CHAPTER 1

INTRODUCTION

House Rental System is a system that provides the list of houses available and at what location which is very beneficial for the tenants who are in search for the houses. The individual who needs a house must contact a rental system and contract out for a room/home/apartment. This system increases customer retention and simplify House and staff management. This system has Tenants who can book houses for rent, give rating to the houses, add members to their houses. Owner who can add houses to the database. House entity which maintains details of all the houses. This system is very reliable, portable and useful.

There are major two types of users who can use our system namely Owners and Tenants. Moreover anyone can view the houses without logging in to the system but in order to book or adding any details, one has to either log in to the system or sign in to it.

OBJECTIVES:

• The world is familiar with the great crowd of students, job seekers and many other people moving to cities to get a good opportunity. This system will help the people especially students,

searching for the best rooms in a good locality. This will help them finding a shelter, which

fits all their requirements at their affordable price.

• This system will also simplify the work for the rental managers so that the work can be efficient.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements :

1. PHP 5 : PHP is a computer programming language originally designed for producing dynamic web pages. The name PHP means PHP Hypertext Preprocessor. Here we have used php for server side programming ie connecting our frontend and backend.

2. MySQL : MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. We have used MYSQL for creating our database.

3. XAMPP SERVER : XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Version used in our system is 3.2.2.

2.2 Hardware Requirements:

1. WINDOWS 10 : This project can be used in windows OS.

2. 520 MB RAM or higher : This project has 520 MB RAM or higher.

2.3 Programming Languages:

1. HTML :

First developed by Tim Berners-Lee in 1990, **HTML** is short for Hypertext Markup Language. **HTML** is used to create electronic documents (called pages) that are displayed on the World Wide Web. We have used HTML for scripting the web page.

2. CSS :

Cascading Style Sheets (**CSS**) is a stylesheet language used to describe the presentation of a document written in HTML or XML. We have used CSS for styling our web pages.

3. BOOTSTRAP :

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation and other interface components. We have used bootsrap so that our system can be used in any of the devices like phones, laptops, PCs etc.

2.4 Functional Requirements

Requirement analysis is a software engineering technique that is composed of the various tasks that determine the needs or conditions that are to be met for a new or altered product, taking into consideration the possible conflicting requirements of the various users. Functional requirements are those requirements that are used to illustrate the internal working nature of the system, the description of the system, and explanation of each subsystem. It consists of what task the system should perform, the processes involved, which data should the system hold and the interfaces with the user. The functional requirements identified are:

- Tenant’s registration: The system should allow new tenant’s to register online.
- Owner’s registration: The system should allow new owner’s to register online.
- Online booking of House: Tenant or Owners should be able to use the system to make booking.
- Adding members who will stay in the rented house booked by the tenants.

- Adding house: The owners can add the houses to the database which can be viewed and booked by everyone.
- Give rating: The tenants can give ratings and comments to the houses.

2.5 Non-Functional Requirements

It describes aspects of the system that are concerned with how the system provides the functional requirements. They are:

a. Security: The subsystem should provide a high level of security and integrity of the data held by the system, only authorized personnel can gain access to the secured page on the system; and only users with valid password and username can login to view user's page.

b. Performance and Response time: The system should have high performance rate when executing user's input and should be able to provide proper response within a short time.

d. Availability: This system should always be available for access at 24 hours, 7 days a week. Also in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days, so that the business process is not severely affected.

e. Portability: This system is very portable and can be carried from one system to another very easily.

Chapter 3

DESIGN

3.1 E-R Diagram

An entity-relationship diagram (ER diagram) is a data modeling technique that graphically illustrates information about system's entities and the relationships between those entities .An ER Diagram is a conceptual and representational model of data used to represent the entity framework infrastructure. This ER diagram has 6 entities namely Tenant who can book houses for rent, give rating to the houses, add members to their houses. Owner who can add houses to the database. House entity which maintains details of all the houses.

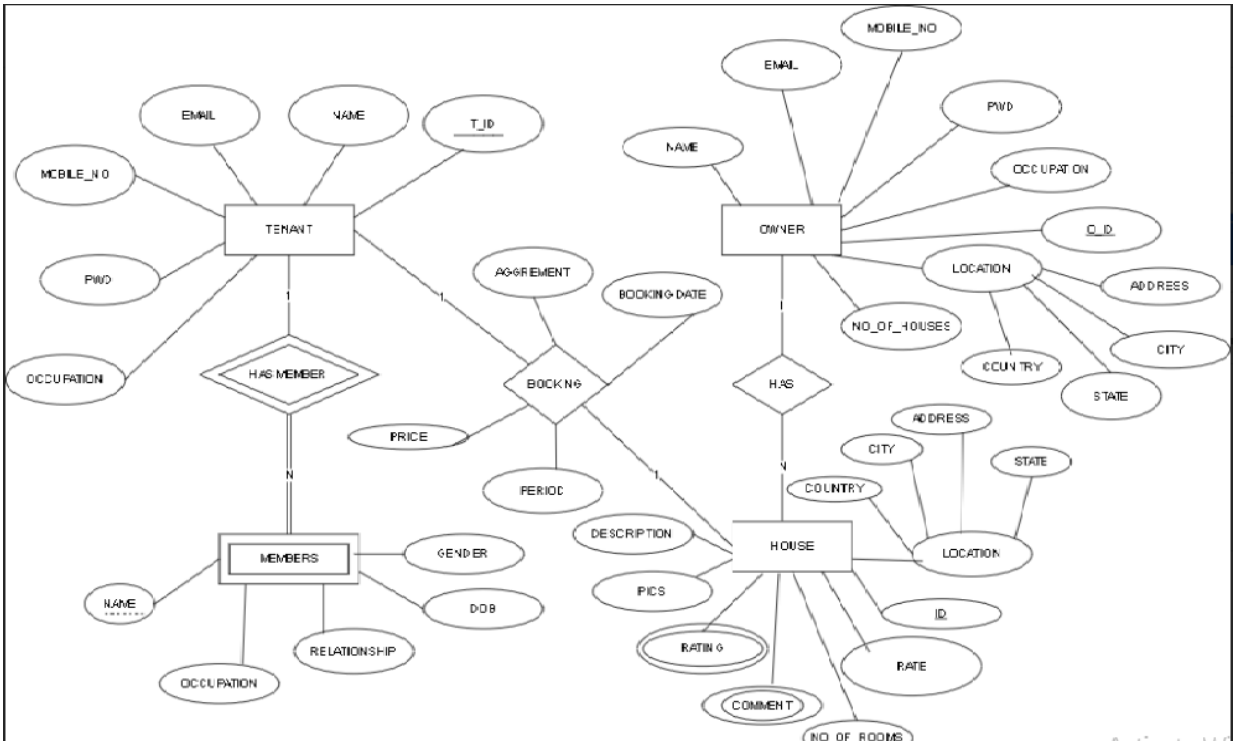


Fig 3.1 ER Diagram House rental management system

3.2 Schema Diagram

A database schema is skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

The schema diagram defines its entities and the relationship among them. In a schema diagram, all database tables are designated with unique columns and special features, e.g., primary key, foreign key, etc... The table's relationships are expressed via a parent table's primary key [illegible] when joined with the child table's corresponding foreign keys.

