"RESIDENTIAL MANAGEMENT SYSTEM"

A PROJECTREPORT SUBMITTED TO

THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU

(An Autonomous Institute under VTU, Belagavi)

ESTD : 1946

In partial fulfillment of the requirements for Project work (Database Laboratory), fifth semester

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by

M Venkateshwar Reddy (4NI17CS032)

Mayur Hebbar T M (4NI17CS036)

Mukund N Acharya (4NI17CS043)

Under the Guidance of

Mr. Yogesh M J
Assistant Professor
Department of CSE
NIE, Mysuru-570008

Ms. Sindhu Devaki R S
Assistant Professor
Department of CSE
NIE, Mysuru-570008

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

Mysore-570008

2019-2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
THE NATIONAL INSTITUTE OF ENGINEERING

ESTD : 1946

CERTIFICATE

This is to certify that the project work entitled "**Residential Management System**" is a work carried out by **M Venkateshwar Reddy (4NI17CS032), Mayur Hebbar T M (4NI17CS036) and Mukund N Acharya (4NI17CS043)** in partial fulfillment for the project work (Database Laboratory), fifth semester, Computer Science & Engineering, The National Institute of Engineering **(**Autonomous Institution under Visvesvaraya Technological University, Belagavi) during the academic year 2019-2020. It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The project work report has been approved in partial fulfillment as per academic regulations of The National Institute of Engineering, Mysuru.

Internal Guides

Mr. Yogesh M J
Assistant Professor

Ms. Sindhu Devaki R S
Assistant Professor

Professor & Head

Dr. V K Annapurna
Professor and Head
Dept. of CS&E
NIE, Mysuru

Examiners

Name:
Designation:

Name:
Designation:

ACKNOWLEDGEMENTS

We would like to take this to express out profound gratitude to all those people who were directly or indirectly involved in the completion of this project. We thank each and every one who encouraged us in every possible way.

We would like to thank our principal **Dr. G Ravi** for letting us to be the part of this prestigious institution and letting us to explore to the fullest.

We would like to extend our sincere gratitude to **Dr. V K Annapurna**, **Head of Department, CSE**, for being the source of inspiration and instilling an enthusiastic spirit in us throughout the process of project making.

We would like to express our heartfelt gratitude to our project guides **Mr. Yogesh M J** and **Ms**. **Sindhu Devaki R S**, for their constant guidance, valuable knowledge and experience.

We are thankful to our parents, family and friends for their constant support, inspiration and encouragement, for their helping hand and all those who supported us directly or indirectly in our academic process.

Thank you

M Venkateshwar Reddy (4NI17CS032)

Mayur Hebbar T M (4NI17CS036)

Mukund N Acharya (4NI17CS043)

ABSTRACT

The Backyard is a Residential Management application aimed to achieve some basic features for residents, tenants and colony administration in one place which will increase efficiency and transpirency in all aspects of community management for a colony, flats or gated community.

This project includes features such as:

1. **Bill Payment:** Easy payment options of maintenance and utility dues and bills from the apllication.
2. **Facility and Activities:** Book and pay for facilities like Swimming Pool, Club House and Gym and participate in many activities.
3. **Effective communication between Residents and Admins:** Community communication for residents such as problem addressing system, helpdesk. The various Contact details are also available to the user as directory.
4. **Visitor Management:** Manage visitors entering and leaving the Society by the Admin.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1:

INTRODUCTION

The main objective of this application is to help the residents to manage the operations and activities of society. This is a simple user interface developed using java. It helps the user to keep track of all the Bills, facilities, problem status and maintenance related information. It helps in reducing the manual work which was earlier required for maintaining these records. Admin can manage bills, problems and also can view tenants utilising the facilities and visitors coming in and out. This application allows effective communication between admin and user through help disk section.

Both user and admin can login using their credentials which will direct them to the home page where the other information needed will be in the form of buttons. The modification of account settings is restricted to only authorized user, making it safe and secure. All the data inserted or deleted is shown in the database immediately making it a dynamic approach.

There are three types of facilities namely clubhouse, swimming pool, gym. User can choose the plans and batch according to their convenience, plans include 1 month, half year, year and for this duration he can discern the estimated amount to be paid.

The application provides all the details of visitors who visits the residents to admin. Admin can see the information of visitors like name, date of moving in and out etc. Application also provides the facility like help desk in which user can describe their problems in the message box and admin will give the solution for that problem in shorter period of time. Admin can generate and print all the bills payment of all the users.

There will be a robust database which stores all the data related to the residents of the society, their maintenance related information etc. It helps to keep track of all the payments done by users towards maintenance, bills and facilities acquired. This tool will ease the admin to manage huge data of different users who own falt. For the maintenance part of the interface it shall have the data of all the maintenance charges paid by the users, whether it is done monthly or not.

Chapter 2:

SYSTEM ANALYSIS

2.1:EXISTING AND PROPOSED SYSTEM

The existing records were manually managed by the resources of the society. It was a huge task as they had to maintain a large database of customers who have bought the flats along with their dates, amount paid etc. They also had to manage the data related to the maintenance activities of all the users. It required lot of time and resources. There was no way to find easily when the user had to pay the installment for maintenance, bills etc details. The admin had to manually search for all the records pertaining to each flat and then prepare a report of all the outstanding payments. Admin had to manage all the details of the users manually which was very difficult to maintain. There is no sit and doing the work, user have to go and pay the bills. Lot of energy and time being lost by doing it manually.

Every year admin is generating the bills for all the users and it is very difficult to manage the records of each and every flat in manual system. It will not only take a a lot of time but it is also increase the chance of errors. Sometimes even after repeated cross checks errors are found which lead to wrong calculation of accounts and balance sheets. All these problems leads to the rise of an alternative option.

This is a simple user interface developed using java. The interface helps the user to login through the system and add the details regarding the flat. The interface takes as input the name of the user, Phone number, address, the flat number. These details are stored in the database. If the user is paid the bills, then these details are stored in the bill table. It will have a unique flat number generated for each of the flats. The interface also has maintenance panel. This panel holds information such as name of the user, the maintenance charges to be paid, quarterly or annually, payment etc. These details are stored in the bills tables in the database. The user can query on the database and fetch this information very easily. This interface helps the user to make all these payments regarding maintenance and facilities etc.

This application will reduce manual work and maintain updates in database from time to time. It is easy handle works related many flats at a time without any confusion. Data is well secured and easy to retrieve old records in a shorter time.

2.2:SYSTEM REQUIREMENTS

Hardware requirements

- OS-Windows
- RAM 512 MB
- 20 GB Hard disk
- Key board
- Mouse

Software configuration

- Front end: Java, java swing
- Operating system: Windows 10
- Back end: MySQL

Front End

We have implemented Java Swing for all the Client side validations. Client side java swing is designed to reside inside java & ensure they run properly. It is object based, event driven, platform independent. These are important parts of any Web application to implement Client side Validations and the invalid data is not submitted. The form is not submitted until user fills in correct data. It is extremely useful to restrict mistakes by user.

Back End

- We have used MySQL. MySQL provides efficient/effective solution for major database tech.
- Large database and space management.
- Many concurrent database users.
- High transaction processing requirement
- High Availability
- Industry accepted standards

Chapter 3:

SYSTEM DESIGN

3.1:TOOLS USED

MySQL:

MySQL is an open source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality. MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, Simple Machines Forum, phpBB, MyBB, and Drupal. MySQL is also used in many high-profile, large scale websites, including Google (though not for searches), Facebook, Twitter, Flickr, and YouTube Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well. It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP-based web application or as a standalone database server.

JAVA:

Java is a general-purpose programming language that is, class-based, object-oriented and designed to have as few implementation dependencies as possible. It is intended to let application developers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them. Programs written in Java have a reputation for being slower and requiring more memory than those written in C++. However, Java programs execution speed improved significantly with the introduction of just-in-time compilation. Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use.

JAVA SWING:

Swing is a GUI toolkit for Java. It is one part of the Java Foundation Classes (JFC). Swing includes graphical user interface (GUI) widgets such as text boxes, buttons, split-panes, and tables. Swing widgets provide more sophisticated GUI components than the earlier Abstract Window Toolkit. Since they are written in pure Java, they run the same on all platforms, unlike the AWT which is tied to the underlying platform's windowing system. Swing supports pluggable look and feel – not by using the native platform's facilities, but by roughly emulating them. This means you can get any supported look and feel on any platform. The disadvantage of lightweight components is slower execution. The advantage is uniform behavior on all platforms. Swing is a set of classes that provides more powerful and flexible functionality that is possible with standard and advanced AWT (Abstract Window Toolkit) components. These are not implemented by platform specific code. Instead they are written entirely in Java, therefore, they are platform independent. Fundamentals of swing is the JApplet class that extends the Applet class. Applets that use Swing must be subclasses of the Japplet class.

