NORMALIZATION

In relational database design, the process of organizing data to minimize redundancy. Normalization usually involves dividing a database into two or more tables and defining relationships between the tables.

Fisrt Normal Form:

A table is said to be in first (1NF) if and only if each attribute of the relation is atomic, each row in a table should be identified by primary key (a unique column value or group of unique column values) no rows of data should have replacing group of column values.

Second Normal Form:

A relation is in second normal form if it is in 1NF and every non key attribute is fully functionally dependent on the primary key.

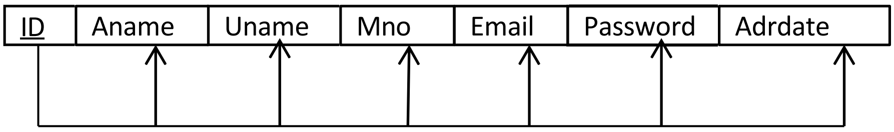
Third Normal Form:

Third normal form(3NF) is the third step in normalizing a database and it builds on the first and second normal forms, 1NF and 2NF. 3NF states that all column reference in referenced data that are not dependent on the primary key should be removed.

Boyce-Codd Normal Form:

Boyce-Codd Normal Form (BCNF) is a stricter version of the Third Normal Form (3NF). A table is in BCNF if it is in 3NF and, for every functional dependency (X → Y), the determinant (X) is a superkey. This means that every determinant must uniquely identify a row in the table. BCNF eliminates redundancy and anomalies by ensuring no non-trivial dependency violates this rule.

Admin Table



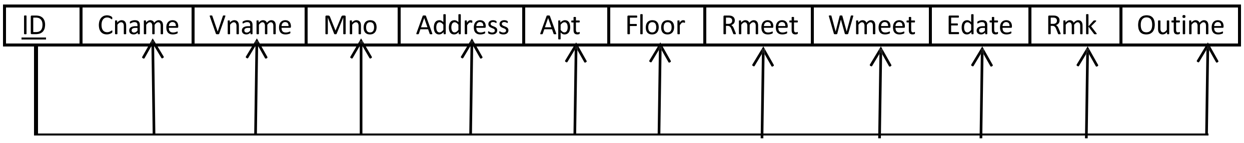
FD: ID→ {Aname,Uname,Mno,Email,Password,Adrdate}

The given schema is in 1NF because it does not have any multi valued or composite attribute.

The given schema is in 2NF because all non-prime attributes have full functional dependency with the primary key.The given schema is in 3NF because it does not have any transitive dependency.

The given schema is in Boyce-Codd Normal Form (BCNF) because, for every functional dependency, the determinant is a superkey, ensuring that there are no violations of dependency rules.

Visitor Table

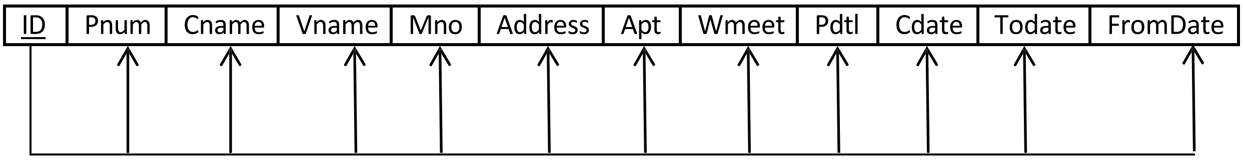


FD:ID → {Cname,Vname,Mno,Address,Apt,Floor,Rmeet,Wmeet,Edate,Rmk,Outime}

The given schema is in 1NF because it does not have any multi valued or composite attribute. The given schema is in 2NF because all non-prime attributes have full functional dependency with the primary key.The given schema is in 3NF because it does not have any transitive dependency.

The given schema is in Boyce-Codd Normal Form (BCNF) because, for every functional dependency, the determinant is a superkey, ensuring that there are no violations of dependency rules.

Visitor Pass Table



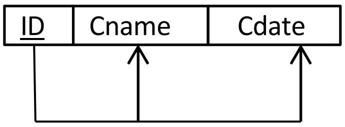
FD:ID → {Pnum,Cname,Mno,Address,Apt,Wmeet,Pdtl,Cdate,ToDate,FromDate}

The given schema is in 1NF because it does not have any multi valued or composite attribute.

The given schema is in 2NF because all non-prime attributes have full functional dependency with the primary key.The given schema is in 3NF because it does not have any transitive dependency.

The given schema is in Boyce-Codd Normal Form (BCNF) because, for every functional dependency, the determinant is a superkey, ensuring that there are no violations of dependency rules.