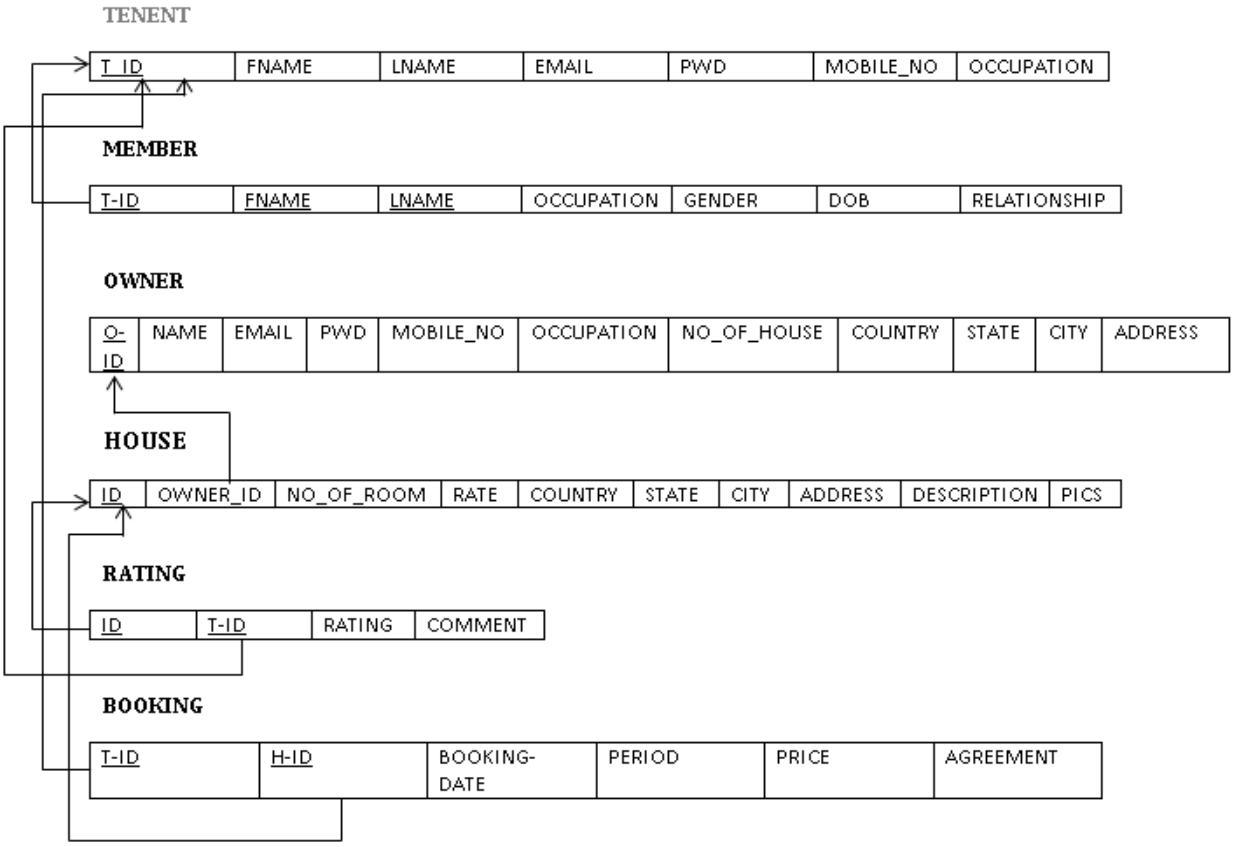


Fig 3.2 Schema Diagram of House rental management system

CHAPTER 4

IMPLEMENTATION

Implementation includes the Table Creations, Table Insertions, Table Queries, Stored Procedure and Triggers, all written in SQL. It also includes explanations as to why Stored Procedures and Triggers are used in this project.

4.1 Table creations

Table Creation for tenant

```
CREATE TABLE `tenant` (
  `t_id` int(11) NOT NULL,
  `fname` varchar(20) DEFAULT NULL,
  `lname` varchar(20) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  `pwd` varchar(30) DEFAULT NULL,
  `mobile_no` bigint(20) DEFAULT NULL,
  `occupation` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	t_id	int(11)			No	None			Change Drop Primary Unique More
2	fname	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
3	lname	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
4	email	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
5	pwd	varchar(30)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
6	mobile_no	bigint(20)			Yes	NULL			Change Drop Primary Unique More
7	occupation	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More

Fig 4.1 Table Creation for tenant

Table Creation for member

```
CREATE TABLE `member` (
  `t_id` int(11) NOT NULL,
  `fname` varchar(20) NOT NULL,
  `lname` varchar(20) NOT NULL,
  `occupation` varchar(50) DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `dob` date DEFAULT NULL,
  `relationship` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

RELATIONSHIPS FOR TABLE `member`:

`t_id` -- `tenant` -> `t_id`

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	t_id	int(11)			No	None			Change Drop Primary Unique More
2	fname	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
3	lname	varchar(20)	latin1_swedish_ci		No	None			Change Drop Primary Unique More
4	occupation	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
5	gender	varchar(10)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
6	dob	date			Yes	NULL			Change Drop Primary Unique More
7	relationship	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More

Fig 4.2 Table Creation for member

Table creation for owner

```
CREATE TABLE `owner` (

  `o_id` int(11) NOT NULL,

  `name` varchar(20) DEFAULT NULL,

  `email` varchar(50) DEFAULT NULL,

  `pwd` varchar(30) DEFAULT NULL,

  `mobile_no` bigint(20) DEFAULT NULL,

  `occupation` varchar(50) DEFAULT NULL,

  `no_of_houses` int(11) DEFAULT NULL,

  `country` varchar(20) DEFAULT NULL,

  `state` varchar(20) DEFAULT NULL,

  `city` varchar(30) DEFAULT NULL,

  `address` varchar(50) DEFAULT NULL
```

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	o_id	int(11)			No	None			Change Drop Primary Unique More
2	name	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
3	email	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
4	pwd	varchar(30)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
5	mobile_no	bigint(20)			Yes	NULL			Change Drop Primary Unique More
6	occupation	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
7	no_of_houses	int(11)			Yes	NULL			Change Drop Primary Unique More
8	country	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
9	state	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
10	city	varchar(30)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
11	address	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More

Fig 4.3 Table Creation for owner

Table creation for house

```
CREATE TABLE `house` (

  `id` int(11) NOT NULL,

  `owner_id` int(11) DEFAULT NULL,

  `no_of_rooms` int(11) DEFAULT NULL,

  `rate` int(11) DEFAULT NULL,

  `pics` blob,

  `country` varchar(20) DEFAULT NULL,

  `state` varchar(20) DEFAULT NULL,

  `city` varchar(30) DEFAULT NULL,

  `address` varchar(50) DEFAULT NULL,
```

```
  `description` varchar(300) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

RELATIONSHIPS FOR TABLE `house`:

`owner_id` -- `owner` -> `o_id`

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None			Change Drop Primary Unique More
2	owner_id	int(11)			Yes	NULL			Change Drop Primary Unique More
3	no_of_rooms	int(11)			Yes	NULL			Change Drop Primary Unique More
4	rate	int(11)			Yes	NULL			Change Drop Primary Unique More
5	pics	blob			Yes	NULL			Change Drop Primary Unique More
6	country	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
7	state	varchar(20)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
8	city	varchar(30)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
9	address	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More
10	description	varchar(300)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More

Fig 4.4 Table Creation for house

Table creation for rating

```
CREATE TABLE `rating` (

  `id` int(11) NOT NULL,

  `t_id` int(11) NOT NULL,

  `rating` int(11) DEFAULT NULL,

  `comment` varchar(300) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

RELATIONSHIPS FOR TABLE `rating`:

`id` -- `house` -> `id`

`t_id` -- `tenant` -> `t_id`

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None			Change Drop Primary Unique More
2	t_id	int(11)			No	None			Change Drop Primary Unique More
3	rating	int(11)			Yes	NULL			Change Drop Primary Unique More
4	comment	varchar(300)	latin1_swedish_ci		Yes	NULL			Change Drop Primary Unique More

