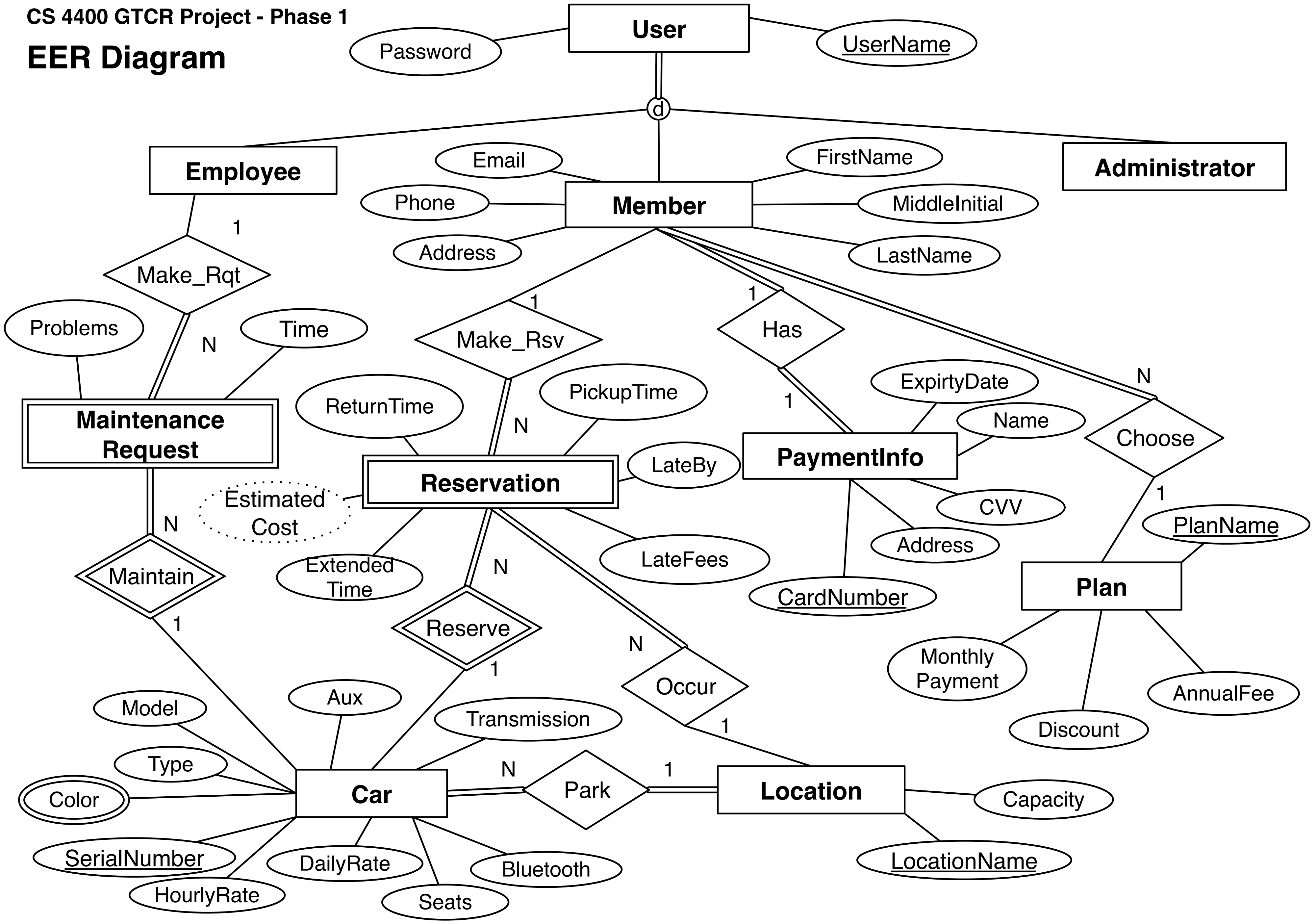