Category Table



FD:ID →{Cname,Cdate}

The given schema is in 1NF because it does not have any multi valued or composite attribute. The given schema is in 2NF because all non-prime attributes have full functional dependency with the primary key.The given schema is in 3NF because it does not have any transitive dependency.

The given schema is in Boyce-Codd Normal Form (BCNF) because, for every functional dependency, the determinant is a superkey, ensuring that there are no violations of dependency rules.

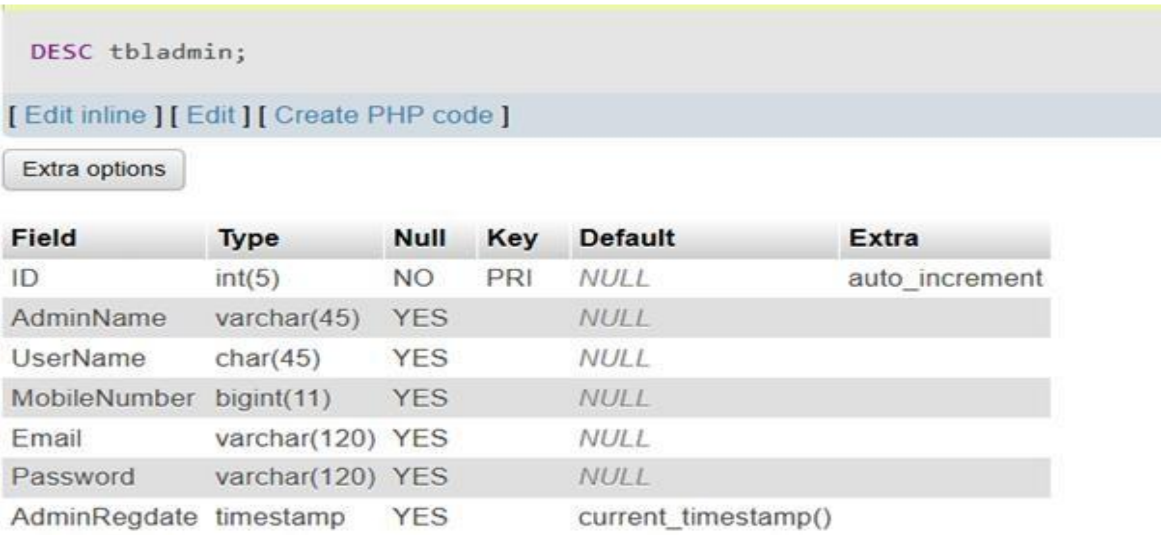
IMPLEMENTATION

Database and Table Structures

1.Create Table and its Structures

Admin Table:

```
CREATE TABLE `tbladmin` (
  `ID` int(5) NOT NULL PRIMARY KEY,
  `AdminName` varchar(45) DEFAULT NULL,
  `UserName` char(45) DEFAULT NULL,
  `MobileNumber` bigint(11) DEFAULT NULL,
  `Email` varchar(120) DEFAULT NULL,
  `Password` varchar(120) DEFAULT NULL,
  `AdminRegdate` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```



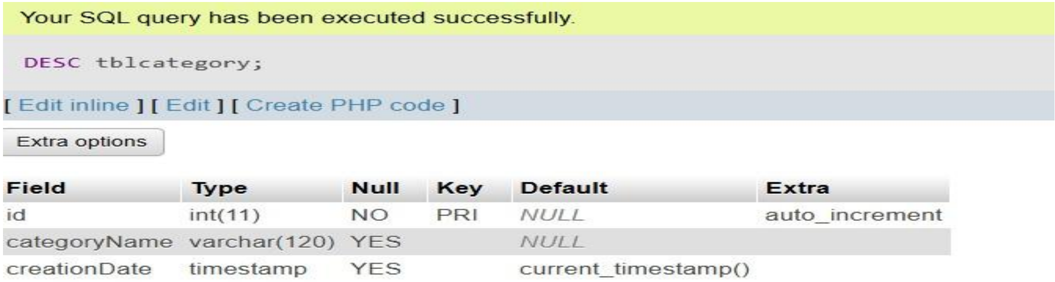
```
DESC tbladmin;
```

Field	Type	Null	Key	Default	Extra
ID	int(5)	NO	PRI	*NULL*	auto_increment
AdminName	varchar(45)	YES		*NULL*	
UserName	char(45)	YES		*NULL*	
MobileNumber	bigint(11)	YES		*NULL*	
Email	varchar(120)	YES		*NULL*	
Password	varchar(120)	YES		*NULL*	
AdminRegdate	timestamp	YES		current_timestamp()	

Fig 1:Admin Table Description

Category Table:

```
CREATE TABLE `tblcategory` (
  `id` int(11) NOT NULL  PRIMARY KEY,
  `categoryName` varchar(120) DEFAULT NULL,
  `creationDate` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



Your SQL query has been executed successfully.