Fig 4.5 Table Creation for rating

Table creation for booking

```
CREATE TABLE `booking` (
  `t_id` int(11) NOT NULL,
  `h_id` int(11) NOT NULL,
  `booking_date` date DEFAULT NULL,
  `period` int(11) DEFAULT NULL,
  `price` int(11) DEFAULT NULL,
  `agreement` mediumblob
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

RELATIONSHIPS FOR TABLE `booking`:

`t_id` -- `tenant` -> `t_id`

`h_id` -- `house` -> `id`

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	t_id	int(11)			No	None			Change Drop Primary Unique Index More
2	h_id	int(11)			No	None			Change Drop Primary Unique Index More
3	booking_date	date			Yes	NULL			Change Drop Primary Unique Index More
4	period	int(11)			Yes	NULL			Change Drop Primary Unique Index More
5	price	int(11)			Yes	NULL			Change Drop Primary Unique Index More
6	agreement	mediumblob			Yes	NULL			Change Drop Primary Unique Index More

Fig 4.6 Table Creation for booking

4.7 Display Table

¬ Select * from member; (fig 4.7)

	t_id	fname	lname	occupation	gender	dob	relationship
Edit Copy Delete	1	Raksha	Sarkar	Student	female	2009-11-10	sister
Edit Copy Delete	3	Amrutha	GP	student	female	2009-12-10	sister
Edit Copy Delete	3	Shourya	Mishra	student	male	2019-11-05	friend
Edit Copy Delete	3	Umang	Shrivastava	student	male	2019-12-04	friend
Edit Copy Delete	4	Aditya	Singh	engineer	male	2019-05-06	brother

Fig 4.7

¬ Select * from rating; (Fig 4.8)

	id	t_id	rating	comment
Edit Copy Delete	1	1	5	good
Edit Copy Delete	2	5	5	cool
Edit Copy Delete	3	1	5	good
Edit Copy Delete	4	4	3	ok ok
Edit Copy Delete	5	2	2	bad

Fig 4.8

¬ Select * from booking; (Fig 4.9)

	t_id	h_id	booking_date	period	price	agreement
Edit Copy Delete	2	1	2019-11-10	10	5000	[BLOB - 13 B]
Edit Copy Delete	2	5	2019-12-20	6	5000	[BLOB - 13 B]
Edit Copy Delete	3	2	2020-01-23	5	8000	[BLOB - 787 B]
Edit Copy Delete	4	4	2019-07-08	12	6000	[BLOB - 694 B]
Edit Copy Delete	5	5	2019-12-25	10	1000	[BLOB - 2.8 KiB]

Fig 4.9

¬ Select * from owner; (fig 4.10)

	o_id	name	email	pwd	mobile_no	occupation	no_of_houses	country	state	city
Edit Copy Delete	1	Megha	megha@gmail.com	ppppp	8660826138	software developer	3	india	karnataka	bangalore
Edit Copy Delete	2	Khushi	khushi@gmail.com	kkkkk	4444555556	Doctor	2	India	Karnataka	Bangalore
Edit Copy Delete	3	Nisha	nisha@gmail.com	nnnnn	6764387428	Teacher	0	india	kerela	tiruvanthapur
Edit Copy Delete	4	Yash	yash@gmail.com	yyyyy	8797879798	IAS	0	india	jharkhand	ranchi
