CS355 MINI PROJECT

PROJECT DOCUMENTATION

Team 03

Gul Jain 1901CS30

Ishita Singh 1901CS27

Kavya Goyal 1901CS30

Link: CS355 Mini Project

1901CS22

Gul Jain

Guest House

ER DIAGRAM

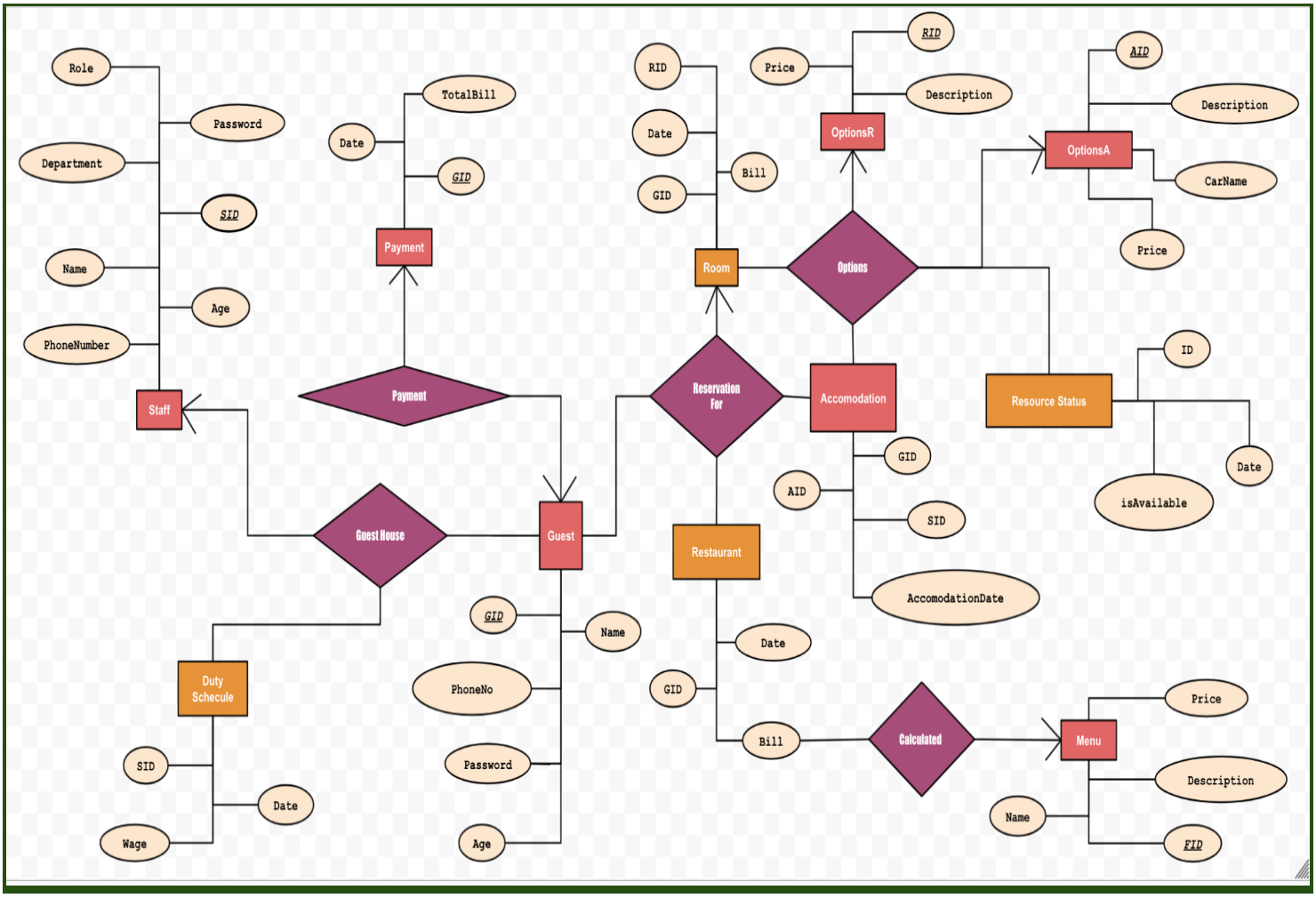
Database Relational Schema

FOLDER (Video, Code, Readme File, ScreenShots, data files, dump files etc)

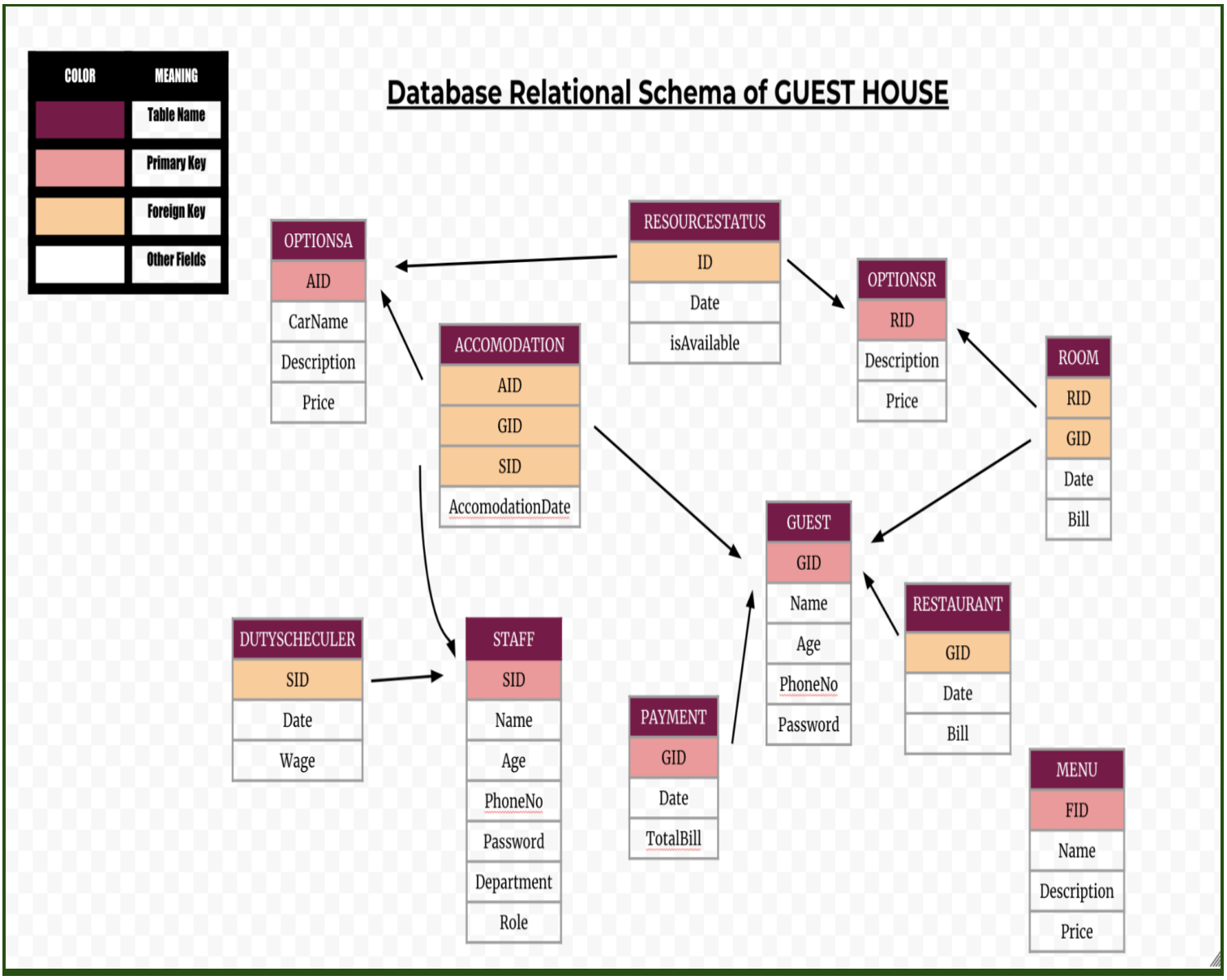
- ➔ All working of the project is included in video and detailed is included in the screenshots
- ➔ All data related files are stored in folder " DATA RELATED FILES "
  - ◆ Dump file : sql format
  - ◆ Table file : pdf format
  - ◆ All data : csv format
- ➔ Documentation is here
- ➔ Code : " Guest House "
- ➔ Screenshots : "Screenshots of project working"
- ➔ ER Diagram and Relational Schema link is above and also as screenshot

ANALYSIS OF ER-MODELS AND RELATIONS

ER Diagram



DATABASE RELATIONAL SCHEMA



ENTITIES

1. GUEST Entity

   Insight :

   Stores information about Guest

   Attributes :

   - GID : varchar(100), Primary Key
   - Name : varchar(100)
   - Age : int(11)
   - PhoneNo : int(11)
   - Password : varchar(100)

   Index :

   - GID

     ```
     ALTER TABLE guest ADD INDEX idx_guest_GID (GID);
     ```

2. ROOM Entity

   Insight :

   Stores information about which Room is occupied by which Guest and when was the payment made

   Attributes :

   - RID : varchar(100), Foreign Key to @OPTIONSR
   - GID : varchar(100), Foreign Key to @GUEST
   - Bill : int(11), bill for booking room
   - Date : date

   Index :

   - GID

     ```
     ALTER TABLE room ADD INDEX idx_room_GID (GID);
     ```

   Foreign Key :

   - RID

     ```
     ALTER TABLE room ADD CONSTRAINT fk_room_RID FOREIGN KEY(RID) REFERENCES optionsR(RID);
     ```

   - GID

     ```
     ALTER TABLE room ADD CONSTRAINT fk_room_GID FOREIGN KEY(GID) REFERENCES guest(GID);
     ```

3. RESTAURANT Entity

Insight :

Stores information about which Guest occupied which Table, what did the Guest order and Bill generated

Attributes :

- GID : varchar(100), Foreign Key to @GUEST
- Date : date
- Bill : int(11), bill for ordering food

Index :

- GID

```
ALTER TABLE restaurant ADD INDEX idx_restaurant_GID (GID);
```

Foreign Key :

- GID

```
ALTER TABLE restaurant ADD CONSTRAINT fk_restaurant_GID FOREIGN KEY(GID)
REFERENCES guest(GID);
```

4. ACCOMODATION Entity

Insight :

Stores information about which Guest has which accomodation booked, which staff has been assigned the duty and date.

Attributes :

- AID : varchar(100), Foreign Key to @OPTIONSA
- GID : varchar(100), Foreign Key to @GUEST
- SID : varchar(100), Foreign Key to @STAFF
- AccomodationDate : date

Index :

- GID

```
ALTER TABLE accomodation ADD INDEX idx_accomodation_GID (GID);
```

Foreign Key :

- AID

```
ALTER TABLE accomodation ADD CONSTRAINT fk_accomodation_AID FOREIGN
KEY(AID) REFERENCES optionsA(AID);
```

- GID

```
ALTER TABLE accomodation ADD CONSTRAINT fk_accomodation_GID FOREIGN
KEY(GID) REFERENCES guest(GID);
```

- SID

```
ALTER TABLE accomodation ADD CONSTRAINT fk_accomodation_SID FOREIGN
KEY(SID) REFERENCES staff(SID);
```

5. RESOURCESTATUS Entity

Insight :
Stores ID of resources (rooms and accomodation option) and availability status on a date.

Attributes :

- ID : varchar(100), either RID or AID
- Date : date
- isAvailable : int(11), '1' if resource is available for a particular date, else '0'

Index :

- ID

```
ALTER TABLE resourceStatus ADD INDEX idx_resourceStatus_ID (ID);
```

6. PAYMENT Entity

Insight :
Stores information the total payment by guest and the date.

Attributes :

- GID : varchar(100), Primary Key, Foreign Key @GUEST
- Date : date, most recent date of payment
- TotalBill : int(11), total bill for a particular guest

Index :

- GID

```
ALTER TABLE payment ADD INDEX idx_payment_GID (GID);
```

Foreign Key :

- GID

```
ALTER TABLE payment ADD CONSTRAINT fk_payment_GID FOREIGN KEY(GID)
REFERENCES guest(GID);
```

7. STAFF Entity

Insight :

Stores information of the Staff

Attributes :

- SID : varchar(100), Primary Key
- Name : varchar(100)
- Age : int(11)
- PhoneNumber : int(11)
- Department : either “RoomStaff”, “RestaurantStaff” or “Accommodation Staff”
- Role : varchar(100), Further Role in the department
- Password : varchar(100)

Index :

- SID

```
ALTER TABLE staff ADD INDEX idx_staff_SID (SID);
```

8. DUTYSCHECULE Entity

Insight :

Stores information about which staff has worked on which date and wage of that day

Attributes :

- SID : varchar(100), Foreign Key to @STAFF
- Date : date
- Wage : int(11)

Index :

- SID

```
ALTER TABLE dutySchecule ADD INDEX idx_dutySchecule_SID (SID);
```

Foreign Key :

- SID

```
ALTER TABLE dutySchecule ADD CONSTRAINT fk_dutySchecule_SID FOREIGN
KEY(SID) REFERENCES staff(SID);
```

9. OPTIONSA Entity

Insight :

Stores information about all cars

Attributes :

- AID : varchar(100), Primary Key
- carName : varchar(100)
- Description : varchar(500)
- Price : int(11)

10. OPTIONSR Entity

Insight :

Stores information about all rooms

Attributes :

- RID : varchar(100), Primary Key
- Description : varchar(500)
- Price : int(11)

11. MENU Entity

Insight :

Stores information about all food items

Attributes :

- FID : varchar(100), Primary Key
- Name : varchar(100)
- Description : varchar(500)
- Price : int(11)

RELATIONSHIPS

- *<Reservation For>*

    - ***Degree*** : 4-ary relationship between GUEST, ROOM, ACCOMODATION and RESTAURANT.
    - ***Relationships and Cardinality*** :
        - One-to-one relationship between ROOM and GUEST, one guest can book at most one room. Cardinality => (1,1)
        - One-to-many relationship between GUEST and RESTAURANT, a guest can order food as many times as he desires. Cardinality => (1,n)
        - One-to-many relationship between GUEST and ACCOMODATION, a guest can book a car as many times as he desires, given that car and driver should be available and booking is done for a single day only. Cardinality => (1,n)

- *<Guest House Duty>*
    - ***Degree*** : Ternary Relationship between GUEST, STAFF and STAFFSCHECULE
    - ***Relationships and Cardinality*** :
        - One-to-many relationship between STAFF and STAFFSCHECULE, one staff can have multiple duties either scheduler or completed. Cardinality => (1,n)
        - Many-to-many relationship between STAFF and GUEST, a staff can work for a particular guest many times and some guests can have the same staff working for them. Cardinality => (n,n)

- *<Payment>*
    - ***Degree*** : Binary Relationship between GUEST and PAYMENT
    - ***Relationships and Cardinality*** :
        - One-to-one relationship between GUEST and PAYMENT, one guest can have one entry in PAYMENT which tells the total amount paid by the guest. Cardinality => (1,1).

- *<Options>*
    - ***Degree*** : 5-ary Relationship between ROOM, OPTIONSA, OPTIONSR, ACCOMODATION and RESOURCESTATUS.
    - ***Relationships and Cardinality*** :
        - One-to-many relationship between OPTIONSR and ROOM, one room can have multiple bookings by different guest. Cardinality => (1,n).
        - One-to-many relationship between OPTIONSA and ACCOMODATION, one car can have multiple bookings. Cardinality => (1,n).

    - One-to-many relationship between OPTIONSR and RESOURCESTATUS, one room can have multiple entries to know which rooms are/were occupied at a certain date. Cardinality => (1,n).
    - One-to-many relationship between OPTIONSA and RESOURCESTATUS, one car can have multiple entries to know which cars are/were occupied at a certain date. Cardinality => (1,n).

- *<Calculated>*
    - ***Degree*** : Binary Relationship between MENU and RESTAURANT
    - ***Relationships and Cardinality*** :
        - Many-to-many relationship between MENU and RESTAURANT, a guest can have multiple orders of a food item and a food item can be ordered by multiple guests. Cardinality => (n,n).