3.2:ENTITY RELATIONSHIP DIAGRAM

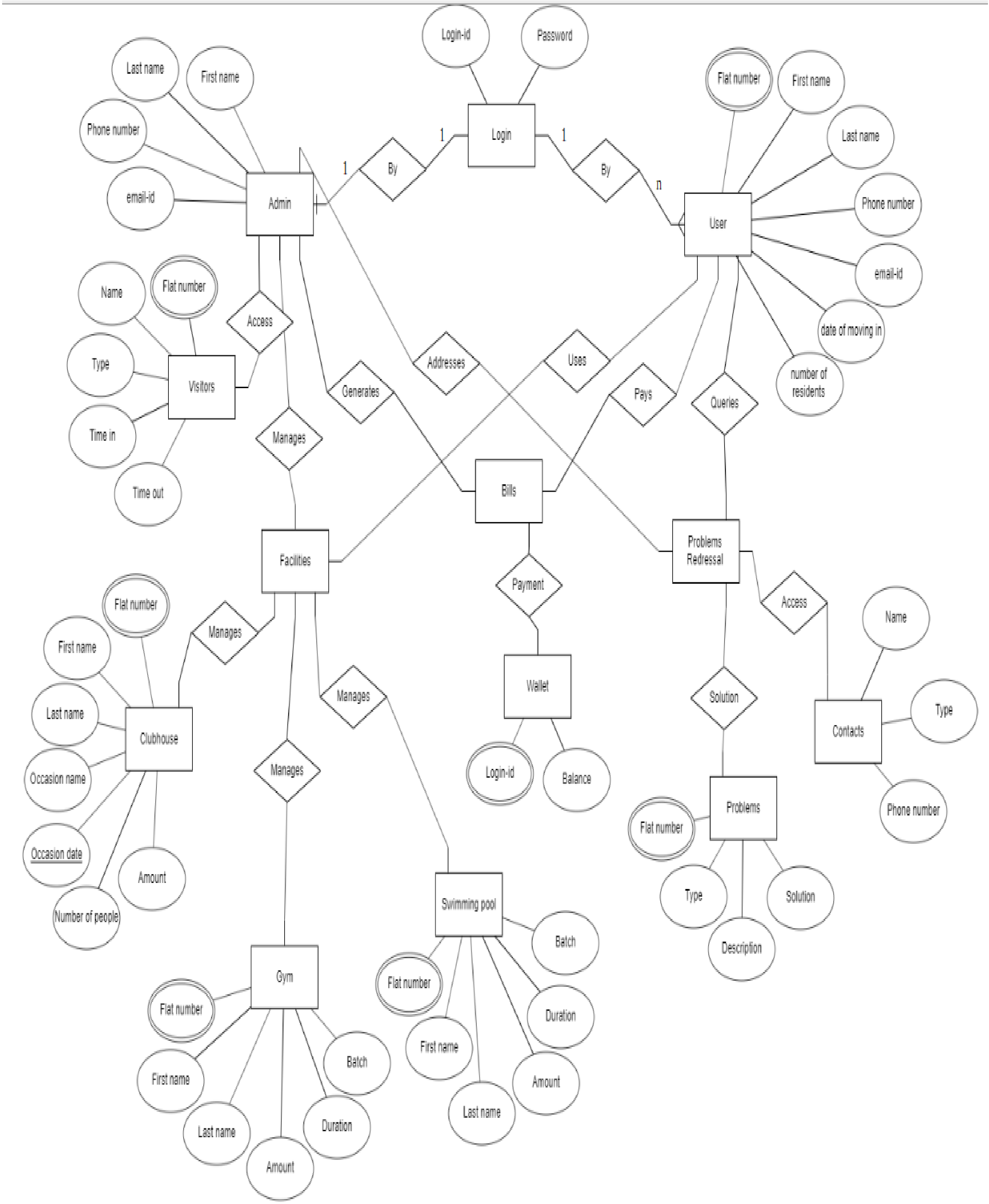


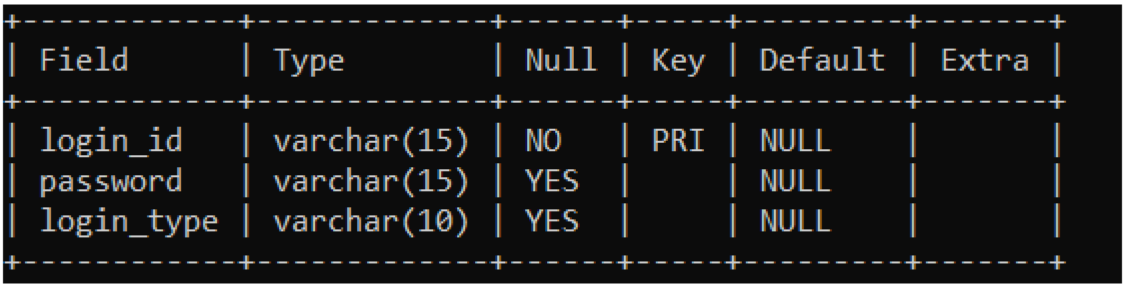
Fig-3.1:ER diagram

Chapter 4:

IMPLEMENTATION

4.1:TABLES USED IN DATABASE

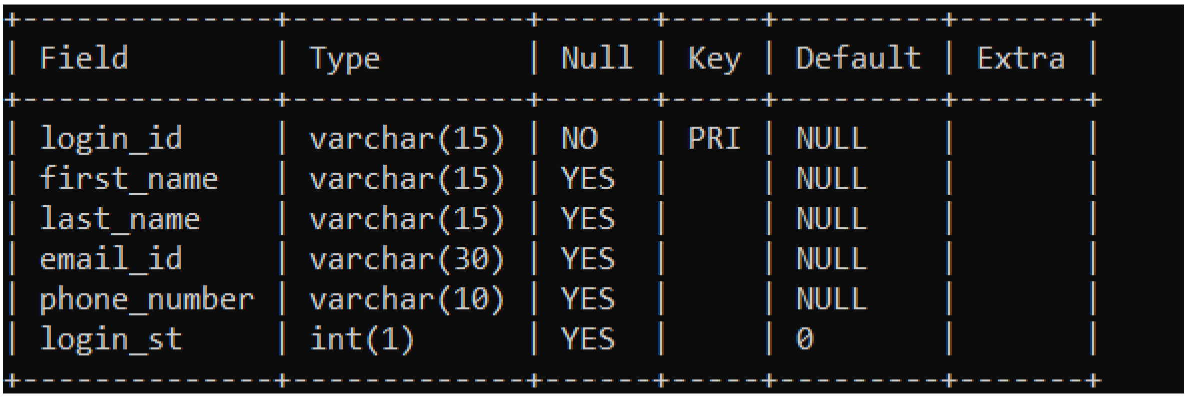
1. login



```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| login_id   | varchar(15) | NO   | PRI | NULL    |       |
| password   | varchar(15) | YES  |     | NULL    |       |
| login_type | varchar(10) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

The login table stores the login credential of all the users and admin. Login-id is primary key which is unique to each user and admin.

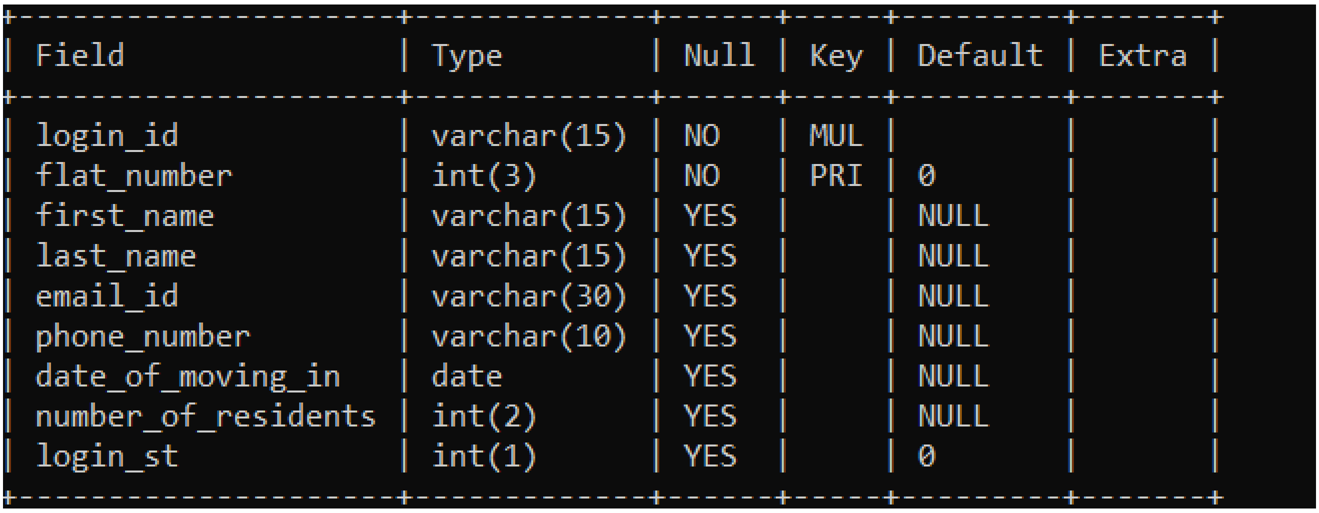
2. admin



