## VISVESVARAYA TECHNOLOGICAL UNIVERSITY

### BELAGAVI, KARNATAKA, INDIA

****

**A MINI-PROJECT REPORT**

#### ON

**APARTMENT VISITORS MANAGEMENT SYSTEM**

##### A report submitted in the partial fulfillment of the requirements for the award of the degree of

***Bachelor of Engineering in***

***Information Science & Engineering***

***Submitted by***

|  |  |
| --- | --- |
| **Name1** | **USN** |
| **Name2** | **USN** |

***Under the guidance of***

### Prof.Name

****

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

**SAPTHAGIRI COLLEGE OF ENGINEERING**

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)

(IAO9001-2015 and ISO14001-2015 certified Institute)

#14/5, Chikkasandra, Hesaraghatta Main Road, Bengaluru – 560057

**2021-22**

SAPTHAGIRI COLLEGE OF ENGINEERING

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)

(IAO9001-2015 and ISO14001-2015 certified Institute)

#14/5, Chikkasandra, Hesaraghatta Main Road, Bengaluru – 560057



### Department of Information Science & Engineering

**CERTIFICATE**

This is to Certify that the DBMS Mini-Project work entitled **APARTMENT VISITORS MANAGEMENT SYSTEM** carried out by **Ms. Name1(USN) and Ms. Name2(USN)** bonafide students of 5th semester, department of **Information Science & Engineering** carried out at our college **Sapthagiri College of Engineering**, Bengaluru in partial fulfillment of the award of **Bachelor of Engineering** in **Information Science & Engineering** of the **Visvesvaraya Technological University,** Belagavi during the year 2020-21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Mini-Project work prescribed for the said Degree.

|  |  |  |
| --- | --- | --- |
| **Signature of the Guide** |  | **Signature of the HOD** |

|  |  |
| --- | --- |
| **External Viva**  **Name of the Examiners**  1. ……………………………………  2. …………………………………… | **Signature of the Examiners with date**  1. ……………………………….  2. ………………………………. |

## ACKNOWLEDGEMENT

Any achievement doesn’t depend solely on the individual efforts but on the guidance, encouragement and co-operation of intellectuals, elders and friends. A number of personalities have helped us. We would like to take this opportunity to thank them all.

We would like to express our heart-felt gratitude to **Dr. H Ramkrishna**, Principal, Sapthagiri College of Engineering, Bengaluru, for his help and inspiration during the tenure of the course.

It is great privilege to extend our deep sense of gratitude to **Dr. H R Ranganatha**, Head of the Department, Information Science and Engineering, Sapthagiri College of Engineering, Bengaluru, who patronized throughout our career, for his constant support and encouragement and for the facilities provided to carry out this work successfully.

We wish to express our sincere thanks to our guides **Prof. Ramya R**, Assistant Professor, **Prof. Chaitanya V,** Assistant Professor, Information Science and Engineering, Sapthagiri College of Engineering, Bengaluru for helping us throughout and guiding us from time to time.

We also extend our sense of gratitude and sincere thanks to all faculty members and non-teaching staff members of Information Science and Engineering, Sapthagiri College of Engineering, Bengaluru for their views and encouraging ideas.

Finally, we also thank our family and friends for their co-operation and motivation.

**Name1**

**Name2**

# **TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Chapter Name** | **Page No.** |
| **1** | **INTRODUCTION** | **07** |
| 1.1 | Overview | 07 |
| 1.2 | Problem Statement | 07 |
| 1.3 | Objectives | 08 |
| 1.4 | Limitations | 08 |
| 1.5 | Literature Review | 09 |
| **2** | **ANALYSIS** | **10** |
| 2.1 | Proposed System | 10 |
| 2.2 | Requirements Specification | 10 |
| 2.2.1 | Hardware Requirements | 10 |
| 2.2.2 | Software Requirements | 11 |
| 2.3 | Development Environment | 11 |
| 2.3.1 | Apache | 11 |
| 2.3.2 | Php | 12 |
| 2.3.3 | My SQL | 12 |
| **3** | **DESIGN** | **13** |
| 3.1 | Design introduction | **13** |
| 3.2  3.3  3.4  3.5 | Usercase diagram  Class diagram  ER diagram  Schema diagram | **14**  **15**  **16**  **18** |
| **4** | **IMPLEMENTATION** | **20** |
| 4.1 | Tables | 20 |
| 4.2 | Triggers | 23 |
| **5** | **SNAPSHOTS** | **24** |
| **6** | **CONCLUSION & FUTURE ENHANCEMENTS** | **29** |
| 6.1 | Conclusion | 29 |
| 6.2 | Future Enhancements | 29 |
|  | **BIBLIOGRAPHY** | 30 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page No.** |
| 3.4 | ER Diagram | 16 |
| 3.5 | Schema Diagram | 18 |
| 5.1 | login Page | 24 |
| 5.2 | Dashboard Page | 24 |
| 5.3 | Manage apartment Page | 25 |
| 5.4 | Entry visitors Page | 25 |
| 5.5 | Checkout visitors Page | 26 |
| 5.6 | Visitors management Page | 26 |
| 5.7 | Reporters profile Page | 27 |
| 5.8 | Admin profile Page | 27 |
| 5.9 | Change Password Page | 28 |

**Tables**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Name** | **Page No** |
| 4.1.1 | Apartment table | 23 |
| 4.1.2 | Admin table | 23 |
| 4.1.3 | Entry visitor table | 24 |
| 4.1.4 | Manage apartment table | 24 |
| 4.1.5 | Checkout visitor table | 25 |
| 4.1.6 | Visitor management table | 25 |
| 4.1.7 | Reports table | 26 |

## CHAPTER 1

**INTRODUCTION**

**1.1 Overview**

Apartment Visitor Management system is a web-based technology that will

revolutionize the way your society manages visitors. Apartment Visitor Management system is more important to security guards or security society. This web application provides a way to effectively control record & track society visitor traffic.

In AVMS we use PHP and MySQL database. This is the project which keeps records of visitors who visited in  the Society. **AVMS has one module i.e. admin**

* Dashboard : In this section, admin can briefly view how many visitors visited in a particular period.
* Visitors : In this section, admin adds new visitors by filling their information in add visitors sections and also view and manage visitor’s records. Admin also put visitors out time in the manage records section.
* Search : In this bar, admin can search a particular person by their name and phone number.
* Reports : In this section admin can generate visitor’s reports between two dates.

Admin can also update his profile, change password and recover password.

**1.2 Problem Statement**

Managing visitors at apartment accommodations is one of the major problems faced by the administration that deals with apartment’s visitors accommodation. Present system of working by the security management is that the security staff at the guard houses of these apartment accommodation, visitors come and provide identity proof and then they provide unit and block number which they wish to visit. The security guard at the guard house will call the resident of the unit number given by the visitor then the resident will acknowledge that the visitor at the guard house is his visitor, once the resident has been acknowledged the security guard shall allow the visitor to enter the accommodation. Visitor management at the apartment right now is manual which consumes a lot of time of the visitors who need to wait for the security guard to call the resident to acknowledge he has a visitor. All this is time consuming more so when there is a rush of visitors at the guard house. For the apartment resident it is more discomfort when there is a get together of friends since for each visitor he has to attend phone calls from the guard house which leads to consume much of his party time attending calls.