Tasks and Queries to be performed with SQL representations

1. Details of all the tables

```
select * from Guest;
select * from Room;
select * from Accomodation;
select * from Restaurant;
select * from ResourceStatus;
select * from Payment;
select * from Staff;
select * from DutySchedule;
select * from Menu;
select * from OptionsA;
select * from OptionsR;
```

2. Queries of “Register User”

-> Checking if the user already exists.

```
  $user_check_query = "SELECT * FROM guest WHERE GID = '$username'";
```

-> If user does not exist, we insert into table “guest”

```
 $query = "INSERT INTO guest (GID, name, age, phoneNo, password) VALUES
( '$username', '$name', '$age', '$mobileNumber', '$password_1' ) ";
```

3. Login User Queries

```
                          $query = "SELECT * FROM guest WHERE GID =
'$username' AND password = '$password' ";
```

4. Checking if particular room is available on dates given

-> Checking for every room if it is available between all given dates which can be obtained by “resourceStatus” table

```
$query1 = "Select date from resourceStatus where id ='R1' and date
between '$date1' AND '$date2'";
```

```
$query2 = "Select date from resourceStatus where id ='R2' and date
between '$date1' AND '$date2'";
$query3 = "Select date from resourceStatus where id ='R3' and date
between '$date1' AND '$date2'";
$query4 = "Select date from resourceStatus where id ='R4' and date
between '$date1' AND '$date2'";
$query5 = "Select date from resourceStatus where id ='R5' and date
between '$date1' AND '$date2'";
$query6 = "Select date from resourceStatus where id ='R6' and date
between '$date1' AND '$date2'";
$query7 = "Select date from resourceStatus where id ='R7' and date
between '$date1' AND '$date2'";
```

5. Checking if particular car is available on certain date or not

-> Checking for every car if it is available between all given dates which can be obtained by "resourceStatus" table

```
$query1 = "Select date from resourceStatus where id ='A1' and date
between '$date1' AND '$date2'";
$query2 = "Select date from resourceStatus where id ='A2' and date
between '$date1' AND '$date2'";
$query3 = "Select date from resourceStatus where id ='A3' and date
between '$date1' AND '$date2'";
$query4 = "Select date from resourceStatus where id ='A4' and date
between '$date1' AND '$date2'";
$query5 = "Select date from resourceStatus where id ='A5' and date
between '$date1' AND '$date2'";
$query6 = "Select date from resourceStatus where id ='A6' and date
between '$date1' AND '$date2'";
```

6. When User pays, Following queries are checked

-> Check if the GuestID entered is same as the login user

```
 $query = "SELECT * from guest where gid = '$gid'";
```

-> Update "Payment" table by adding the new amount, if user has at least one payment initially

```
$check = mysqli_query($db, "Select * from payment where gid = '$gid' ");
 $update_ = "update payment set totalbill = totalbill + '$bill' , date =
'$timezone' where gid = '$gid' ";
```

-> If user has its first payment, we enter details into “payment” table

```
              $insert = "Insert into payment (GID, date, totalBill)
VALUES ( '$gid' , '$timezone', '$bill' )";
```

-> If the user ordered food, “restaurant” table and “dutyschedule” table is updated

```
             $insert = "INSERT INTO restaurant (gid, date, bill) VALUES
('$gid', '$timezone', '$bill')";
             $staff = "INSERT INTO dutySchecule (sid, date, wage)
VALUES ('S3', '$timezone', 50)";
             $staff_ = "INSERT INTO dutySchecule (sid, date, wage)
VALUES ('S1', '$timezone', 200)";
```

-> If the user books room, “room” table, “dutyschedule” table and “resourcestatus” is updated

```
             $insert = "INSERT INTO Room (rid, gid, date, bill) VALUES
('$rid', '$gid', '$timezone', '$bill' )";
             while($days--) {
                 echo $date1;
                 echo " ";
                  $insert_ = "INSERT INTO resourceStatus (id, date,
isAvailable) VALUES ('$rid','$date1', 0 )";
                  $date1 = date('Y-m-d', strtotime("+1 day",
strtotime($date1)));
                  $A = mysqli_query($db, $insert_ );
             }              $staff = "INSERT INTO dutySchecule (sid,
date, wage) VALUES ('S6', '$timezone', 100)";
```

-> If the user books car, “accomodation” table, “dutyschedule” table and “resourcestatus” is updated

```
              $insert = "INSERT INTO Accomodation (aid, gid, sid,
accomodationDate) VALUES ('$aid', '$gid', 'S4', '$order' )";
              $insert_ = "INSERT INTO resourceStatus (id, date,
isAvailable) VALUES ('$aid','$order', 0 )";
              $staff = "INSERT INTO dutySchecule (sid, date, wage)
VALUES ('S4', '$order', 1000)";
```

7. Before User books Room, we need to check if user already has a booked room or not

```
$see = "Select rid from Room where gid = '$gid' " ;
```

8. Before User books car, we need to check if driver is available or not.

```
$query = "Select SID from staff where role = 'Driver' and SID not in (select sid from dutySchecule where date = '$date1' ) ";
```

9. Staff Login

```
$query = "SELECT * FROM staff WHERE sid = '$username' AND password = '$password'";
```

10. Outputting required tables for particular month

-> restaurant

```
$query=mysqli_query($db, "select * from restaurant where MONTH(date) = '$monthNo'" ); }
```

-> room

```
$query=mysqli_query($db, "select * from Room where MONTH(date) = '$monthNo'" );
```

-> payment

```
$query=mysqli_query($db, "select * from payment where MONTH(date) = '$monthNo'" ); }
```

-> car

```
$query=mysqli_query($db, "select * from Accomodation where MONTH(accomodationDate) = '$monthNo'" ); }
```

-> duty schedule

```
$query=mysqli_query($db, "select * from dutySchecule where MONTH(date) = '$monthNo'" ); }
```

11. Total bill for a particular guest

```
$payment = mysqli_fetch_array( mysqli_query($db, "select totalBill from payment where GID = '$gid' ")); ?>
```

12. Total wage for a particular staff

```
              $payment = mysqli_fetch_array( mysqli_query($db,  "select
sum(wage) as A from dutySchecule where sid = '$sid' "));
```

13. Total Bill for a particular car

```
            $payment = mysqli_fetch_array(mysqli_query($db, "select
count(*) as A from Accomodation where AID = '$aid' "));
```

1901CS27

Ishita Singh

Landscaping

Landscaping_ERD : ER Diagram
Landscaping_Relational : Relational Model
Landscaping : Google Drive Folder (Video, Code, Readme File, ScreenShots, data files, dump files etc)

- ➔ The working of the project is included in video and explained via further screenshots for the working.
- ➔ All data related files are stored in folder " DATABASE DATA "
  - ◆ Dump file : sql format
  - ◆ Table file : pdf format
  - ◆ All data : csv format
- ➔ Documentation is here
- ➔ Code : " Code Files "
- ➔ Screenshots : "Landscaping Screenshots"
- ➔ ER Diagram and Relational Models' links are given above and are also shown in the drive

LANDSCAPING DEPARTMENT

DATABASE INFORMATION:

DATABASE NAME: **CSPROJECT**
NUMBER OF TABLES: 8

TABLE INFORMATION:

1) **ATTENDANCE**: Stores attendance details as (Present/Absent/NULL), with NULL implying attendance hasn't been marked yet.

   Attributes
   Task_ID : (varchar(10)) Task ID, **indexed**
   GrID : (varchar(10)) Gardener ID
   attendance_mark:(varchar(15)) To store attendance
   ->Primary Key is (Task_ID, GrID)
   ->Foreign Key is (Task_ID) references Roster (Task_ID)

```
--
-- Table structure for table `attendance`
--

CREATE TABLE `attendance` (
  `Task_ID` varchar(10) NOT NULL,
  `GrID` varchar(10) NOT NULL,
  `attendance_mark` varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `attendance`
--

INSERT INTO `attendance` (`Task_ID`, `GrID`, `attendance_mark`) VALUES
('1', '1a', 'Absent'),
('2', '1a', 'Present'),
('4', '1a', 'Absent');

-- --------------------------------------------------------
```

Indexes used in Table

```
ALTER TABLE `attendance` ADD INDEX (`Task_ID`);
```

Primary Key in Table

```
ALTER TABLE `attendance` ADD PRIMARY KEY (`Task_ID`,`GrID`);
```

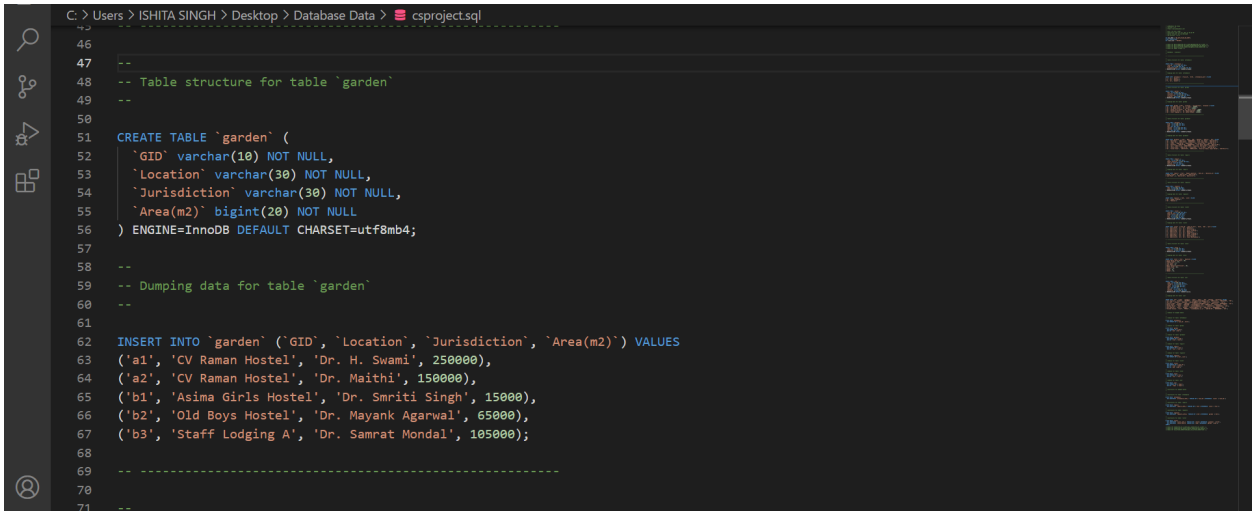
Foreign Key in Table

```
ALTER TABLE `attendance` ADD CONSTRAINT `attendance_ibfk_1` FOREIGN KEY
(`Task_ID`) REFERENCES `roster` (`Task_ID`);
```

2) **GARDEN**: Stores garden/ location area that comes under Landscaping Department

Attributes

GID	:	(varchar(10)) **Primary Key (Indexed)** Garden ID
Location	:	(varchar(30)) Location of Garden
Jurisdiction	:	(varchar(30)) Jurisdiction Head of Landscaping Area
Area(m2)	:	(bigint) Area of Garden in terms of metres squared



Indexes used in Table

```
ALTER TABLE `garden` ADD INDEX (`GID`);
```

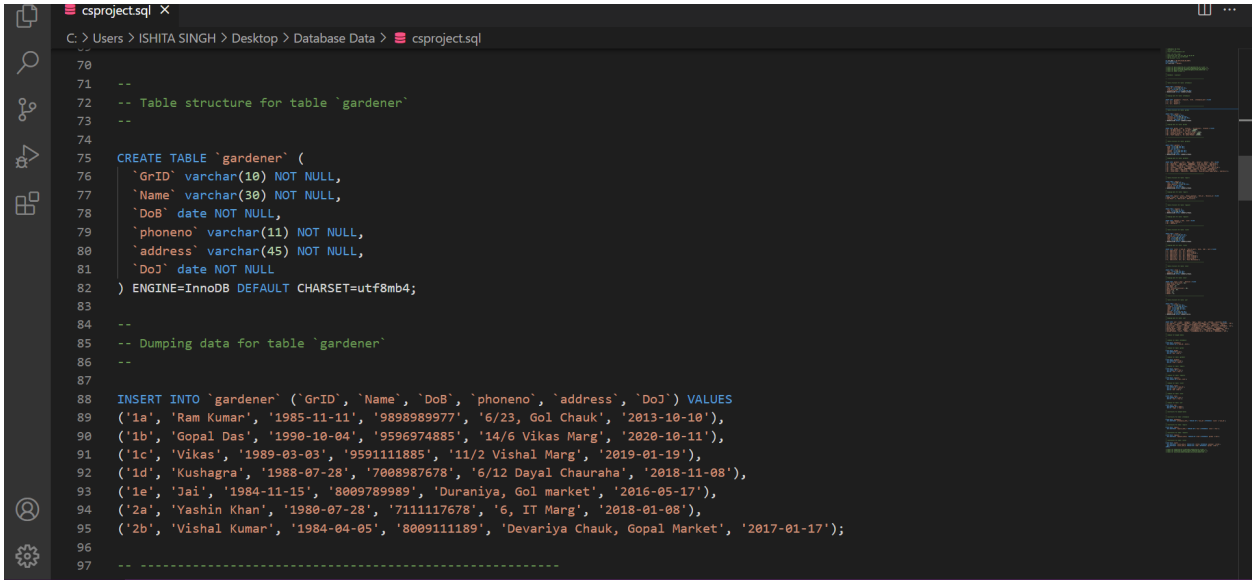
Primary Key in Table

```
ALTER TABLE `garden` ADD PRIMARY KEY (`GID`);
```

3) **GARDENER**: Stores Gardener Information

Attributes

GrID	:	(varchar(10)) **Primary Key (Indexed)** Gardener ID
Name	:	(varchar(30)) Name of the Gardener
DoB	:	(date) Date of Birth of Gardener
Phoneno	:	(varchar(11)) Contact Number of Gardener
Address	:	(varchar(45)) Address of Gardener
DoJ	:	(date) Date when Gardener joined IIT Patna landscaping



Indexes used in Table

```
ALTER TABLE `gardener` ADD INDEX (`GrID`);
```

Primary Key in Table

```
ALTER TABLE `gardener` ADD PRIMARY KEY (`GrID`);
```

4) **REPAIRS**: Stores information on the Tools/ Equipments sent for repair.

Attributes

Tool : (varchar(30)) **Primary Key (Indexed)** Tool/ Equipment Name
Repair_quantity:(int) Quantity sent for Repairs
Sent_On : (date) Date when sent for repair
Receival_On : (date) Date of receival of Tools sent for repair
->Foreign Key is (Tool) references Stock (Tool)

Indexes used in Table

```
ALTER TABLE `repairs` ADD INDEX (`Tool`);
```

Primary Key in Table

```
ALTER TABLE `repairs` ADD PRIMARY KEY (`Tool`),
```

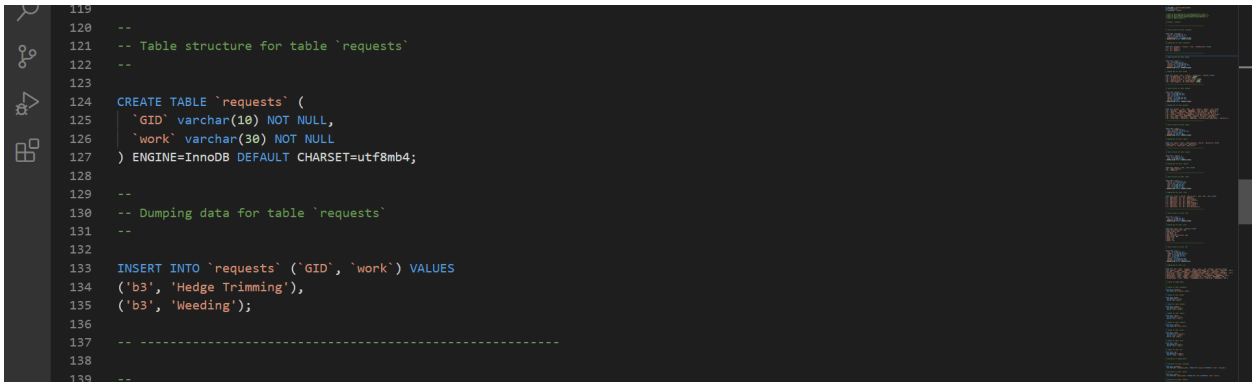
Foreign Key in Table

```
ALTER TABLE `repairs` ADD CONSTRAINT `repairs_ibfk_1` FOREIGN KEY (`Tool`)
REFERENCES `stock` (`Tool`);
```

5) **REQUESTS**: Stores the task requests made by registered users.