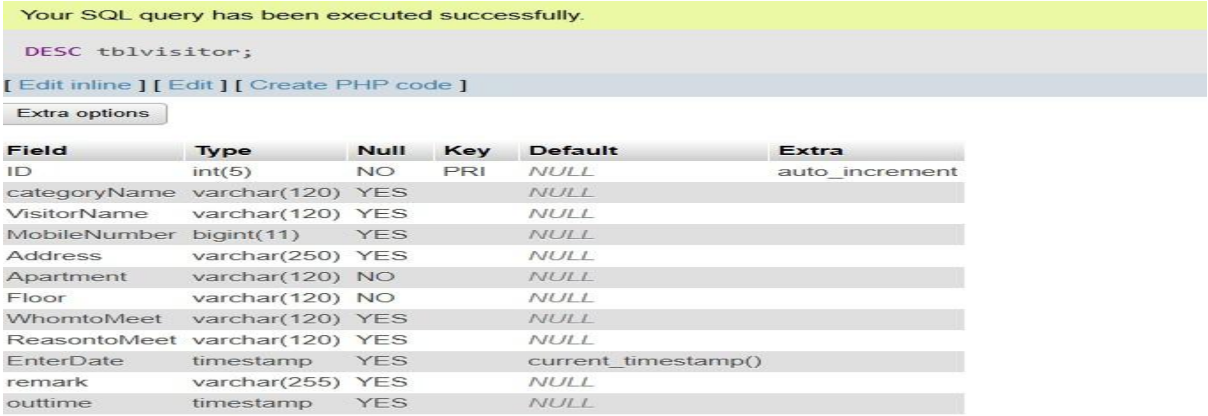
```
DESC tblcategory;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
categoryName	varchar(120)	YES		*NULL*	
creationDate	timestamp	YES		current_timestamp()	

Fig 2: Table Category Description

Table Visitor:

```
CREATE TABLE `tblvisitor` (
  `ID` int(5) NOT NULL PRIMARY KEY,
  `categoryName` varchar(120) DEFAULT NULL,
  `VisitorName` varchar(120) DEFAULT NULL,
  `MobileNumber` bigint(11) DEFAULT NULL,
  `Address` varchar(250) DEFAULT NULL,
  `Apartment` varchar(120) NOT NULL,
  `Floor` varchar(120) NOT NULL,
  `WhomtoMeet` varchar(120) DEFAULT NULL,
  `ReasontoMeet` varchar(120) DEFAULT NULL,
  `EnterDate` timestamp NULL DEFAULT current_timestamp(),
  `remark` varchar(255) DEFAULT NULL,
  `outtime` timestamp NULL DEFAULT NULL ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```



Your SQL query has been executed successfully.

