**1.PROJECT SYNOPSIS**

INTRODUCTION

Finance is the process of raising funds or capitals for any kind of expenditure. It is a financial

entity which belongs to its members , who are at the same time the owners and the customers

of their finance. Finance are often created by persons belonging to the same local or

professional community or sharing a common interest. finance generally provide their

members with financial services ( loans, deposits, accounts…..).

The web application will draw in some new customers and will notify the existing customers about the recent happenings in the finance. It will also help the customers to get a variety of information about the services and facilities of the finance . The web application will ease certain tasks like locating branches and getting the exact monthly instalment on loans. The potential customers can also create a account online using the web application.

OBJECTIVES:

“E-FINANCE SYSTEM”, is a Dynamic web application which provides information about

the finance to its customers. THE OBJECTIVE is to prepare a software or application, which

could maintain data & provide a user friendly interface for retrieving customer related details

just in few seconds, with 100% accuracy. Software is completely computerized, so it is not

time consuming process. No paper work required & can be implemented further..

INPUTS OF THE PROJECT

* As this system is going to be computerized we need all the information about the finance’s services.
* The types of accounts the finance provides along with the respective rate of interests.
* We need all the information about the different types of loans provided and the interest on them so as to calculate monthly EMI and the rate of interest.
* We even need info about guidelines to open different types of account.

OUTPUT OF THE PROJECT

* This system provides the info about the different types of finance accounts, guidelines to open accounts, different rate of interests for the respective account.
* The system also provides info about the different types of loans provided by the finance.
* It provides a very helpful tool called “loan calculator”, to calculate the amount of interest and the total loan amount.

PROCESS LOGIC:

**INPUTS:**

**1) ACCOUNT DETAILS:**

**i) Holder name**

**2) LOAN DETAILS:**

**i) Loan amount**

**ii) Principle**

**iii) Loan period**

**iv) Rate of interest**

**3) SCHEMES:**

**i) Actual amount**

**ii) Scheme amount**

**iii) Eligibility**

**OUTPUT:**

1. **GENERATION OF ACCOUNT NUMBER**
2. **Loan EMI Chart**
3. **New Schemes Available**

LIMITATION OF THE PROJECT:

* This application doesn’t imply net banking as there is no ATM service available yet in the finance.
* It doesn’t display the particulars of the customer’s account details to another customer.

FUTURE SCOPE OF THE PROJECT:

* Android application can be developed.
* Online payment can be implemented.

**2 .FRAMEWORK**

2.1. Introduction

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy and it is a process of having the systems personnel check out and put new products into use and construct any files of data needed to use it.

2.1. Why you need WAMP, MySQL and PHP?

WAMP server allows developers to test PHP scripts locally. Additionally dynamic websites are dependent on stored information that can be accessed easily; this is the main difference between a dynamic site and a static HTML site. PHP is a powerful scripting language that can be run by itself in the command line of any computer with PHP installed. However PHP alone is not enough in order to build dynamic web sites. To use PHP on a web site, you need a server that can process PHP scripts However PHP does not provide a simple, efficient way to store data. This is where a relational database management system like MySQL comes into play.

2.2. PHP:

PHP originally stood for “Personal Home Page” and was released as a free and open source project. Over time, the language was reworked to meet the needs of its users. In 1997, PHP was renamed to the current “PHP:Pre-processorHypertext.” PHP is generally used as a server-side scripting language; it is especially well-suited for creating dynamic web pages and client-side GUI applications. PHP generally runs on web server, taking PHP code as its input and creating web pages as output. The scripting language features integrated support for interfacing with databases such as MySQL, which makes it a prime candidate for building all manner of web applications, from simple personal web sites to complex enterprise-level applications.

Unlike HTML, which is parsed by a browser when a page loads, PHP is pre-processed by the machine that serves the document .all PHP code contained with the document processed by the server before the document is sent to the visitor’s browser. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems. . It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use. Many programming languages require that you compile that file into machine code before they can be run, which is a time consuming process. Bypassing the need to compile means you are able to edit and test code much more quickly because PHP is server side language, running PHP scripts on your local machine requires installing a server on your local machine.

PHP is free software released under the PHP License; however it is incompatible with the GNU General Public License (GPL), due to restrictions on the usage of the term PHP. It is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking PHP code as its input and creating web pages as output. It can be deployed on most web servers and on almost every operating system and platform free of charge. PHP is installed on more than 20 million websites and 1 million web servers. PHP originally stood for Personal Home Page.