Attributes

GID : (varchar(10)) Garden ID
work : (varchar(30)) Work requested on GID
->Primary Key is (GID,work)
->Foreign Key is (GID) references Garden (GID)



```
--
-- Table structure for table `requests`
--

CREATE TABLE `requests` (
  `GID` varchar(10) NOT NULL,
  `work` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `requests`
--

INSERT INTO `requests` (`GID`, `work`) VALUES
('b3', 'Hedge Trimming'),
('b3', 'Weeding');

-- --------------------------------------------------------

--
```

Primary Key in Table

```
ALTER TABLE `requests` ADD PRIMARY KEY (`GID`,`work`);
```

Foreign Key in Table

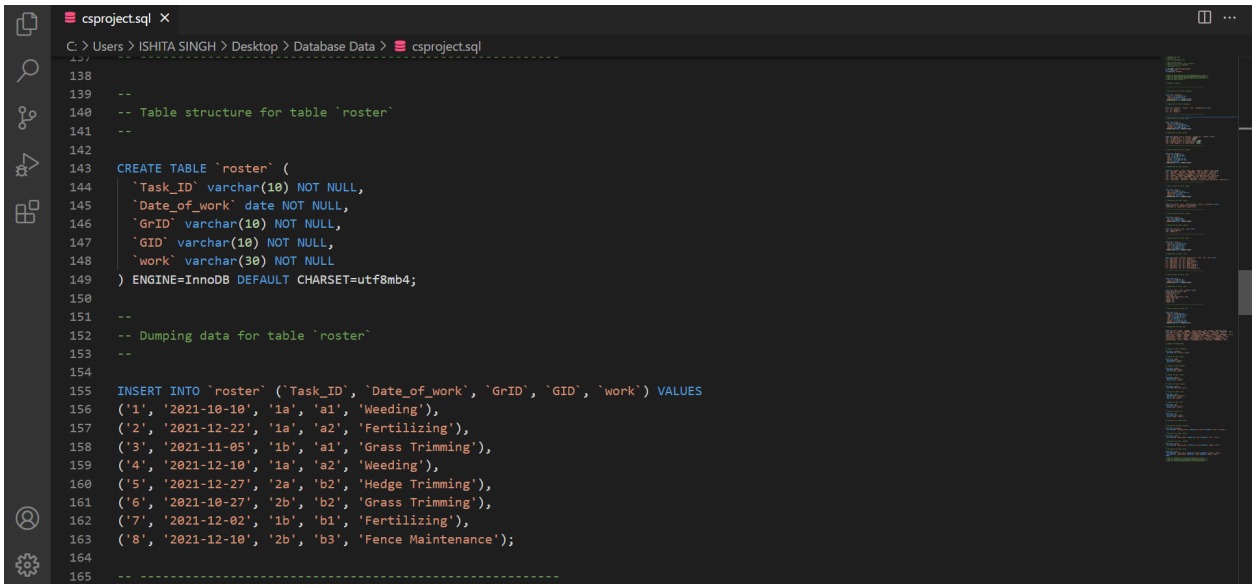
```
ALTER TABLE `requests` ADD CONSTRAINT `requests_ibfk_1` FOREIGN KEY `GID`)
REFERENCES `garden` (`GID`);
```

6) **ROSTER**: Stores the Time table for the tasks to be performed, when to be performed, where to be performed and by whom.

Attributes

Task_ID : (varchar(10)) **Primary Key** Task ID
Date_of_Work : (date) Date to do the mentioned Task
GrID : (varchar(10)) Gardener ID
GID : (varchar(10)) Garden ID

work : (varchar(30)) Assigned Task
->Foreign Key (GrID) references Gardener (GrID)
->Foreign Key (GID) references Garden (GID)



Primary Key in Table

```
ALTER TABLE `roster` ADD PRIMARY KEY (`Task_ID`);
```

Foreign Key in Table

```
ALTER TABLE `roster` ADD CONSTRAINT `roster_ibfk_1` FOREIGN KEY (`GrID`)
REFERENCES `gardener` (`GrID`), ADD CONSTRAINT `roster_ibfk_2` FOREIGN
KEY (`GID`) REFERENCES `garden` (`GID`);
```

7) **STOCK**: Stores the total tools and equipment stock information.

Attributes

Tool : (varchar(30)) **Primary Key (Indexed)** Tool/ Equipment Name
quantity : (int) Net Quantity In Stock

```sql
-- --------------------------------------------------------

--
-- Table structure for table `stock`
--

CREATE TABLE `stock` (
  `Tool` varchar(30) NOT NULL,
  `Quantity` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `stock`
--

INSERT INTO `stock` (`Tool`, `Quantity`) VALUES
('Gloss White Fence Paint', 10),
('Hedge Trimmer', 15),
('Lawn Mower', 5),
('Leaf Rake', 5),
('Magic Potassium Fertilizer', 10),
('Paint Brush', 10),
('Showel', 6),
('Spade', 6),
('Weeder', 6);

-- --------------------------------------------------------
```

Indexes used in Table

```sql
ALTER TABLE `stock` ADD INDEX (`Tool`);
```

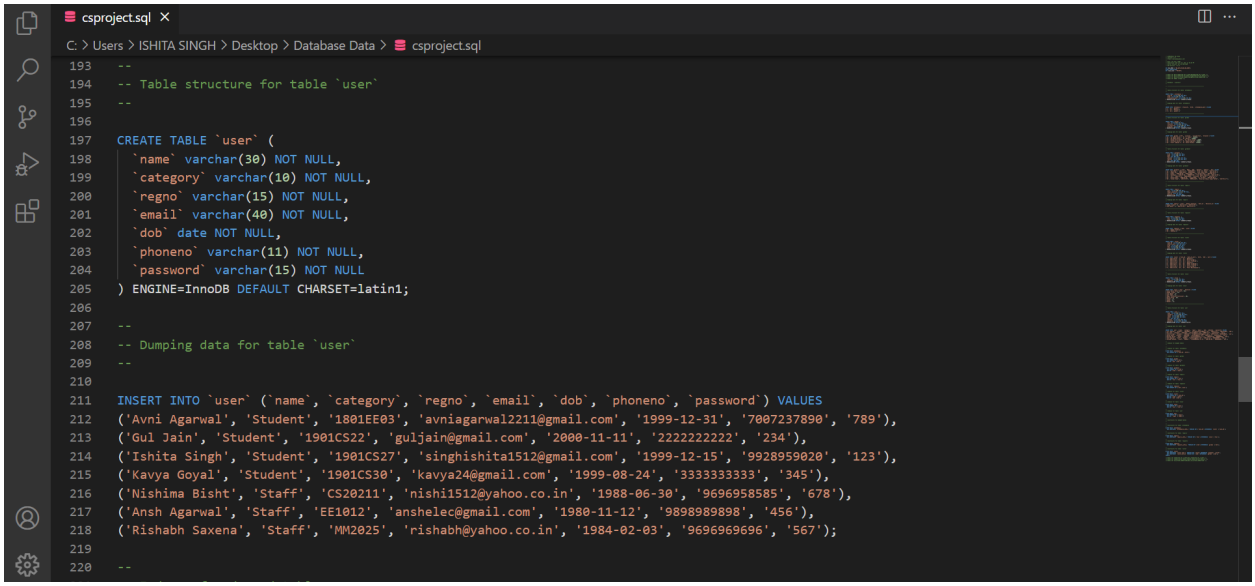
Primary Key in Table

```sql
ALTER TABLE `stock` ADD PRIMARY KEY (`Tool`);
```

8) **USER**: Stores User Information, which allows them to login and send landscaping related requests to the Admin, post Registration.

Attributes

regno	:	(varchar(15)) **Primary Key (Indexed)** Registration Number
name	:	(varchar(30)) Name of the User
category	:	(varchar(10)) Staff/ Student
email	:	(varchar(40)) Email ID of User
dob	:	(date) Date of Birth of User
phoneno	:	(varchar(11)) Contact Number of Information
password	:	(varchar(15)) Password set by User for Login



Indexes used in Table

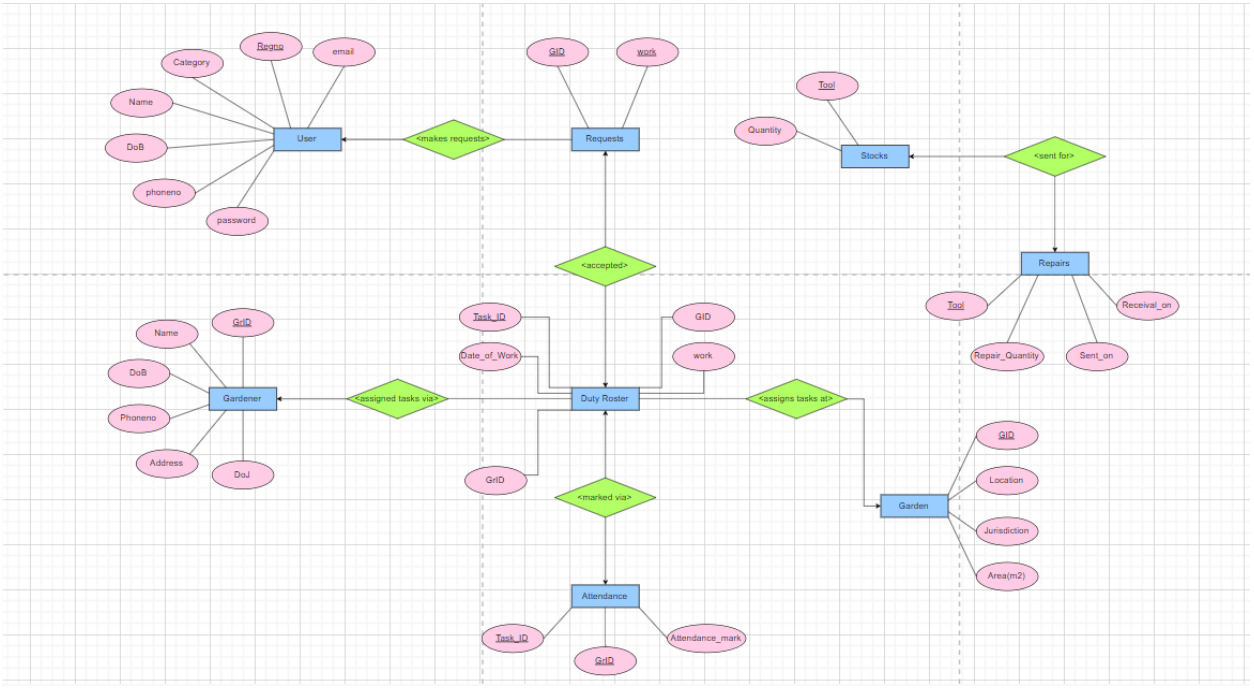
```
ALTER TABLE `user` ADD INDEX (`regno`),
```

Primary Key in Table

```
ALTER TABLE `user` ADD PRIMARY KEY (`regno`),
```

ER DIAGRAM AND ITS ANALYSIS:

ER DIAGRAM:



Number of Entities = 8
Number of Relations = 6

Link to complete ER Model:
[Landscaping_ERD]

ENTITIES' INFORMATION:

All entities' descriptions are the same as explained in the 'Table Information' Part.
Note- Entity corresponding Table 'roster' is shown as 'Duty Roster' in the ER diagram
Entity corresponding Table 'stock' is shown as 'stocks' in the ER diagram
Rest names are same as the corresponding table in 'Table Information' and the descriptions for the 8 tables in the 'Table Information' part suffices for the 8 entities shown in the ER diagram.

RELATIONS' INFORMATION: (Primarly Cardinality Study)

1) **<makes requests>:** Corresponds to users making requests for tasks like Fertilizing, Grass Trimming, Hedge Trimming, Fence Maintenance, Weeding, etc, to be done on Garden with entered GID.

   Binary, One to many relationship between 'User' and 'Requests'.
   One request can be associated with at most one user, whereas one user can be associated with any number of requests.

2) **<accepted>:** Corresponds to user's requests being accepted by the admin.

   Binary, One to one relationship between 'Requests' and 'Roster'/'Duty Roster' [Roster=Duty Roster]
   One request can only be accepted once and hence can only be inserted in roster via this relationship once.

3) **<assigned tasks via>:** Corresponds to tasks assigned to Gardener via Duty Roster.

   Binary, One to many relationship between 'Gardener' and 'Duty Roster'.
   One roster entry can be associated with at most one gardener, whereas one gardener can be associated with any number of roster entries.

4) **<assigns tasks at>:** Corresponds to tasks assigned at Garden via Duty Roster.
   Binary, One to many relationship between 'Garden' and 'Duty Roster'.
   One roster entry can be associated with at most one garden, whereas one garden can be associated with any number of roster entries.

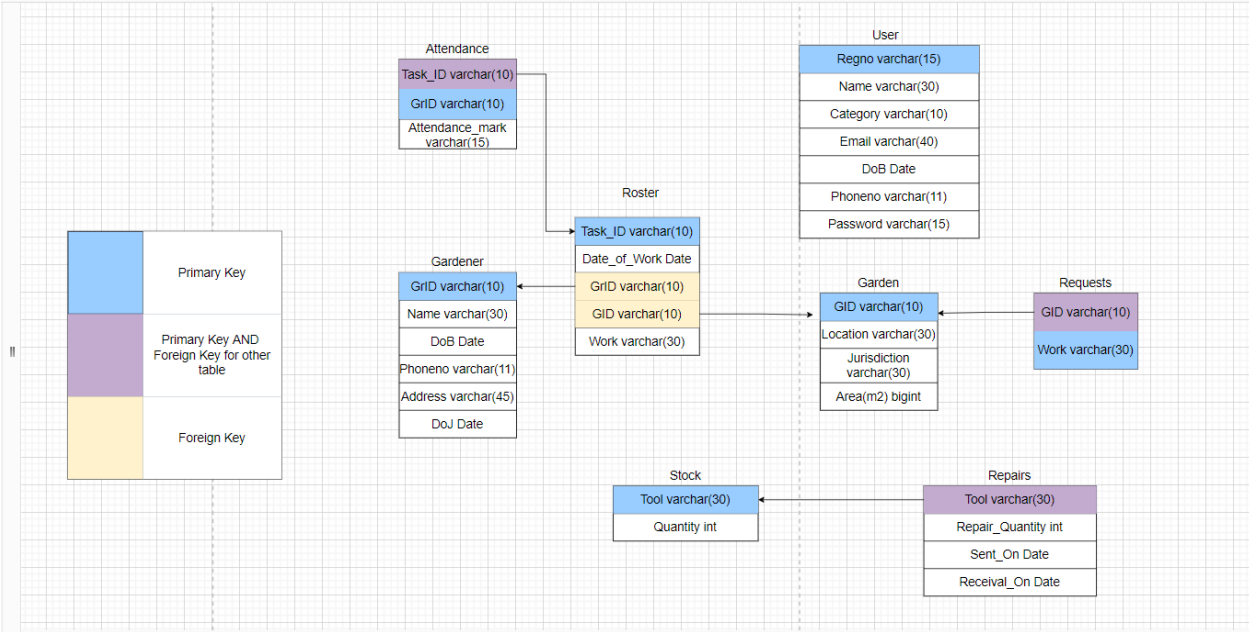
5) **<marked via>:** Corresponds to attendance marked using Duty Roster entries.

   Binary, One to one relationship between 'Attendance' and 'Duty Roster'.
   One roster entry can be associated with at most one attendance result, and vice versa.

6) **<sent for>:** Corresponds to tools/equipment sent for repairs amongst those in stock.

   Binary, One to one relationship between 'Stock' and 'Repairs'.
   One stock entry is associated with one repairs entry and vice versa.

RELATIONAL MODEL:



Link to complete Relational Model:
[Landscaping_Relational]

QUERIES AND INTERFACE:

The stakeholders involved in the Landscaping part of the project are:

1) Admin: Can view and modify stored information, can mark and check monthly attendance, can keep track of requests and decide between accepting and rejecting requests, while also determining how the requests gets completed and thereby adding it into the duty roster, can view monthly duty roster, can keep track of the equipments stock and the details of each item sent for repairs.

2) User: Can make requests regarding tasks they want to get done, for example Fertilizing in Garden with GID 2a and figure out if the tasks they want to request have already been requested, or have already been inserted into the duty roster, thereby would be getting completed in near future so no requesting required.