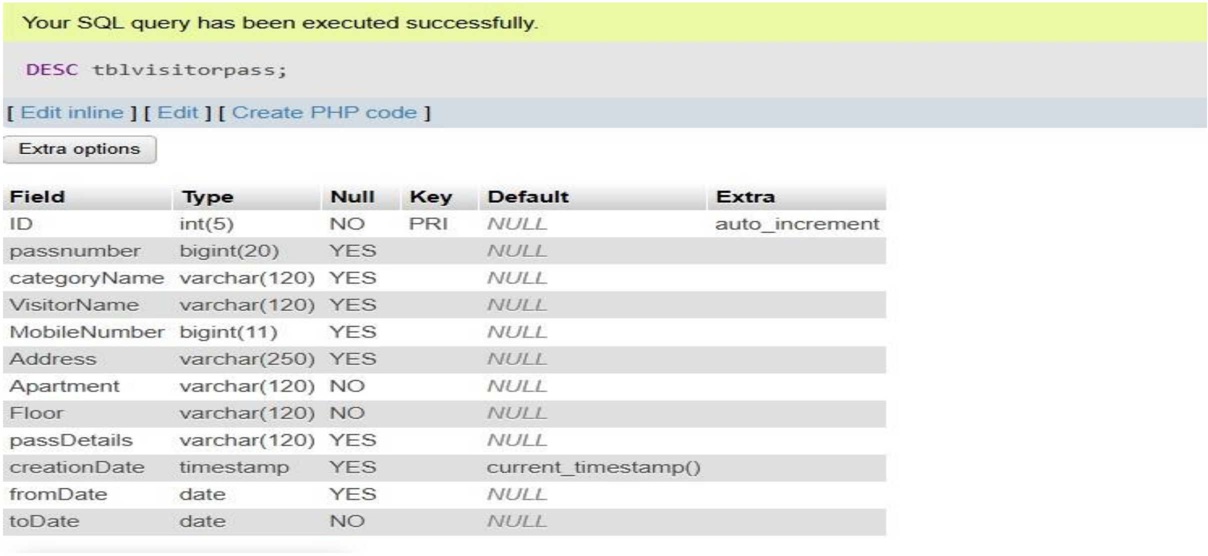
```
DESC tblvisitor;
```

Field	Type	Null	Key	Default	Extra
ID	int(5)	NO	PRI	NULL	auto_increment
categoryName	varchar(120)	YES		NULL	
VisitorName	varchar(120)	YES		NULL	
MobileNumber	bigint(11)	YES		NULL	
Address	varchar(250)	YES		NULL	
Apartment	varchar(120)	NO		NULL	
Floor	varchar(120)	NO		NULL	
WhomtoMeet	varchar(120)	YES		NULL	
ReasontoMeet	varchar(120)	YES		NULL	
EnterDate	timestamp	YES		current_timestamp()	
remark	varchar(255)	YES		NULL	
outtime	timestamp	YES		NULL	

Fig 3: Table Visitor Description

Table VistorPass:

```
CREATE TABLE `tblvisitorpass` (
  `ID` int(5) NOT NULL PRIMARY KEY,
  `passnumber` bigint(20) DEFAULT NULL,
  `categoryName` varchar(120) DEFAULT NULL,
  `VisitorName` varchar(120) DEFAULT NULL,
  `MobileNumber` bigint(11) DEFAULT NULL,
  `Address` varchar(250) DEFAULT NULL,
  `Apartment` varchar(120) NOT NULL,
  `Floor` varchar(120) NOT NULL,
  `passDetails` varchar(120) DEFAULT NULL,
  `creationDate` timestamp NULL DEFAULT current_timestamp(),
  `fromDate` date DEFAULT NULL,
  `toDate` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```



Your SQL query has been executed successfully.

```
DESC tblvisitorpass;
```

Field	Type	Null	Key	Default	Extra
ID	int(5)	NO	PRI	NULL	auto_increment
passnumber	bigint(20)	YES		NULL	
categoryName	varchar(120)	YES		NULL	
VisitorName	varchar(120)	YES		NULL	
MobileNumber	bigint(11)	YES		NULL	
Address	varchar(250)	YES		NULL	
Apartment	varchar(120)	NO		NULL	
Floor	varchar(120)	NO		NULL	
passDetails	varchar(120)	YES		NULL	
creationDate	timestamp	YES		current_timestamp()	
fromDate	date	YES		NULL	
toDate	date	NO		NULL	

Fig 4:Table Visitor Pass Description

Functionality:

Connection to the Database:

```
<?php

$con=mysqli_connect("localhost", "root", "", "avmsdb");

if(mysqli_connect_errno()){

echo "Connection Fail".mysqli_connect_error();

}

?>
```

INSERT:

Table Admin:

INSERT INTO `tbladmin`(`ID`, `AdminName`, `UserName`, `MobileNumber`, `Email`, `Password`, `AdminRegdate`)

VALUES ('[value-1]','[value-2]','[value-3]','[value-4]','[value-5]','[value-6]','[value-7]')

Table Category:

INSERT INTO `tblcategory`(`id`, `categoryName`, `creationDate`)

VALUES ('[value-1]','[value-2]','[value-3]')

Table Visitor:

```
INSERT INTO `tblvisitor`(`ID`, `categoryName`, `VisitorName`, `MobileNumber`, `Address`, `Apartment`, `Floor`, `WhomtoMeet`, `ReasontoMeet`, `EnterDate`, `remark`, `outtime`)
```

```
VALUES ('[value-1]','[value-2]','[value-3]','[value-4]','[value-5]','[value-6]','[value-7]','[value-8]','[value-9]','[value-10]','[value-11]','[value-12]')
```

Table VisitorPass:

```
INSERT INTO `tblvisitorpass`(`ID`, `passnumber`, `categoryName`, `VisitorName`, `MobileNumber`, `Address`, `Apartment`, `Floor`, `passDetails`, `creationDate`, `fromDate`, `toDate`)
```

```
 VALUES('[value-1]','[value-2]','[value-3]','[value-4]','[value-5]','[value-6]','[value-7]','[value-8]','[value-9]','[value-10]','[value-11]','[value-12]')
```

DELETE:

```
DELETE FROM `tbladmin` WHERE condition;
```

```
DELETE FROM `tbcategory` WHERE condition;
```

```
DELETE FROM `tblvisitor` WHERE condition;
```

```
DELETE FROM `tblvisitorpass` WHERE condition;
```

UPDATE:

```
UPDATE `tbladmin` SET `ID`='[value-1]',`AdminName`='[value-2]',`UserName`='[value-3]',`MobileNumber`='[value-4]',`Email`='[value-5]',`Password`='[value-6]',`AdminRegdate`='[value-7]' WHERE 1
```

```
UPDATE `tblcategory` SET `id`='[value-1]',`categoryName`='[value-2]',`creationDate`='[value-3]' WHERE 1
```

```
UPDATE `tblvisitor` SET `ID`='[value-1]',`categoryName`='[value-2]',`VisitorName`='[value-3]',`MobileNumber`='[value-4]',`Address`='[value-
5]',`Apartment`='[value-6]',`Floor`='[value-7]',`WhomtoMeet`='[value-8]',`ReasontoMeet`='[value-9]',`EnterDate`='[value-10]',`remark`='[value-11]',`outtime`='[value-12]' WHERE 1
```

```
UPDATE `tblvisitorpass` SET `ID`='[value-1]',`passnumber`='[value-2]',`categoryName`='[value-3]',`VisitorName`='[value-4]',`MobileNumber`='[value-5]',`Address`='[value-6]',`Apartment`='[value-7]',`Floor`='[value-8]',`passDetails`='[value-9]',`creationDate`='[value-10]',`fromDate`='[value-11]',`toDate`='[value-12]' WHERE 1
```

SELECT:

```
SELECT `ID`, `AdminName`, `UserName`, `MobileNumber`, `Email`, `Password`, `AdminRegdate` FROM `tbladmin` WHERE 1

SELECT `id`, `categoryName`, `creationDate` FROM `tblcategory` WHERE 1

SELECT `ID`, `categoryName`, `VisitorName`, `MobileNumber`, `Address`, `Apartment`, `Floor`, `WhomtoMeet`, `ReasontoMeet`, `EnterDate`, `remark`, `outtime` FROM `tblvisitor` WHERE 1

SELECT `ID`, `passnumber`, `categoryName`, `VisitorName`, `MobileNumber`, `Address`, `Apartment`, `Floor`, `passDetails`, `creationDate`, `fromDate`, `toDate` FROM `tblvisitorpass` WHERE 1
```

ASSERTION:

-- Assertion for email validation in `tbladmin

```
DELIMITER $$

CREATE TRIGGER before_insert_tbladmin_email

BEFORE INSERT ON tbladmin

FOR EACH ROW

BEGIN
```

```
  IF NOT NEW.Email REGEXP '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$' THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Invalid Email Address in tbladmin.';
  END IF;
END$$
```

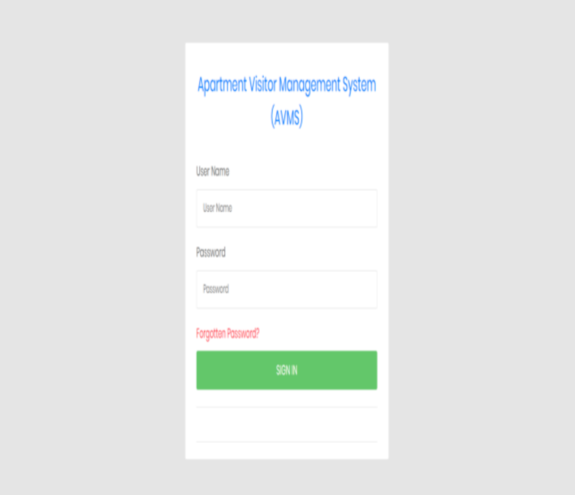
Assertion for `fromDate` < `toDate` in `tblvisitorpass`

```
-- Trigger to check valid date ranges in `tblvisitorpass`
--
DELIMITER $$

CREATE TRIGGER before_insert_tblvisitorpass_datecheck
BEFORE INSERT ON tblvisitorpass
FOR EACH ROW
BEGIN
  IF NEW.fromDate >= NEW.toDate THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Invalid Date Range: `fromDate` must be earlier than `toDate`.';
  END IF;
END$$

DELIMITER ;
```

TRIGGER:

```
DELIMITER ;

-- Trigger to auto-update `outtime` in `tblvisitor`

--

DELIMITER $$


CREATE TRIGGER before_update_tblvisitor_outtime

BEFORE UPDATE ON tblvisitor

FOR EACH ROW

BEGIN

  IF NEW.outtime IS NULL THEN

    SET NEW.outtime = CURRENT_TIMESTAMP;

  END IF;

END$$


CREATE TABLE visitorlogs (

    id INT AUTO_INCREMENT PRIMARY KEY,

    visitor_id INT NOT NULL,

    action VARCHAR(255) NOT NULL,

    cdate DATETIME NOT NULL

);