* 1. **Objectives**

The objectives of developing apartment visitor management system is to

* Provide resident’s safety and monitoring
* Computerized the tradition way of visitors.
* Generate the report automatically.
* Developed as a web application and it will work over web.

**1.4 Limitations**

* **Not user friendly:** The present system not user friendly because data is not stored in structure and proper format.
* **Manual Control:** All report calculation is done manually so there is a chance of error.
* **Lots of paper work:** Visitors maintain in the register so lots of paper require storing details.
* **Time consuming.**
  1. **Literature review**

Visitor Management Software is working system that allow the visitor to call the inmates from the gate requires someone to answer from inside the house. Visitor Management Software is the outcome of motivation that has come out of the drawbacks or loopholes of the present system of working. Apartment Visitor management system (AVMS) is developed using the trends in information technology which provides a feature to capture visitor information by using identification proof of the visitor and save them to a database that is centralized.

In addition to physical barriers limiting apartment access to one point, apartment have discovered the need to improve their process for recording who is entering and exiting the facility. The basic clipboard method is limiting because it is slow and provides little information other than name and time. Several computer based systems are available that provide a wealth of information to enhance both security and front office efficiency. Many systems have a kiosk type environment where a visitor essentially checks himself in much like the clipboard method, but the system has the ability to record much more information quickly. This additional information can include a photo of the visitor, the reason he is there, to what location he is going, a time stamp, and more.

**CHAPTER 2**

**ANALYSIS**

**2.1 Proposed System**

In present all visitor work done on the paper. The whole year visitor is stored in the registers. We can’t generate reports as per our requirements because its take more time to calculate the visitors report.

**2.2 Requirement Specification**

**2.2.1 Hardware Configuration :**

**Client Side:**

|  |  |
| --- | --- |
| **RAM RAMfgdf RAM** | 512 MB |
|  |  |
| **Hard disk** | 10 GB |
|  |  |
| **Processor** | 1.0 GHz |
|  |  |

**Server side:**

|  |  |
| --- | --- |
| **RAM** | 1 GB |
| **Hard disk** | 20 GB |
| **Processor** | 2.0 GHz |

**2.2.2 Software Requirement:**

**Client Side:**

|  |  |
| --- | --- |
| **Web Browser** | Google Chrome or any compatible browser |
| **Operating System** | Windows or any equivalent OS |

**Server Side:**

|  |  |
| --- | --- |
| **Web Server** | APACHE |
| **Server side Language** | PHP5.6 or above version |
| **Database Server** | MYSQL |
| **Web Browser** | Google Chrome or any compatible browser |
| **Operating System** | Windows or any equivalent OS |

**2.3 Development Environment**

**2.3.1 APACHE**

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

The Apache HTTP Server ("httpd") was launched in 1995 and it has been the most popular web server on the Internet since April 1996. It has celebrated its 20th birthday as a project in February 2015.

**2.3.2 PHP**

* PHP stands for PHP: Hypertext Preprocessor.
* PHP is a server-side scripting language, like ASP.
* PHP scripts are executed on the server.
* PHP supports many databases (MYSQL, Informix, Oracle, Sybase, Solid, Generic ODBC, etc.).
* PHP is open source software.
* PHP is free to download and use.

**2.3.3 MYSQL**

* MYSQL is a database server
* MYSQL is ideal for both small and large applications
* MYSQL supports standard SQL
* MYSQL compiles on a number of platforms
* MYSQL is free to download and use
* How to access MySQL: <http://localhost/phpmyadmin>

**Chapter 3**

**DESIGN**

**3.1 Design Introduction:**

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data

UML Diagrams:

Actor:  
 A coherent set of roles that users of use cases play when interacting with the use cases.

Use case:A description of sequence of actions, including variants, that a system performs that yields an observable result of value of an actor.

UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

**3.2 USECASE DIAGRAMS:**

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what’s called an actor.

Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can’t do.

Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

* The purpose is to show the interactions between the use case and actor.
* To represent the system requirements from user’s perspective.
* An actor could be the end-user of the system or an external system.

**USECASE DIAGRAM:**

A Use case is a description of set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors. An actor represents a real-world object. Primary Actor – Sender, Secondary Actor Receiver.

**Use Case Diagrams:**

Dashboard

**Admin**

Manage Visitors

(Update Details)

Ad

Generate Reports

Update Profile

Change Password

Add New Visitors

Password Recovery

**3.3 Class Diagram:**

A description of set of objects that share the same attributes operations, relationships, and semantics

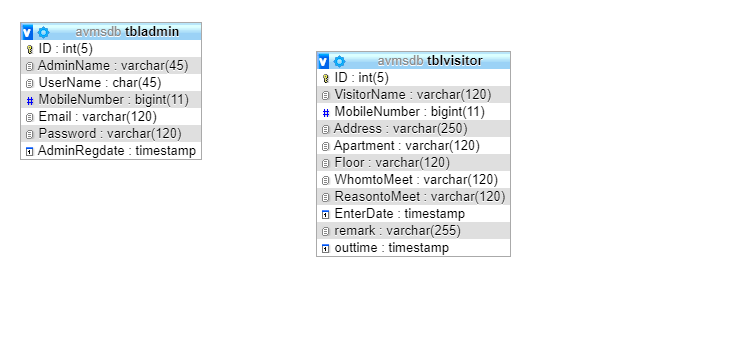


Fig 3.3 Class Diagram

**3.4 ER Diagram:**

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [Chen76] as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database design for the database designer, the utility of the ER model is:

* It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
* It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.
* In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

**ER Notation**

There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. The original notation used by Chen is widely used in academics texts and journals but rarely seen in either CASE tools or publications by non-academics. Today, there are a number of notations used; among the more common are Bachman, crow's foot, and IDEFIX.

All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The notation used in this document is from Martin. The symbols used for the basic ER constructs are:

* **Entities** are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.
* **Relationships** are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs
* **Attributes**, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.
* **Cardinality** of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.