FORM FOR USER REGISTRATION:

CHECKING THAT DETAILS ENTERED BY THE USER SUFFICE:

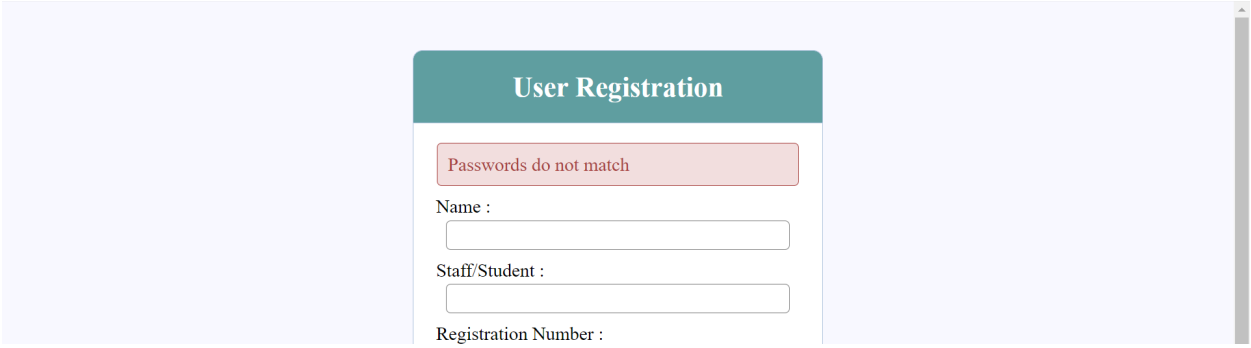
1. **Ensuring that all fields are filled:**

```
if (isset($_POST['reg_user'])) {

        // storing the user input in variables
        $name1 = mysqli_real_escape_string($db, $_POST['name1']);
        $category = mysqli_real_escape_string($db, $_POST['category']);
        $regno = mysqli_real_escape_string($db, $_POST['regno']);
        $email = mysqli_real_escape_string($db, $_POST['email']);
        $dob = mysqli_real_escape_string($db, $_POST['dob']);
        $phoneno = mysqli_real_escape_string($db, $_POST['phoneno']);
        $password1 = mysqli_real_escape_string($db, $_POST['password1']);
        $password2 = mysqli_real_escape_string($db, $_POST['password2']);

        // form validation
        if (empty($name1)) {
            array_push($errors, "Name is a required field");
        }
        if (empty($category)) {
            array_push($errors, "Category is a required field");
        }
        if (empty($regno)) {
            array_push($errors, "Registration Number is a required
field");
```

```
        }
        if (empty($email)) {
            array_push($errors, "Email ID is a required field");
        }
        if (empty($dob)) {
            array_push($errors, "Date of Birth is a required field");
        }
        if (empty($phoneno)) {
            array_push($errors, "Phone Number is a required field");
        }
        if (empty($password1) || empty($password2)) {
            array_push($errors, "Password is required");
        }
```

2. **Ensuring that the password and confirm- password match:**

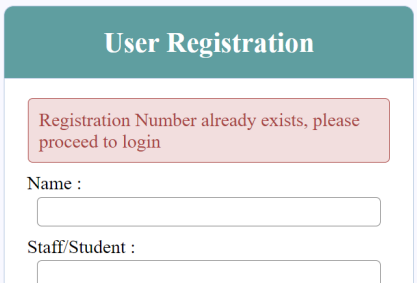
```
        if ($password1 != $password2) {
            array_push($errors, "Passwords do not match");
        }
```



3. **Ensuring that the Registration Number entered by the user is not already registered:**

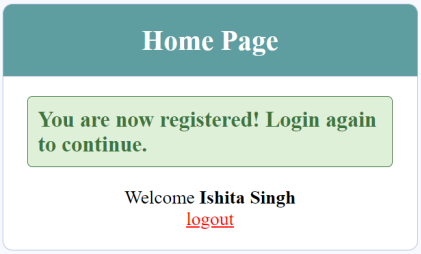
```
$user_check_query = "SELECT * FROM user WHERE regno = '$regno'";
        $results = mysqli_query($db, $user_check_query);

        //if( $results) echo "YES";
        //else echo "NO";

        $user1 = mysqli_fetch_assoc($results);

        if ($user1) {
```

```
            array_push($errors, "Registration Number already exists,
please proceed to login");
        }
```



4. **Query to insert into 'users' table:**

```
 $query = "INSERT INTO user (name, category, regno, email, dob, phoneno,
password) VALUES ( '$name1', '$category', '$regno', '$email', '$dob',
'$phoneno', '$password1') ";
            $check = mysqli_query($db, $query);
```



USER LOGIN:

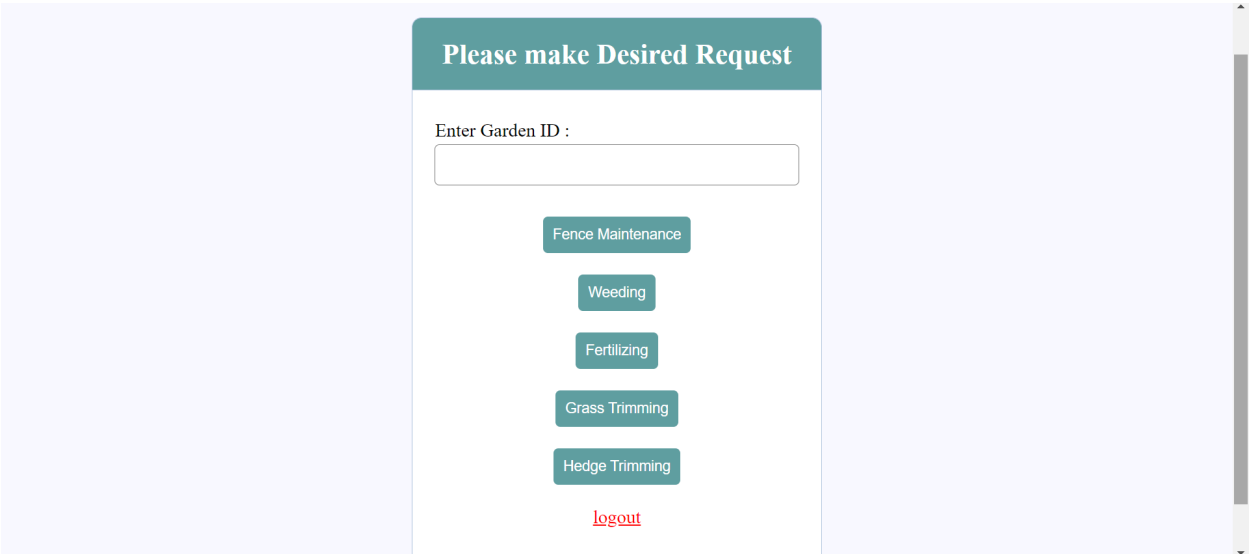
1. **Ensuring that all fields are filled:**

```
if (empty($regno)) {
            array_push($errors, "Registration Number is required");
        }
if (empty($password1)) {
      array_push($errors, "Password is required");
        }
```

2. **Ensuring that entered details are correct:**

```
$query = "SELECT * FROM user WHERE regno = '$regno' AND BINARY password =
'$password1' ";
$results = mysqli_query($db, $query);
```

USER REQUESTS:

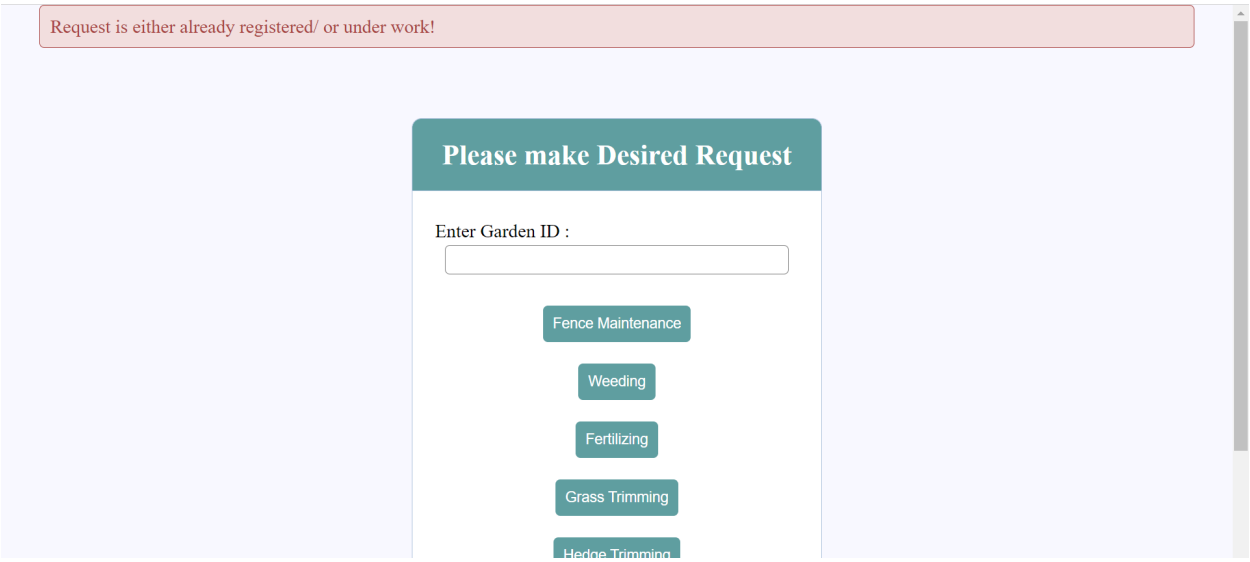


1. **Checking if request already exists:**

```
$row = mysqli_fetch_assoc($results);
            $query2 = "SELECT * FROM requests WHERE GID = '$GID' AND work
= '$request' ";
```

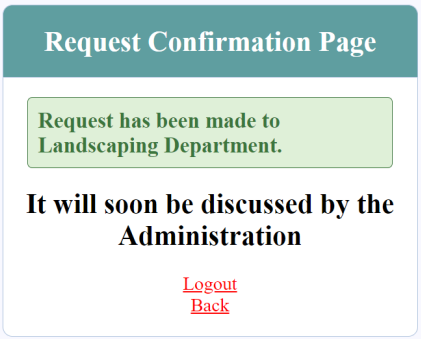
2. **Checking if the task requested already exists in the duty roster, and hence has already been assigned to someone for the future:**

```
 $results2 = mysqli_query($db, $query2);
            $query12 = "SELECT * FROM `roster` WHERE GID = '$GID' AND work
= '$request' and Date_of_work> Current_date() ";
```



3. **Inserting into table 'Requests' for Admin to see:**

```
$query3 = "INSERT INTO requests (GID, work) VALUES ( '$GID', '$request')
";
                $check = mysqli_query($db, $query3);

                $_SESSION['success1'] = "Request has been made to
Landscaping Department.";

                header('location: requestconfirmed.php');
```

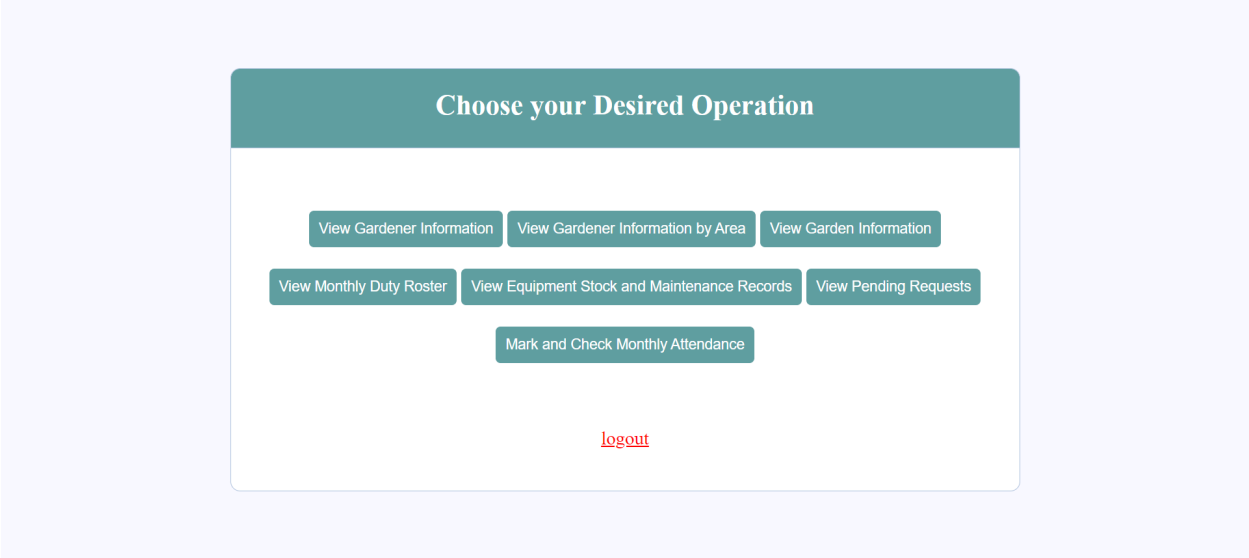


ADMIN LOGIN:

1. **Checking Password:**

```php
if(isset($_POST['admin_login'])) {
        $pass = mysqli_real_escape_string($db, $_POST['password']);
        if($pass == '1234') {
            $_SESSION['admin'] = "good";
            header('location: admin.php');
        }
        else {
            array_push($errors, "Wrong Password");
        }

    }
```

ADMIN FUNCTIONALITIES:



1. **To View Gardener Details**

```php
<?php $query = mysqli_query($db, "Select * from gardener"); ?>
```

2. **To View Garden Details**

```php
<?php $query = mysqli_query($db, "Select * from garden"); ?>
```

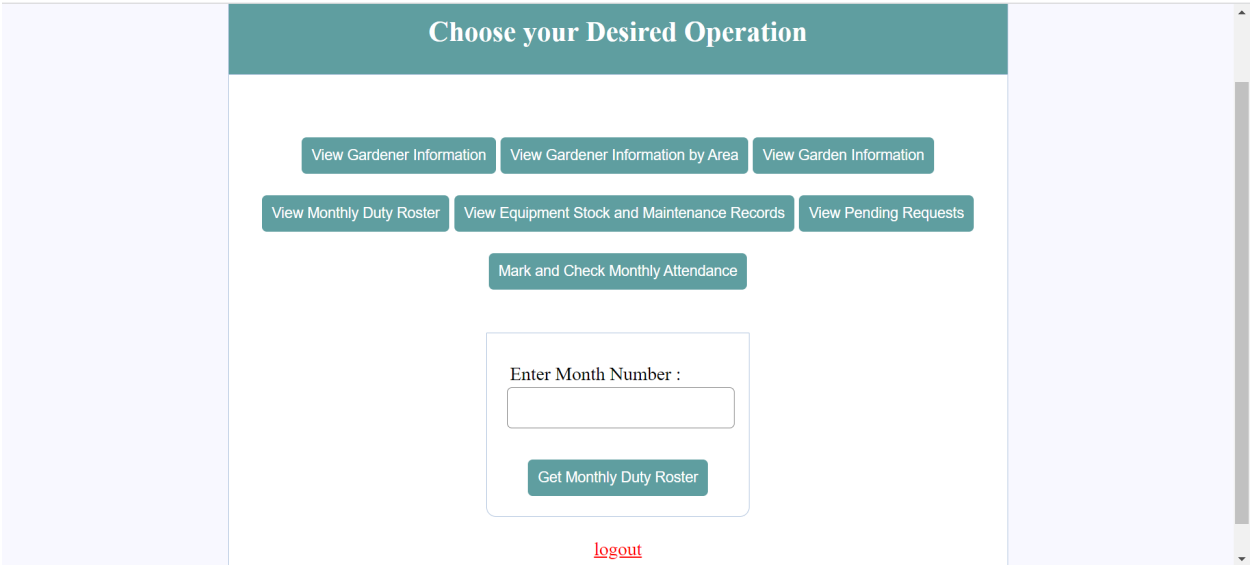
3. **To View Gardener Details using GID i.e., using Garden ID**
   -> Checking that entered GID is valid:

```php
 $checkquery=mysqli_query($db, "select * from garden where GID = '$ar'" );
                if($ar===false || empty($ar) ) {
                    array_push($errors, "Please enter valid Garden ID" );
}
```

->viewing:

```
$query=mysqli_query($db, "select Date_of_Work,GrID,work from roster where
GID = '$ar'" );
                                if($query==false ||
mysqli_num_rows($query) == 0)  {
                                array_push($errors, "No gardener assigned
in this area" );
```