2.3.HTML

**2.3.1.Hyper Text Mark-up Language (HTML)** is used to creating the web page either of Static or of Dynamic and used to develop the user friendly web pages.

HTML is used for developing web pages.HTML is popularly used in World Wide Web(WWW).It uses ASCII characters for both the main text and formatting instructions. The main text is data and the whole information is used by the browser to format the data. A HTML document is simply a text file, which contains certain information you would like to publish.

A set of instructions embedded in a document is called Markup language. These instructions describe what the document text means and how it should look in a display. The language also tells you how to make a document with other document on your local system, the World Wide Web and other Internet resources such as FTP.

The global publishing format of the Internet is HTML. It allows authors to use not only text but also format that text with headings, list and tables, and also includes still images videos, and sound within text. Readers can access pages information from anywhere in the world at the click of mouse button information can be downloaded to readers own PC or workstations HTML pages can also be used for entering a data and as a front end for commercial transaction.

2.3.2. Features of HTML

* It is a programming language.
* It is not a data description language.
* It is simple to understand and implement.
* HTML constructs are very easy to comprehend, and can be used effectively by anybody.
* The methodology used by HTML to mark up information is independent of its representation on a particular hardware or software architecture.

2.4. Usage

PHP is a general-purpose scripting language that is especially suited for web development. PHP generally runs on a web server, taking PHP code as its input and creating web pages as output. It can also be used for command-line scripting and client-side GUI applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

PHP primarily acts as a filter, taking input from a file or stream containing text and/or PHP instructions and outputs another stream of data; most commonly the output will be HTML. It can automatically detect the language of the user. From PHP 4, the PHP parser compiles input to produce byte code for processing by the Zend Engine, giving improved performance over its interpreter predecessor. Originally designed to create dynamic web pages, PHP’s principal focus is server-side scripting, and it is similar to other server-side scripting languages that provide dynamic content from a web server to a client, such as Microsoft’s Active Server Pages, Sun Microsystems’ Java Server Pages, and mod\_perl. PHP has also attracted the development of many frameworks that provide building blocks and a design structure to promote rapid application development (RAD). Some of these

includeCakePHP, Symphony, Code Igniter, and Zend Framework, offering features similar to other web application frameworks.

The WAMP architecture has become popular in the web industry as a way of deploying web applications. PHP is commonly used as the PHP in this bundle alongside Linux, Apache and MySQL, although the P may also refer to Python or Perl.As of April 2007, over 20 million Internet domains were hosted on servers with PHP installed, and PHP was recorded as the most popular Apache module. Significant websites are written in PHP including the user-facing portion of

Face book, Wikipedia (MediaWiki), Yahoo!, My Yearbook, , Digg, Wordpress and Tagged.In addition to server-side scripting, PHP can be used to create stand-alone, compiled applications and libraries, it can be used for shell scripting, and the PHP binaries can be called from the command line.

2.5.MY SQL:

What is a database? Quite simply, it’s an organized collection of data. A database management system (DBMS) such as Access, FileMaker Pro, Oracle or SQL Server provides you with the software tools you need to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents.

MySQL is a multithreaded, multi-user SQL database management system (DBMS). The basic program runs as a server providing multi-user access to a number of databases. Originally financed in a similar fashion to the JBoss model, MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL LAB now a subsidiary of Sun Micro system , which holds the copyright to most of the codebase.

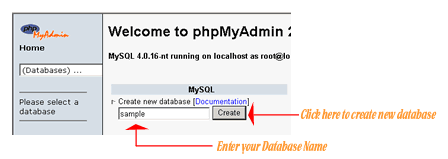
The project’s source code is available under terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is a database. The data in MySQL is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows. Databases are useful when storing information categorically.

Security

View are basically used as a part of security, means in many organizations ,the end user will never be given original tables & all data entry will be done with the help of views only. But the data base administrator will be able to see everything because all the operations done by the different users will come to the same table.

Steps to create a database in PHPMyAdmin:

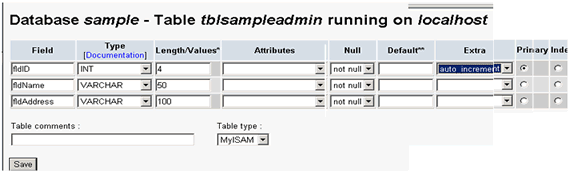
1) The following figure shows your PHPMyAdmin interface, just enter your database name and click the 'Create' button to create your database.