**Existence** is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional.

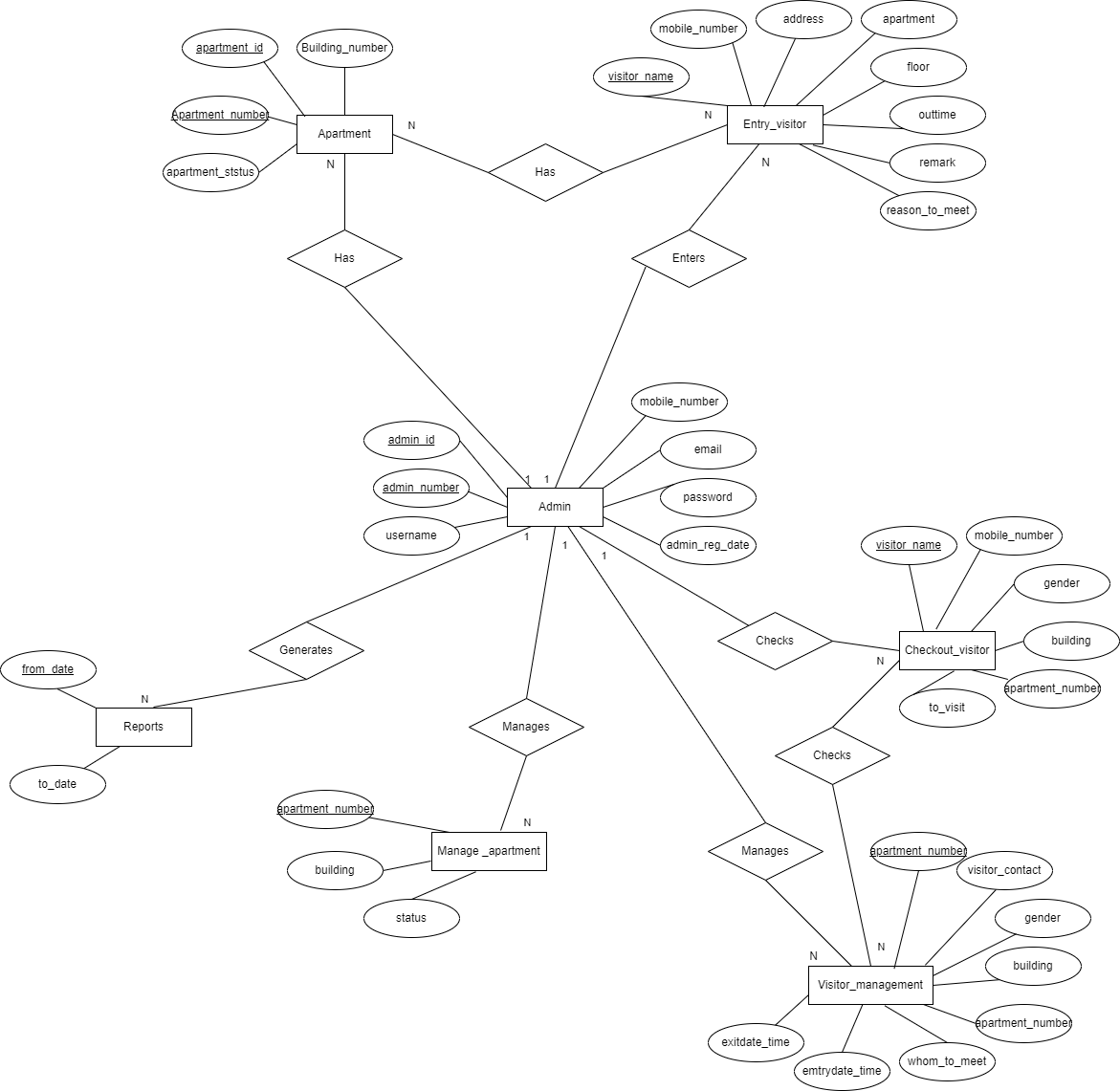
****

Fig 3.4 ER Diagram

**3.5 SCHEMA DIAGRAM**

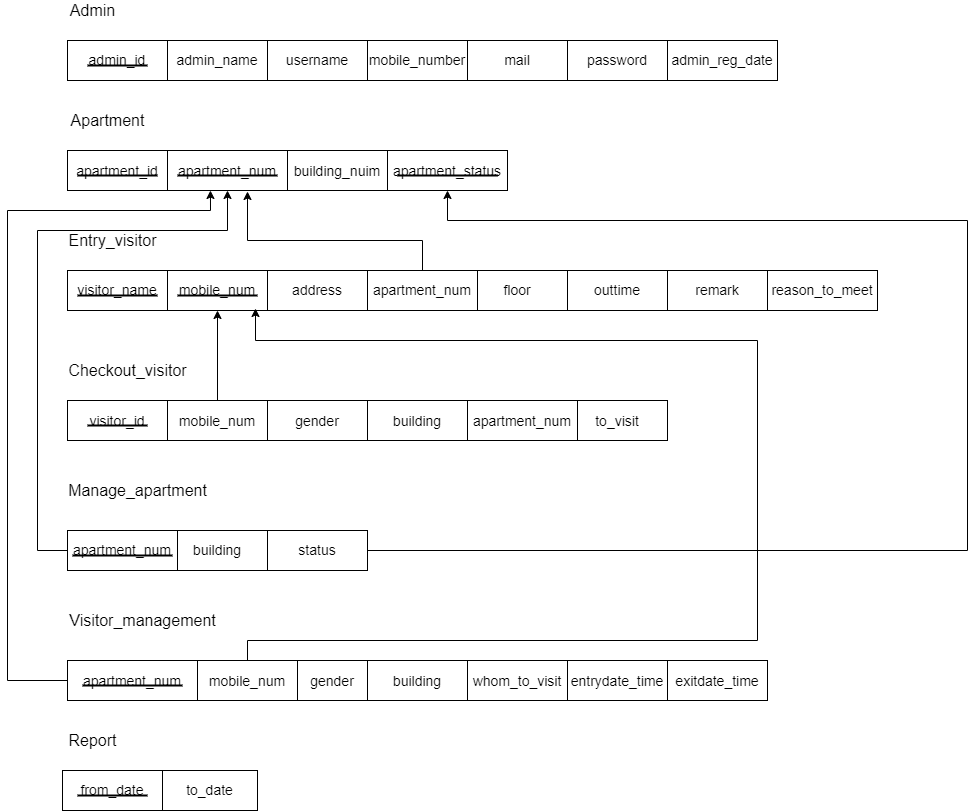


Fig 3.5 Schema Diagram

**Relational Schema**

A relational database schema is the tables, columns and relationships that make up a relational database. A relational database schema helps you to organize and understand the structure of a database. This is particularly useful when designing a new database, modifying support more functionality, or building integration between databases. There are two steps to creating a relational database schema: creating the logical schema and creating the physical schema. The logical schema depicts the structure of the database, showing the tables, columns and relationships with other tables in the database and can be created with modelling tools or spread sheet and drawing software. The physical schema is created by actually generating the tables, columns and relationships in the relational database management software (RDBMS).

.

**Chapter 4**

**IMPLEMENTATION**

**4.1 Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Id | Int | 11 | Id of the apartment |
| Apartment\_number | Varchar | 255 | Number of apartment |
| Building\_number | Varchar | 20 | Number of building |
| Apartment\_status | Varchar | 20 | Status of apartment |

**4.1.1 Apartment Table**

4.1.1 Apartment table

CREATE TABLE apartment (

Id int(11) PRIMARY KEY,Apartment\_number varchar(255),Building\_number varchar(20),Apartment\_status varchar(20));

**4.1.2 Admin Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Id | Int | 5 | Admin id |
| Admin\_name | Varchar | 45 | Name of admin |
| User\_name | Varchar | 45 | Name of the user |
| Security\_code | Int | 50 | Security code |
| Mobile\_number | Bigint | 11 | Mobile number |
| Email | Varchar | 120 | Email of admin |
| Password | Varchar | 120 | Password |
| Admin\_reg\_date | Timestap | - | Admin registration date |

4.1.2 Admin table

CREATE TABLE admin (

Id int(5) PRIMARY KEY,admin\_name varchar(45),user\_name varchar(45),security\_code int(50),mobile\_number bigint(11),email varchar(120),password varchar(120),admin\_reg\_date timestamp);