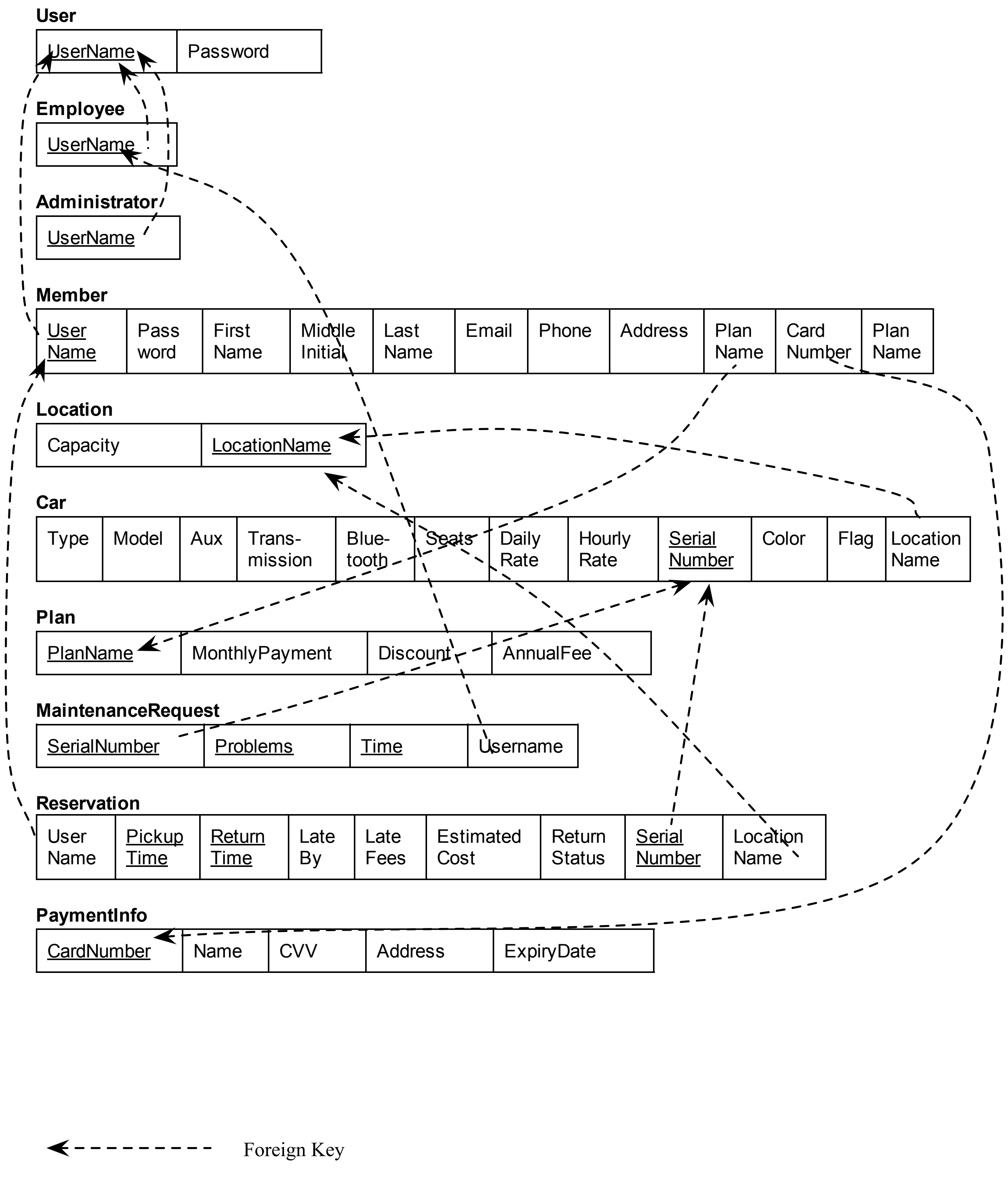