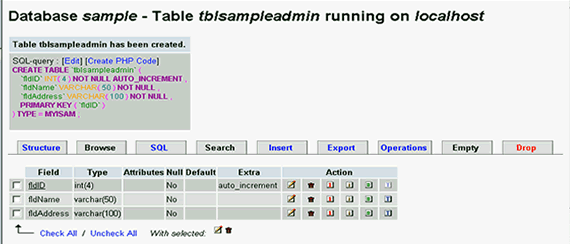
2) Now to create a new table enter your table name and the number of fields in the table, then click the 'Go' Button.



3) The next step is to create the fields, just enter values for each field name, type, length of the field, null option and mention whether it is a primary key or not. Then click the 'Save' button to complete your table creation.



4) The following figure is displayed upon successful creation of your table



**3. PROJECT SUBJECT**

3.1 OBJECTIVE

“E-FINANCE SYSTEM”, is a Dynamic web application which provides information about

the finance to its customers. THE OBJECTIVE is to prepare a software or application, which

could maintain data & provide a user friendly interface for retrieving customer related details

just in few seconds, with 100% accuracy. Software is completely computerized, so it is not

time consuming process. No paper work required & can be implemented further.

**3.2 EXISTING SYSTEM:**

There is no such system which provides information of the finance. If a customer wants to know the work procedure or information of the finance, he needs to visit the finance personally. Hence it is a tedious task for the employees to provide the customers about the bank details or account details again and again, hence affecting the flow of their regular work.

**3.3 PROPOSED SYSTEM:**

THIS SYSTEM HELPS TO OVERCOME THE PROBLEM LIKE:

* It gives the whole and sole control of the application to the admin
* It avoids the misguidance or inappropriate info about the finance to the customers.
* It establish the accurate way of providing appropriate info about various services of the finance.
* It helps in reduction of clerical work as all the info is available online.
* It is very easy to maintain as the system is user friendly and is easily understandable.

3.4 MODULES

* ADMIN :-

He is the person who manages the whole & sole activities of the branches. The admin updates all the necessary activities for the customer related to the that finance.

* PIGMY COLLECTOR:-

He is the person work under the admin and he is collect the amount from the account

holders to their doorstep.

* END USER :-

They are the customers who waits to gain knowledge regarding the finance they can download the a/c filling forms. They can give out feedback and suggestions and also file enquiries.

* CUSTOMER REGISTRATION:-

In this module admin specifies all the details of the account with the facility to register the account form and account opening related data.The End user or the customer is provided with an account number.

* LOAN DETAILS :-

To this module the loans provided by the finance will be updated with the facility to calculate the loans or interest rate or EMI’s related to the loan a/c will be calculated. The monthly installments can also be specified.

**4.SYSTEM REQUIREMENT AND ANALYSIS**

4.1 HARDWARE REQUIREMENT:

Processor : Pentium Dual Core with 2.0 GHz or Higher

Hard Disk : 30GB and Higher

Ram : 1 GB and Higher

4.2 SOFTWARE REQUIREMENT:

* Front end : HTML, CSS
* Designing tool : Dreamweaver
* Middle ware : PHP
* Back end : My SQL
* Server :Xampp Server
* Operating system : Windows XP and more

**4.3 Definitions , Acronyms and Abbreviations**

**Abbreviations:**

a)SRS: Software Requirement specification.

b) MSMS: Medical Store Management System.

c) HTML: Hyper Text Markup Language.

d) CSS: Cascading Style Sheets.

e) PHP: Personal Home Page or Hypertext Preprocessor.

f)xampp:Cross-,Apache, Mysql, and PHP and perl.

g) CGST: Central Goods and Services Tax.

h) SGST: State Goods and Services Tax.

**Definitions:**

SRS: SRS is a document that completely describes all of the functions of a proposed system and the constraints under which it must operate.

Database: Database is collection of all the information monitored by the system.

User: User is the person who uses software.

Admin: Admin is the one who maintains, administrates and updates the information.

**4.4.Specific Requirements**

**4.4.1 Functional Requirements :**

Functional requirements are statements of the services of the services that the system must provide or are description of how some computations must be carried out. The plan for implementing functional requirements is detailed in the system design.

We describe the functional requirements by giving various use cases.