4. **To view Monthly Duty Roster**



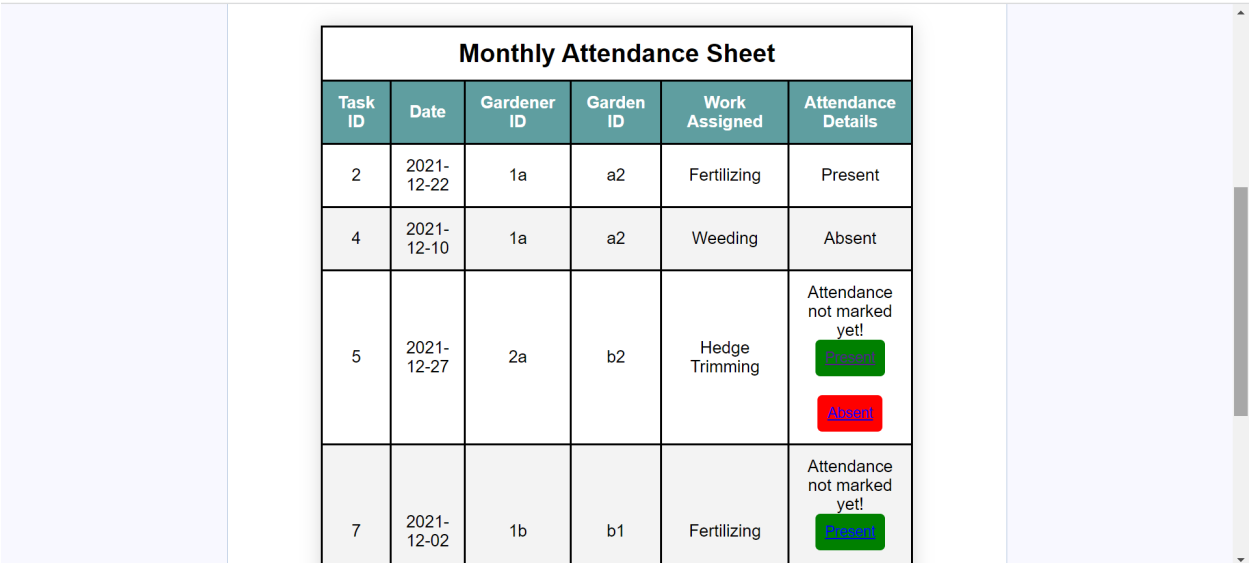
->Ensuring that a valid month has been entered:

```
if($monthNo===false || empty($monthNo) || (!($monthNo >=1 && $monthNo <=
12))) {
                    array_push($errors, "Please enter valid month number
between 1 and 12" ); }
```

->Viewing:

```
$query=mysqli_query($db, "select * from roster where MONTH(date_of_work) =
'$monthNo'" );
                    if(empty($errors) && mysqli_num_rows($query) == 0) {
                        array_push($errors, "No work assigned in this
month" );
```

5. To view and mark attendance



Monthly Attendance Sheet

Task ID	Date	Gardener ID	Garden ID	Work Assigned	Attendance Details
2	2021-12-22	1a	a2	Fertilizing	Present
4	2021-12-10	1a	a2	Weeding	Absent
5	2021-12-27	2a	b2	Hedge Trimming	Attendance not marked yet! Present Absent
7	2021-12-02	1b	b1	Fertilizing	Attendance not marked yet! Present

->Ensuring valid month is entered:

```
if(empty($monthNo2) || (!($monthNo2 >=1 && $monthNo2 <= 12))) {
                    array_push($errors, "Please enter valid month number
between 1 and 12" ); }
```

->Viewing:

```
$query=mysqli_query($db, "select roster.Task_ID, roster.Date_of_Work,
roster.GrID, roster.GID, roster.work, attendance_mark from `roster` left
join `attendance` on roster.Task_ID=attendance.Task_ID where
MONTH(roster.Date_of_work) = '$monthNo2';" );
                    if(mysqli_num_rows($query) == 0) {
                        array_push($errors, "No tasks in this month" );
                    }
```

->Giving options to mark attendance

```
                <?php while(($row=mysqli_fetch_array($query)))  { ?>
                <tr>
                    <td><?php echo $row[0] ?></td>
```

```
                    <td><?php echo $row[1] ?></td>
                    <td><?php echo $row[2] ?></td>
                    <td><?php echo $row[3] ?></td>
                    <td><?php echo $row[4] ?></td>
                    <?php if($row[5]===NULL){ ?>
                        <td>
                        <?php echo "Attendance not marked yet!";?>
                        <button type="submit" name="Present" class="btn"
button style='background-color: green';><a href="admin.php?id2=<?php echo
$row[0]; ?>">Present</a></button>
                        <br><br><button type="submit" name="Absent"
class="btn" button style='background-color: red';><a
href="admin.php?id3=<?php echo $row[0]; ?>">Absent</a></button></td>
                    <?php } else { ?>
                        <td> <?php echo $row[5];?> </td>
                        <?php } ?>
                </tr>
                <?php } ?>
```

->Marking Absent:

```
    $query_attend = mysqli_query($db, " insert into `attendance` values
('$row[0]','$row[1]','Absent');");
```

->Marking Present:

```
    $query_attend = mysqli_query($db, " insert into `attendance` values
('$row[0]','$row[1]','Present');");
```

6. **To view stock maintenance and repair details:**

```
<?php $query = mysqli_query($db, "select stock.tool, quantity,
repair_quantity, sent_on, receival_on from `stock` left join `repairs` on
stock.tool=repairs.tool;"); ?>
```

Stock and Maintenance Record Details

Equipment Picture	Tool	Total Quantity	Quantity for Repair	Sent On	Receival On
	Gloss White Fence Paint	10	0	N/A	N/A
	Hedge Trimmer	15	5	2021-11-20	2021-12-01
	Lawn Mower	5	1	2021-11-20	2021-12-15
	Leaf Rake	5	0	N/A	N/A

7. To view Pending Requests and Accept/Reject them

View Gardener Information | View Gardener Information by Area | View Garden Information

View Monthly Duty Roster | View Equipment Stock and Maintenance Records | View Pending Requests

Mark and Check Monthly Attendance

Pending Requests

Garden ID	Work Requested	Accept	Reject
b3	Hedge Trimming	Accept	Reject
b3	Weeding	Accept	Reject

logout

->Viewing:

```
<?php $query = mysqli_query($db, "Select * from requests"); ?>
```

->Giving admin option to accept/ reject:

```
<?php while(($row=mysqli_fetch_array($query)))  { ?>
                <tr>
                    <td><?php echo $row[0]?></td>
                    <td><?php echo $row[1] ?></td>
                    <td><button type="submit" name="Accept" class="btn"
button style='background-color: green';><a href="admin.php?id5=<?php echo
$row[0]?>&id55=<?php echo $row[1]?>">Accept</a></button></td>
```

```
                    <td><button type="submit" name="Reject" class="btn"
button style='background-color: red';><a href="admin.php?id6=<?php echo
$row[0]?>&id66=<?php echo $row[1]?>">Reject</a></button></td>
                </tr>
                <?php } ?>
```

->Rejecting:

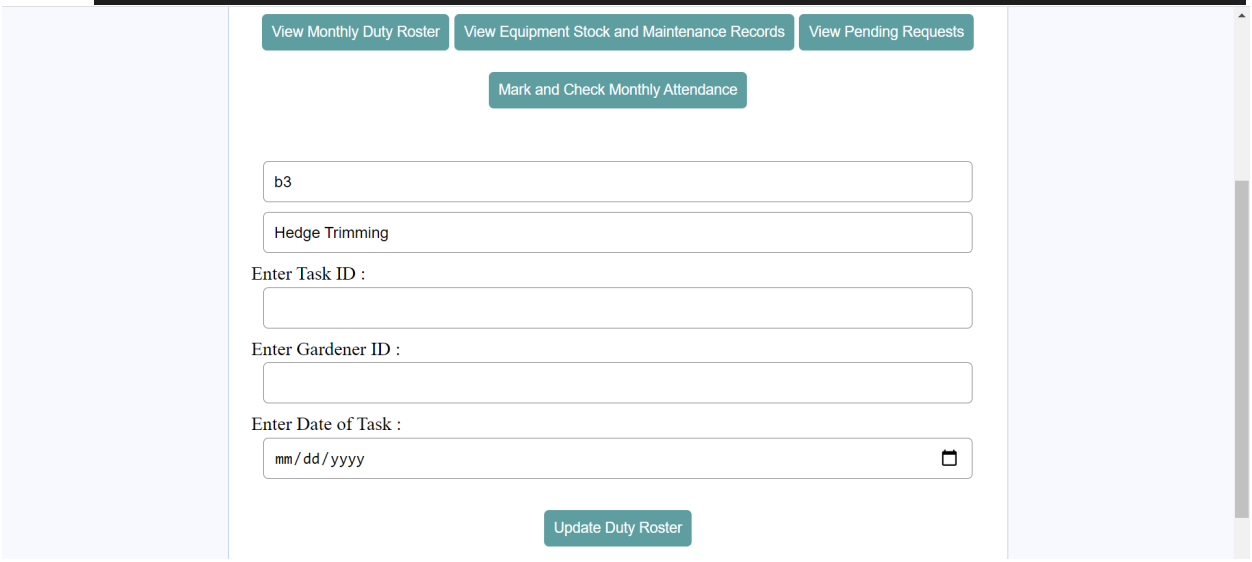
```
$id6 = $_GET['id6'];
    unset($_GET['id6']);
    $id66 = $_GET['id66'];
    unset($_GET['id66']);

    $querry1 = mysqli_query($db, "delete from `requests` where
`GID`='$id6' and `work`='$id66' ;" );
```

->Accepting:

->Taking input for how to update request in Duty Roster

```
<div class="container">
                <div class="input-group">
                    <input type="text" name="key1" value="<?php echo
$id5;?>" />
                </div>
                <div class="input-group">
                    <input type="text" name="key2" value="<?php echo
$id55;?>" />
                </div>
                <div class="input-group">
                    <label>Enter Task ID : </label>
                    <input type="text" name="tsk">
                </div>
                <div class="input-group">
                    <label>Enter Gardener ID : </label>
                    <input type="text" name="grd">
                </div>
                <div class="input-group">
                    <label>Enter Date of Task : </label>
                    <input type="date" name="dtsk">
                </div>
```

```
            </div>
            <br>
            <button type="submit" name="prf" class="btn">Update Duty
Roster</button>
```



->Checking none of the fields are wrong or empty:

```
if($tsk===false || $dtsk===false || $grd===false || empty($tsk) ||
empty($dtsk) || empty($grd)) {
                    array_push($errors, "Please enter valid details" );
                    }
```

->Checking that entered GrID is valid, Task_ID is unique and the inserting into ‘roster’

```
else {
                        $check1= mysqli_query($db, "select `GrID` from
`gardener` where `GrID`= '$grd';" );
                        $check2= mysqli_query($db, "select `Task_ID` from
`roster` where `Task_ID`= '$tsk';" );
                        if($check1 === false || mysqli_num_rows($check1)
==0 ){
                            array_push($errors, "Non-existant Garden ID"
);
                        }
                        else{
                            if($check2 === false ||
mysqli_num_rows($check2) ==0 ){
```

```
                                $query=mysqli_query($db, "INSERT INTO
`roster`(`Task_ID`, `Date_of_work`, `GrID`, `GID`, `work`) VALUES
('$tsk','$dtsk','$grd','$key1','$key2');" );
                                $quer=mysqli_query($db, "delete from
`requests` where `GID`='$key1' and `work`='$key2' ;" );
                            }
                            else{
                                array_push($errors, "Task ID already
exists" );
                            }
                        }
                    }
```

FOR ERRORS:

```
<?php  if (count($errors) > 0) : ?>

<div class="error">

    <?php foreach ($errors as $error) : ?>
    <p><?php echo $error ?></p>
    <?php endforeach ?>

</div>

<?php  endif ?>
```

USING HYPERLINKS IN HOME:

```
 <p><a href="adminlogin.php" style="color: red;">Admin Login</a> </p>
            <p><a href="userregistration.php" style="color: red;">User
Registration</a> </p>
            <p><a href="userlogin.php" style="color: red;">User Login</a> </p>
```

TO ENSURE LOGIN, AND LOGOUT FUNCTIONALITIES

```
<?php
  session_start();

  if (!isset($_SESSION['name1'])) {
    $_SESSION['msg'] = "You must log in first";
    header('location: userlogin.php');
```

```
  }
  if (isset($_GET['logout'])) {
    session_destroy();
    unset($_SESSION['name1']);
    header("location: userlogin.php");
  }
?>
```

CSS:

```
* {
    margin: 0px;
    padding: 0px;
  }
  body {
    font-size: 120%;
    text-align: center;
    background: #f8f8ff;
  }

  .header {
    width: 30%;
    margin: 50px auto 0px;
    color: white;
    background: #5f9ea0;
    text-align: center;
    border: 1px solid #b0c4de;
    border-bottom: none;
    border-radius: 10px 10px 0px 0px;
    padding: 20px;
  }
  .form,
  .content {
    width: 30%;
    margin: 0px auto;
    padding: 20px;
    border: 1px solid #b0c4de;
    background: white;
    border-radius: 0px 0px 10px 10px;
  }
```

```css
.input-group {
  margin: 10px 0px 10px 0px;
}
.input-group label {
  display: block;
  text-align: left;
  margin: 3px;
}
.input-group input {
  height: 30px;
  width: 93%;
  padding: 5px 10px;
  font-size: 16px;
  border-radius: 5px;
  border: 1px solid gray;
}
.btn {
  padding: 10px;
  font-size: 15px;
  color: white;
  background: #5f9ea0;
  border: none;
  border-radius: 5px;
}
.error {
  width: 92%;
  margin: 0px auto;
  padding: 10px;
  border: 1px solid #a94442;
  color: #a94442;
  background: #f2dede;
  border-radius: 5px;
  text-align: left;
}
.success {
  color: #3c763d;
  background: #dff0d8;
  border: 1px solid #3c763d;
  margin-bottom: 20px;
```

```
}

table,
th,
td {
  width: 80%;
  border: 2px solid black;
}
```

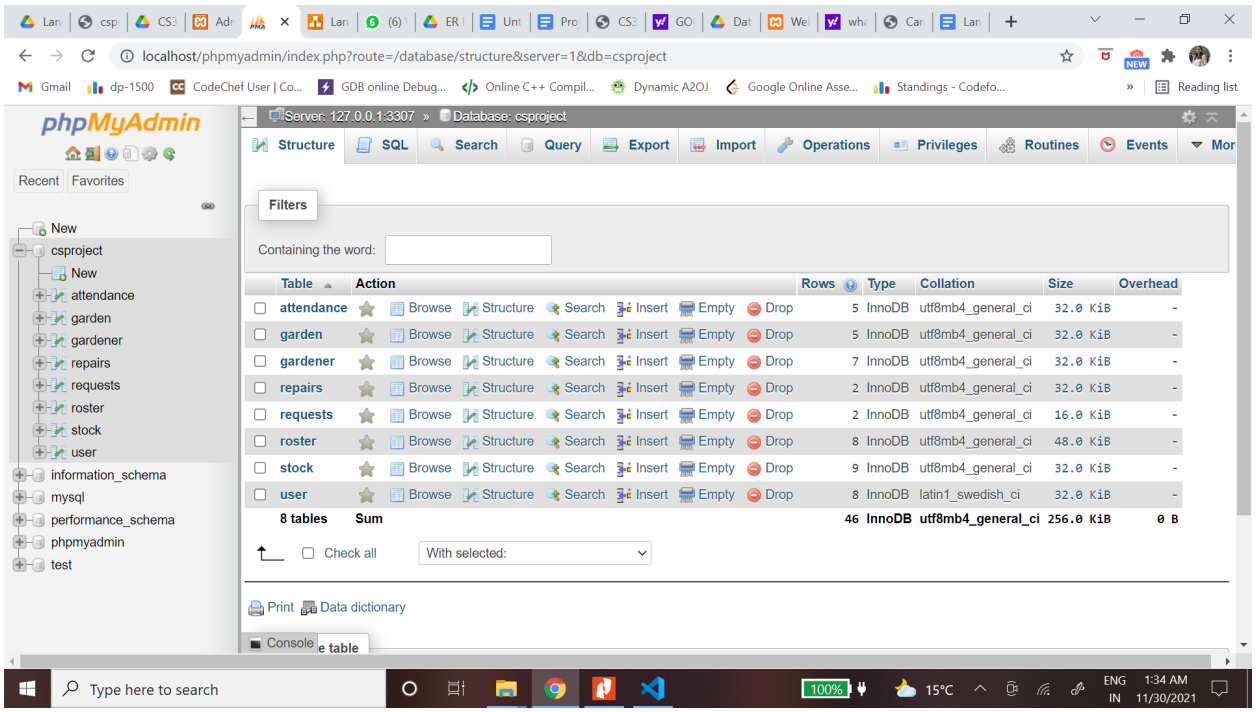
->FOR TABLE:

```
.styled-table {
    border-collapse: collapse;
    margin: 25px 0;
    font-size: 0.9em;
    font-family: sans-serif;
    table-layout: relative;
    border: 1px solid black;
    margin-left: auto;
    margin-right: auto;
    min-width: 100px;
    max-width: 500px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
}

.styled-table thead tr {
    background-color: #5f9ea0;
    color: #ffffff;
    text-align: center;
}

.styled-table th,
.styled-table td {
    padding: 12px 15px;
}

.styled-table tbody tr {
    border-bottom: 1px solid #dddddd;
}
```

```css
.styled-table tbody tr:nth-of-type(even) {
  background-color: #f3f3f3;
}

.styled-table tbody tr:last-of-type {
  border-bottom: 2px solid #009879;
}

.styled-table tbody tr.active-row {
  font-weight: bold;
  color: #009879;
}
```