Edit Copy Delete	5	Ravi	ravi@gmail.com	papa	9898989898	Pilot	0	india	punjab	jalandhar

Fig 4.10

¬ Select * from tenant; (Fig 4.11)

	t_id	fname	lname	email	pwd	mobile_no	occupation
Edit Copy Delete	1	Sameeksha	Sen	sen@gmail.com	sen123	8888888888	student
Edit Copy Delete	2	Sunetra	Sarkar	sunetra@gmail.com	qqqqq	1234567890	Student
Edit Copy Delete	3	Shwetha	GP	shwetha@gmail.com	sssss	9796858463	Student
Edit Copy Delete	4	Vinuta	Bhat	vinuta@gmail.com	vvvvv	7654387697	Doctor
Edit Copy Delete	5	Mehul	Jain	mehul@gmail.com	mmmmm	8888855555	Poet

Fig 4.11

¬ Select * from house;(fig 4.12)

	id	owner_id	no_of_rooms	rate	pics	country	state	city	address	description
Edit Copy Delete	1	1	3	10000	[BLOB - 11.7 KiB]	india	karnataka	bangalore	jalahalli	good house
Edit Copy Delete	2	1	3	20000	[BLOB - 10.2 KiB]	india	karnataka	bangalore	jalahalli west	beautiful house
Edit Copy Delete	3	1	5	15000	[BLOB - 11.2 KiB]	india	bihar	patna	patna garden	beautiful house
Edit Copy Delete	4	2	3	5000	[BLOB - 14.3 KiB]	India	Bengal	kolkata	kolkata	good
Edit Copy Delete	5	2	3	20000	[BLOB - 5.6 KiB]	india	goa	goa	goa	seaview

Fig 4.12

4.13 Queries

Query 1: Update the no of houses in owner table based on the data in house table.

Q1: update owner o set no_of_houses=(SELECT count(*) from house h where o.o_id=h.owner_id);

select * from owner;

OUTPUT (4.13)

opy	Delete	1	Megha	megha@gmail.com	ppppp	8660826138	software developer	3	india	karnataka	bangalore	vrpura
opy	Delete	2	Khushi	khushi@gmail.com	kkkkk	4444555556	Doctor	2	India	Karnataka	Bangalore	Bel Circle
opy	Delete	3	Nisha	nisha@gmail.com	nnnnn	6764387428	Teacher	0	india	kerela	tiruvanthapuram	tiruvanthapuram
opy	Delete	4	Yash	yash@gmail.com	yyyyy	8797879798	IAS	0	india	jharkhand	ranchi	ranchi
opy	Delete	5	Ravi	ravi@gmail.com	papa	9898989898	Pilot	0	india	punjab	jalandhar	jalandhar

Fig 4.13

Query 2. Create a view showing all the vacant houses.

Q 2: create view vacant houses AS

select * from house where id not in (SELECT h_id from booking);

OUTPUT (Fig 4.14**)**

					id	owner_id	no_of_rooms	rate	pics	country	state	city	address	description
Edit	Copy	Delete	3	1	5	15000	[BLOB - 11.2 KiB]	india	bihar	patna	patna garden	beautiful house

Fig 4.14

Query 3. Display the owners who stay in a cities and have houses in other cities.

Q 3 : SELECT * from owner where not city=any(SELECT city from house where o_id=owner_id) and no_of_houses>0;

OUTPUT (Fig 4.15)

	o_id	name	email	pwd	mobile_no	occupation	no_of_houses	country	state	city	address