**Use Case1:** Registration

Primary Actor: User

Pre Condition: Username should not exist already.

Main Scenario:

1. User will enter required information like username, contact number, mobile number and password etc.
2. Then clicks “submit” Button to store the details.

Alternate Scenario:

1. If username already exists then prompt “ Username exists” allow user to re-enter his username.
2. If password is incorrect Prompt “Incorrect password” then allow user to re-enter password**.**

**Use Case 2:** Login

Primary Actor: User

Pre Condition: User must be registered

Main Scenario:

1. Start the application. User prompted for login and password.

2. User gives the login and password.

3. System does authentication.

4. Main screen is displayed.

Alternate Scenario:

5. Authorization fails

5(a). Prompt the user that he typed the wrong password.

5(b). Allow him to re -enter the password. Give him 3 chances.

**Use Case 3:**Forgotten Password

Primary Actor: User

Main Scenario:

1. User forgotten the password.

2. System gives hint question.

3. System does Authentication.

4. password is displayed on the system.

Alternate Scenario:

5(a). Authorization fails.

5(b). System does not allow him to login.

**Use case related to Pigmy details:**

**Use case 4** : Adding pigmy details.

Primary Actor: Pigmy Agent

Pre Condition: Pigmy Agent logged in.

Main Scenario:

1. Admin initiates “Add new” functionality.

2. System displays Add new page where user enters each and every information about loan.

3. Admin clicks on “ADD”. New loan details is added.

Alternate Scenario:

4. loan with same name and conditions exist.

4(a). System asks Admin to enter different Loan details.

4(b). Admin enters different name for the Loan details.

4(c). Loan details is added.

**Use case 5:** Updating Pigmy Collection

Primary Actor: Pigmy Agent

Pre Condition: Agent logged in.

Main Scenario:

1. Agent initiates “Update Collection” functionality.

2. Agent updates information about pigmy.

3. Clicks on “save” ,prompt “Updated successfully”.

Alternate Scenario:

4. If not found.

4(a). Prompt “Pigmy details is not collected”.

**Use case 6**: Deleting Loan details.

Primary Actor: Admin

Pre Condition: Admin logged in / Loan details must be exist.

Main Scenario:

1. Admin initiates “Delete Loan details” functionality.

2. Admin Deletes Loan details. 3.

3.Clicks on “save” prompt “Deleted successfully”.

Alternate Scenario:

4. If not found.

4(a). Prompt “Product not found”.

**Use case 7:** Display Loan details.

Primary Actor: Admin

Pre Condition: Admin logged in / Loan details must exist.

Main Scenario:

1.Admin selects option “Loan details”.

2.System displays details of Loan with its name, description with etc.

**4.5 Non- Functional Requirements**

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that specific behaviour or functions. The plan for implementing non-functional requirements is detailed in the system architecture.

**Performance Requirements**

To run the application, users will require an internet connection-oriented phone or system with minimum configuration of

• Any of the browsers.

• Strong internet connection (Wi-Fi, 3G, or 4G)

• Accessibility permissions for notifications, storage, camera, Internet.

**Safety Requirements**

This web application will not affect data stored outside of its servers nor will it affect any other applications installed on the user’s device. It cannot cause any damage to the user’s device.

**Security Requirements**

This application requires username and password of each users or super user, login logic provides more security with highly encrypted passwords so that every user’s username and password should be checked in all manners. .

**4.6 Design constraints:**

In case of any error handling mechanisms have been provided

* Without proper authorization the data in the data base can’t be modified.
* Complete validation has been done so that no mandatory fields are empty by chance if any field is left empty then appropriate alert message will be displayed.
* In case of any errors Error handling mechanism have been provided

1. **Security:** The files in which information regarding securities should be secured against malicious deformations.
2. **Fault tolerance:** Data should not become corrupted incase of system crash or power failures.

**4.7 External interface requirements:**

**\*User Interface:**

The user interface allow the users to remotely access the system via several applications Users will be able to use the service through applications such as Mozilla Firefox, Microsoft Internet Explorer, etc. Allow the user to access the information fast and easily from remote locations

There will be two user types – the application user and the application administrator – each of which will have different interface with login id and password.

**User**

The minimal requirements are that the customers should be able to interact with the system through the interface provided by Constraint layout or Relative layout. There will be a different command for each of the following actions:

• Register and login to the web application.

• Search particular products.

• View related description and price of different medicines.