TABLE SCHEMA DIAGRAM

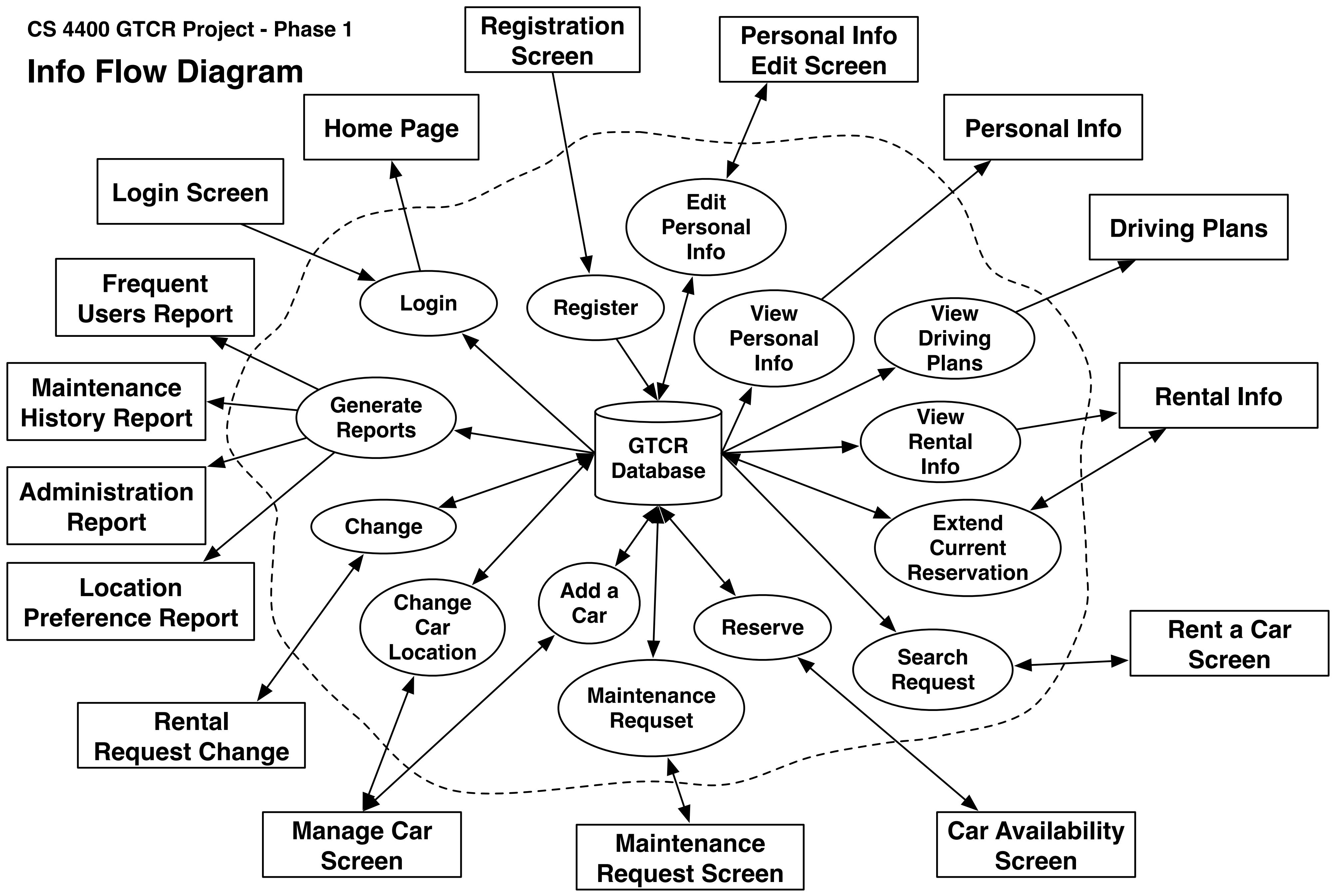


QUERIES OF CREATING TABLES

```
CREATE TABLE User   (
        UserName varchar(50) not null,
        Password varchar(50) not null,
        PRIMARY KEY (UserName));

CREATE TABLE Employee   (
        UserName varchar(50) not null,
        PRIMARY KEY (UserName),
        FOREIGN KEY (UserName) REFERENCES User(UserName));

CREATE TABLE Administrator   (
        UserName varchar(50) not null,
        PRIMARY KEY (UserName),
        FOREIGN KEY (UserName) REFERENCES User(UserName));

CREATE TABLE Member   (
        UserName varchar(50) not null,
        FirstName varchar(50) not null,
        MiddleInitial varchar(50) not null,
        LastName varchar(50) not null,
        Email varchar(50) not null,
        Phone varchar(20),
        CardNumber char(20),
        PlanName varchar(50) not null,
        PRIMARY KEY (UserName),
        FOREIGN KEY (UserName) REFERENCES User(UserName),
        FOREIGN KEY (CardNumber) REFERENCES PaymentInfo(CardNumber),
        FOREIGN KEY (PlanName) REFERENCES Plan(PlanName));

CREATE TABLE Location (
        Capacity int,
        LocationName varchar(50) not null,
        PRIMARY KEY (LocationName));

CREATE TABLE Car (
        Type varchar(20) not null,
        Model varchar(50) not null,
        Aux varchar(3) not null,
        Transmission varchar(10) not null,
        Bluetooth varchar(3) not null,
        Seats int,
        HourlyRate float,
        DailyRate float,
        SerialNumbervarchar(20) not null,
        Color varchar(20) not null,
        LocationNamevarchar(50) not null,
        Flag varchar(5) not null DEFAULT 1,
        PRIMARY KEY (SerialNumber)
        FOREIGN KEY (LocationName) REFERENCES Location(LocationName));

CREATE TABLE Plan (
        PlanNameVarchar(20) not null,
        MonthlyPayment   float,
        Discount float,
        AnnualFee float,
        PRIMARY KEY(PlanName));

CREATE TABLE MaintenanceRequest(
        SerialNumber varchar(20) not null,
        Problems varchar(255),
        Time datetime,
        UserName varchar(50) not null,
        PRIMARY KEY (SerialNumber, Time, Problems),
        FOREIGN KEY (SerialNumber) REFERENCES Car (SerialNumber),