```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| login_id     | varchar(15) | NO   | PRI | NULL    |       |
| first_name   | varchar(15) | YES  |     | NULL    |       |
| last_name    | varchar(15) | YES  |     | NULL    |       |
| email_id     | varchar(30) | YES  |     | NULL    |       |
| phone_number | varchar(10) | YES  |     | NULL    |       |
| login_st     | int(1)      | YES  |     | 0       |       |
+--------------+-------------+------+-----+---------+-------+
```

The admin table contains all the details of the admin. It includes login-id, first name, last name, email-id, phone number. Login-id is made as primary key in this.

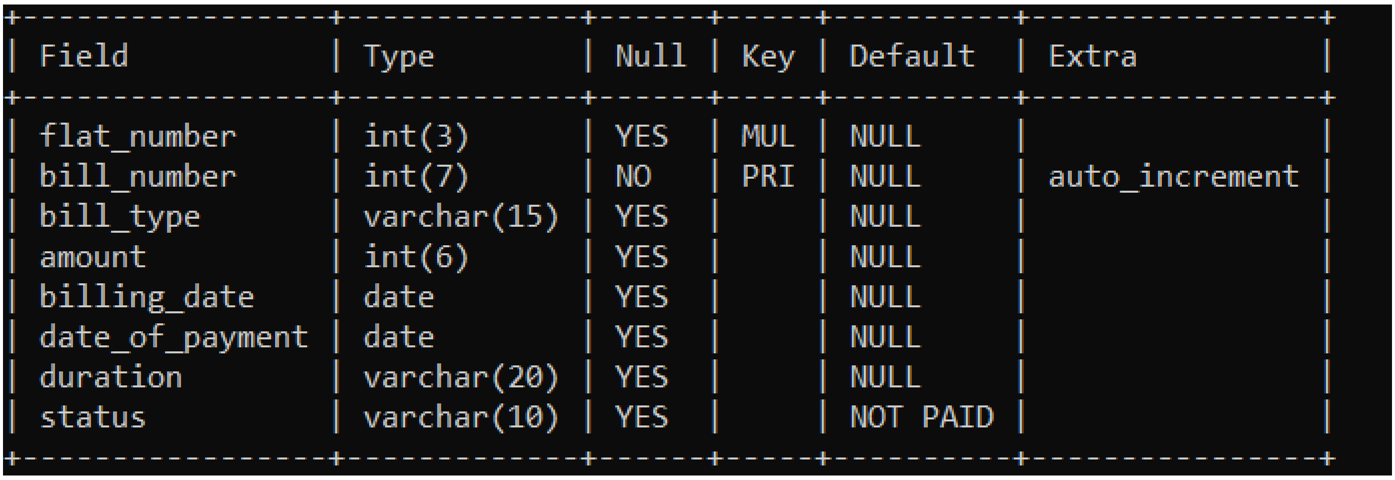
3. user



```
+---------------------+-------------+------+-----+---------+-------+
| Field               | Type        | Null | Key | Default | Extra |
+---------------------+-------------+------+-----+---------+-------+
| login_id            | varchar(15) | NO   | MUL |         |       |
| flat_number         | int(3)      | NO   | PRI | 0       |       |
| first_name          | varchar(15) | YES  |     | NULL    |       |
| last_name           | varchar(15) | YES  |     | NULL    |       |
| email_id            | varchar(30) | YES  |     | NULL    |       |
| phone_number        | varchar(10) | YES  |     | NULL    |       |
| date_of_moving_in   | date        | YES  |     | NULL    |       |
| number_of_residents | int(2)      | YES  |     | NULL    |       |
| login_st            | int(1)      | YES  |     | 0       |       |
+---------------------+-------------+------+-----+---------+-------+
```

The user table contains all the details of the users. It includes login-id, flat number, first name, last name, email-id, phone number, date of moving in, number of residents.

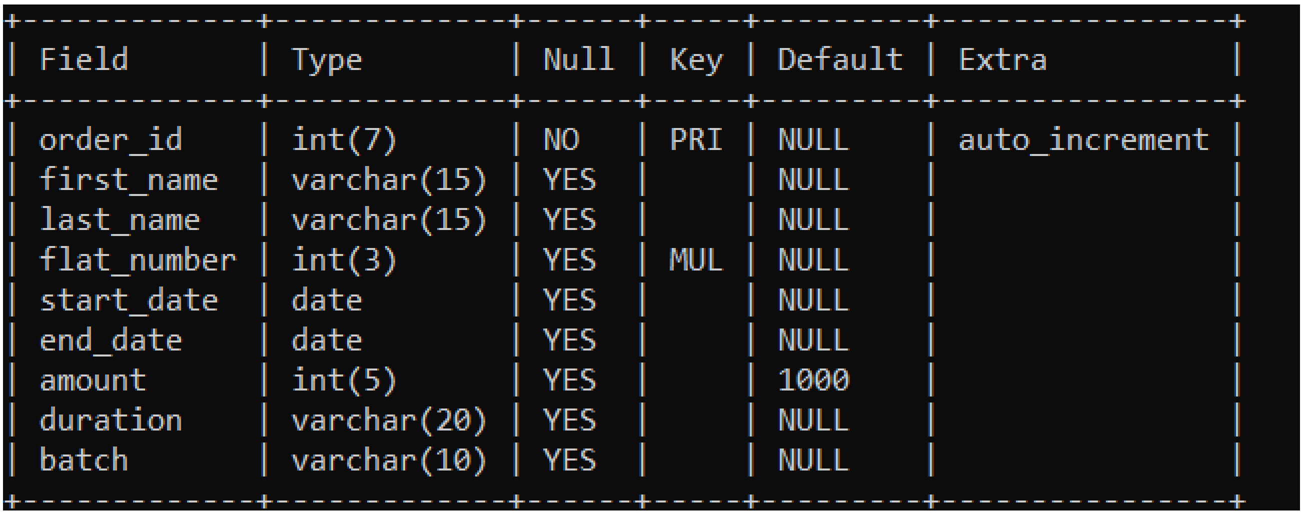
4. bills



```
+-----------------+-------------+------+-----+----------+----------------+
| Field           | Type        | Null | Key | Default  | Extra          |
+-----------------+-------------+------+-----+----------+----------------+
| flat_number     | int(3)      | YES  | MUL | NULL     |                |
| bill_number     | int(7)      | NO   | PRI | NULL     | auto_increment |
| bill_type       | varchar(15) | YES  |     | NULL     |                |
| amount          | int(6)      | YES  |     | NULL     |                |
| billing_date    | date        | YES  |     | NULL     |                |
| date_of_payment | date        | YES  |     | NULL     |                |
| duration        | varchar(20) | YES  |     | NULL     |                |
| status          | varchar(10) | YES  |     | NOT PAID |                |
+-----------------+-------------+------+-----+----------+----------------+
```

The bills table contains all the details of bills along with the status whether the bill is paid or not paid. The bill number is auto generated in this table which can be used for future reference.

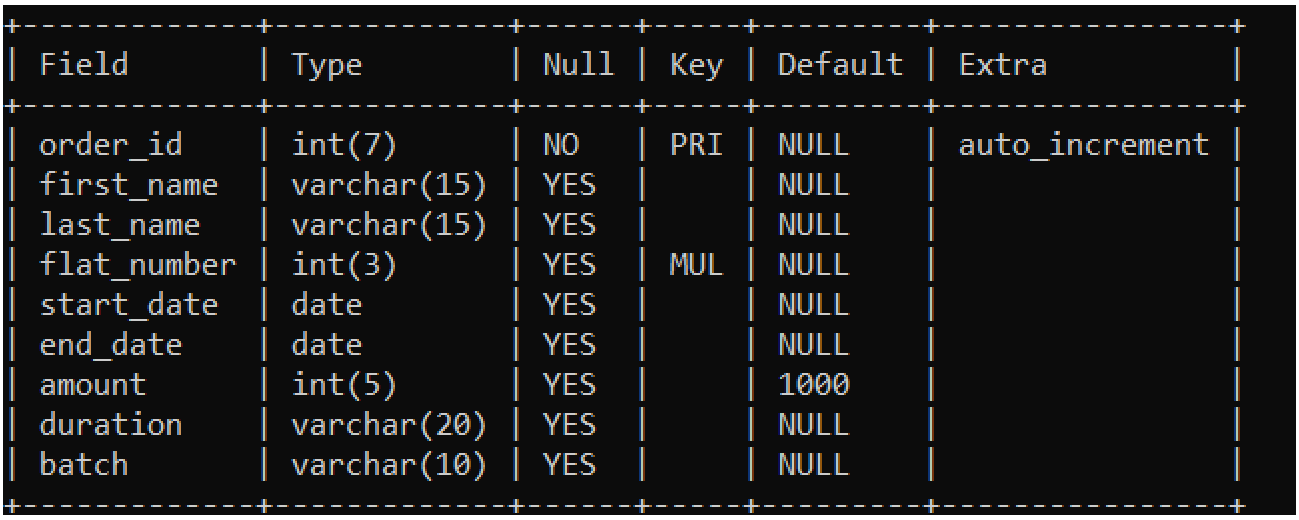
5. swimming



```
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| order_id    | int(7)      | NO   | PRI | NULL    | auto_increment |
| first_name  | varchar(15) | YES  |     | NULL    |                |
| last_name   | varchar(15) | YES  |     | NULL    |                |
| flat_number | int(3)      | YES  | MUL | NULL    |                |
| start_date  | date        | YES  |     | NULL    |                |
| end_date    | date        | YES  |     | NULL    |                |
| amount      | int(5)      | YES  |     | 1000    |                |
| duration    | varchar(20) | YES  |     | NULL    |                |
| batch       | varchar(10) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
```

The swimming table contains the details of users who applied for this facility. The order-id is auto generated which can be used for future references.

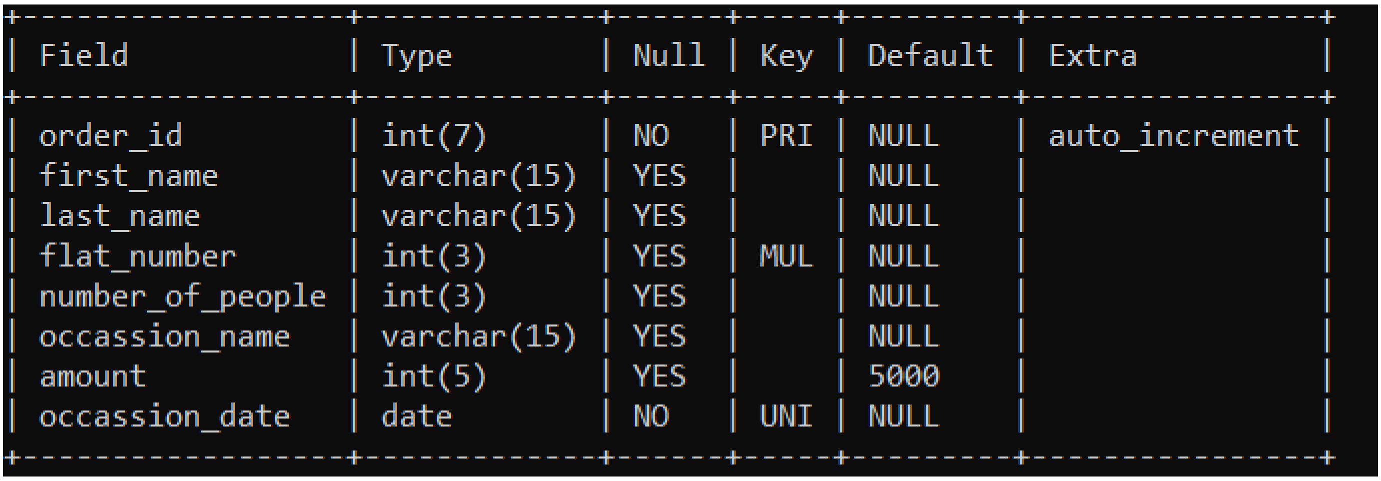
6. gym