• Order the required medicines and get invoice.

• Administrator will get the enquired details.

**Administrator**

The minimal requirements are that the administrator should have a separate application and in addition to that, additional commands will allow the administrator to:

• Check the status of order

• Add products to the database

• Alter the cost structure of the products, respond to the queries.

• Send appropriate notifications.

**\*Hardware Interfaces**

Hardware Requirements (minimum) for the development of the web application:

• Intel Core i3 and above

• 4GB RAM and above

• 5GB HDD space and above

Application Hardware Requirement:

• Access to storage.

**\*Software Interfaces**

The development of the web application requires a system with the following software:

• Windows 7 + /Mac OSX 10.8.5 + /Ubuntu 12.04+ • MySQL as a database.

• Xampp server.

• Dreamweaver and Sublime as editor.

**5. DATA FLOW DIAGRAMS**

**Data Flow Diagram:**

The data flow diagram (DFD) is one of the important modeling tools. It shows the user of the data pictorially. DFD represents the flow of the data between different transformations and processes in the system. The data flow diagram shows logical flow of the data. It represents the functional dependencies within a system. It shows output values in a computation are derived from input values. It is a simple pictorial representation or model for system behaviour. It specifies, “What is to be done but not how is to be done”. It describes the logical structure of the system. It relates data information to various processes of the system. It follows top-down approach.

**Data Flow Diagram Notations:**

**Data Flow:**

It may be from file-to-file or file-to-process or process- process.

There may be either an input data flow or output data flow.

**Functional Processing:**

The process is nothing but the transformation of data it starts with the subject and it has the verb followed by the subject.

**Data store:**

It includes file, data base and repository. To parallel lines represent it or a one end closed rectangle.

**Actor/source/sink:**

The files which are outside the system and used by the process or the processes of the system. Generally

Source/Sink in the actor.

**Objectives:**

• To graphically document boundaries of a system.

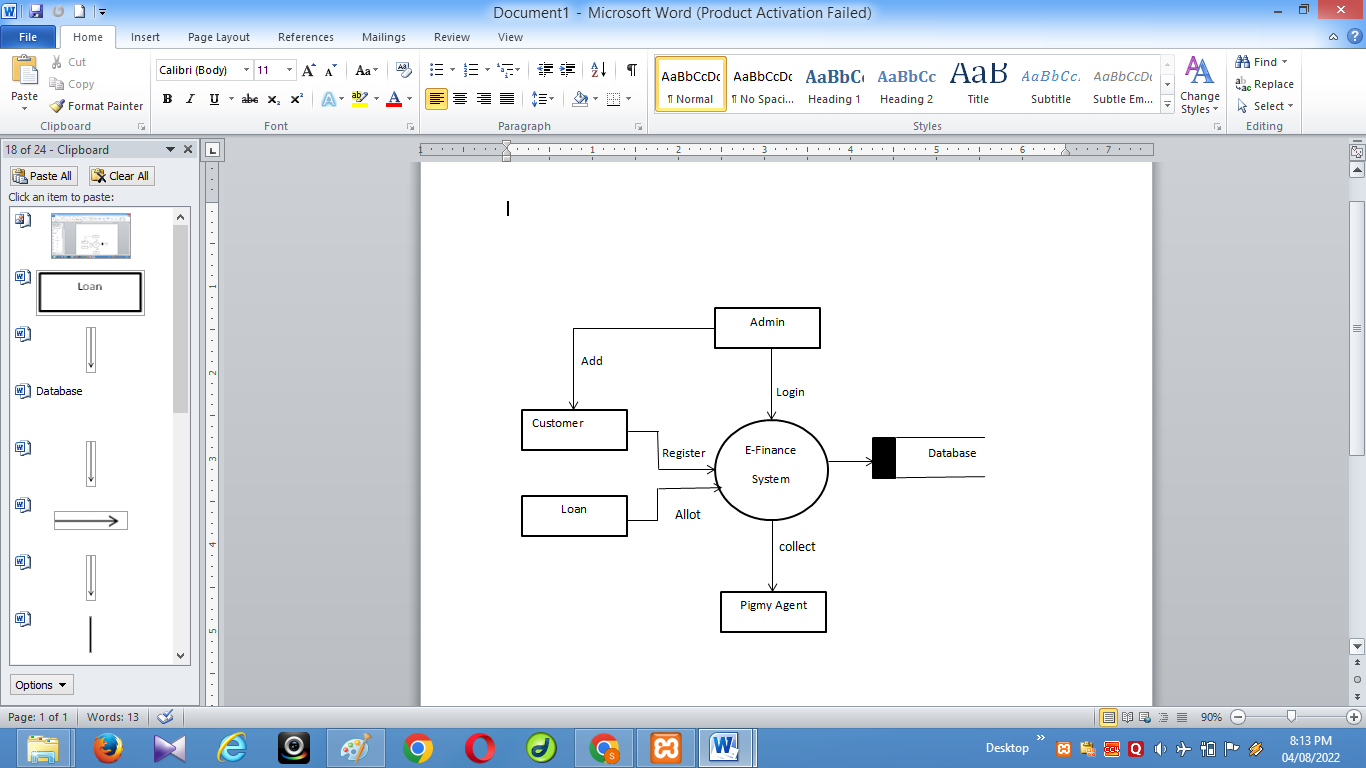
• To provide hierarchy breakdown of the system.