        FOREIGN KEY (UserName) REFERENCES Employee(UserName));

CREATE TABLE Reservation(
        UserName varchar(50) not null,
        SerialNumber varchar(50) not null,
        LocationName varchar(50) not null,
        PickupTime datetime,
        ReturnTime datetime,
        LateBy float DEFAULT 0,
        LateFees float DEFAULT 0,
        Seats integer,
        ExtendedTime datetime,
        EstimatedCost float,
        ReturnStatus varchar(10),
        PRIMARY KEY (SerialNumber, PickupTime, ReturnTime),
        FOREIGN KEY (UserName) REFERENCES Member (UserName),
        FOREIGN KEY (SerialNumber) REFERENCES Car (SerialNumber),
        FOREIGN KEY (LocationName) REFERENCES Location(LocationName));

CREATE TABLE PaymentInfo(
        CardNumberchar(20),
        Address varchar(50) not null,
        CVV varchar(3) not null,
        Name varchar(50) not null,
        ExpiryDate date,
        PRIMARY KEY (CardNumber));


CREATE ASSERTION EXPIRYDATE_CONSTRAINT
CHECK (NOT EXISTS (
        SELECT *
        FROM PaymentInfo
        WHERE ExpiryDate<CURDATE()
)    );



CREATE ASSERTION BOOKTIME_CONSTRAINT
CHECK (NOT EXISTS (
        SELECT *
        FROM Reservation
        WHERE TIMESTAMPDIFF(HOUR, PickupTime,ReturnTime) > 24
)    );



CREATE ASSERTION NOOVERLAP_CONSTRAINT
CHECK (NOT EXISTS (
        (SELECT PickupTime, ReturnTime   FROM Reservation) r1
        INNER JOIN
        (SELECT PickupTime, ReturnTime   FROM Reservation) r2
        ON r1.PickupTime<r2.ReturnTime AND r1.ReturnTime>r2.PickupTime
)    );


CREATE ASSERTION PERIODAVAILAB_CONSTRAINT
CHECK (NOT EXISTS (
        SELECT *
        FROM Reservation
        WHERE PickupTime>ReturnTime
)    );
```

CS 4400 GTCR Project - Phase 1

Info Flow Diagram



QUERIES OF TASKS

In this chapter, we list queries for tasks, which are categorized and presented in the following order.

Almost all queries are successfully tested in our group's MYSQL database:
https://academic-mysql.cc.gatech.edu/phpmyadmin
Username: cs4400_Group_36

Variables with '$' in front are assigned from outer application (such as PHP _REQUEST() variables). For example, in 'Login' task, the variable '$UserName' is assigned by a PHP login page.

We strongly suggest you check the queries in the above database.