Edit Copy Delete	2	Khushi	khushi@gmail.com	kkkkk	4444555556	Doctor	2	India	Karnataka	Bangalore	Bel Circle

Fig 4.15

Query 4. Find the tenant who has no members.

Q 4: SELECT * FROM tenant t where not exists (SELECT * from member m where t.t_id =m.t_id);

OUTPUT (Fig 4.16)

	t_id	fname	lname	email	pwd	mobile_no	occupation
Edit Copy Delete	2	Sunetra	Sarkar	sunetra@gmail.com	qqqqq	1234567890	Student
Edit Copy Delete	5	Mehul	Jain	mehul@gmail.com	mmmmm	8888855555	Poet

Fig 4.16

Query 5 . Find the house with the maximum rating.

Q 5: SELECT *,max(rating) FROM house h, rating r where rating in(SELECT max(rating) from rating) and r.id=h.id;

OUTPUT (Fig 4.17)

id	owner_id	no_of_rooms	rate	pics	country	state	city	address	description	id	t_id	rating	comment	max(rating)
1	1	3	10000	[BLOB - 11.7 KiB]	india	karnataka	bangalore	jalahalli	good house	1	1	5	good	5

Fig 4.17

Query 6. Display the names of tenants and members who have taken owner Megha's house.

Q 6: SELECT t.fname,t.lname from tenant t,house h,owner o,booking b where o.o_id=h.owner_id and b.h_id=h.id and b.t_id =t.t_id and o.name="Megha" union SELECT m.fname,m.lname from member m, tenant t,house h,owner o,booking b where m.t_id=t.t_id and o.o_id=h.owner_id and b.h_id=h.id and b.t_id =t.t_id and o.name="Megha";

OUTPUT (Fig 4.18)

+ Options

fname	lname
Sunetra	Sarkar
Shwetha	GP
Amrutha	GP
Shourya	Mishra
Umang	Shrivastava

Fig 4.18

Query 7. Display all the tenants who are students.

Q 7: SELECT * from tenant where occupation="student";

OUTPUT (Fig 4.19)

+ Options

←T→	t_id	fname	lname	email	pwd	mobile_no	occupation
Edit Copy Delete	1	Sameeksha	Sen	sen@gmail.com	sen123	8888888888	student
Edit Copy Delete	2	Sunetra	Sarkar	sunetra@gmail.com	qqqqq	1234567890	Student
Edit Copy Delete	3	Shwetha	GP	shwetha@gmail.com	sssss	9796858463	Student

Fig 4.19

Query 8: Show the reviews of tenant Sameeksha.

Q 8: SELECT id as house_id ,rating,comment from rating r join tenant t on t.t_id=r.t_id where fname="Sameeksha";

OUTPUT (Fig 4.20)

+ Options

house_id	rating	comment
1	5	good
3	5	good

Fig 4.20

Query 9. Display all the houses booked in 2019.

Q 9: SELECT h_id as house_id from booking where booking_date like '2019%';

OUTPUT (Fig 4.21)

+ Options

house_id
1
5
4
5

Fig 4.21

Query 10. Display all the houses booked in 2019.

Q 10 : SELECT h_id as house_id from booking where booking_date like '2019%';

OUTPUT (Fig 4.22)

+ Options

tenant_id	house_id	price
3	2	8000

Fig 4.22