These above codes are the same codes as I have uploaded in the 'Landscaping' folder in the google drive link given and a better idea of the implementation and the various features can be seen from the drive

DATABASE:



1901CS30

Kavya Goyal

Market Shop

ER DIAGRAM

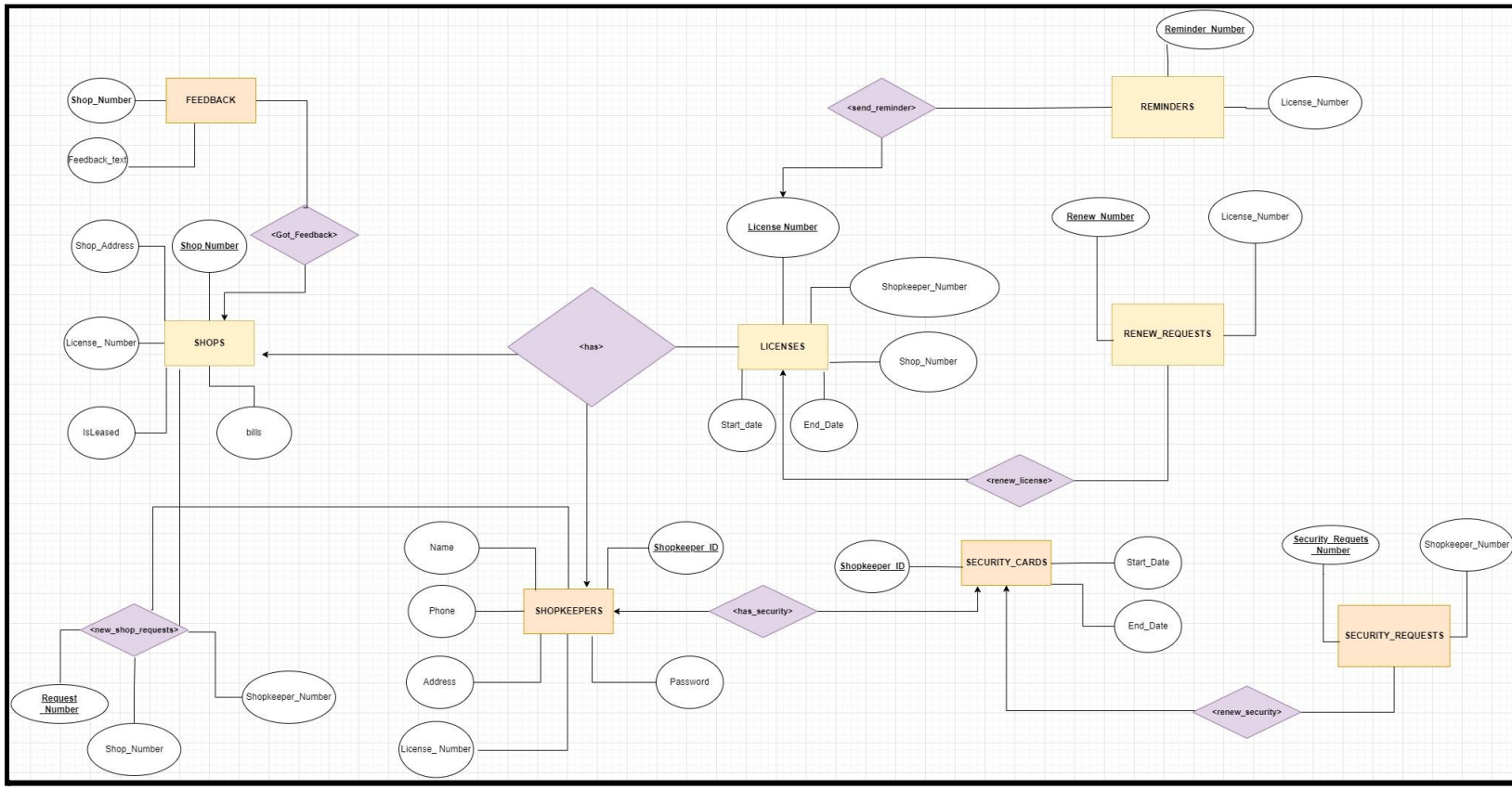
DATABASE RELATIONAL MODEL

FOLDER (Video, Code, Readme File, ScreenShots, data files, dump files etc)

- ➔ All working of the project is included in video and detailed in the screenshots
- ➔ All data related files are stored in folder "DATABASE DATA"
  - ◆ Dump file : sql format
  - ◆ Table file : pdf format
  - ◆ All data : csv format
- ➔ Documentation is here
- ➔ Code : "CODE FILES"
- ➔ Screenshots : "Screenshots of Project"
- ➔ ER Diagram link and screenshot attached

ANALYSIS OF ER-MODELS AND RELATIONS

ER DIAGRAM



Number of Entities = 9
Number of Relations = 7

The complete ER Model with all the relations is given below
https://drive.google.com/file/d/1-YrJY3f-sCoHKNTNSUqlUgDZkG3mihzb/view?usp=sharing

ENTITIES

a) SHOPS

The details of shops in the market management portal

Attributes

1. Shop_Number : (int) Primary Key for identifying shops (AUTO INCREMENT)
2. Shop_Address : (varchar(255)) Address of the shop
3. isLeased : (int) Tells whether the shop is under any license DEFAULT 0
4. License_Number : (int) Foreign Key to @LICENSE DEFAULT NULL
5. Bills : (int) Total payable on the shop currently DEFAULT 0

Indexes used in Table

1. License_Number

```
ALTER TABLE `shops` ADD INDEX idx_shop_lic (`License_Number`);
```

Foreign Key used in Table

1. License_Number

```
ALTER TABLE shops ADD CONSTRAINT fk_lic_num FOREIGN KEY(License_Number)
REFERENCES licenses(License_Number);
```

b) FEEDBACK

Feedbacks on the shops. Given by the admin panel.

Attributes

1. Shop_Number : (int) Foreign Key to @SHOPS
2. Feedback_Text : varchar(255)

Indexes used in Table

1. Shop_Number

```
ALTER TABLE `feedback` ADD INDEX idx_feedback_shop (`Shop_Number`);
```

Foreign Key used in Table

1. Shop_Number

```
ALTER TABLE feedback ADD CONSTRAINT fk_shop_num_feedback FOREIGN
KEY(Shop_Number) REFERENCES shops(Shop_Number);
```

c) SHOPKEEPERS

The details of shopkeepers registered within the market management portal

Attributes

1. Shopkeeper_Number : (int) Primary Key AUTO INCREMENT
2. Name : varchar(20)
3. Phone : varchar(10)
4. Address : varchar(255)
5. License_Number : (int) Foreign Key to @LICENSE
6. Password : varchar(10)

Indexes used in Table

1. License_Number

```
ALTER TABLE `shopkeepers` ADD INDEX idx_shopkeeper_lic (`License_Number`);
```

Foreign Key used in Table

1. License_Number

```
ALTER TABLE shopkeepers ADD CONSTRAINT fk_lic_num_shopkeepers FOREIGN
KEY(License_Number) REFERENCES licenses(License_Number);
```

d) NEW_SHOP_REQUESTS

The entity storing the new requests to admin from the shopkeepers to get license for shops

Attributes

1. Request_Number : (int) Primary Key AUTO INCREMENT
2. Shop_Number : (int) Foreign Key to @SHOPS
3. Shopkeeper_Number : (int) Foreign Key to @SHOPKEEPERS

Foreign Key used in Table

1. Shopkeeper_Number

```
ALTER TABLE NEW_SHOP_REQUESTS ADD CONSTRAINT fk_new_shop_keeper FOREIGN
KEY(Shopkeeper_Number) REFERENCES shopkeepers(Shopkeeper_Number);
```

2. Shop_Number

```
ALTER TABLE NEW_SHOP_REQUESTS ADD CONSTRAINT fk_new_shop_shop FOREIGN
KEY(Shop_Number) REFERENCES shops(Shop_Number);
```

e) SECURITY_CARDS

Attributes

1. Shopkeeper_Number : (int) Foreign Key to @SHOPKEEPERS
2. Start_Date : date
3. End_Date : date

Foreign Key used in Table

1. Shopkeeper_Number

```
ALTER TABLE SECURITY_CARDS ADD CONSTRAINT fk_sec_cards FOREIGN
KEY(Shopkeeper_Number) REFERENCES shopkeepers(Shopkeeper_Number);
```

f) SECURITY_REQUESTS

The entity storing the security card renewal requests to admin from the shopkeepers

Attributes

1. Security_Request_Number : (int) Primary Key AUTO INCREMENT
2. Shopkeeper_Number : (int) ForeignKey to @SHOPKEEPERS

Foreign Key used in Table

1. Shopkeeper_Number

```
ALTER TABLE SECURITY_REQUESTS ADD CONSTRAINT fk_security_keeper FOREIGN
KEY(Shopkeeper_Number) REFERENCES shopkeepers(Shopkeeper_Number);
```

g) LICENSES

Every shopkeeper can have a shop under the relation license.

Attributes

1. License_Number : (int) Primary Key AUTO INCREMENT
2. Shopkeeper_Number : (int) Foreign Key TO @SHOPKEEPERS
3. Shop_Number : (int) Foreign Key TO @SHOPS
4. Start_Date : date
5. End_Date : date

Foreign Key used in Table

1. Shopkeeper_Number

```
ALTER TABLE LICENSES ADD CONSTRAINT fk_keeper_num_lic FOREIGN
KEY(Shopkeeper_Number) REFERENCES shopkeepers(Shopkeeper_Number);
```

2. Shop_Number

```
ALTER TABLE LICENSES ADD CONSTRAINT fk_shop_num_lic FOREIGN
KEY(Shop_Number) REFERENCES shops(Shop_Number);
```

h) RENEW_REQUESTS

The entity storing the license renewal requests to admin from the shopkeepers

Attributes

1. Renew_Number : (int) Primary Key AUTO INCREMENT
2. License_Number : (int) ForeignKey to @LICENSES

Foreign Key used in Table

1. License_Number

```
ALTER TABLE RENEW_REQUESTS ADD CONSTRAINT fk_renew_lic FOREIGN
KEY(License_Number) REFERENCES licenses(License_Number);
```

i) REMINDERS

The entity storing the active reminder requests issued from admin to shopkeepers

Attributes

1. Reminder_Number : (int) Primary Key AUTO INCREMENT
2. License_Number : (int) Foreign Key to @LICENSES

Foreign Key used in Table

1. License_Number

```
ALTER TABLE REMINDERS ADD CONSTRAINT fk_remainder_lic FOREIGN
KEY(License_Number) REFERENCES licenses(License_Number);
```

RELATIONS

1. <Got_Feedback>
One to many relationship between SHOPS and FEEDBACK, that is a shop may have multiple feedbacks, however a unique feedback will be associated with a single shop.
Cardinality : No lower or upper limit on the number of feedbacks per shop

2. <Has_Security>
One to one relationship between SHOPKEEPERS and SECURITY_CARDS, that is one shopkeeper may have one security card, and one security card will have a unique shopkeeper
Cardinality : Assuming that a security card is necessary for the shopkeepers, (1,1)

3. <has>
Ternary Relationship between SHOPS, SHOPKEEPERS , LICENSES

- ➔ One to Many <Has> relationship between SHOPS and LICENSE indicating that for for a shop, many licenses can exist owing to different shopkeepers, but a single license is for a single unique shop
- ➔ One to Many <Has> relationship between SHOPKEEPER and LICENSE indicating that a shopkeeper may have many licenses, but a single license is for a single unique shopkeeper.
- ➔ One to One <Has> relationship between SHOPS and SHOPKEEPER indicating that a shopkeeper may belong to only one shop and a shop can have only one shopkeeper

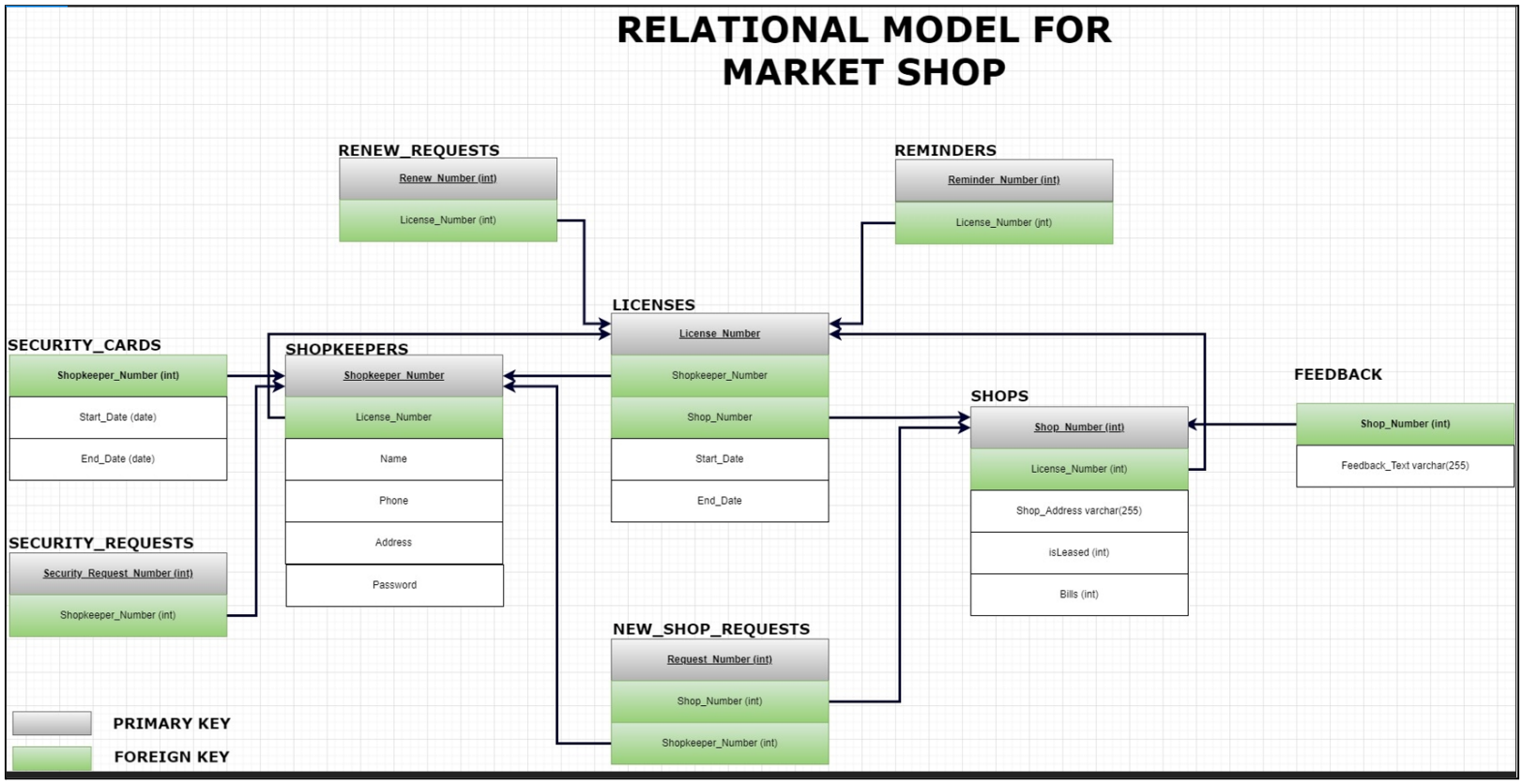
Cardinality : A shop and a shopkeeper must have at least 1 license. Also, a license can be owned by only 1 shop and shopkeeper

4. <Renew_License>
One to many relationship between LICENSES and RENEW_REQUESTS , that is a license may get renewed any number of times, however a unique renewal request will be associated with a single license.
Cardinality : No lower or upper limit on the number of renewal requests

5. <Renew_Security>
One to many relationship between SECURITY_CARDS and SECURITY_REQUETS , that is a security_card may get renewed any number of times, however a unique security renewal request will be associated with a single security_card.
Cardinality : No lower or upper limit on the number of renewal requests

6. <New_Shop_Requests>
Many to many relationship between SHOPS and SHOPKEEPERS, that is a shop can be requested by many shopkeepers, and similarly, a shopkeeper can request as many shops he wants
Cardinality : No lower or upper limit on the number of requests

7. <send_reminder>
One to many relationship between LICENSES and REMINDERS, that is a license can get multiple reminders, but a reminder will be associated with only one license
Cardinality : No lower or upper limit on the number of requests



QUERIES AND INTERFACE

The stakeholders involved for the market management portal are

1. Admin
    - Responsible for adding shops in the database
    - Acts as the authority to accept or reject the requests from the shopkeepers to provide licenses automatically
    - Manages the license renewals and security card validity requests of the shops and shopkeepers
    - Keeps track of the feedbacks/performance and pending bills of the shops
    - Send Reminders to shopkeepers if licenses have expired or close to expiry

2. Shopkeepers
    - Basic Registration and Login
    - Can send requests to update/extend security card
    - Can request for adding shops. A shopkeeper can only manage one shop at a time
    - Request for renewing licenses and pay bills of the shop
    - Check the feedbacks given to the shop
    - View the reminders sent by admin

1. ADMIN RELATED QUERIES

a. ACCEPT NEW SHOP

The shopkeepers can request to get licenses for new shops. The admin has the authority to accept or reject these requests.