```
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| order_id    | int(7)      | NO   | PRI | NULL    | auto_increment |
| first_name  | varchar(15) | YES  |     | NULL    |                |
| last_name   | varchar(15) | YES  |     | NULL    |                |
| flat_number | int(3)      | YES  | MUL | NULL    |                |
| start_date  | date        | YES  |     | NULL    |                |
| end_date    | date        | YES  |     | NULL    |                |
| amount      | int(5)      | YES  |     | 1000    |                |
| duration    | varchar(20) | YES  |     | NULL    |                |
| batch       | varchar(10) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
```

The gym table contains the details of users who applied for this facility. The order-id is auto generated which can be used for future references.

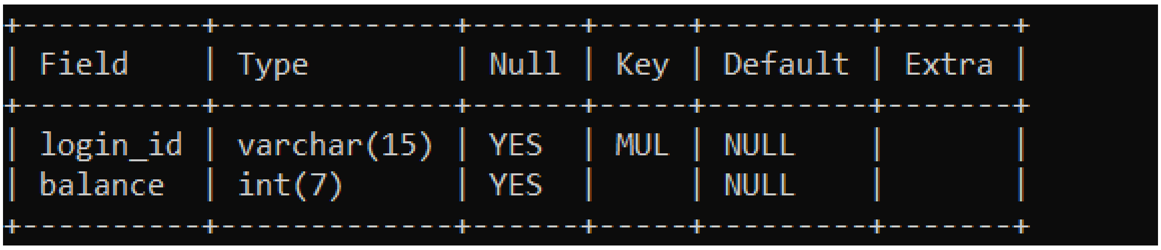
7. clubhouse



```
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| order_id         | int(7)      | NO   | PRI | NULL    | auto_increment |
| first_name       | varchar(15) | YES  |     | NULL    |                |
| last_name        | varchar(15) | YES  |     | NULL    |                |
| flat_number      | int(3)      | YES  | MUL | NULL    |                |
| number_of_people | int(3)      | YES  |     | NULL    |                |
| occassion_name   | varchar(15) | YES  |     | NULL    |                |
| amount           | int(5)      | YES  |     | 5000    |                |
| occassion_date   | date        | NO   | UNI | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
```

The clubhouse table contains the details of users who applied for this facility. The order-id is auto generated which can be used for future references.

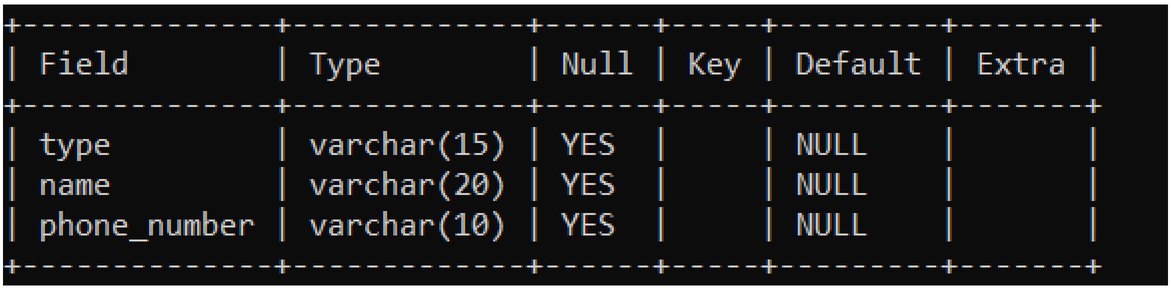
8. wallet



```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| login_id | varchar(15) | YES  | MUL | NULL    |       |
| balance  | int(7)      | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

The wallet contains balance amount of each user based on login-id. Contents of this table are used for the payment of bills.

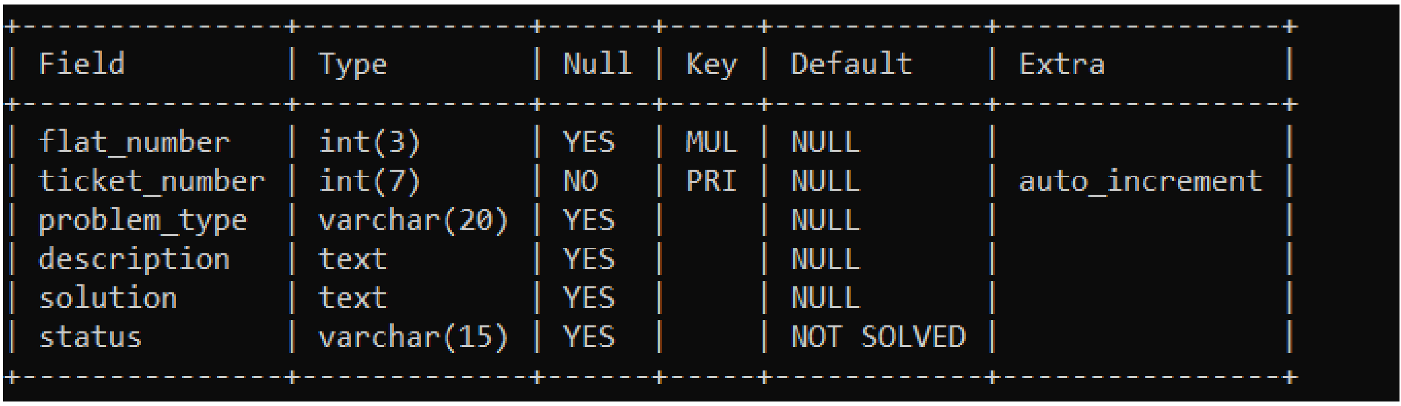
9. contacts



```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| type         | varchar(15) | YES  |     | NULL    |       |
| name         | varchar(20) | YES  |     | NULL    |       |
| phone_number | varchar(10) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

The contacts table contains contact details of the various necessary general services available.

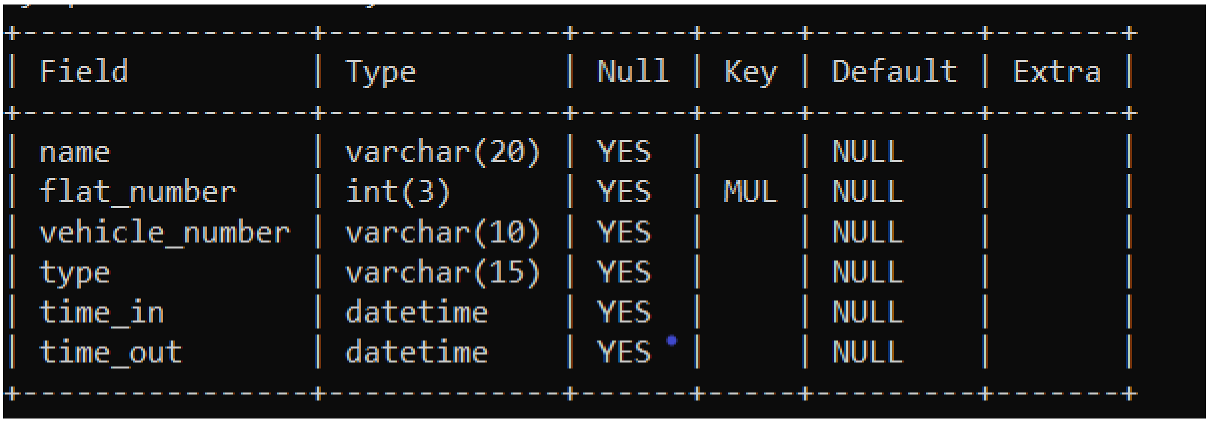
10. problems



```
+---------------+-------------+------+-----+------------+----------------+
| Field         | Type        | Null | Key | Default    | Extra          |
+---------------+-------------+------+-----+------------+----------------+
| flat_number   | int(3)      | YES  | MUL | NULL       |                |
| ticket_number | int(7)      | NO   | PRI | NULL       | auto_increment |
| problem_type  | varchar(20) | YES  |     | NULL       |                |
| description   | text        | YES  |     | NULL       |                |
| solution      | text        | YES  |     | NULL       |                |
| status        | varchar(15) | YES  |     | NOT SOLVED |                |
+---------------+-------------+------+-----+------------+----------------+
```

The problems table contain the column for the problem type, problem description and solution which is used by both user and admin. The ticket number is auto generated for future references.

11. visitors



```
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| name           | varchar(20) | YES  |     | NULL    |       |
| flat_number    | int(3)      | YES  | MUL | NULL    |       |
| vehicle_number | varchar(10) | YES  |     | NULL    |       |
| type           | varchar(15) | YES  |     | NULL    |       |
| time_in        | datetime    | YES  |     | NULL    |       |
| time_out       | datetime    | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

The visitors table contains the details of the people entering and leaving the society. It contains time in and time out details of the visitors.