CREATE TRIGGER `insert_log` AFTER INSERT ON `tblvisitor`

FOR EACH ROW INSERT INTO visitorlogs VALUES(null, NEW.id, 'Inserted', NOW())

CREATE TRIGGER `update_log` AFTER UPDATE ON `tblvisitor`

 FOR EACH ROW INSERT INTO visitorlogs VALUES(null, NEW.id, 'Updated', NOW())
```

Results:



Apartment Visitor Management System (AVMS)

User Name

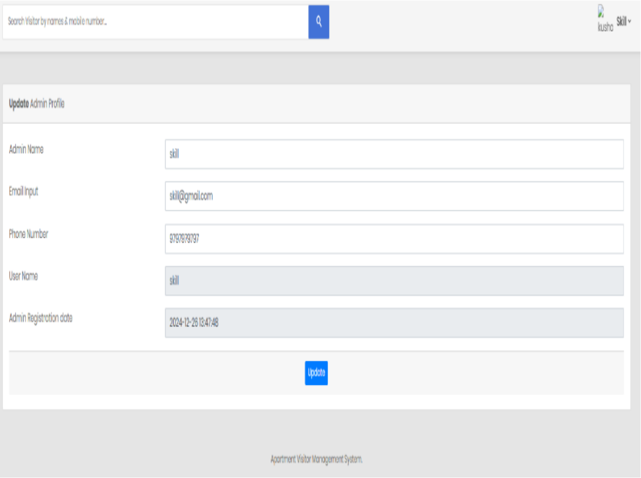
User Name

Password

Password

Forgotten Password?

Fig 1:Login Page



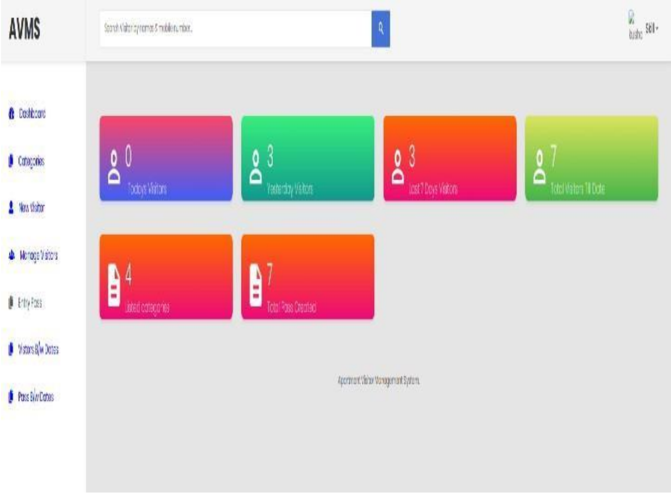
Search Visitor by names & mobile number...
kusha
Skill

Update Admin Profile

Admin Name	skill
Email Input	skill@gmail.com
Phone Number	9797979797
User Name	skill
Admin Registration date	2024-12-26 13:47:48

Update
Apartment Visitor Management System.

Fig 2:Admin Page



0 Todays Visitors

3 Yesterday Visitors

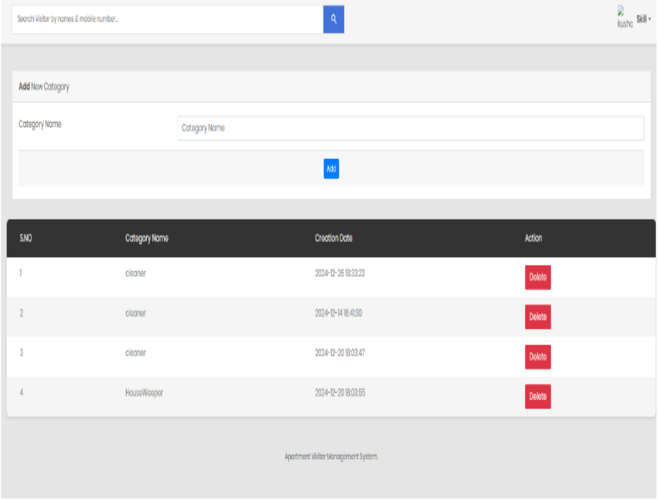
3 Last 7 Days Visitors

7 Total Visitors Till Date

4 Listed categories

7 Total Pass Created

Fig 3:DashBoard



Search Visitor by names & mobile number...
kusha
Skill

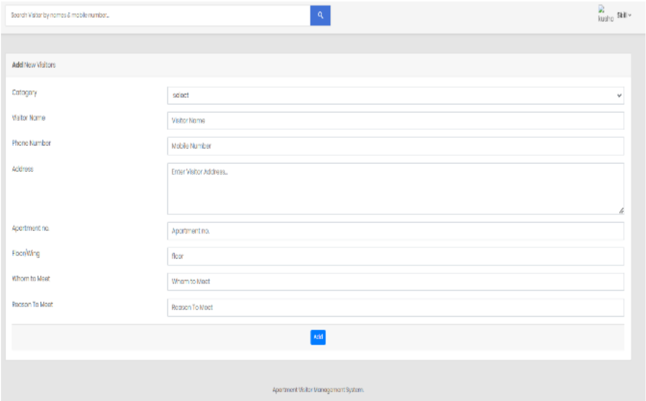
Add New Category

Category Name	Category Name

S.NO	Category Name	Creation Date	Action
1	cleaner	2024-12-26 18:33:23	Delete
2	cleaner	2024-12-14 16:41:30	Delete
3	cleaner	2024-12-20 18:03:47	Delete
4	HouseWeeper	2024-12-20 18:03:55	Delete

Apartment Visitor Management System.

Fig 4:Categories



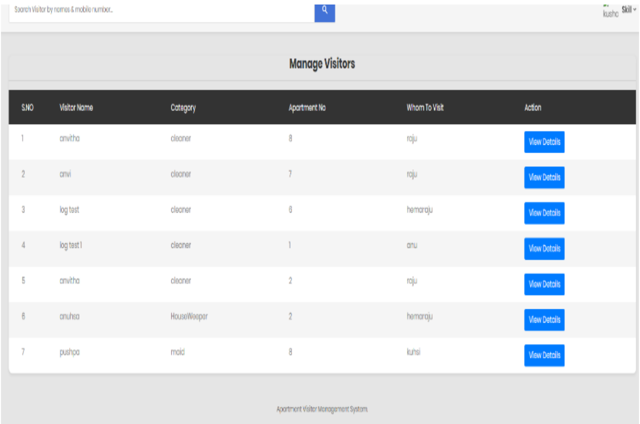
Search Visitor by names & mobile number...
kusha
Skill

Add New Visitors

Category	select
Visitor Name	Visitor Name
Phone Number	Mobile Number
Address	Enter Visitor Address...
Apartment no.	Apartment no.
Floor/Wing	floor
Whom to Meet	Whom to Meet
Reason To Meet	Reason To Meet

Add
Apartment Visitor Management System.

Fig 5:New Visitors



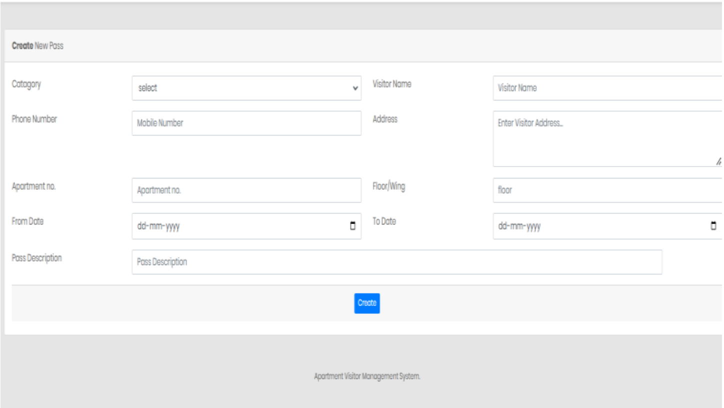
Search Visitor by names & mobile number...