**4.1.3 Entry Visitor Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Visitor\_name | Varchar | 120 | Name of visitor |
| Mobile\_number | Bigint | 11 | Mobile number |
| Address | Varchar | 250 | Visitor address |
| Gender | Varchar | 11 | Visitor gender |
| Apartment | Varchar | 120 | Apartment number |
| Building\_number | Varchar | 55 | Building number |
| Whomto\_meet | Varchar | 120 | Whom to meet |
| Reason | Varchar | 120 | Reason of visitor |

4.1.3 Entry visitor table

CREATE TABLE entry visitor (

Visitor\_name varchar(120) PRIMARY KEY,mobile\_number bigint(11),address varchar(250),gender varchar(11),apartment varchar(120),

building\_number varchar(55),whomto\_meet varchar(120),reason varchar(120));

**4.1.4 Manage apartment table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Apartment\_number | Int | 255 | Apartment number |
| Building | Varchar | 120 | Building of apartment |
| Status | Varchar | 120 | Status of the apartment |

4.1.4 Manage apartment table

CREATE TABLE manage apartment (

Apartment\_number int(255) PRIMARY KEY,building varchar(120),status varchar(120));

**4.1.5 Checkout Visitors Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Visitor\_name | Varchar | 250 | Name of visitor |
| Mobile\_number | Bigint | 11 | Contact number |
| Gender | Varchar | 11 | Gender of visitor |
| Building | Varchar | 120 | Building name |
| Apartment\_number | Int | 255 | Apartment number |
| Tovisit | Varchar | 120 | Whom to visit |
| Entry\_time | Timestap | - | Entry time of visitor |

4.1.5 Checkout visitors table

CREATE TABLE checkout visitor (  
visitor\_name varchar(250) PRIMARY KEY,mobile\_number bigint(11),gender varchar(11),building varchar(120),apartment\_number int(255),tovisit varchar(120),entry\_time timestamp);

**4.1.6 Visitor Management Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Visitors\_name | Varchar | 120 | Name of visitor |
| Visitor\_contact | Bigint | 11 | Contact of visitor |
| Gender | Varchar | 11 | Gender of visitor |
| Building | Varchar | 120 | Building id |
| Apartment\_number | Int | 250 | Apartmet number |
| Whomto\_visit | Varchar | 120 | Whom to visit |
| Entrydate\_Entrytime | Int | 20 | Entry date and time |
| Exitdate\_Exittime | Int | 20 | Exit date and time |

4.1.6 Visitors management table

CREATE TABLE visitor management (

Visitor\_name varchar(120),visitor\_contact varchar(11),gender varchar(11),building varchar(120),apartment\_number int(250),whomto\_visit varchar(120),entrydate\_entrydate int(20),exitdate\_exittime int(20));

**4.1.7 Report Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| From\_date | Int | 120 | Report from the date |
| To\_date | Int | 120 | Report to the date |

4.1.7 Reports table

CREATE TABLE report (  
 from\_date int(120),to\_date int(120));

* 1. **Triggers**

TRIGGER specifies an event, a condition and an action. It specifies type of the action to be taken when certain events occur and when certain condition are satisfied.

1.Start apartment visitor database system.

2.Create trigger trigger\_name for table visitor with action to be taken before update.

3.Increment the visitor number.