4.2:CONNECTING CODE

The connecting code for login page is wriiten below:

```
private void sign_in_buttonActionPerformed(java.awt.event.ActionEvent evt) {
    if(jCheckBox1.isSelected())
    {
  try
  {
  String usern=jTextField1.getText();
  String pass=new String(jPasswordField1.getPassword());
  if(usern.equals("") || pass.equals(""))
  {
     JOptionPane.showMessageDialog(rootPane, "Username or password Empty! ");
  }
  else
  {
  Class.forName("java.sql.DriverManager");
  con=DriverManager.getConnection("jdbc:mysql://localhost:3306/residential","root","password");
  String sql="select password,login_type from login where login_id='" + usern + "'";
  st=con.createStatement();
  rs=st.executeQuery(sql);
  if(!rs.next())
  {
  int p=JOptionPane.showConfirmDialog(null, " No Record Found. Please proceed to Sign-up
","EXIT",JOptionPane.YES_NO_OPTION );
  if(p==0){
```

```
new user_sign_up_1().setVisible(true);
this.setVisible(false);
}
}
else
{
String passdb=rs.getString("password");
String type=rs.getString("login_type");
if(pass.equals(passdb))
{
   if(type.equals("user"))
   {
String sql1="Update user set login_st=1 where login_id='" + usern + "'";
st=con.createStatement();
st.executeUpdate(sql1);
new User_homepage().setVisible(true);
this.setVisible(false);
   }
   else
   {
      String sql1="Update admin set login_st=1 where login_id='" + usern + "'";
st=con.createStatement();
st.executeUpdate(sql1);
new Admin_homepage().setVisible(true);
this.setVisible(false);
```

```
    }
}
else
{
    JOptionPane.showMessageDialog(null, " Password did not match the username ");
    jCheckBox1.setSelected(false);
    jPasswordField1.setText("");
}
}
}}
catch(Exception e)
  {
  JOptionPane.showMessageDialog(null, "ERROR:" + e.getMessage());
  }
  }
  else
  {
      JOptionPane.showMessageDialog(rootPane, " Continue by agreeing to the checkbox above. ");
  }
}
```

Chapter 5:

TESTING

The application was testedfor various permutations and combination. It included a multitude of cases to check whether the things run as expected.

The various test cases involved during login of user or admin are:

- Correct login-id and correct password:
  If the login credentials are entered correctly by the user or admin, then it directs to the home page.
- Correct login-id and incorrect password:
  If login-id is entered correctly but password entered is wrong by the user or admin, then it prompts the password error and ask to enter the password correctly.
- Incorrect login-id and incorrect password:
  If login credentials are entered incorrectly by the user or admin, then it prompts the login error and ask to retry or go to signup.

These test cases are used to ensure security for all the users as well as the database to ensure modification of data in the database is restricted only to authorised personnel.

5.1:HOW IT WORKS

The first step in using the application is logging in. So, on opening the application the login page is displayed where the user enters his/her login-id and password. Based on this, the actions that can take place are divided into 2 sections:

1. A Resident logs in:
   The residents are initially needs to create an account by providing all the details during signup. Once the resident logs in, he/she is directed to home page. The home page contains four sections- Bill payment, clubhouse, help desk and account settings. In bill payment section, the user can view, pay and manage bills. The bills table is used here for this purpose. The clubhouse section includes three facilities namely banquet hall, swimming and gym. The resident can book the banquet hall on the specific day by choosing the free dates available. He/she can apply for gym or swimming services by choosing specific plans provided. The bills will be generated for

the facility availed by the user. A trigger is used here to insert into the bills table. The help desk section provides access to the phone directory and the residents can post their problems related to society or any facility. The admin takes care of the problems raised. The account settings section contains the account details of the resident where he can view and edit any of his details.

2. Admin logs in:

   The admin will be provided with the login credentials with which he/she logs into the application. Once the admin logs in, he/she will be directed to home page. The homepage contains four sections namely- Bill management, facilities, help desk and visitors management. In bill management section, the admin can add, view and sort bills of the residents. The bills table is used for this purpose. In the facilities section, the admin can get the information about the facilities and schedule activities. In help desk section, the admin can view the problems raised by the residents and provide the solutions to their problems. In the visitor management section, the admin gets the data of all the visitors arriving and leaving the place along with name and other details.

Chapter 6:

SCREENSHOTS AND RESULTS

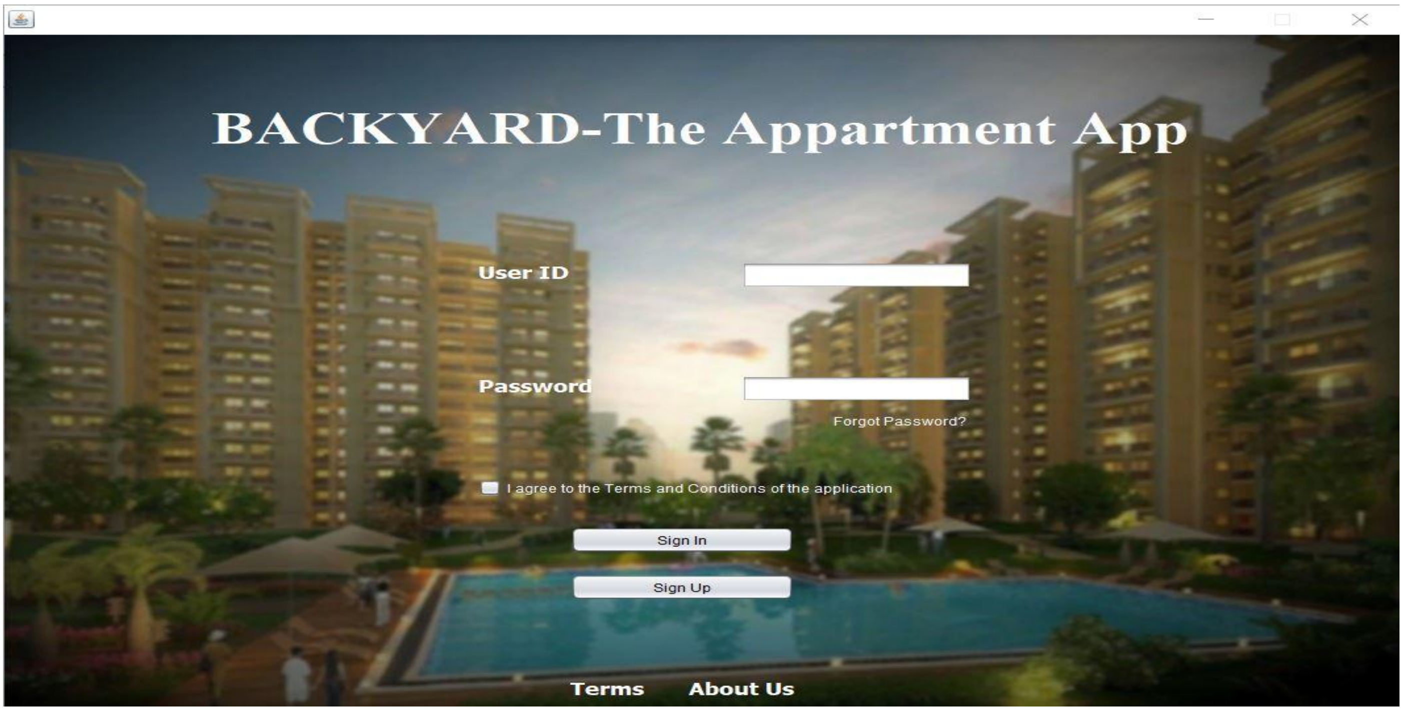

BACKYARD-The Appartment App
User ID
Password
Forgot Password?
□ I agree to the Terms and Conditions of the application
Sign In
Sign Up
Terms
About Us