General Tasks

- **Login**
- **Register (Create an Account)**

// **Login**

```
SELECT *
FROM User
WHERE UserName=$UserName AND Password = $Password
```

// **Register (Create an Account)**

```
INSERT INTO member
    ( UserName, Password, FirstName, MiddleInitial, LastName, Email, Phone,
    Address, PlanName, CardNumber )
VALUES
    ( $UserName, $Password, $FirstName, $MiddleInitial, $LastName, $Email,
    $Phone, $Address, $PlanName, $CardNumber )
```

Member's Tasks

- **View Personal Info**
- **Edit Personal Info**
- **Edit Payment Information**
- **View Driving plans**
- **Search Request**
- **Reserve**
- **Extend Current Reservation**
- **View Rental Info**

```
// View Personal Info
SELECT
    FirstName=$FirstName, MiddleInitial=$MiddleInitial, LastName=$LastName,
    Email=$Email, Phone=$Phone, Address=$Address
FROM Member
WHERE UserName=$UserName


// Edit Personal Info
UPDATE Member
SET
    FirstName=$FirstName, MiddleInitial=$MiddleInitial, LastName=$LastName,
    Email=$Email, Phone=$Phone, Address=$Address
WHERE UserName=$UserName;

UPDATE Member
SET
    PlanName=$PlanName
WHERE UserName=$UserName;
```

// Edit Payment Information (3 queries)

```
DELETE FROM PaymentInfo
WHERE CardNumber=$CardNumber;

INSERT INTO PaymentInfo
    ( CardNumber, Name, CVV, Address, ExpiryDate )
VALUE
    ( $CardNumber, $Name, $CVV, $Address, $ExpiryDate );

UPDATE Member
SET CardNumber=$CardNumber
WHERE UserName=$UserName;
```

// View Driving plans

```
SELECT *
FROM Plan
WHERE PlanName=$PlanName
```

// Search Request

```
SET @SRPickupTime := CAST('2013-3-26 12:00:00' AS datetime);
SET @SRReturnTime := CAST('2013-3-27 12:00:00' AS datetime);
SET @SRResvTime := FORMAT(TIMESTAMPDIFF(SECOND,@SRPickupTime,
@SRReturnTime)/3600,2);
SET @SRLocationName := 'Clough';
SET @SRType := 'Sedan';
SET @UserPlanDiscount := 0.85;
// SET user defined variables to test queries

SELECT
    SerialNumber, Model, Type, LocationName AS 'Location', Color, HourlyRate,
    HourlyRate*0.9 AS 'DiscountedRate (Frequent)',
    HourlyRate*0.85 AS 'DiscountedRate (Daily)',
    DailyRate, Seats, Transmission, Bluetooth, Aux, AvailableTill,
    FORMAT(HourlyRate*@UserPlanDiscount*@SRResvTime, 2) AS   'Estimated
    Cost'
FROM(
    (Car c)
    INNER JOIN
    (
    SELECT SNumber, MIN(PickupTime) AS AvailableTill
    FROM (
        (
          SELECT DISTINCT
               SerialNumber AS SNumber, LocationName AS LName
          FROM Car
          WHERE SerialNumber NOT IN (
               SELECT DISTINCT SerialNumber
               FROM Reservation
               WHERE (SerialNumber, PickupTime, ReturnTime) NOT IN (
                    SELECT SerialNumber, PickupTime, ReturnTime
                    FROM Reservation
                    WHERE
                         ReturnTime<@SRPickupTime
                         OR
                         PickupTime >@SRReturnTime))
                         AND
                         LocationName=@SRLocationName
                         AND
                         Type=@SRType
                         AND
                         Flag=1
                    ) available_car
                    LEFT OUTER JOIN
                    (
                    SELECT * FROM
                    Reservation
                    WHERE
                         PickupTime > @SRReturnTime
                         AND
                         FORMAT(TIMESTAMPDIFF(SECOND, @SRReturnTime,
                         PickupTime)/3600, 2) <= 12
                    ) resv
                    ON available_car.SNumber = resv.SerialNumber
                    AND available_car.LName = resv.LocationName
               )
          GROUP BY(SNumber)
          ) ac
          ON c.SerialNumber = ac.SNumber
     )
```

// Reserve
```
INSERT INTO Reservation
    ( UserName, PickupTime, ReturnTime, LateBy, LateFees, ExtendedTime,
    EstimatedCost, ReturnStatus, SerialNumber, LocationName )
VALUES
    ( $UserName, $PickupTime,$ ReturnTime,$LateBy, $LateFees, $ExtendedTime,
    $EstimatedCost, $ReturnStatus, $SerialNumber, $LocationName )
```

// Extend Current Reservation
```
UPDATE Member
SET ExtendedTime=$ExtendedTime
WHERE (
    UserName=$UserName
    AND
    ReservationTime=$ReservationTime
    AND
    ActualReturnTime=$ActualReturnTime
)
```