• To show movement of information between a system and its environment.

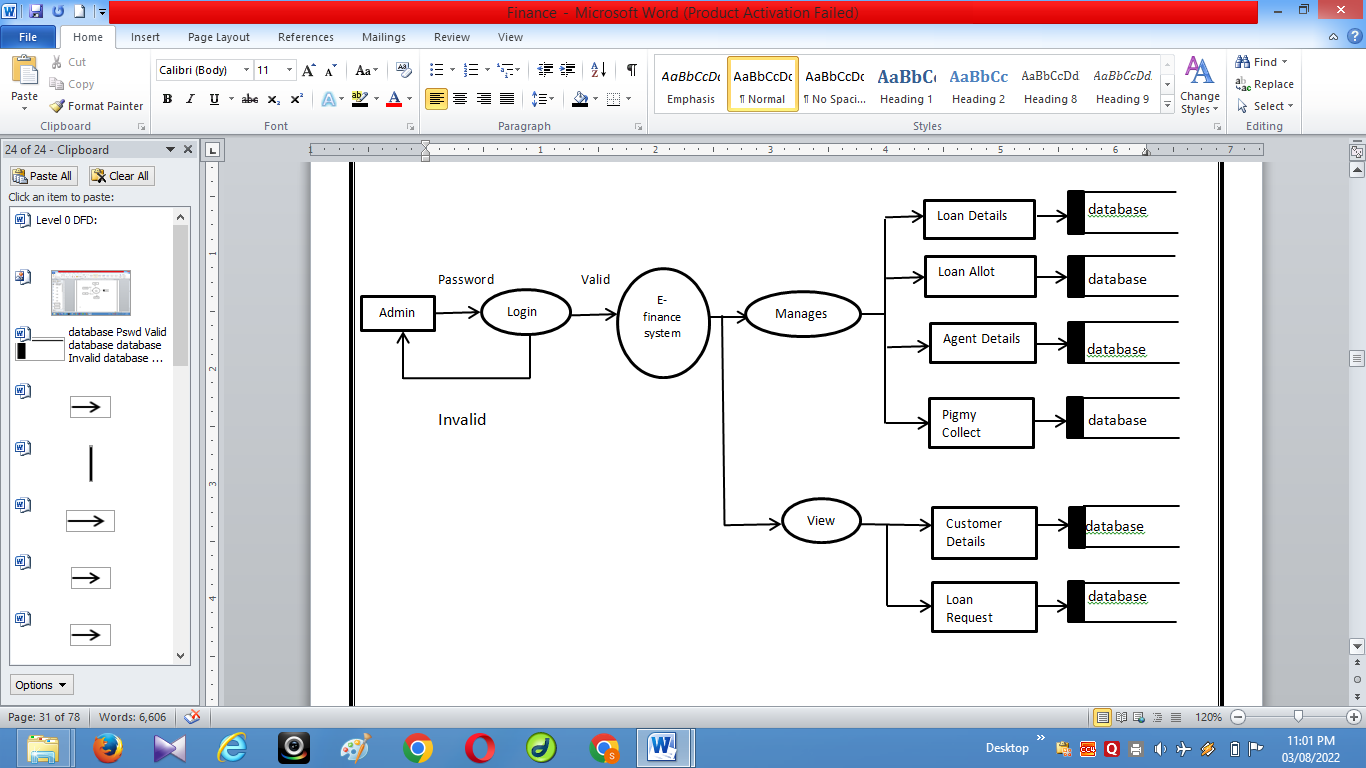
• To document information flows within the system.