In order to accept a new shop, shopkeeper_number, shop_num are passed as inputs to the procedure acceptNewShop. A new license is generated because of this

```
create procedure acceptNewShop (in _shopkeeper int, _shop int, _start date,
_end date, _req int)
begin
   DELETE FROM NEW_SHOP_REQUESTS WHERE Request_Number = _req;
   INSERT INTO LICENSES(Shopkeeper_Number, Shop_Number, Start_Date,
End_Date) VALUES(_shopkeeper,_shop, _start, _end);

end//
```

The newly created license's number is added to shops, and shopkeeper list also using trigger after insert on licenses

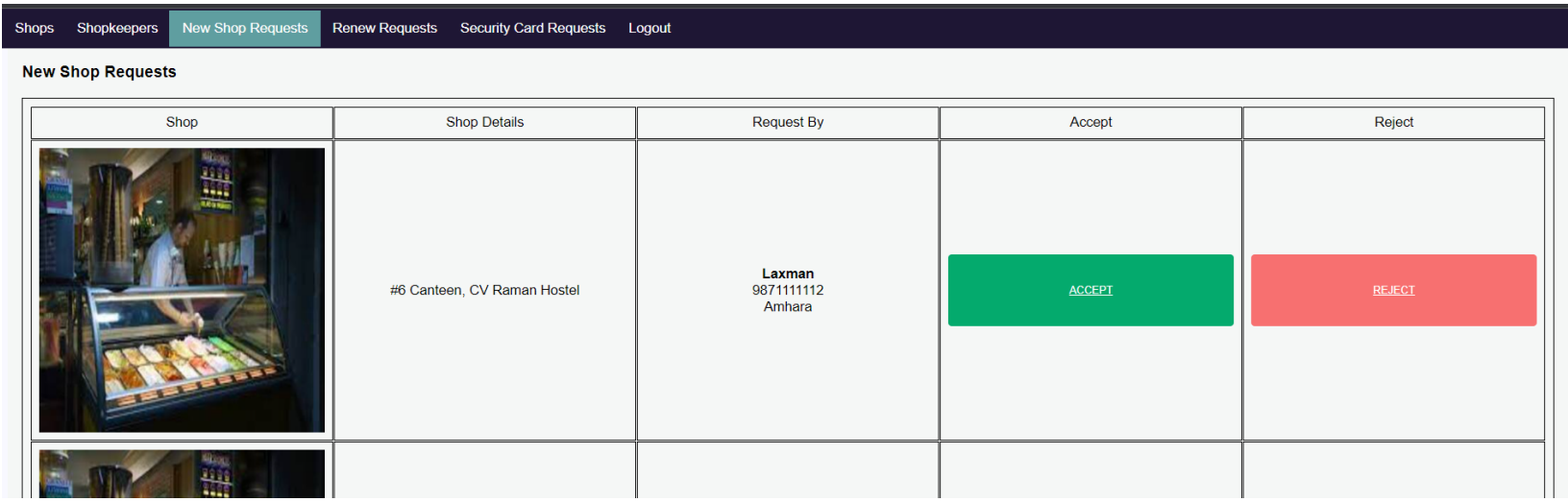
```
create trigger insert_license_trigger
 after insert
 on licenses
 for each row
 begin
   UPDATE SHOPS SET License_Number = NEW.License_Number WHERE SHOP_NUMBER =
NEW.Shop_Number;
  UPDATE SHOPKEEPERS SET License_Number = NEW.License_Number WHERE
SHOPKEEPER_NUMBER = NEW.Shopkeeper_Number;
 end//
```

```
$query_acceptNewShop = mysqli_query($db,
"call acceptNewShop ('$shopkeeper_num', '$shop_num', '$renew_start',
'$renew_end', '$req_num')") or die ($db->error);
```

b. REJECT NEW SHOP

```
$qeury_delete = mysqli_query($db,
"DELETE FROM NEW_SHOP_REQUESTS WHERE Request_Number = '$req_num'")
or die ($db->error);
```

The admin can reject the request the adding new shops under three cases
1. When the shop is already assigned to someone else
2. When the shopkeeper is already assigned a shop
3. Will of admin



c. ADD NEW SHOPS

```
$query_shop = mysqli_query($db,
"INSERT INTO SHOPS(Shop_Address) VALUES ('$shop_addr')")
or die ($db->error);
```

The admin can add new shops in the system by adding their addresses. All the other parameters of the SHOPS entity will be taken by default

d. DETAILS OF ALL THE SHOPS WITH FEEDBACKS, PENDING BILLS AND LICENSE AGREEMENTS

```
$query_shop =
mysqli_query($db, "SELECT * FROM SHOPS") or die ($db->error);
```

For all the shops, certain queries are performed.

```
$query_feedback = mysqli_query($db,
"SELECT * FROM FEEDBACK WHERE Shop_Number = '$shop_num' ")
 or die ($db->error);
```

If the License_Number is NOT NULL, we will have a license and a shopkeeper associated with the shop. Their details will be extracted from the following queries. All this data is viewed in table format using html, php

```
$query_license = mysqli_query($db,
"SELECT * FROM Licenses WHERE License_Number = '$shop_lic'") or die
($db->error);

$query_keeper = mysqli_query($db,
"SELECT * FROM SHOPKEEPERS WHERE License_Number = '$shop_lic'") or die
($db->error);
```

e. ADD NEW BILLS ON SHOPS

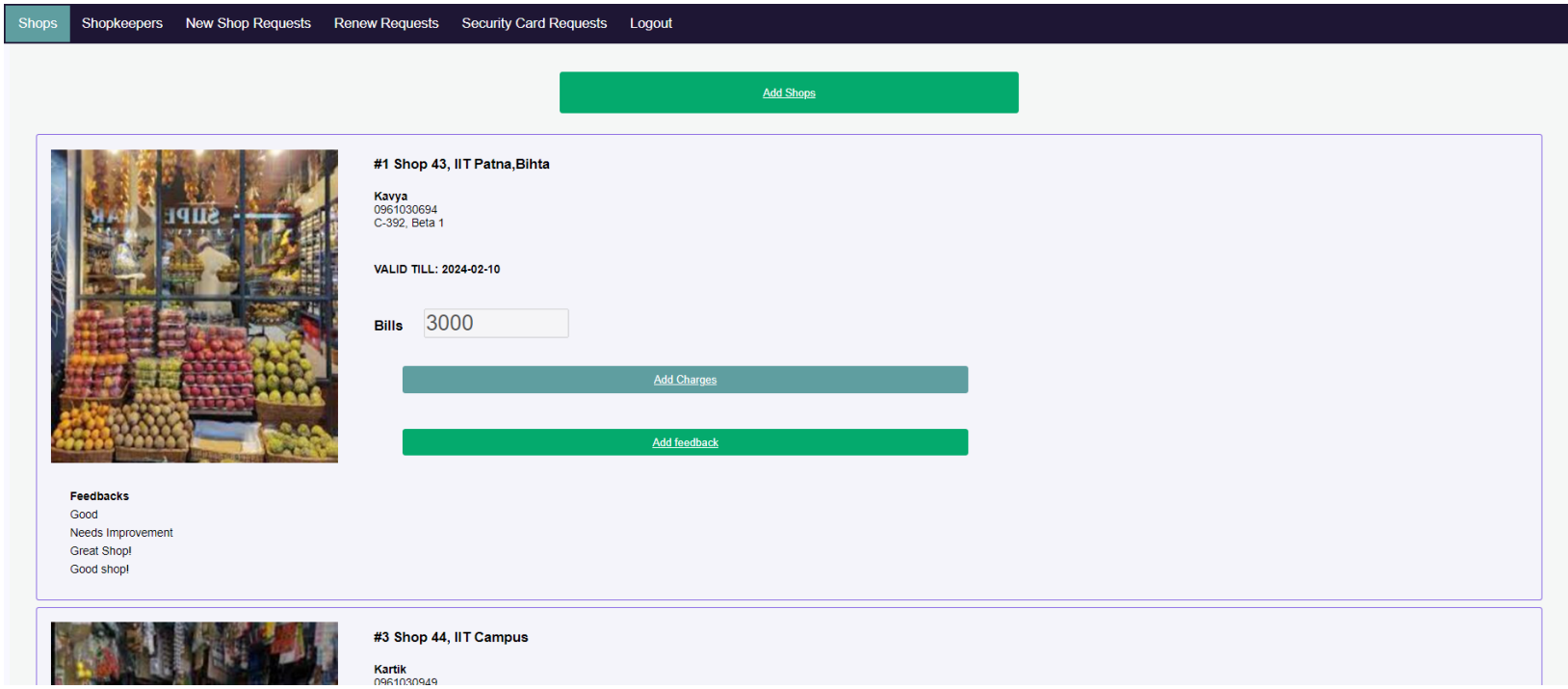
```
$query_pay = mysqli_query($db,
"UPDATE SHOPS SET BILLS = BILLS + 3000 WHERE Shop_Number = '$shop_num' ")
or die ($db->error);
```

The admin will be responsible for adding charges/bills on shops. At a given time, he can add 3000 Rs charge, which can then be paid by shopkeepers later

f. ADD FEEDBACKS

```
$query_feedback = mysqli_query($db,
"INSERT INTO FEEDBACK(Shop_Number, Feedback_Text) VALUES
('$shop_num','$feedback_value')") or die ($db->error);
```

The feedback will be given by the admin to shops.



g. DETAILS OF ALL THE SHOPKEEPERS WITH SECURITY CARD STATUS AND LICENSES

```
$query_keeper = mysqli_query($db,
"SELECT * FROM SHOPKEEPERS") or die ($db->error);

$query_security = mysqli_query($db,
"SELECT * FROM SECURITY_CARDS WHERE SHOPKEEPER_NUMBER = '$keeper_num'")
or die ($db->error);
```

If License_Number of the Shopkeeper is NOT NULL, then he will have a license and shop associated with him

```
$query_current = mysqli_query($db,
"SELECT * FROM LICENSES WHERE Shopkeeper_Number = '$keeper_num'")
or die ($db->error);


$query_shop = mysqli_query($db,
"SELECT * FROM SHOPS WHERE License_Number = '$current[0]'")
or die ($db->error);
```

h. SEND REMINDERS TO SHOPKEEPERS

```
$query_rem = mysqli_query($db,
"INSERT INTO REMINDERS (License_Number) VALUES ('$lic_number')")
or die ($db->error);
```

The reminders can be sent by the admin to shopkeepers to renew their licenses. Reminders can not be sent if a license-reminder is already present and also when there are no shops with a shopkeeper

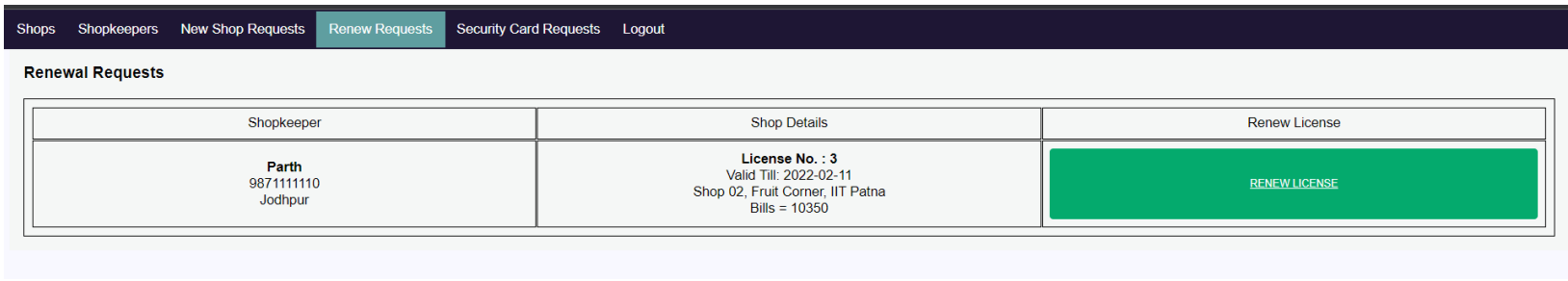
i. RENEW LICENSES

The renewal requests are requested by shopkeepers. Uniquely identified by License_Number

```
create procedure renewLicense (in _license_number int )
begin
   DELETE FROM RENEW_REQUESTS WHERE License_Number = _license_number;
   UPDATE LICENSES SET End_Date =  DATE_ADD(End_Date, INTERVAL 3 MONTH) WHERE
License_Number = _license_number;
end//
```

```
$query_renew_lic =  mysqli_query($db,
"call renewLicense('$license_number');") or die ($db->error);
```

This will renew the current license validity for 3 months more



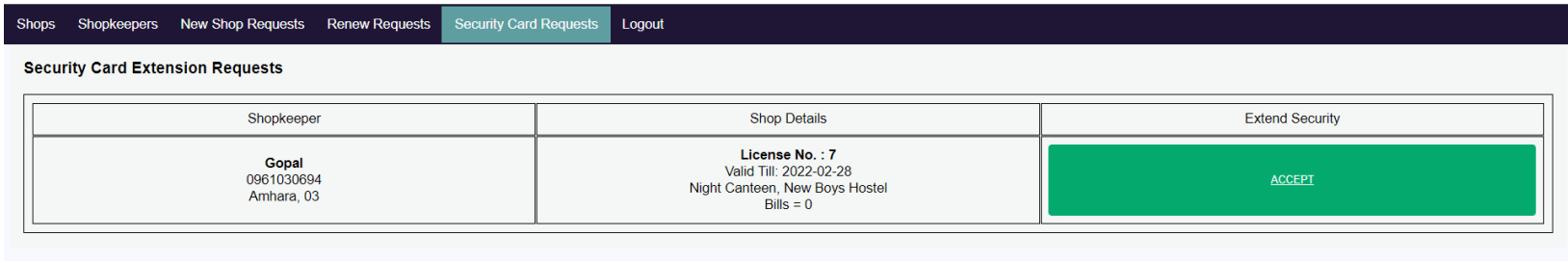
j. RENEW SECURITY VALIDITY

Procedure to renew security cards. Requested by shopkeepers, accepted by admin

```
create procedure renewSecurity (in _shopkeeper_number int )
begin
   DELETE FROM SECURITY_REQUESTS WHERE Shopkeeper_Number = _shopkeeper_number;
   UPDATE SECURITY_CARDS SET End_Date =  DATE_ADD(End_Date, INTERVAL 3 MONTH)
WHERE Shopkeeper_Number = _shopkeeper_number;
end//
```

```
$query_security_extend  = mysqli_query($db,
"call renewSecurity('$keeper_number');") or die ($db->error);
```