4.23 Triggers:

In a **DBMS**, a **trigger** is a **SQL** procedure that initiates an action (i.e., fires an action) when an event (INSERT, DELETE or UPDATE) occurs. Since **triggers** are event-driven specialized procedures, they are stored in and managed by the **DBMS**. ... Each **trigger** is attached to a single, specified table in the database. Here log table has been used which will store the trigger actions and details. The trigger is applied on house table. There are 3 triggers used namely insert,delete and update triggers.

Insert

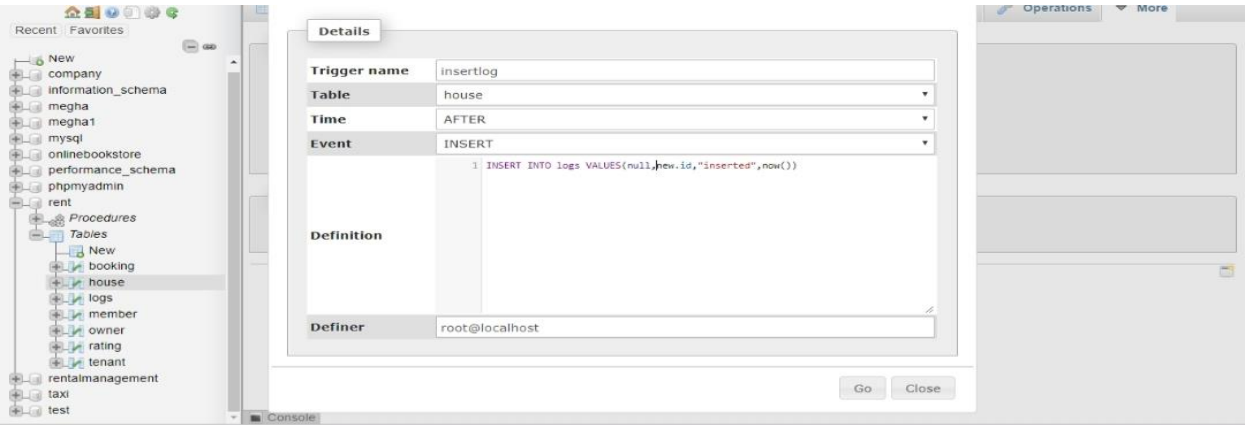


Fig 4.23

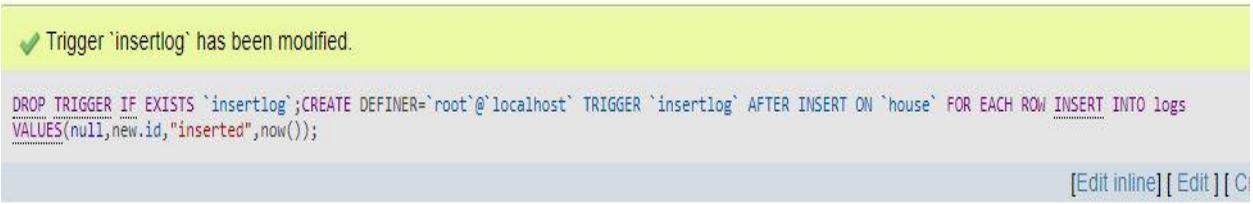


Fig 4.24

Delete

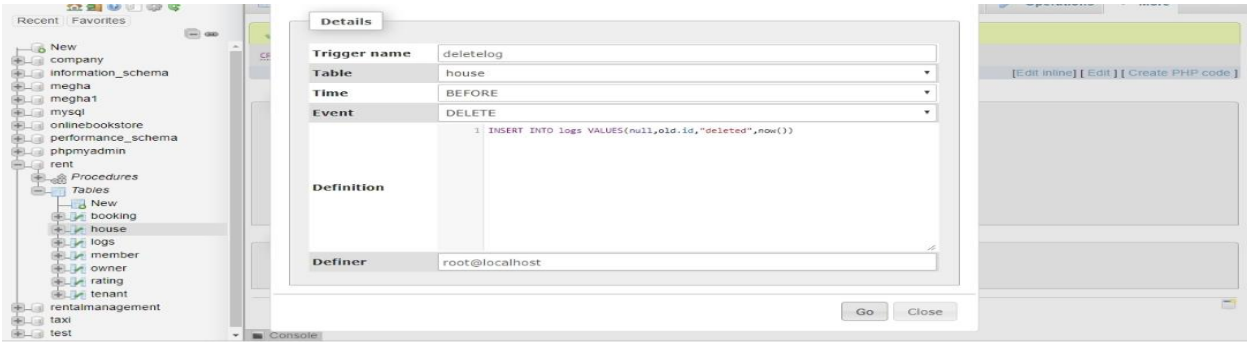


Fig 4.25

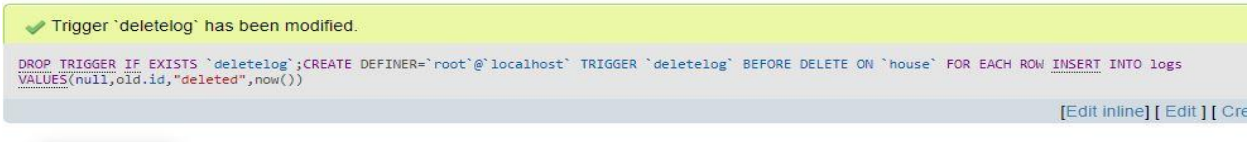


Fig 4.26

Update

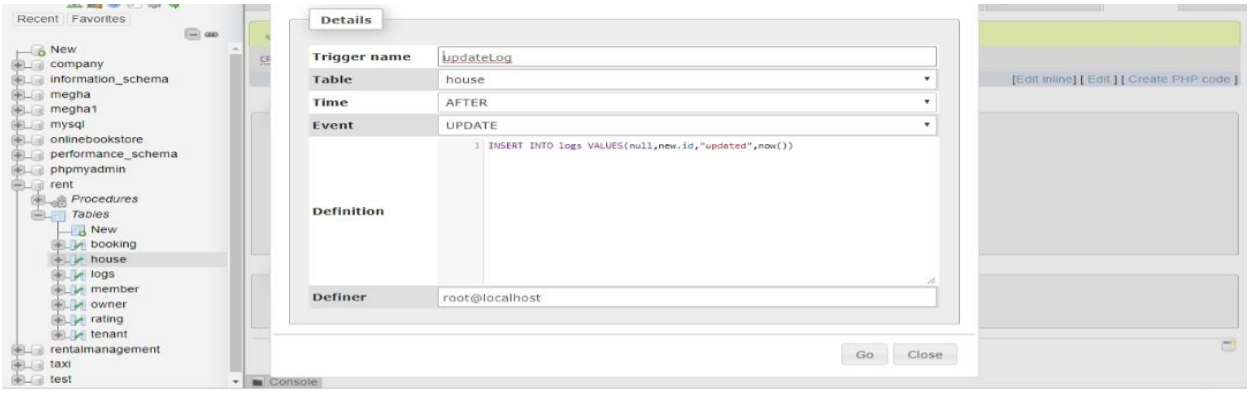


Fig 4.27

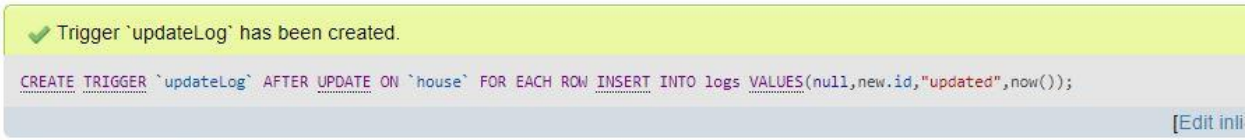


Fig 4.28

Log Table

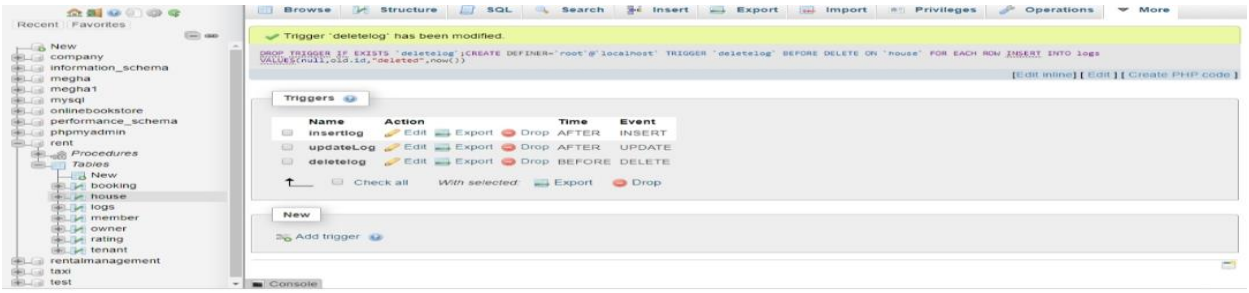


Fig 4.29

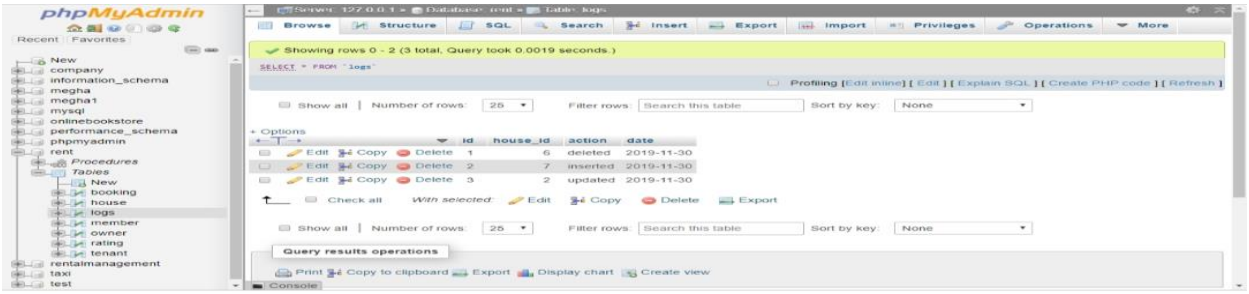


Fig 4.30

4.30 Stored procedure:

A **stored procedure** is a set of Structured Query Language (SQL) statements with an assigned name, which are **stored** in a relational **database management** system as a group, so it can be reused and shared by multiple program A **stored procedure** is a set of Structured Query Language (SQL) statements with an assigned name, which are **stored** in a relational **database management** system as a group, so it can be reused and shared by multiple program.This stored procedure is for booking and will display booking date from booking.

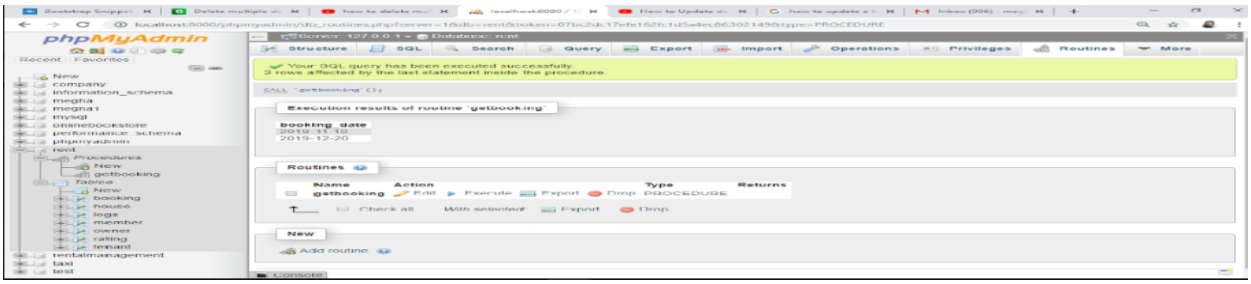


Fig 4.31

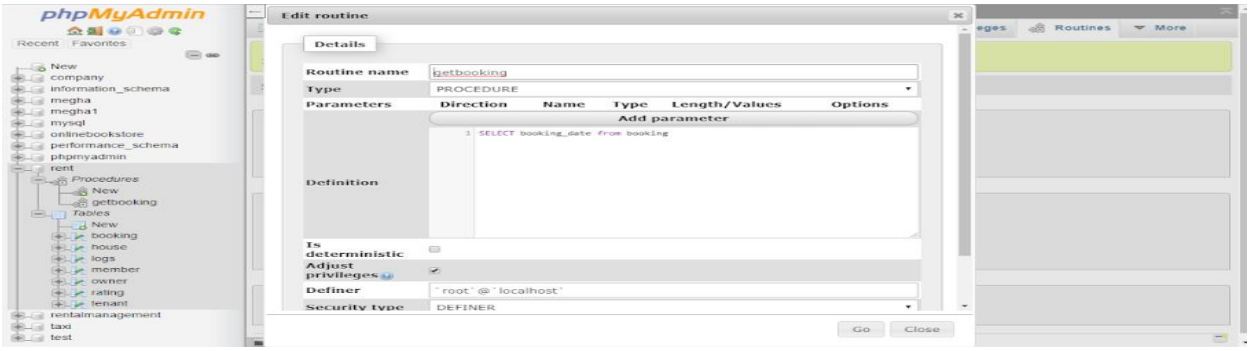


Fig 4.32

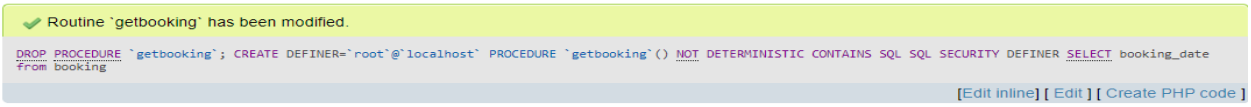


Fig 4.33

CHAPTER 5

RESULTS AND SNAPSHOT

1. Home Page Before Log In:

This fig 5.1 is shown before log in as tenant or as owner.

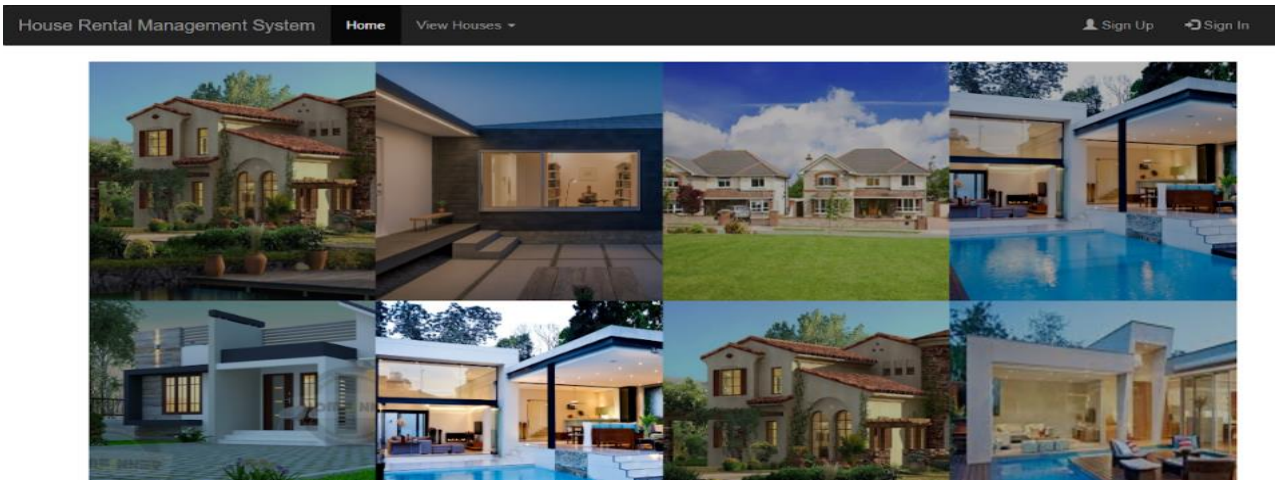


Fig 5.1 Home Page Before Log In

2. Sign up/Sign in page

This Fig 5.2 & Fig 5.3 is a sign up/in page for tenant or owner.

Fig 5.2 Sign up/sign in page for tenant

Fig 5.3 Sign up/sign in page for owner

1. Sign up as new tenant/owner:

This fig 5.4 & Fig 5.5 is sign up page for tenant and owner..

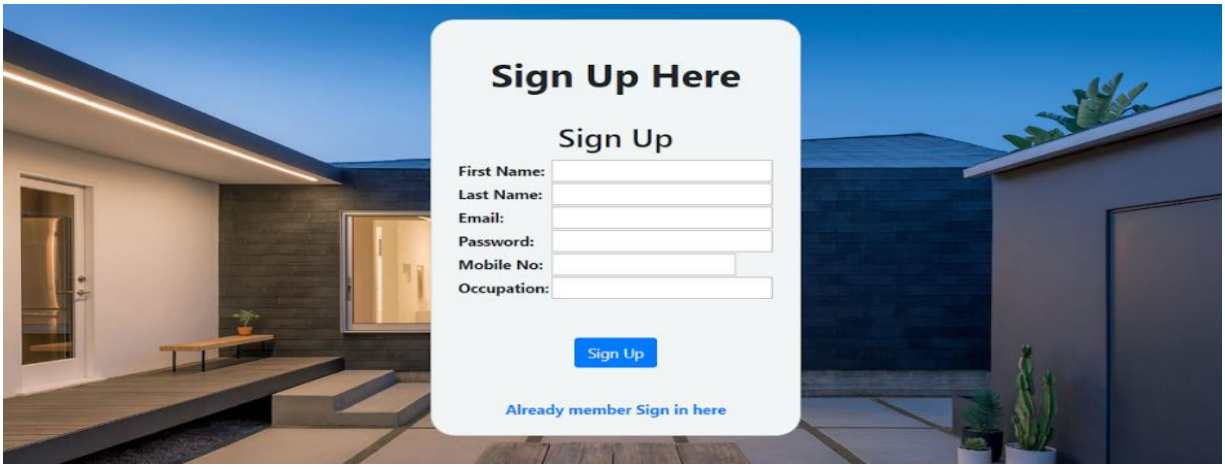


Fig 5.4 Sign up page of new tenant

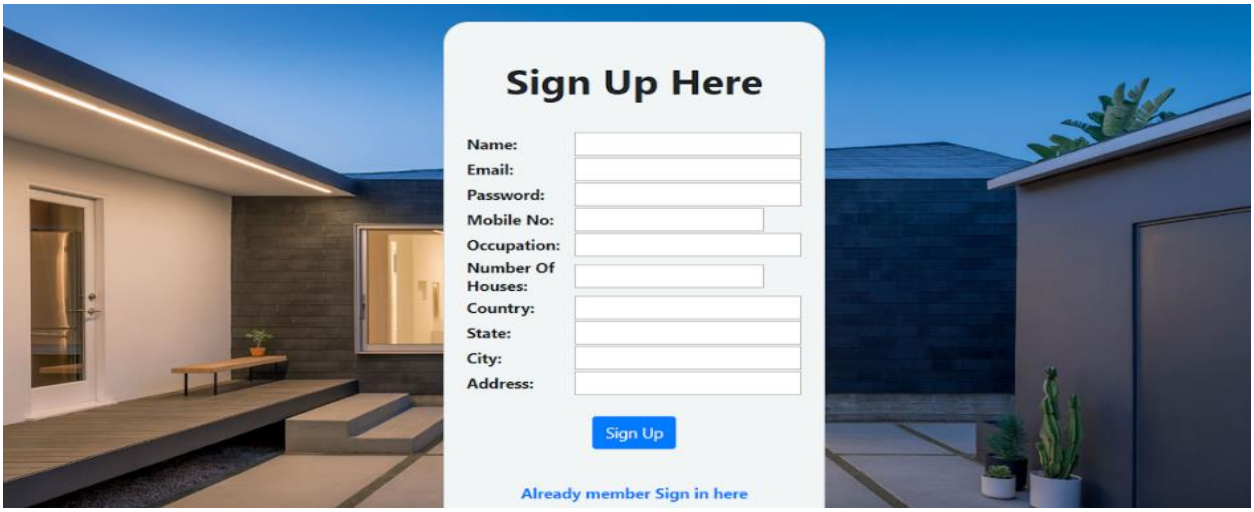


Fig 5.5 Sign up page of new owner

3. Home page after sign in/sign up

This fig 5.6 is after sign in or sign up.

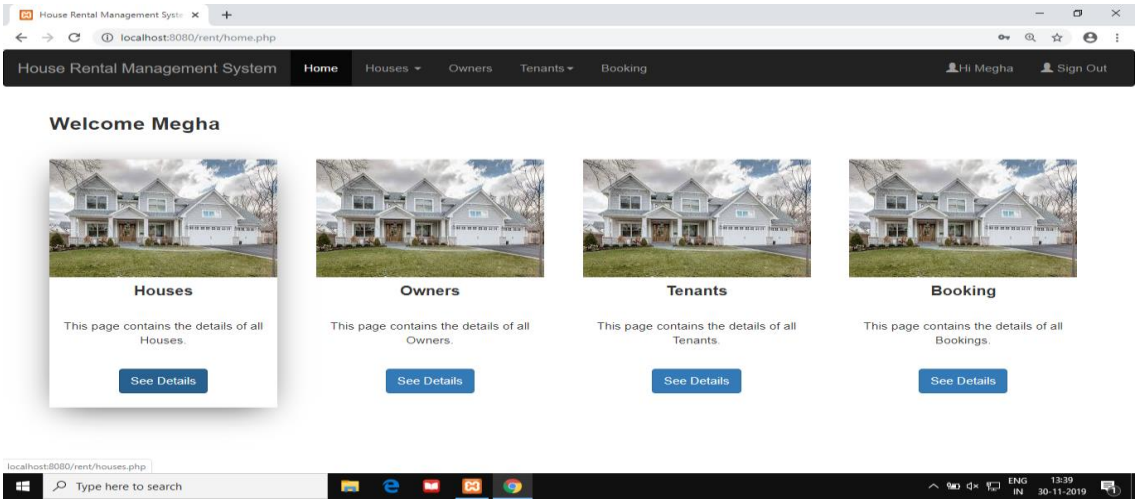