// View Rental Info
```
SELECT EstimatedCost, LocationName, PickupTime, ReturnTime, Model
FROM (
    (
    SELECT
        SerialNumber AS SN, EstimatedCost, LocationName, PickupTime,
        ReturnTime
    FROM Reservation
    WHERE UserName=$UserName AND ReturnTime>NOW()
    ORDER BY ReturnTime DESC
    LIMIT 1
    ) m
    INNER JOIN
    (
    SELECT SerialNumber AS SNo, Model
    FROM Car
    ) c
    ON c.SNo=m.SN
)
SELECT EstimatedCost, LocationName, PickupTime, ReturnTime, Model
FROM (
(
      SELECT
```

```
        SerialNumber AS SN, EstimatedCost, LocationName, PickupTime,
        ReturnTime
        FROM Reservation
        WHERE UserName=$UserName AND ReturnTime<NOW()
        ) m
        INNER JOIN
        (
        SELECT SerialNumber AS SNo, Model
        FROM Car
        ) c
        ON c.SNo=m.SN
)
```

Employee's Tasks

- **Add a Car**
- **Change Car Location**
- **Maintenance Request**
- **Change (Rental Request Change)**

// **Add a Car**

```
INSERT INTO Car
    ( Type, Model, Aux, Transmission, Bluetooth, Seats, DailyRate, HourlyRate,
    SerialNumber, Color, Flag, LocationName )
VALUES
    ( $Type,$Model, $Aux, $Transmission, $Bluetooth, $Seats, $DailyRate,
    $HourlyRate, $SerialNumber, $Color, $Flag, $LocationName )
```

// **Change Car Location**

```
UPDATE Car
    SET LocationName=$NewLocationName
WHERE LocationName=$OldLocationName AND Type=$Type
```

// **Maintenance Request**

```
INSERT INTO MaintenanceRequest
    ( SerialNumber, Problems, Time, UserName)
VALUES
    ( $SerialNumberr, $Problems, $Time, $UserName )
```

// **Change (Rental Request Change)**

```
SELECT SerialNumber, PickupTime
FROM Reservation
WHERE
    UserName = $UserName AND PickupTime<NOW()
GROUP BY PickupTime DESC
LIMIT 1

SELECT Model, LocationName
FROM Car
WHERE SerialNumber= $SerialNumber
```

//if no same-car reservation by other memeber B
//just "INSERT INTO" Reservation - ReturnTime=$ApproxBackTime of this member

```
INSERT INTO Reservation
    ( UserName, PickupTime, ReturnTime, ExtendedTime, EstimatedCost,
    ReturnStatus, SerialNumber, LocationName )
VALUES
    ( $UserName, $PickupTime,$ApproxBackTime,$ExtendedTime, $EstimatedCost,
    $ReturnStatus, $SerialNumber, $LocationName )

//If another reservation is affected(extended not allowed) of member B;
//1. INSERT INTO Reservation "ReturnTime"=$ApproxBackTime, "EstimatedCost",
"LateBy" ,"LateFees" in Reservation of A (Calculation done by PHP)
//2. SELECT member B'info (Reservation)
WHERE SerialNumber=$LateCarSeriNo AND PickupTime<$ApproxBackTime
INSERT INTO Reservation
    ( UserName, PickupTime, ReturnTime, LateBy, LateFees, ExtendedTime,
    EstimatedCost, ReturnStatus, SerialNumber, LocationName)
VALUES
    ( $UserName, $PickupTime,$ApproxBackTime,$LateBy, $LateFees,
    $ExtendedTime, $EstimatedCost, $ReturnStatus, $SerialNumber,
    $LocationName )

SELECT Username, PickupTime, ReturnTime
FROM Reservation
WHERE
    SerialNumber=$LateCarSeriNo AND PickupTime<$ApproxBackTime AND
    PickupTime>NOW()

//PHP do the assignment, $AffectedUserName()
SELECT Email, Phone
FROM Member
WHERE UserName=$AffectedUserName

//If B chooses to cancle,
//3-a. delect the tuple
DELETE FROM Reservation
WHERE
    SerialNumber=$LateCarSeriNo AND UserName=$AffectedUserName AND
    PickupTime=$AffectedPickupTime

//If B chooses to re-book,
//3-b. show car availability
DELETE FROM Reservation
WHERE
    SerialNumber=$LateCarSeriNo AND UserName=$AffectedUserName AND
    PickupTime=$AffectedPickupTime
```

```
//when click "show car availability" button, go to the "Rent a Car Screen" and do
"search request"
INSERT INTO Reservation
    ( UserName, PickupTime, ReturnTime,EstimatedCost, ReturnStatus,
    SerialNumber, LocationName )
VALUES
    ( $UserName, $PickupTime, $ReturnTime, $EstimatedCost, $ReturnStatus,
    $SerialNumber, $LocationName )
```