Fig-6.1:Login page

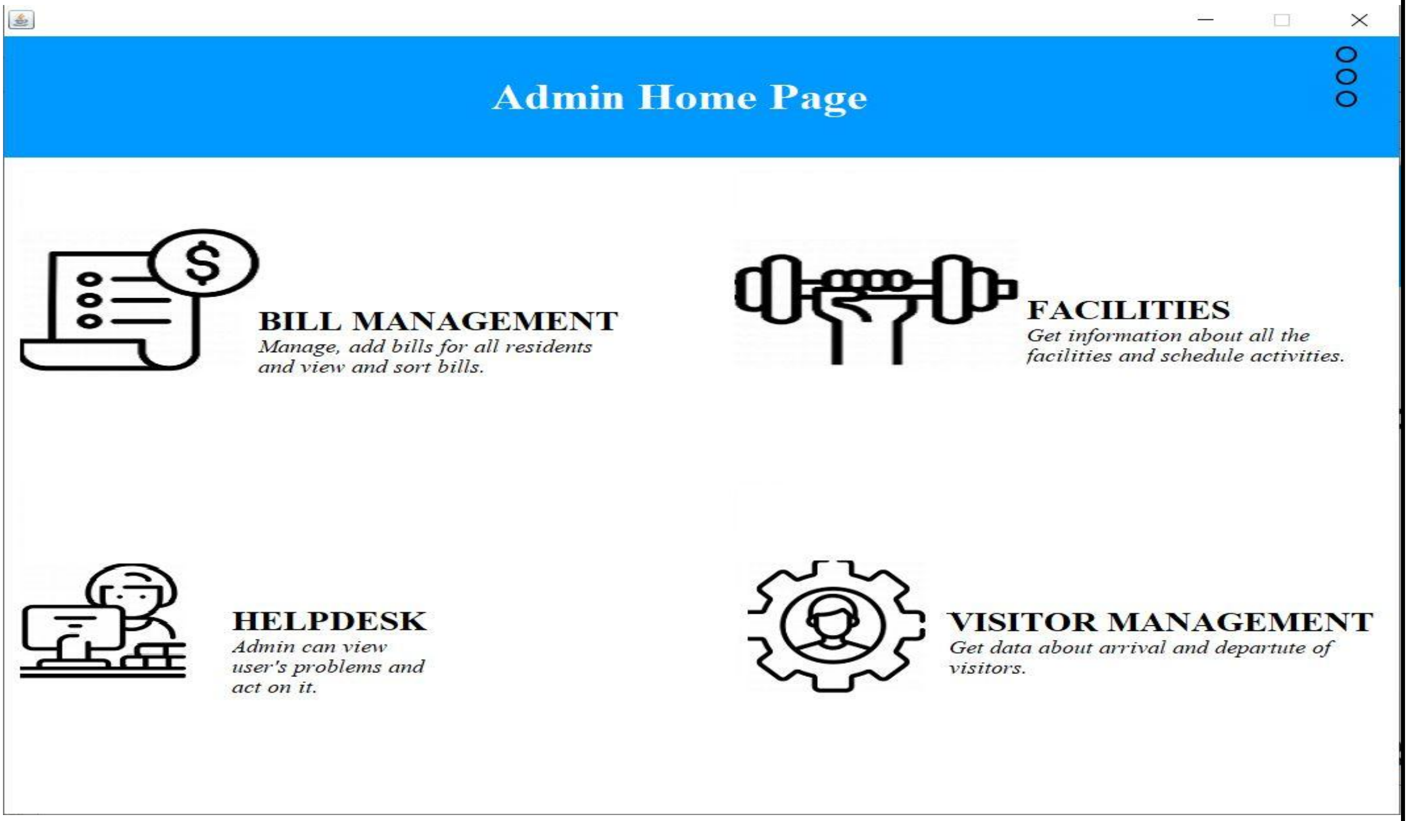
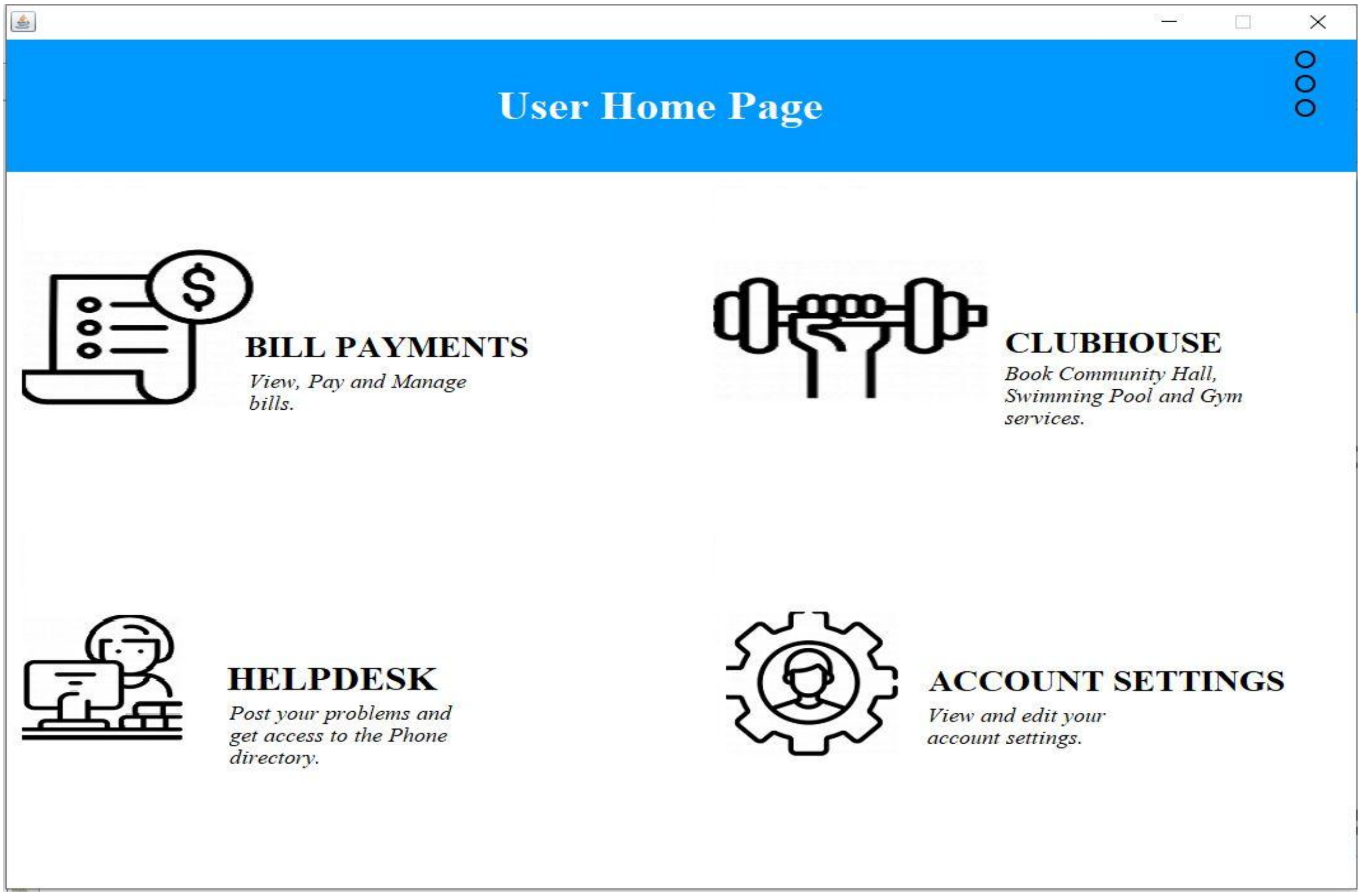

Admin Home Page
BILL MANAGEMENT
Manage, add bills for all residents and view and sort bills.
FACILITIES
Get information about all the facilities and schedule activities.
HELPDESK
Admin can view user's problems and act on it.
VISITOR MANAGEMENT
Get data about arrival and departute of visitors.

Fig-6.2:Admin's homepage



User Home Page

BILL PAYMENTS
View, Pay and Manage bills.

CLUBHOUSE
Book Community Hall, Swimming Pool and Gym services.

HELPDESK
Post your problems and get access to the Phone directory.

ACCOUNT SETTINGS
View and edit your account settings.