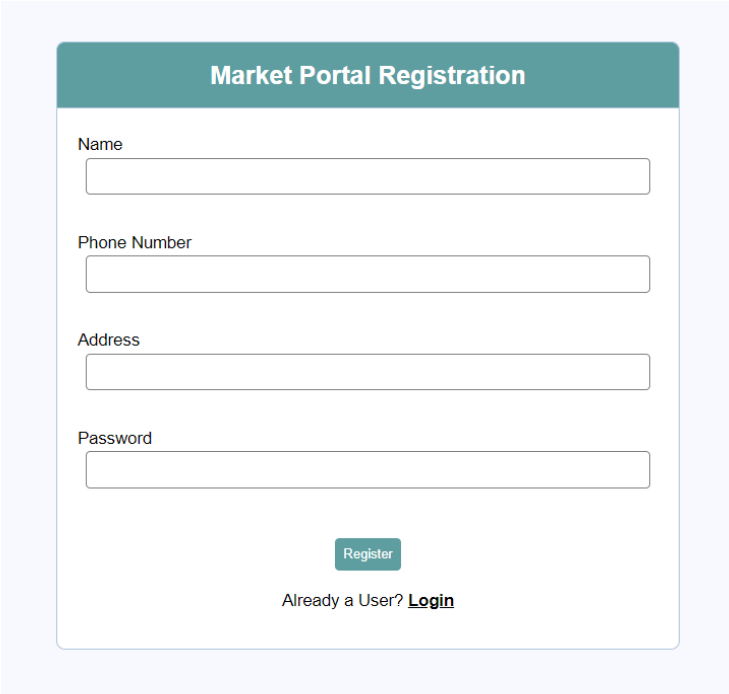
This will update the security card validity for 3 months more



2. SHOPKEEPER RELATED QUERIES AND INTERFACE

a. ADD NEW SHOPKEEPER

```
$add_new_shopkeeper = mysqli_query($db,
"INSERT INTO SHOPKEEPERS (Shopkeeper_Number, Name, Phone, Address,Password)
VALUES
('$_shopkeeper_id','$_name','$_phone','$_address','$_pwd')")
or die ($db->error);
```



Whenever a new shopkeeper is added, a Shopkeeper_Number is generated. Using Shopkeeper_Number and Password, the shopkeeper can log in. On registration, a new security card is also issued with Primary Key as Shopkeeper_Number with 3 months validity

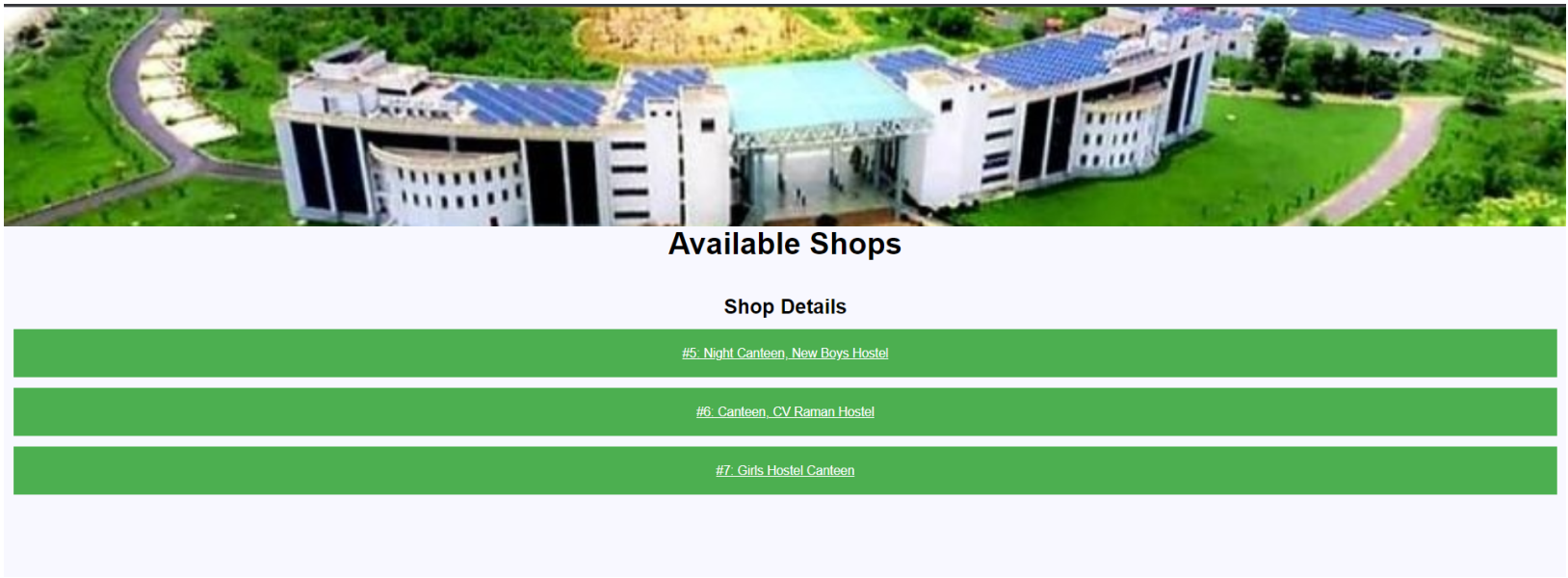
b. REQUEST FOR UPDATE SECURITY CARD

```
 $query= mysqli_query($db, "INSERT INTO SECURITY_REQUESTS(Shopkeeper_Number)
VALUES ('$_shopkeeper_id') ") or die($db->error);
```

The shopkeeper can send request to update the security card extension

c. REQUEST ADD SHOP

```
$query_request_shop =
mysqli_query($db, "INSERT INTO NEW_SHOP_REQUESTS (Shop_Number,
Shopkeeper_Number) VALUES ('$shop_to_buy','$_shopkeeper_id') ")
or die($db->error);
```



Shop can only be requested is the shopkeeper does not have any prior shops

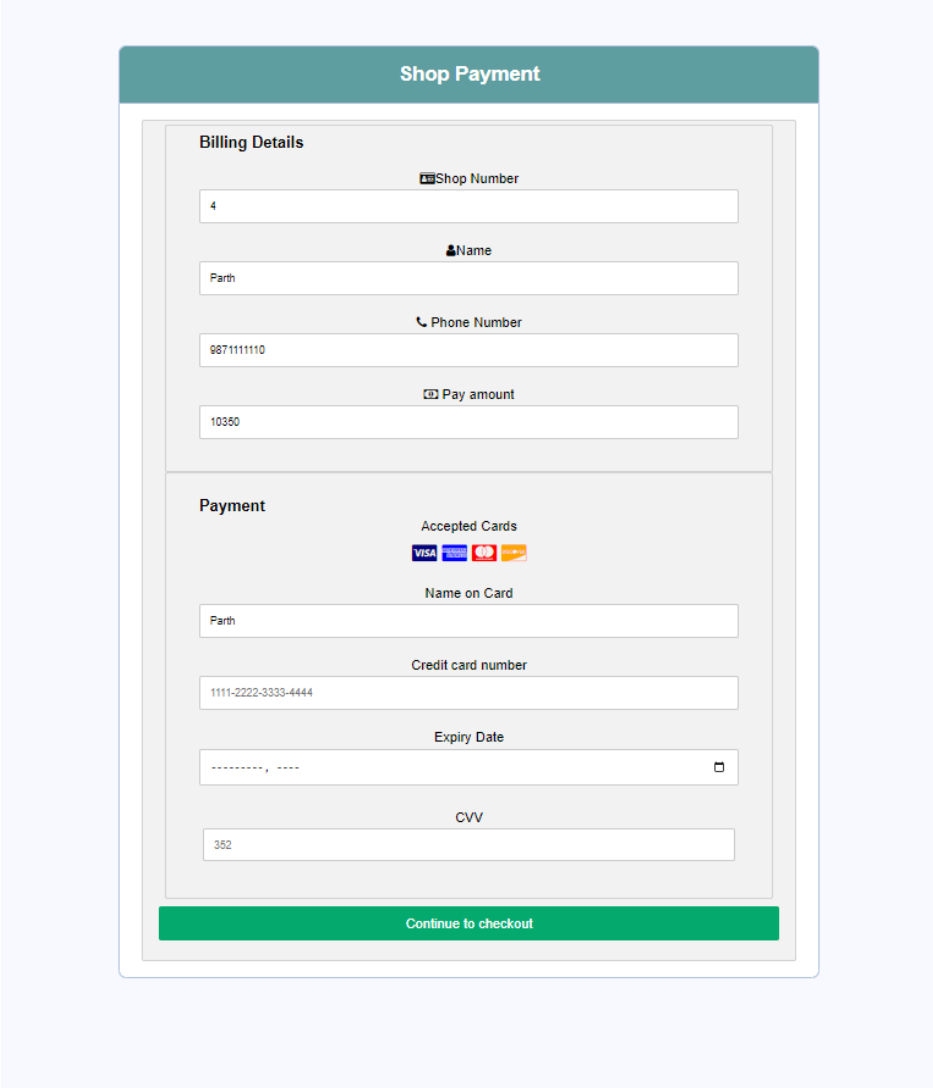
d. REQUEST FOR LICENSE RENEWAL

```
$query_renew= mysqli_query($db, "INSERT INTO RENEW_REQUESTS(License_Number)
VALUES ('$licenses')") or die($db->error);
```

Request for License Renewal can only be done if the pending bills on the shop are less than 10000 and no renew requests are pending

e. PAY BILLS

```
$query_pay= mysqli_query($db, "UPDATE SHOPS SET Bills = Bills-$bill_value WHERE
Shop_Number = '$shop_number'") or die($db->error);
```

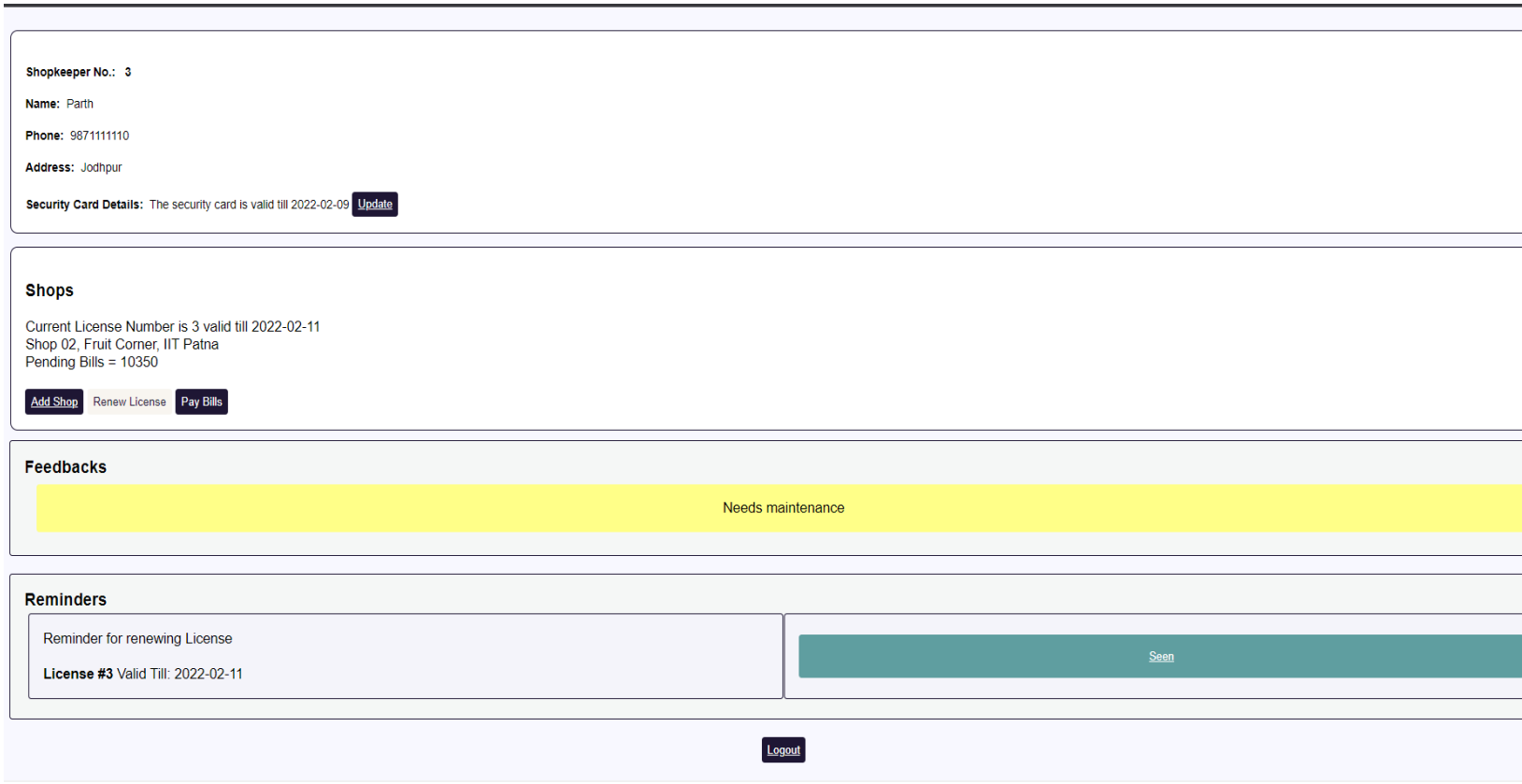


f. ACKNOWLEDGE REMINDERS

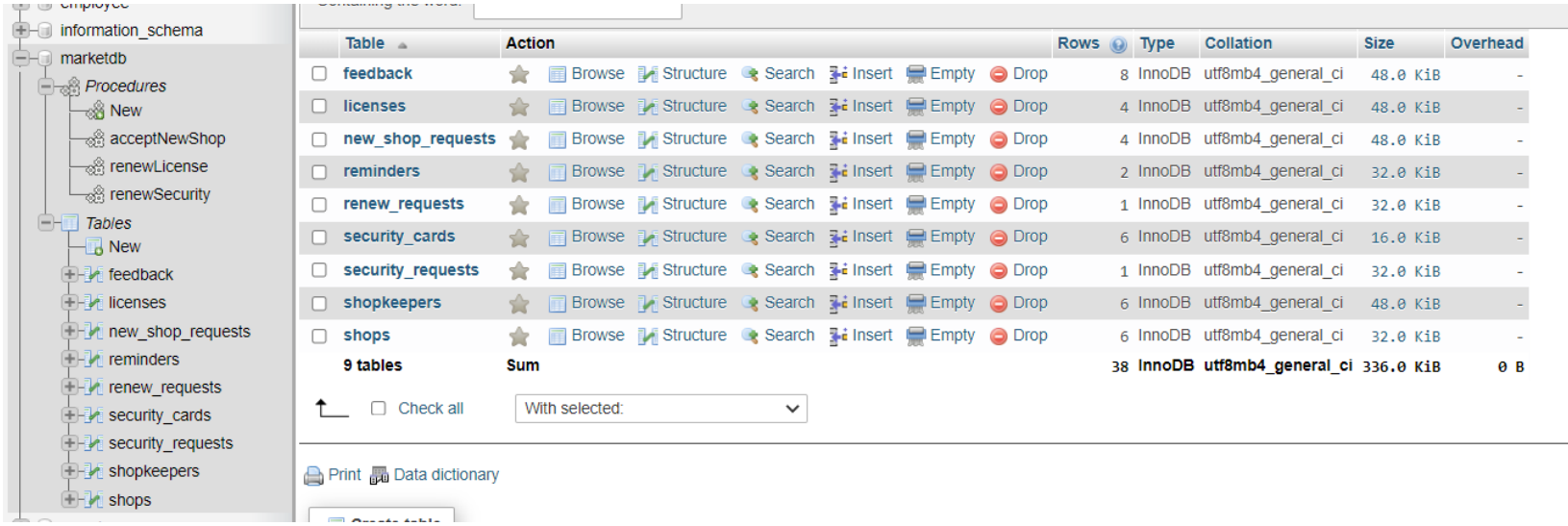
```
$query_rem= mysqli_query($db,
"DELETE FROM REMINDERS WHERE Reminder_Number = '$rem_num' ")
or die($db->error);
```

g. VIEW FEEDBACKS

```
$query_fe = mysqli_query($db,
"SELECT * FROM FEEDBACK WHERE Shop_Number = '$shop_num'")
or die ($db->error);
```



DATABASE



Name of the Database : marketdb

CONCLUSION

All the relevant code files, database, demonstrations and screenshots can be found here