4.Stop

**Chapter 5**

**SNAPSHOTS**

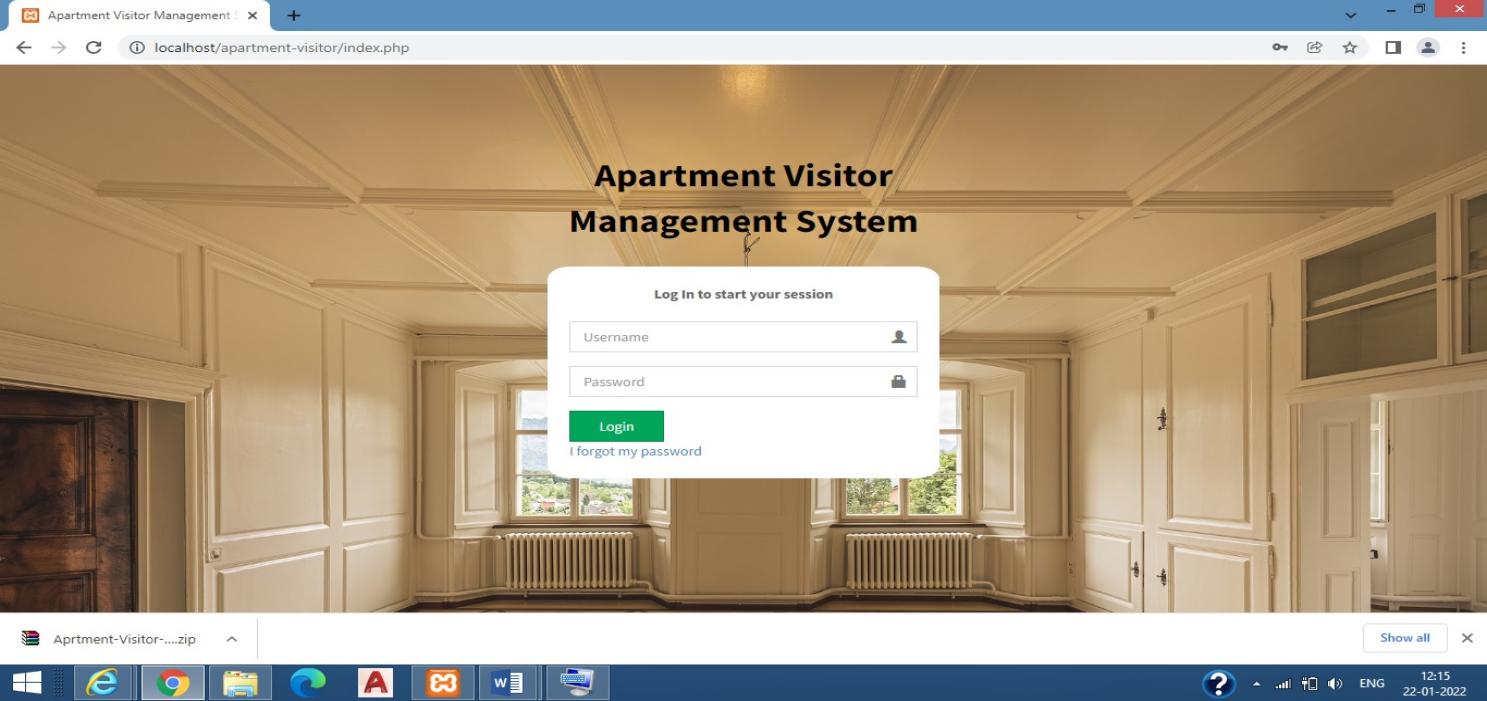
****

Fig 5.1 Login Page

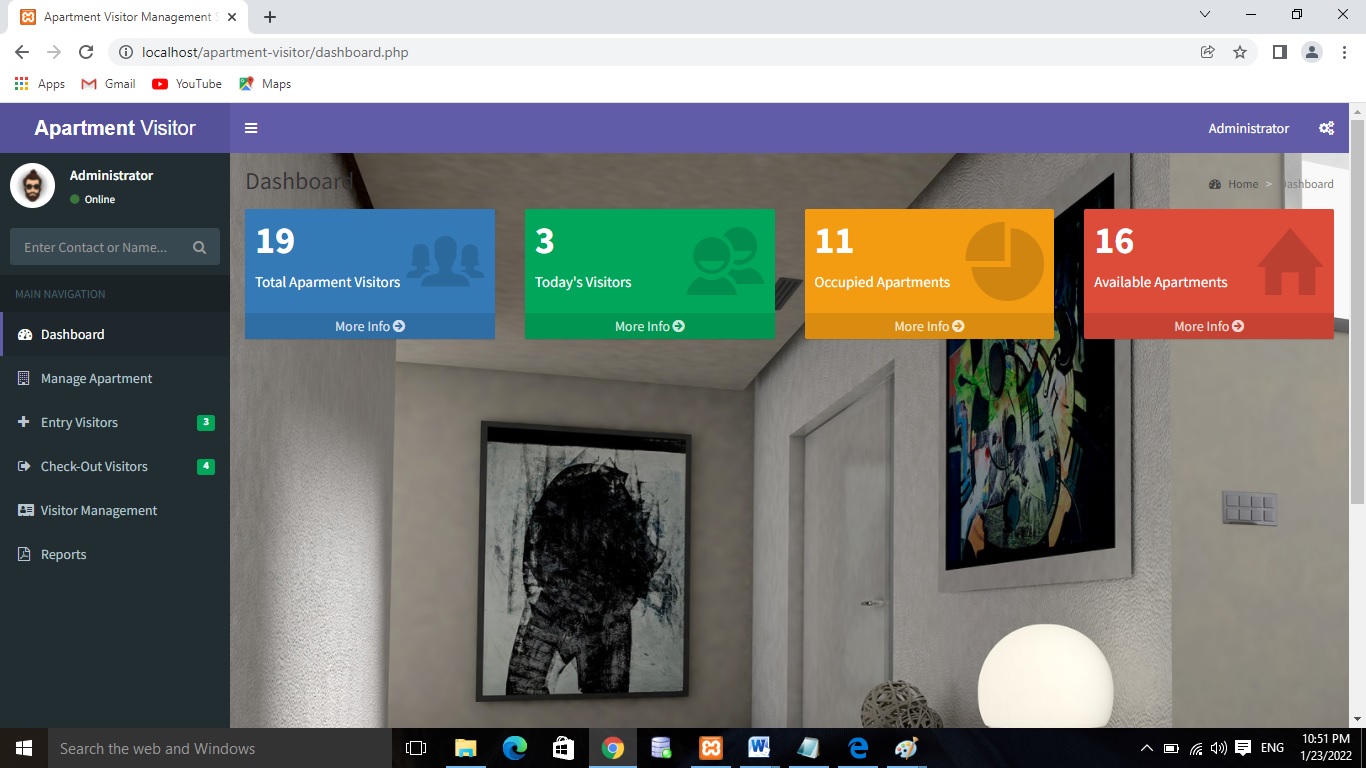


Fig 5.2 Dashboard Page

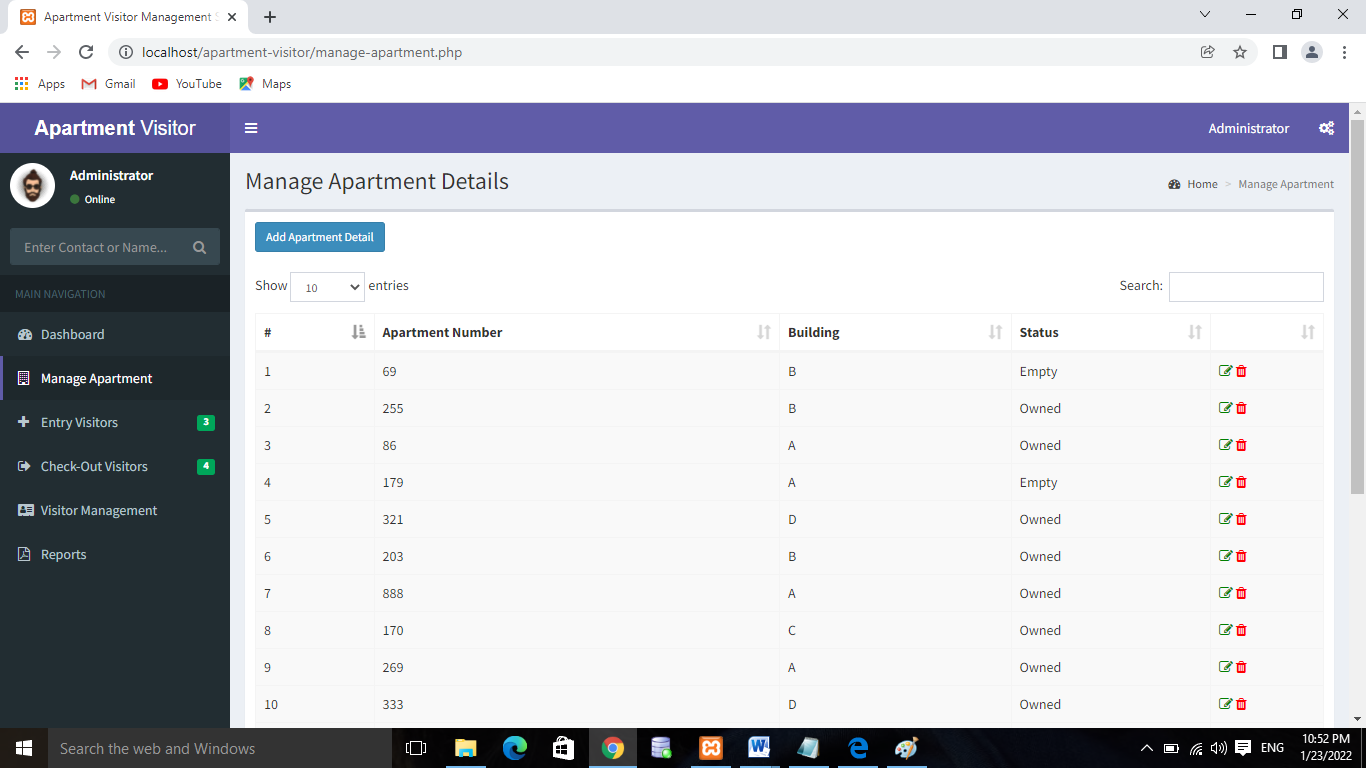


Fig 5.3 Manage apartment Page

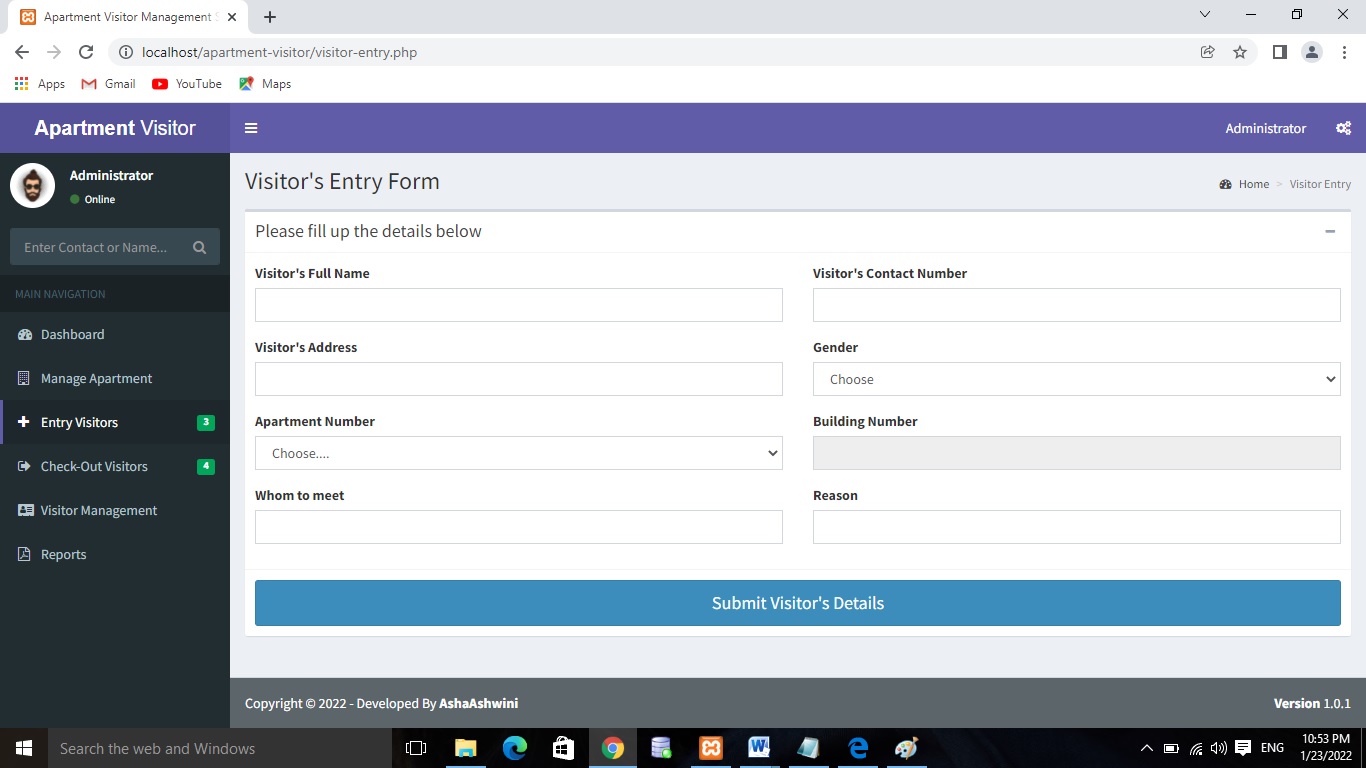


Fig 5.4 Entry visitors Page

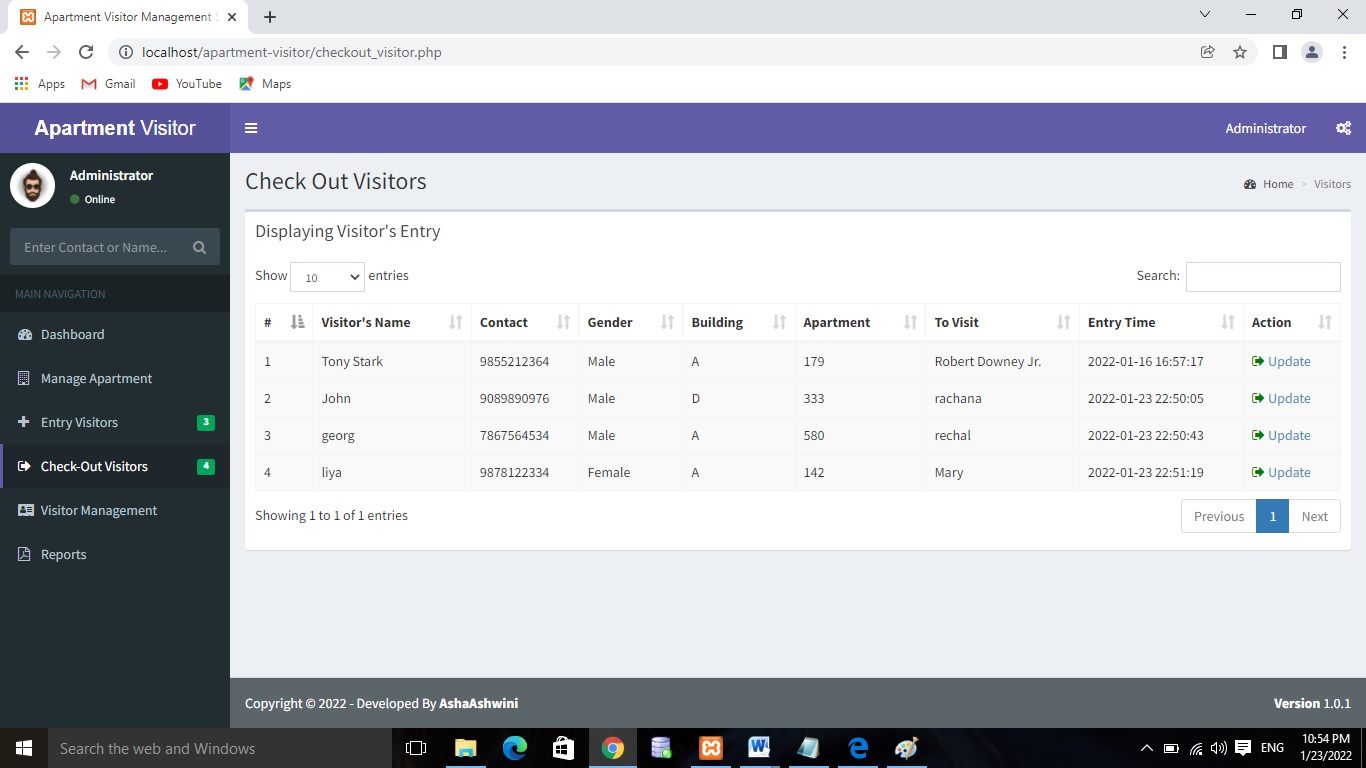


Fig 5.5 Checkout Visitors Page

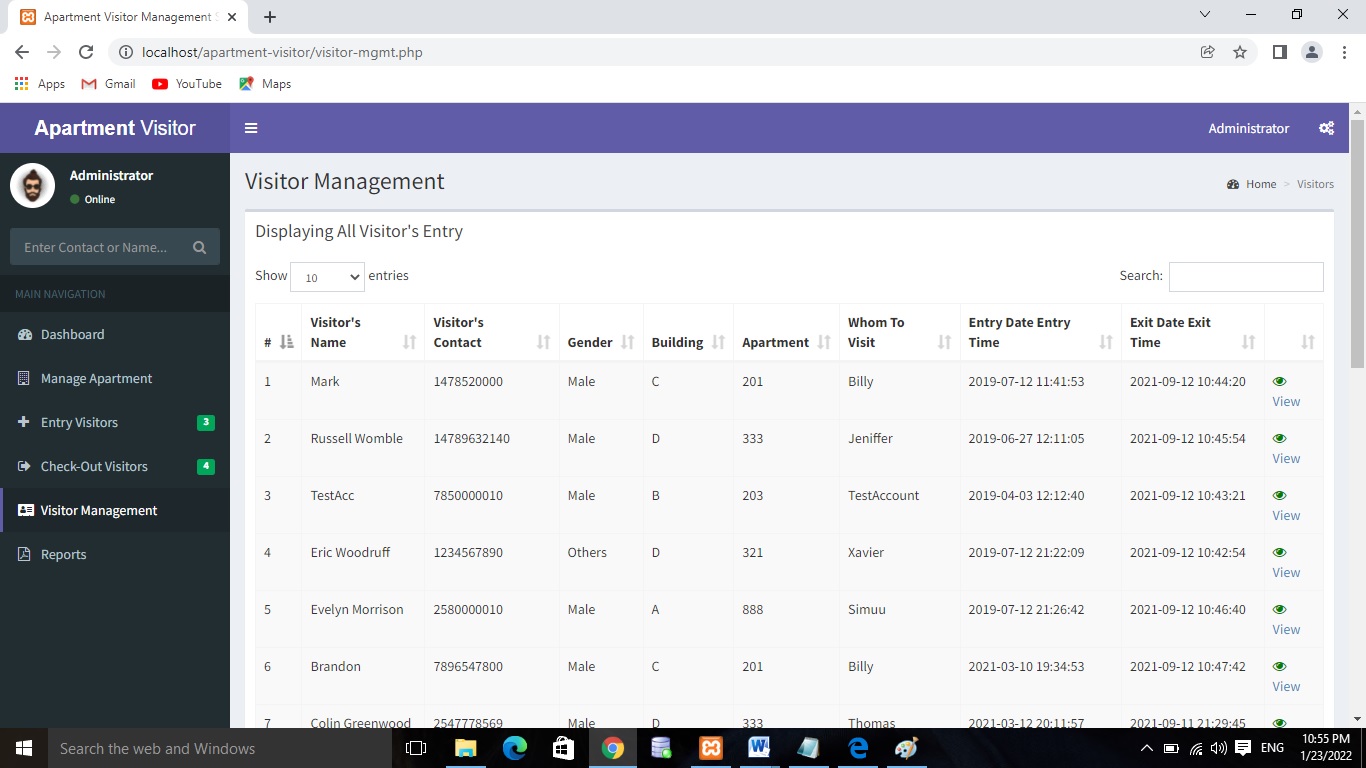


Fig 5.6 Visitor management Page

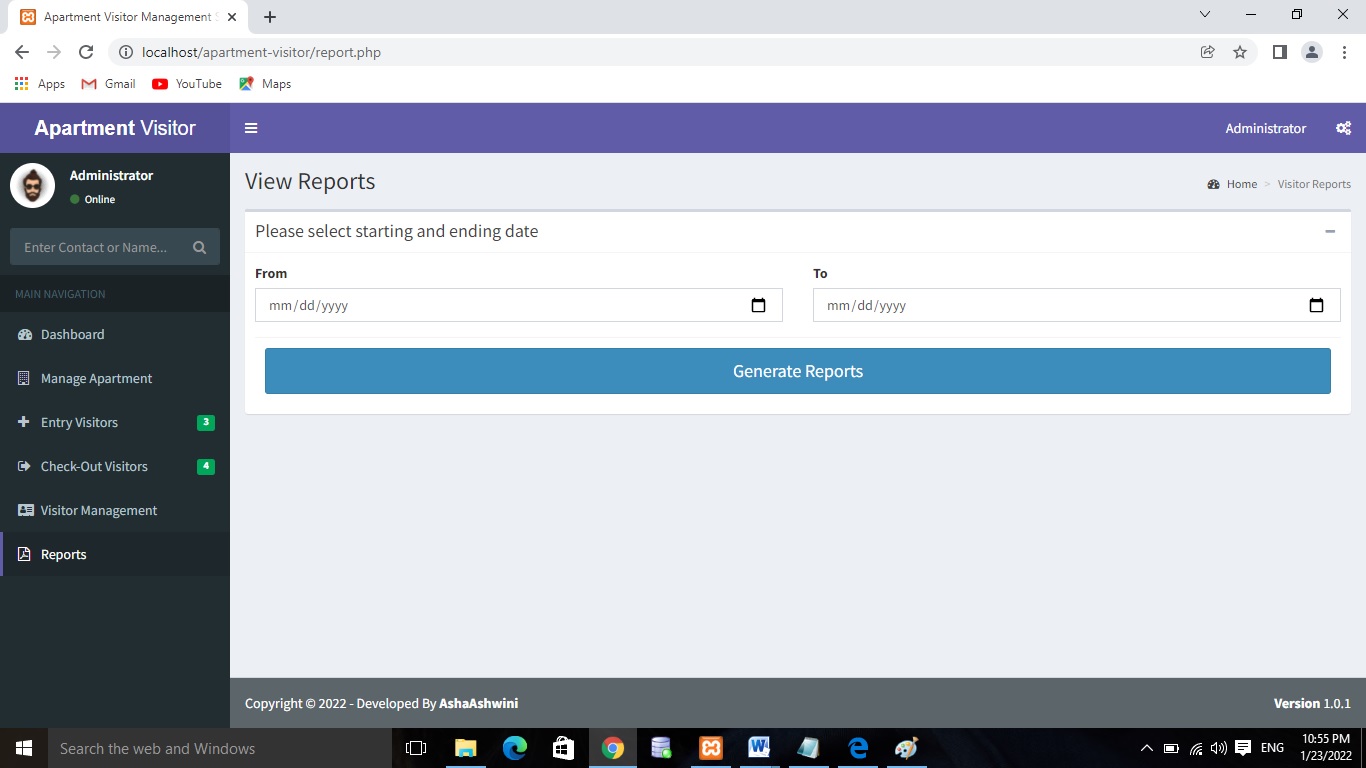


Fig 5.7 Reports Page

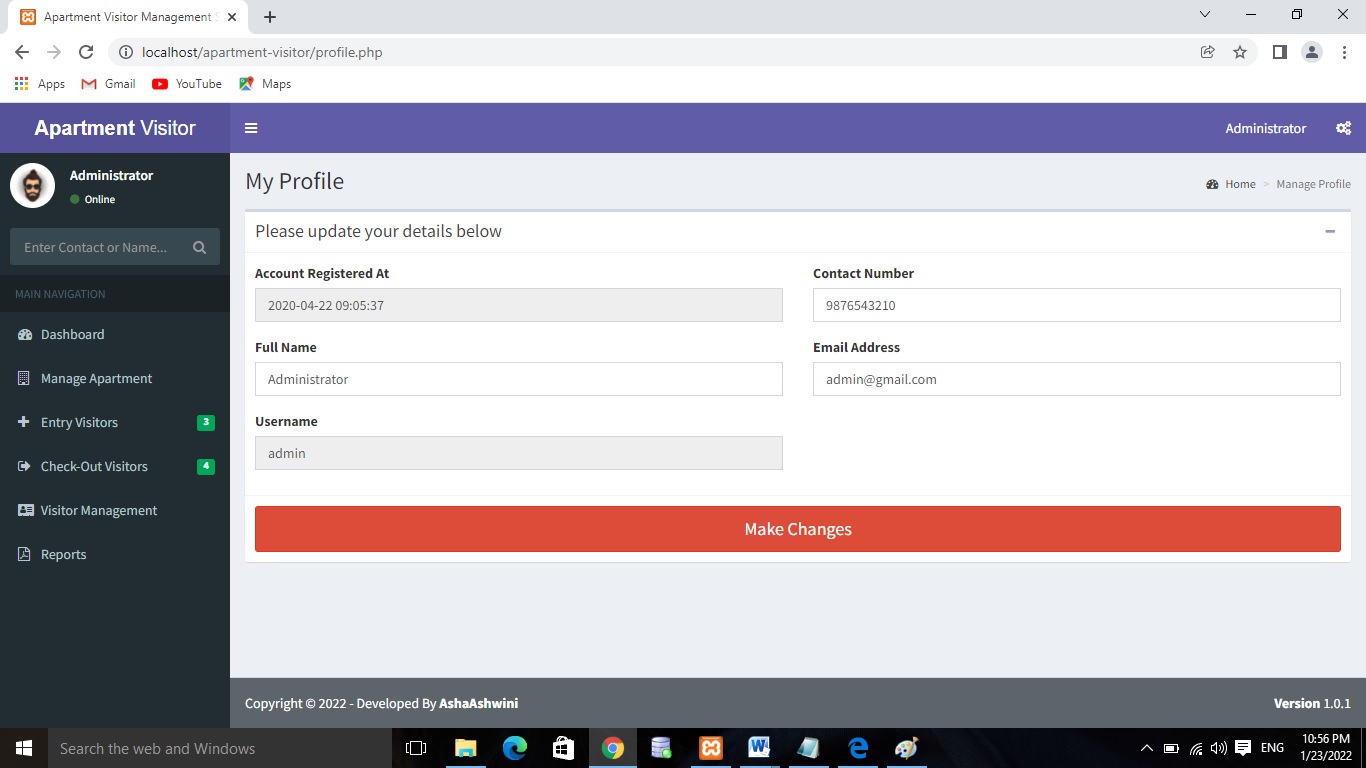