Administrator’s Tasks

- **Frequent User Report**
- **Maintenance History Report**
- **Administration Report**
- **Location Preference Report**

```
// Frequent User Report
SELECT UserName, PlanName, ResvNo
FROM
(
    (
    SELECT UserName As Name, Count(*) AS ResvNo
    FROM Reservation
    GROUP BY UserName
    ) resv
    INNER JOIN
    Member
    ON UserName=Name
)
ORDER BY ResvNo DESC
LIMIT 5


// Maintenance History Report
SELECT
    Car, Time AS 'Date-time', UserName AS 'Employee', Problems AS 'Problem'
FROM
(
    (
    SELECT *
    FROM
    MaintenanceRequest
    ) mt
    INNER JOIN
    (
    SELECT
        SerialNumber AS SNumber, Model AS Car, Time AS T, UserName AS UN,
        COUNT(Problems) AS Count
    FROM
    (
        (
        SELECT SerialNumber AS SN, Model
```

```
		FROM Car
		) c
		INNER JOIN
		MaintenanceRequest mr
		ON c.SN = mr.SerialNumber
	)
	GROUP BY SerialNumber, Time
	) mc
	ON mt.SerialNumber=mc.SNumber AND mt.Time=mc.T
)
ORDER BY Count DESC, SerialNumber, Time
```

// **Administration Report**

```
SELECT SerialNo, Model, Type, ReservationRevenue,LateFeesRevenue
FROM
(
	(
	SELECT
		SerialNumber AS SerialNo, SUM(EstimatedCost) AS
		ReservationRevenue, SUM(LateFees) AS LateFeesRevenue
	FROM Reservation r
	WHERE (
		(MONTH(PickupTime) = (SELECT DISTINCT MONTH(CURDATE())-1
		FROM Reservation))
		OR
		(MONTH(PickupTime) = (SELECT DISTINCT MONTH(CURDATE())-2
		FROM Reservation))
		OR
		(MONTH(PickupTime) = (SELECT DISTINCT MONTH(CURDATE())-3
		FROM Reservation))
		AND
		(YEAR(PickupTime) = (SELECT DISTINCT YEAR(CURDATE())   FROM
		Reservation))
	)
	GROUP BY r.SerialNumber
	) revn
	INNER JOIN
	Car c
	ON c.SerialNumber=revn.SerialNo
)
```

// Location Preference Report

```
SELECT DISTINCT ResvMonth, LocationName, TotalResv,TotalHour
FROM (
    (
    SELECT
        MONTH(PickupTime) AS ResvMonth, LocationName, PickupTime,
        COUNT(*) AS TotalResv,
        SUM(FORMAT(TIMESTAMPDIFF(SECOND,PickupTime,ReturnTime)/3
        600,2) ) AS TotalHour
    FROM Reservation
    WHERE (
        (MONTH(PickupTime) = (SELECT DISTINCT MONTH(CURDATE())-1
        FROM Reservation))
        OR
        (MONTH(PickupTime) = (SELECT DISTINCT MONTH(CURDATE())-2
        FROM Reservation))
        OR
        (MONTH(PickupTime) = (SELECT DISTINCT MONTH(CURDATE())-3
        FROM Reservation))
        AND
        (YEAR(PickupTime) = (SELECT DISTINCT YEAR(CURDATE())   FROM
        Reservation))
    )
    GROUP BY MONTH(PickupTime), LocationName
    ) resvcount
    INNER JOIN
    (
    SELECT DISTINCT MAX(TotalResv) AS ResvMax
    FROM   CountPerMonLoc
    GROUP BY MONTH(PickupTime)
    ) maxresv
    ON resvcount.TotalResv=maxresv.ResvMax
)
```