kusha
Skill

Manage Visitors

S.NO	Visitor Name	Category	Apartment No	Whom To Visit	Action
1	anvitha	cleaner	8	raju	View Details
2	anvi	cleaner	7	raju	View Details
3	log test	cleaner	8	hemaraju	View Details
4	log test1	cleaner	1	anu	View Details
5	anvitha	cleaner	2	raju	View Details
6	anuhsa	HouseWeeper	2	hemaraju	View Details
7	pushpa	maid	8	kuhsi	View Details

Apartment Visitor Management System.

Fig 6:Manage visitors



Create New Pass

Catagory: select

Visitor Name: Visitor Name

Phone Number: Mobile Number

Address: Enter Visitor Address...

Apartment no.: Apartment no.

Floor/Wing: floor

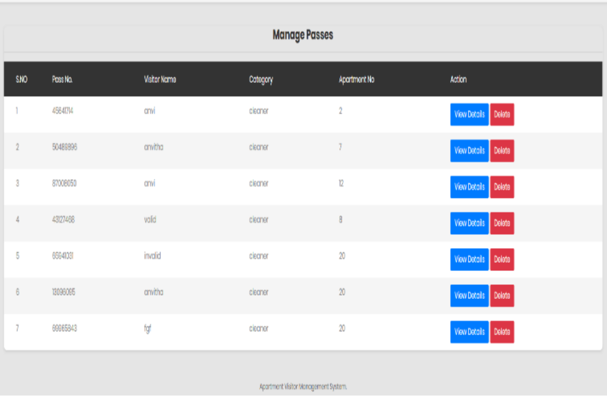
From Date: dd-mm-yyyy

To Date: dd-mm-yyyy

Pass Description: Pass Description

Apartment Visitor Management System.

Fig 7:Create New Pass

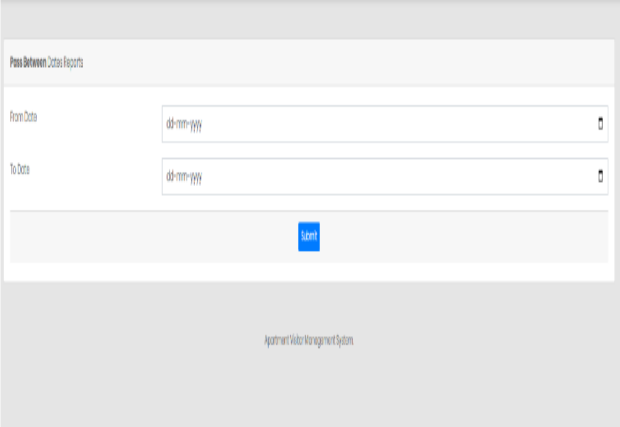


Manage Passes

S.NO	Pass No.	Visitor Name	Category	Apartment No	Action
1	45641714	anvi	cleaner	2	View Details Delete
2	50489896	anvitha	cleaner	7	View Details Delete
3	87008050	anvi	cleaner	12	View Details Delete
4	43127468	valid	cleaner	8	View Details Delete
5	65641331	invalid	cleaner	20	View Details Delete
6	13096095	anvitha	cleaner	20	View Details Delete
7	69965843	fgf	cleaner	20	View Details Delete

Apartment Visitor Management System.

Fig 8:Manage Pass

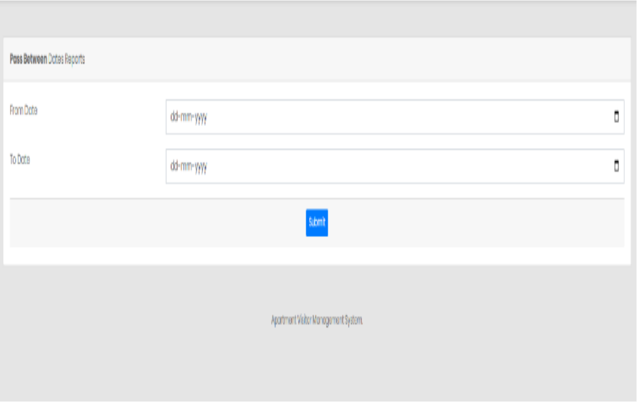


Pass Between Dates Reports

From Date: dd-mm-yyyy

To Date: dd-mm-yyyy

Apartment Visitor Management System.



Pass Between Dates Reports

From Date: dd-mm-yyyy

To Date: dd-mm-yyyy

Apartment Visitor Management System.

Fig 7:Visitor Between Dates

Fig 8:Pass Between Dates

Conclusion