Fig 5.6 Home page after sign in/sign up

4. Houses page for the owner

This fig 5.7 will be displayed when the owner clicks on house tab.

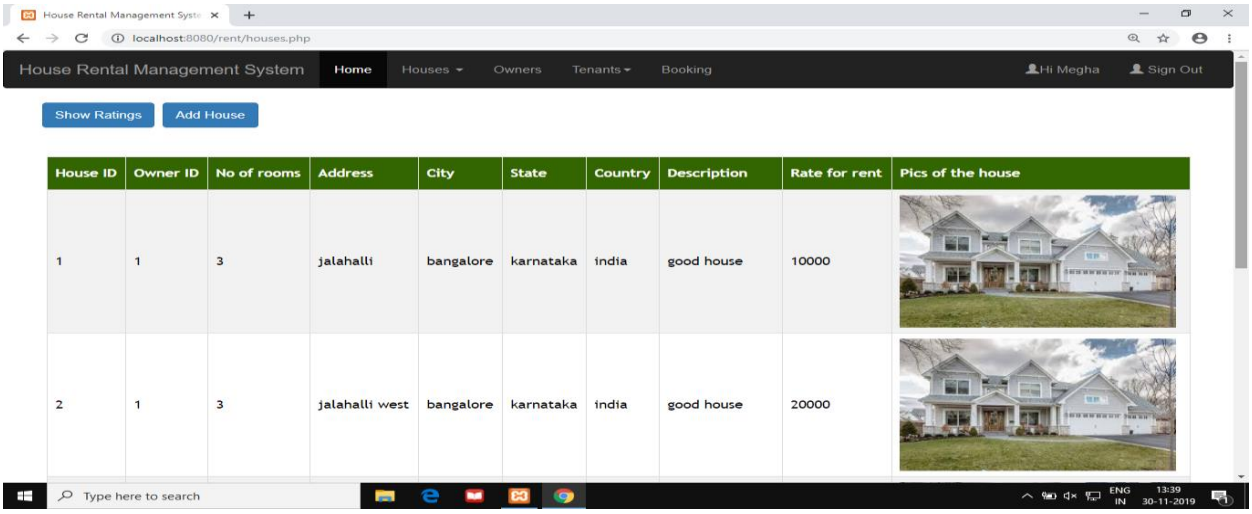


Fig 5.7 Houses page for the owner

5. Add house

This fig 5.8 is to add house by the owner**.**

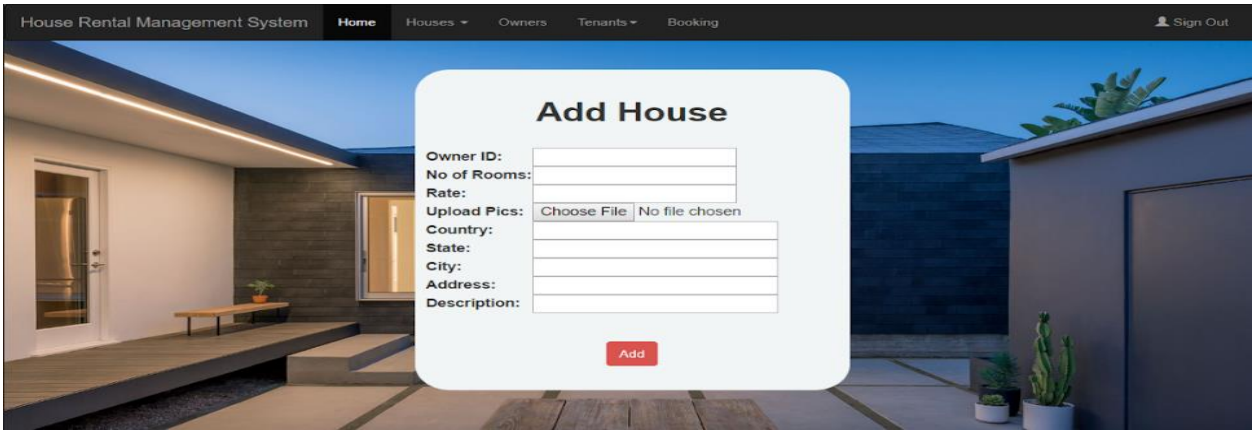


Fig 5.8 Add house

6. Add rating

This fig 5.9 is used to add the ratings of the house.

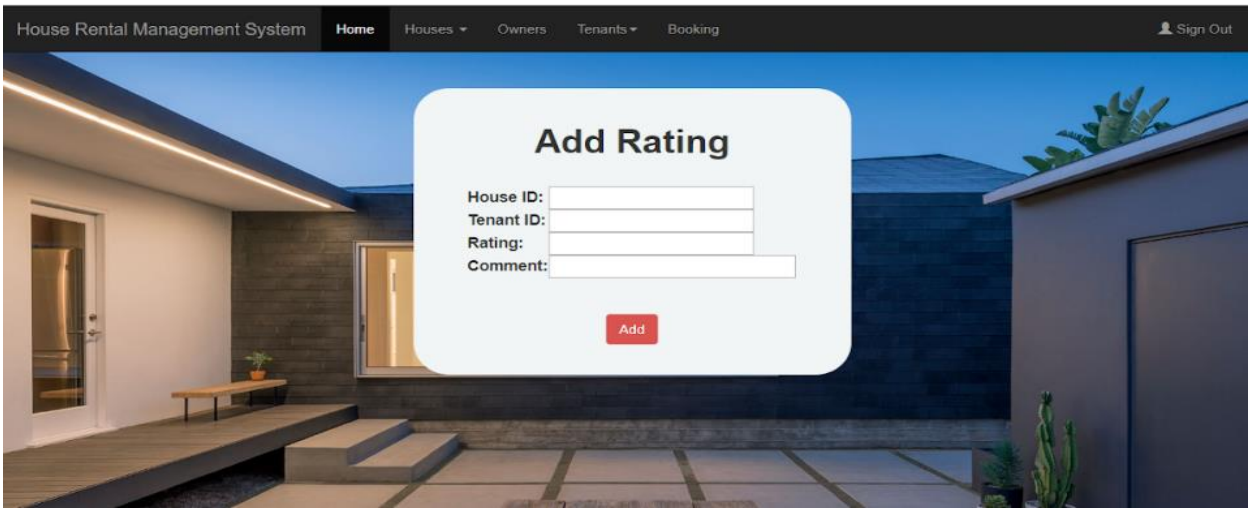
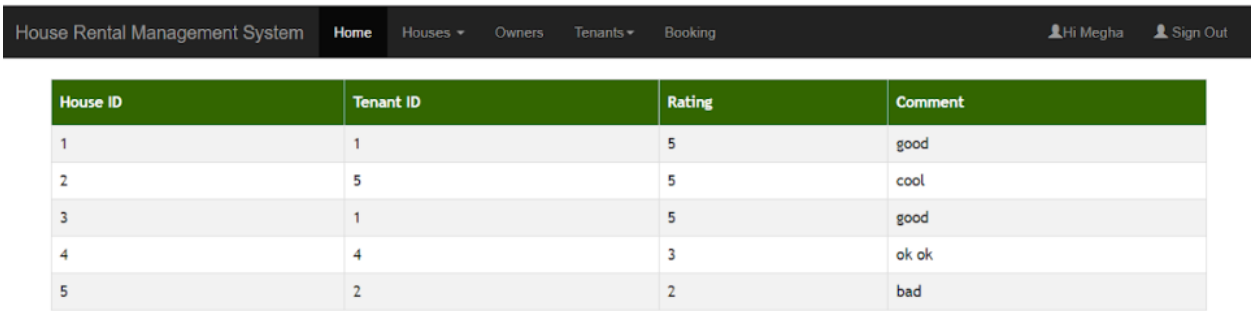


Fig 5.9 To show rating

7. To show rating

This fig 5.10 is used to display rating of the houses.

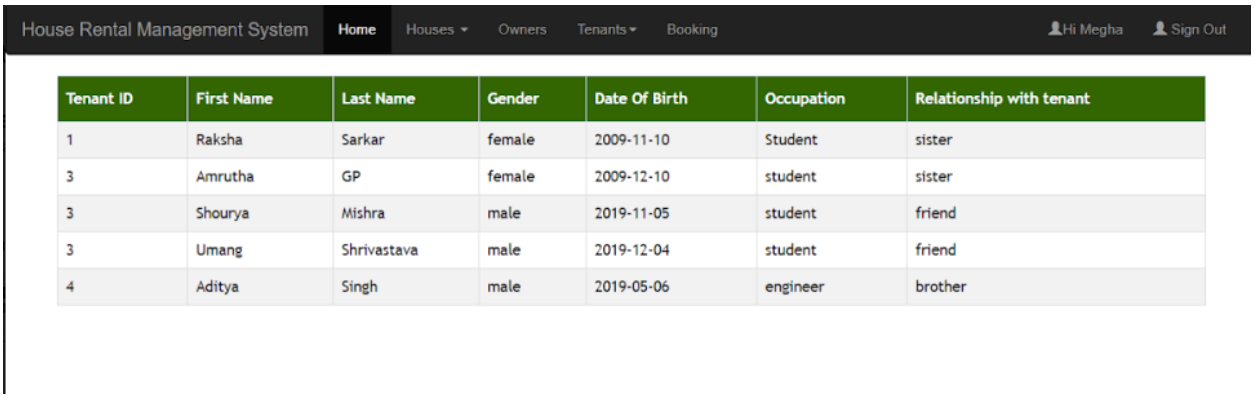


House ID	Tenant ID	Rating	Comment
1	1	5	good
2	5	5	cool
3	1	5	good
4	4	3	ok ok
5	2	2	bad

Fig 5.10 To show rating

8. Show tenant

This fig 5.11 will display the tenant details**.**



Tenant ID	First Name	Last Name	Gender	Date Of Birth	Occupation	Relationship with tenant
1	Raksha	Sarkar	female	2009-11-10	Student	sister
3	Amrutha	GP	female	2009-12-10	student	sister
3	Shourya	Mishra	male	2019-11-05	student	friend
3	Umang	Shrivastava	male	2019-12-04	student	friend
4	Aditya	Singh	male	2019-05-06	engineer	brother

Fig 5.11 Show tenant

9. Add Member

This fig 5.12 will add the member which is done by the tenant.

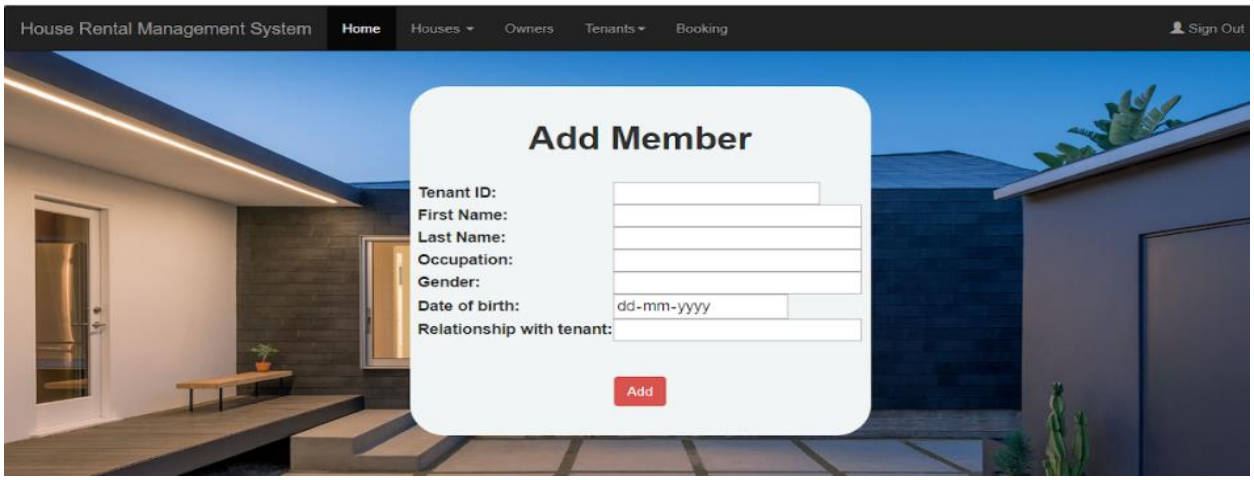


Fig 5.12 Add Member

10. Book house

This fig 5.13 will book the house if any tenant want to book it.

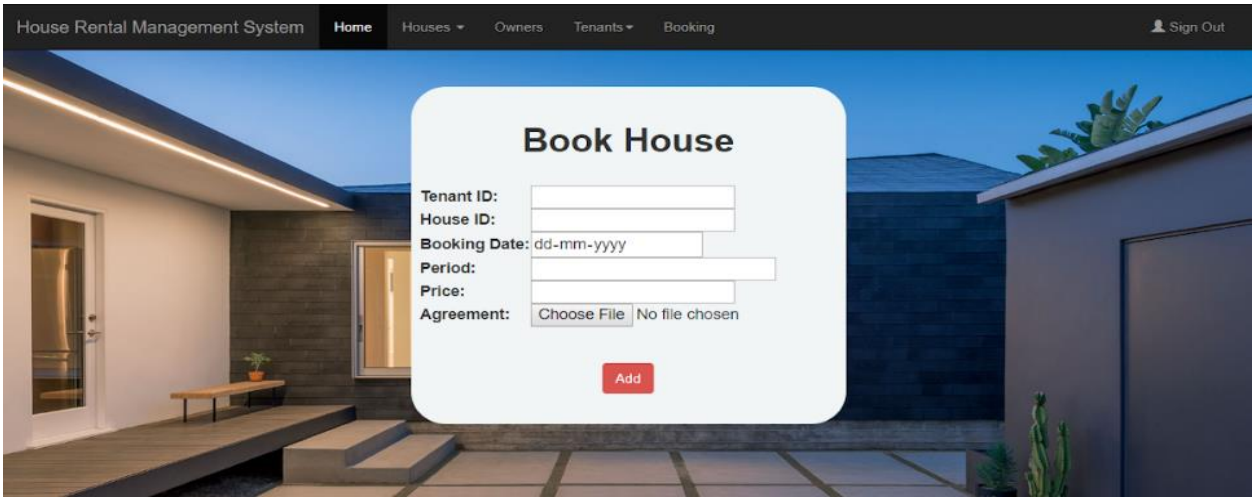
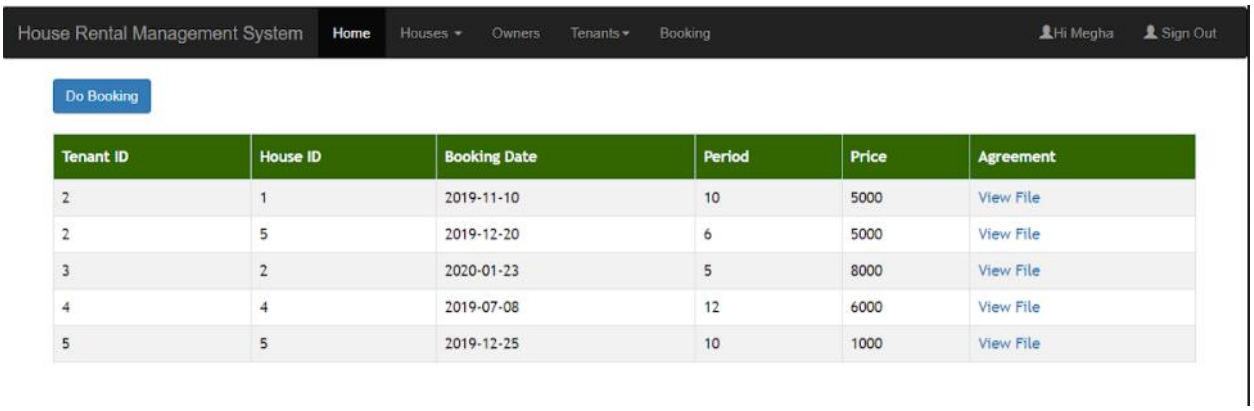


Fig 5.13 Book house

11. Booking By Tenant/owner

This fig 5.12 will displayed tenant want to book a house.

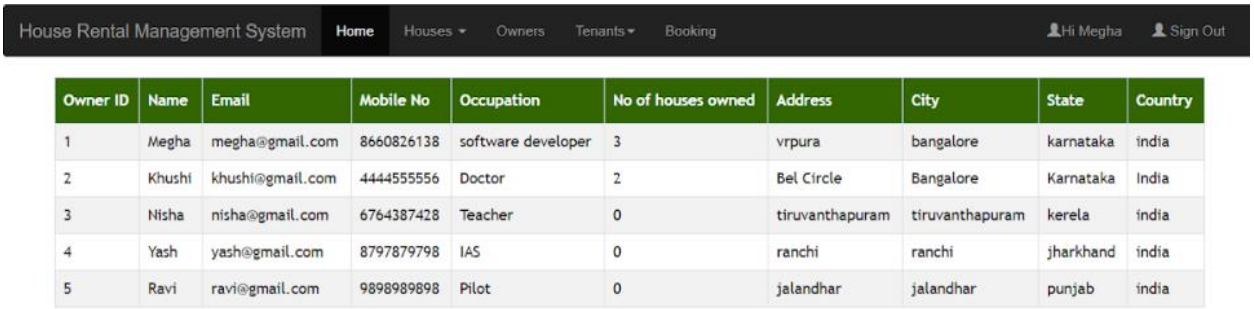


Tenant ID	House ID	Booking Date	Period	Price	Agreement
2	1	2019-11-10	10	5000	View File
2	5	2019-12-20	6	5000	View File
3	2	2020-01-23	5	8000	View File
4	4	2019-07-08	12	6000	View File
5	5	2019-12-25	10	1000	View File

Fig 5.14 Booking By Tenant

12. Show owner details

This fig 5.15 will displayed owner details.

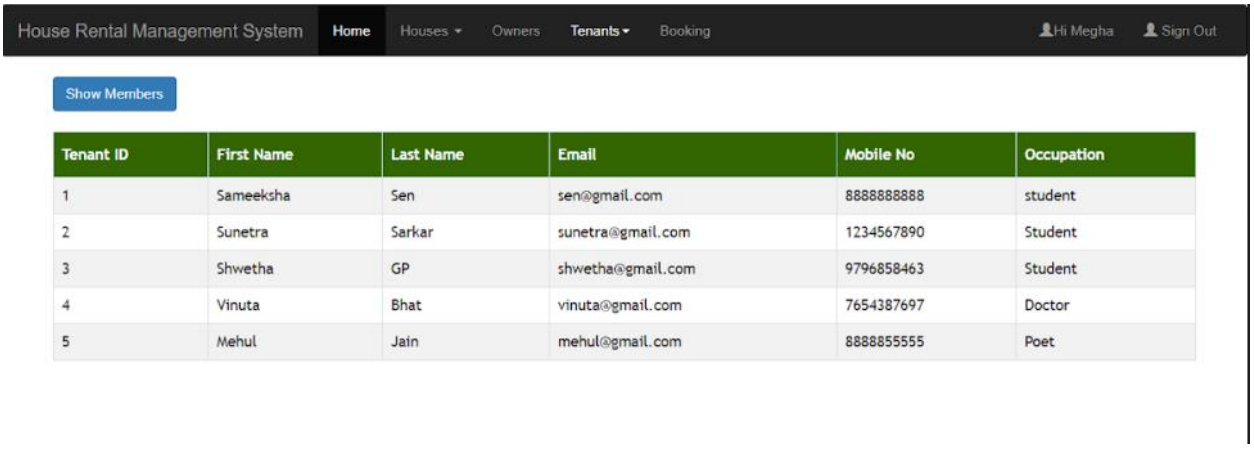


Owner ID	Name	Email	Mobile No	Occupation	No of houses owned	Address	City	State	Country
1	Megha	megha@gmail.com	8660826138	software developer	3	vrpura	bangalore	karnataka	india
2	Khushi	khushi@gmail.com	4444555556	Doctor	2	Bel Circle	Bangalore	Karnataka	India
3	Nisha	nisha@gmail.com	6764387428	Teacher	0	tiruvanthapuram	tiruvanthapuram	kerela	india
4	Yash	yash@gmail.com	8797879798	IAS	0	ranchi	ranchi	jharkhand	india