Fig-6.3:User’s homepage

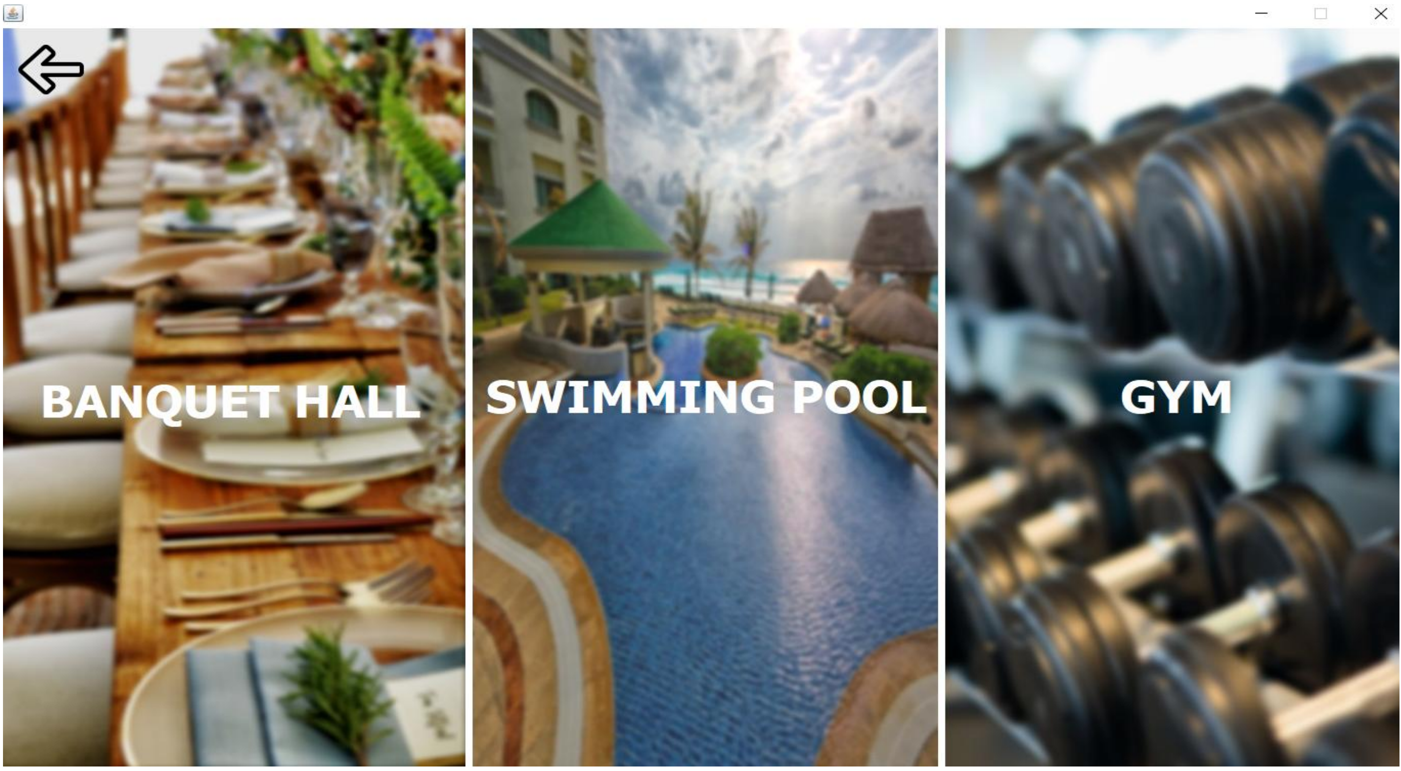


Fig-6.4:Facilities available

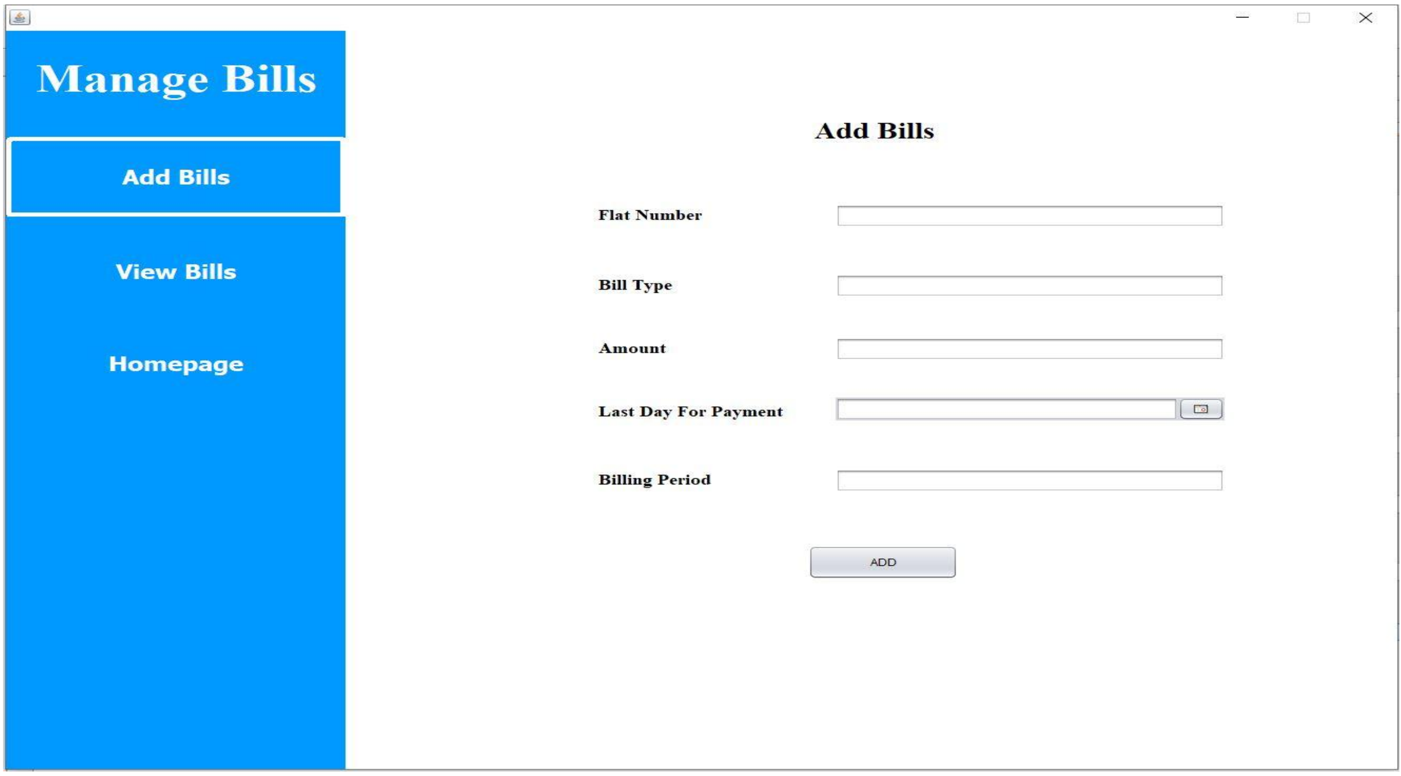
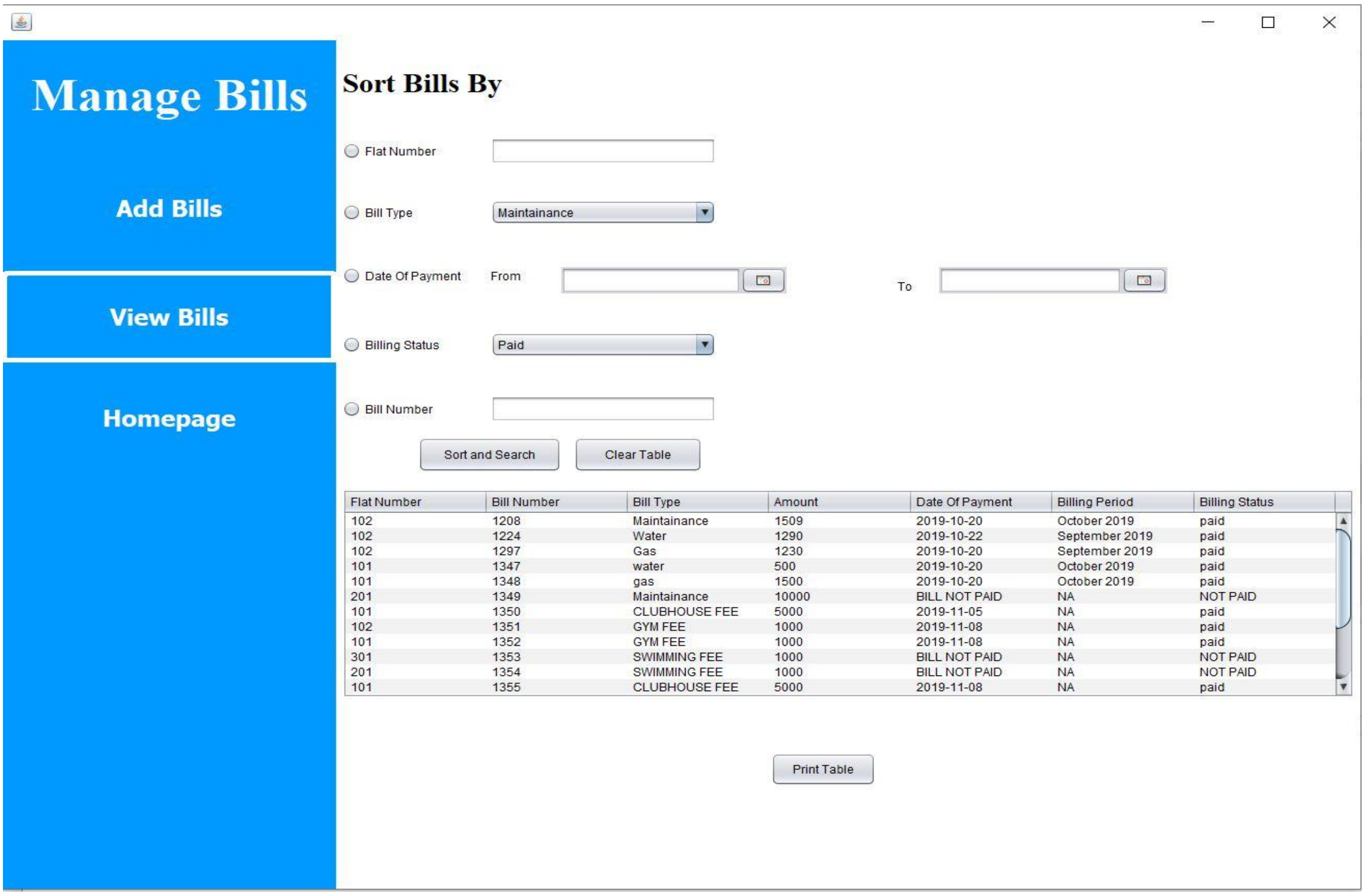