Fig 5.8 Admin Profile Page

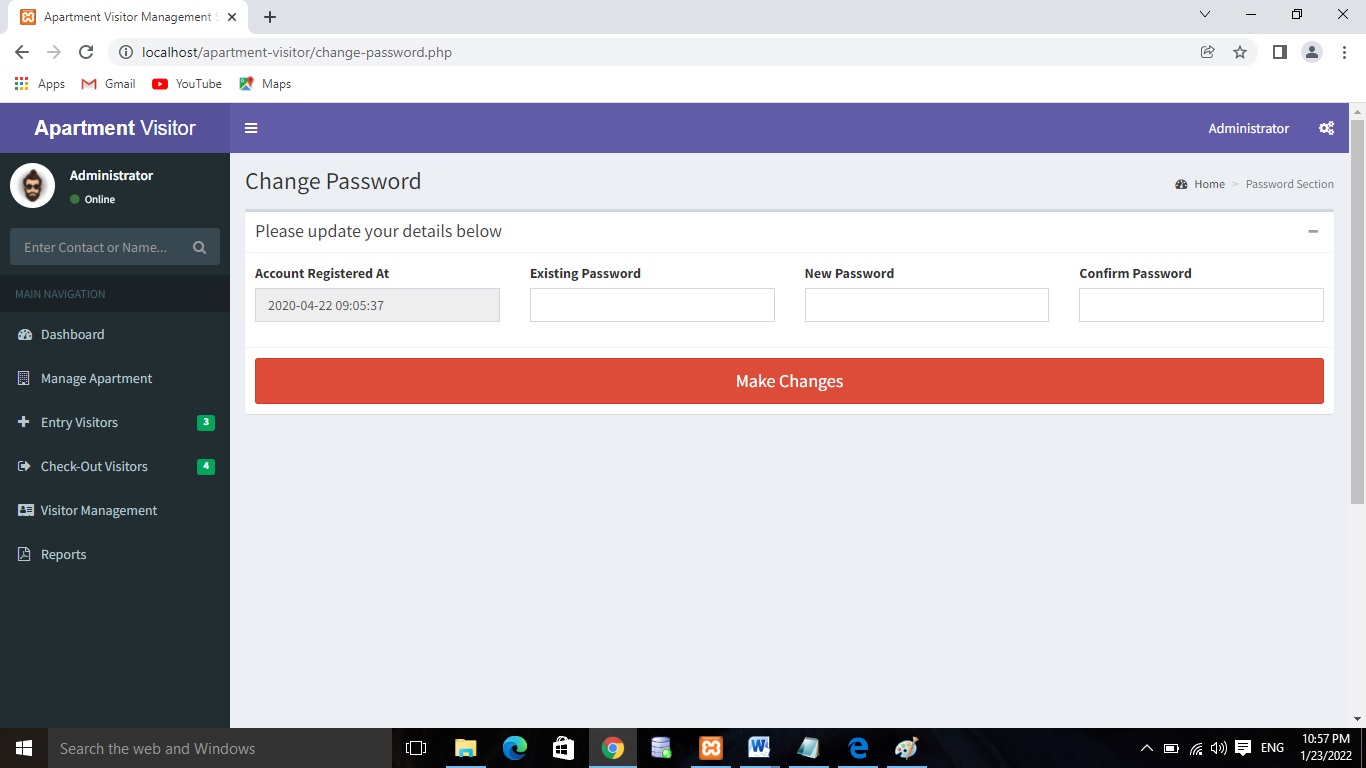


Fig 5.9 Change Password Page

**Chapter 6**

**CONLUSION AND FUTURE ENCHANCEMENTS**

**6.1 Conclusion**

This Application provides a computerized version of Apartment Visitor Management System which will benefit the society.

It makes entire process online and can generate reports. It has a facility of staff’s login where staff can fill the visitor details and generate report.

The Application was designed in such a way that future changes can be done easily. The following conclusions can be deduced from the development of the project.

* Automation of the entire system improves the productivity.
* It provides a friendly graphical user interface which proves to be better when compared to the existing system.
* It gives appropriate access to the authorized users depending on their permissions.
* It effectively overcomes the delay in communications.
* Updating of information becomes so easier.
* System security, data security and reliability are the striking features.
* The System has adequate scope for modification in future if it is necessary.

**6.2 Future Enhancements**

* Pre-registration of any person entering the facility or apartment premise
* Customized workflow maintenance for signing in various types of visitors & guests
* Providing the visitors with digital cards for minimizing contact and controlling in-office access
* Host notifications on employee/user mobile app as and when the visitor arrives
* Visitor reporting and dashboard in real-time

**BIBLIOGRAPHY**

**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](http://www.mysqltutorial.org/)

**For XAMPP**

* <https://www.apachefriends.org/download.html>