5	Ravi	ravi@gmail.com	9898989898	Pilot	0	jalandhar	jalandhar	punjab	india

Fig 5.15 Show owner details

13. Show tenant details

This fig 5.16 will displayed tenant details.



Tenant ID	First Name	Last Name	Email	Mobile No	Occupation
1	Sameeksha	Sen	sen@gmail.com	8888888888	student
2	Sunetra	Sarkar	sunetra@gmail.com	1234567890	Student
3	Shwetha	GP	shwetha@gmail.com	9796858463	Student
4	Vinuta	Bhat	vinuta@gmail.com	7654387697	Doctor
5	Mehul	Jain	mehul@gmail.com	8888855555	Poet

Fig 5.16 Show tenant details

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENT

House Rental business has emerged with a new goodies compared to the past experience where every activity concerning House rental business is limited to a physical location only. Even though the physical location has not been totally eradicated; the nature of functions and how these functions are achieved has been reshaped by the power of internet. Nowadays, customers can reserve book/buy/sale House online, rent House online, and have the house contracted successfully without any sweat once the tenant is a registered member of the House Rental Management System. The web based House rental system has offered an advantage to both Tenants as well as Owners to efficiently and effectively manage the business and satisfies tenants' need at the click of a button.

Future enhancement

In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

- Module of online payment in new system.
- Google map in order to improve functionality.
- We advise people or company to include the above listed module in order to make this product full working software that can be implemented in a professional environment.

  I have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them.

BIBILOGRAPHY

BOOKS REFERRED

• Software Engineering - R.S. Pressman

• PHP for Dummies

• PHP Beginners Guide by McGraw-Hill Publication

• Fundamentals of database system- Ramesh Elmasri and Shamkant B. Navathe,7th Edition

WEBSITES REFERRED

- https://stackoverflow.com

  This is the link of stack overflow which is an open community for anyone that codes, It is help you get answers to your coding questions. This website was referred for all the problems faced.

- https://www.udemy.com

  This is the link of udemy where anyone can study any topic any time.

- https://w3schools.com

  This is the link of w3schools which is very useful to learn languages for building web pages and also you can learn here languages for accessing databases.

.