Fig-6.5:Admin's add bills page

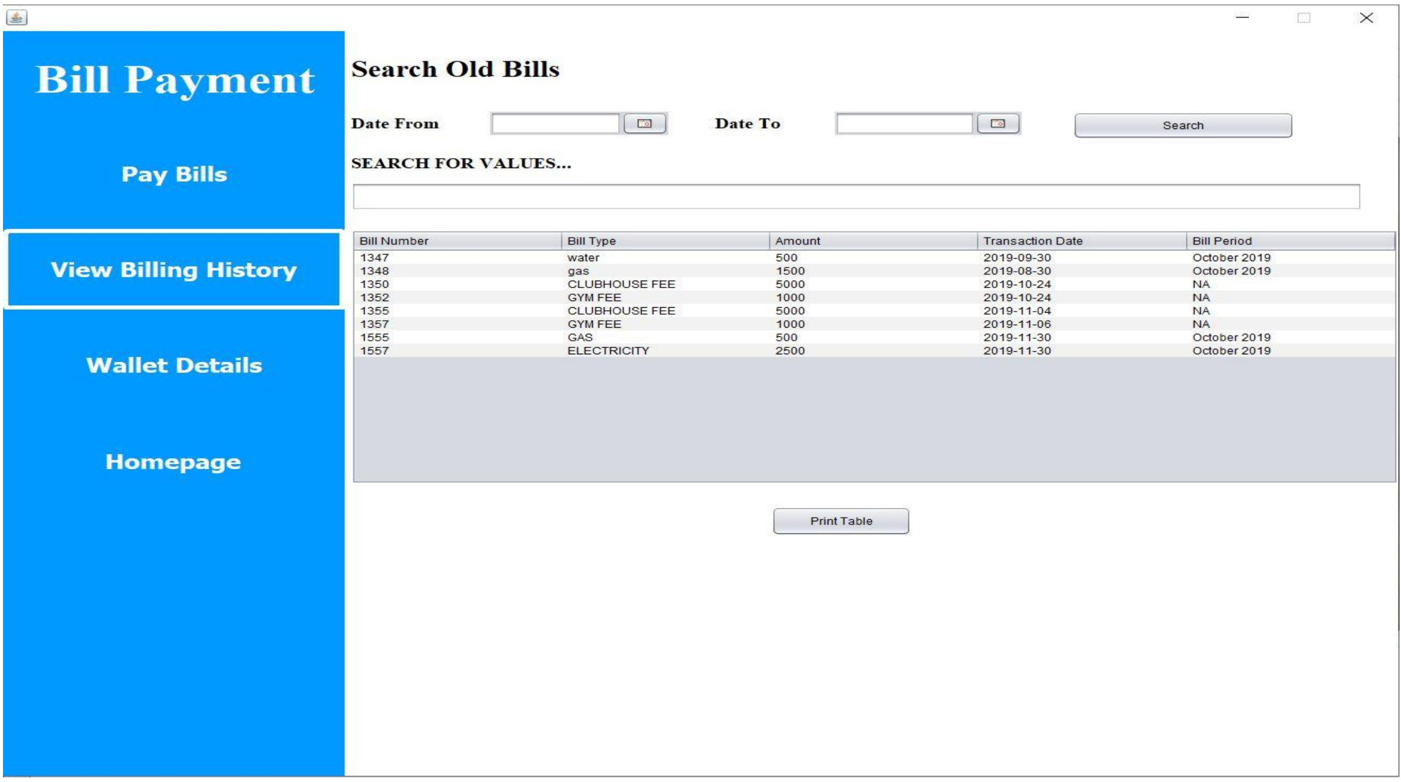


○ Bill Type: Maintainance

○ Billing Status: Paid

Flat Number	Bill Number	Bill Type	Amount	Date Of Payment	Billing Period	Billing Status
102	1208	Maintainance	1509	2019-10-20	October 2019	paid
102	1224	Water	1290	2019-10-22	September 2019	paid
102	1297	Gas	1230	2019-10-20	September 2019	paid
101	1347	water	500	2019-10-20	October 2019	paid
101	1348	gas	1500	2019-10-20	October 2019	paid
201	1349	Maintainance	10000	BILL NOT PAID	NA	NOT PAID
101	1350	CLUBHOUSE FEE	5000	2019-11-05	NA	paid
102	1351	GYM FEE	1000	2019-11-08	NA	paid
101	1352	GYM FEE	1000	2019-11-08	NA	paid
301	1353	SWIMMING FEE	1000	BILL NOT PAID	NA	NOT PAID
201	1354	SWIMMING FEE	1000	BILL NOT PAID	NA	NOT PAID
101	1355	CLUBHOUSE FEE	5000	2019-11-08	NA	paid

Fig-6.6:Admin's view bills page



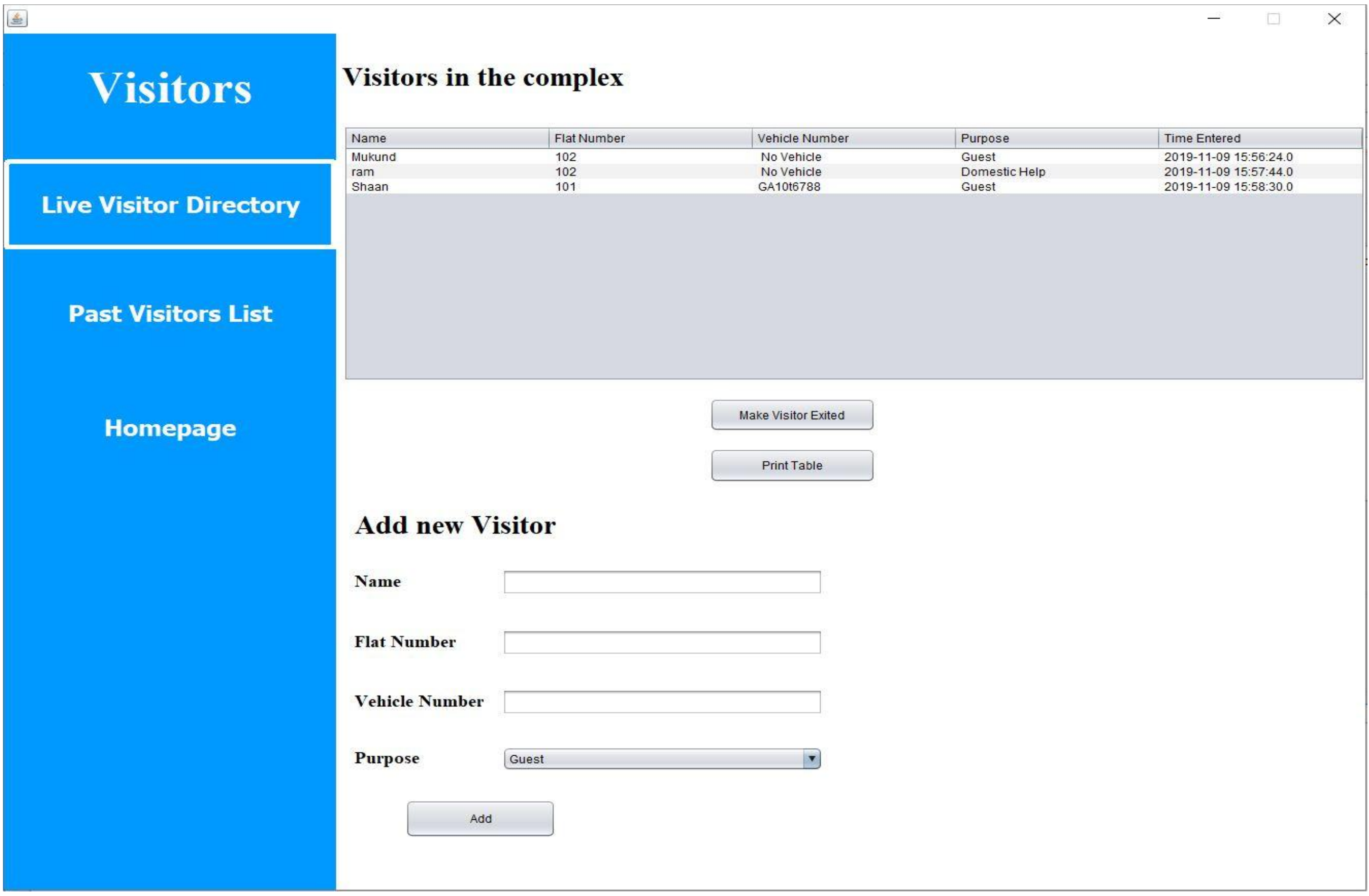
Search Old Bills

Date From Date To

SEARCH FOR VALUES...

Bill Number	Bill Type	Amount	Transaction Date	Bill Period
1347	water	500	2019-09-30	October 2019
1348	gas	1500	2019-08-30	October 2019
1350	CLUBHOUSE FEE	5000	2019-10-24	NA
1352	GYM FEE	1000	2019-10-24	NA
1355	CLUBHOUSE FEE	5000	2019-11-04	NA
1357	GYM FEE	1000	2019-11-06	NA
1555	GAS	500	2019-11-30	October 2019
1557	ELECTRICITY	2500	2019-11-30	October 2019

Fig-6.7:User’s bill payment page



Visitors in the complex

Name	Flat Number	Vehicle Number	Purpose	Time Entered
Mukund	102	No Vehicle	Guest	2019-11-09 15:56:24.0
ram	102	No Vehicle	Domestic Help	2019-11-09 15:57:44.0
Shaan	101	GA10t6788	Guest	2019-11-09 15:58:30.0

Add new Visitor

Name

Flat Number

Vehicle Number

Purpose Guest

Fig-6.8:Admin’s visitor management page

Chapter 7:

CONCLUSION AND FUTURE ENHANCEMENT

The primary motivation and purpose behind developing this application is to reduce and automate the manual tasks which were traditionally being done in the residential community management. We have achieved most of the requirements by the community for the application. As of now, the residents can view and pay their bills without standing in a line. The residents can also book clubhouse activities such as banquet hall, swimming pool and gym from their houses. The inclusion of helpdesk in the application has simplified the process of problem redresal and made it efficient. We have built a software product on which they can run their Apartment Community - elegantly weaving together every aspect of Communication, Facility Management, Security and Accounting among other aspects. Most of the tasks have been achieved which we had set during the planning of the project.

There is a scope for future enhancements for the application. This includes –

- Integrating the software with the concepts of Internet of Things like implementing sensors for visitor management and Vehicle parking management.
- Adding a Payment Gateway for paying bills.
- Making the application more secure by using One Time Passwords and security questions.

Due to money and time constraints, we were not able to implement these features.

Chapter 8:

REFERENCES

- The Database book: Principles and Practice using MySQL, Narain Gehani, Universities press (India) private Limited 2008.
- Fundamentals of Database System, Elmasri and Navathe, Addison-Wesley, 5th edition, 2007.
- http://stackoverflow.com for solving common problems.
- http://google.com for most resources like images and ideas.
- http://flaticon.com for most of the icons in the application.
- https://dev.mysql.com/doc/ for MySQL documentation.
- https://docs.oracle.com/javase/8/docs/ for Java-Swing documentation.