The Apartment Visitor Management System is a modernized solution designed to simplify and automate the process of managing visitor records in apartment complexes. By replacing traditional manual record books, the system significantly enhances productivity and reduces the challenges associated with maintaining physical records. With an intuitive graphical user interface, it offers a user-friendly experience for both staff and administrators. The system enables authorized staff to log visitor details efficiently, generate reports seamlessly, and ensure secure and reliable data management. It eliminates communication delays and allows for quick and hassle-free updates to visitor information. Moreover, the application is built with flexibility in mind, ensuring it can be easily modified to accommodate future needs. This comprehensive system not only improves operational efficiency but also strengthens data security and reliability, making it an invaluable tool for modern apartment management.

Beyond its core functionalities, the Apartment Visitor Management System offers a scalable and adaptable framework that can accommodate the evolving needs of residential communities. By integrating advanced features like real-time updates, secure data storage, and detailed analytics, the system empowers management to maintain a high level of operational efficiency. It also enhances the safety and security of the premises by ensuring accurate tracking of visitor movements and providing swift access to visitor records when required. The system's ability to streamline workflows and reduce manual errors makes it an indispensable tool for modern residential complexes, fostering a secure and well-organized environment for residents and visitors alike.

References:

For PHP

- https://www.w3schools.com/php/default.asp
- https://www.sitepoint.com/php/
- https://www.php.net/

For MySQL

- https://www.mysql.com/
- http://www.mysqltutorial.org

For XAMPP

https://www.apachefriends.org/download.html