• To aid communication between users and developers.

**Level 0 DFD:**



**Level 1 DFD:**



**6. ER-DAIGRAM**

**The ER Module**

An entity-relationship diagram is a data modelling technique that creates a graphical representation of the entities, and the relationships between entities, within an information system. An entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modelling is a database modelling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs. The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database. The data modelling technique can be used to describe any ontology (i.e. an overview and classifications of used terms and their relationships) for a certain area of interest. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model; this in turn is mapped to a physical model during physical design. Sometimes, both of these phases are referred to as "physical design".

**The three main components of an E-R Diagram are:**

• The entity is a person, object, place or event for which data is collected. For example, if we consider the information system for a business, entities would include not only customers, but the customer’s address, and orders as well. The entity is represented by a rectangle and labelled with a singular noun.

• The relationship is the interaction between the entities. In the example above, the customer places an order, so the word “places” defines the relationship between that instance of a customer and the order or orders that they place. A relationship may be represented by a diamond shape, or more simply, by the line connecting the entities. In either case, verbs are used to label the relationships.

• The cardinality defines the relationship between the entities in terms of numbers. An entity may be optional: for example, a sales representative could have no customers or could have one or many customers; or mandatory: for example, there must be at least one product listed in an order .There are several different types of cardinality notations; crow’s foot notation, used here, is a common one. In crow’s foot notation, a single bar indicates one, a double bar indicates one and only one (for example, a single instance of a product can only be stored in one warehouse), a circle indicates zero, and a crow's foot indicates many. The three main cardinal relationships are: one-to-one, expressed as 1:1; one-to-many, expressed as 1: M; and many-to-many, expressed as M: N.

**Entity Relationship Diagram Notations:**

Peter Chen developed ER Diagram’s in 1976. Since then Charles Bachman and James Martin have added some sleigh refinements to the basic ERD principle**.**

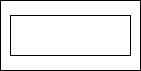
**Entity:**

An entity is an object or concept about which we want to store information. An entity is a real-world item or concept that exists on its own. In an ER model, we diagram an entity type as a rectangle containing the type name.

Description: entity.jpg

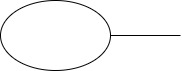
**Weak Entity:**

A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes.



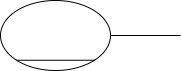
**Attribute:**

Each entity has attributes, or particular properties that describe the entity. Most of the data in a database consists of values of attributes. The set of all possible values of an attribute is the attribute domain. In an ER model, an attribute name appears in an oval that has a line to the corresponding entity box.



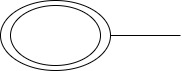
**Key attributes:**

A key attribute is the unique, distinguishing characteristic of the entity. An attribute or set of attributes that uniquely identifies a particular entity is a key. A key attribute in an ER Diagram is represented by an oval that has a line inside it and a line to the corresponding entity box. For example, an employee's social security number might be the employee's key attribute.



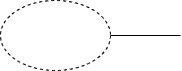
**Multi-valued attribute:**

A multi-valued attribute can have more than one value. We indicate this with a double oval. For example, an employee entity can have multiple skill values.

****

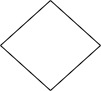
**Derived attribute:**

A derived attribute is based on another attribute. It is denoted by a oval and dotted line within it. For example, an employee's monthly salary is based on the employee's annual salary.



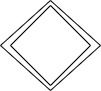
**Relationships:**

Relationships illustrate how two entities share information in the database structure.An association among entities is called a relationship. An attribute can also be a property of a relationship set. The association among the entities is described as one-to-one, one-to-many, many-to-many. A relationship is indicated by a rhombus**.**

****

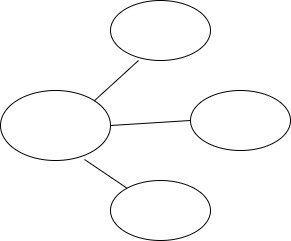
**Identifying relationship:**

Identifying relationship is denoted by double rhombus



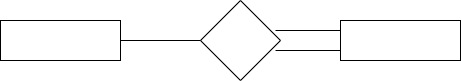
**Composite Attribute**

A composite attribute has multiple components and each component is atomic or composite. We illustrate this composite nature in the ER model by branching off the component attributes.



**Total Participation:**

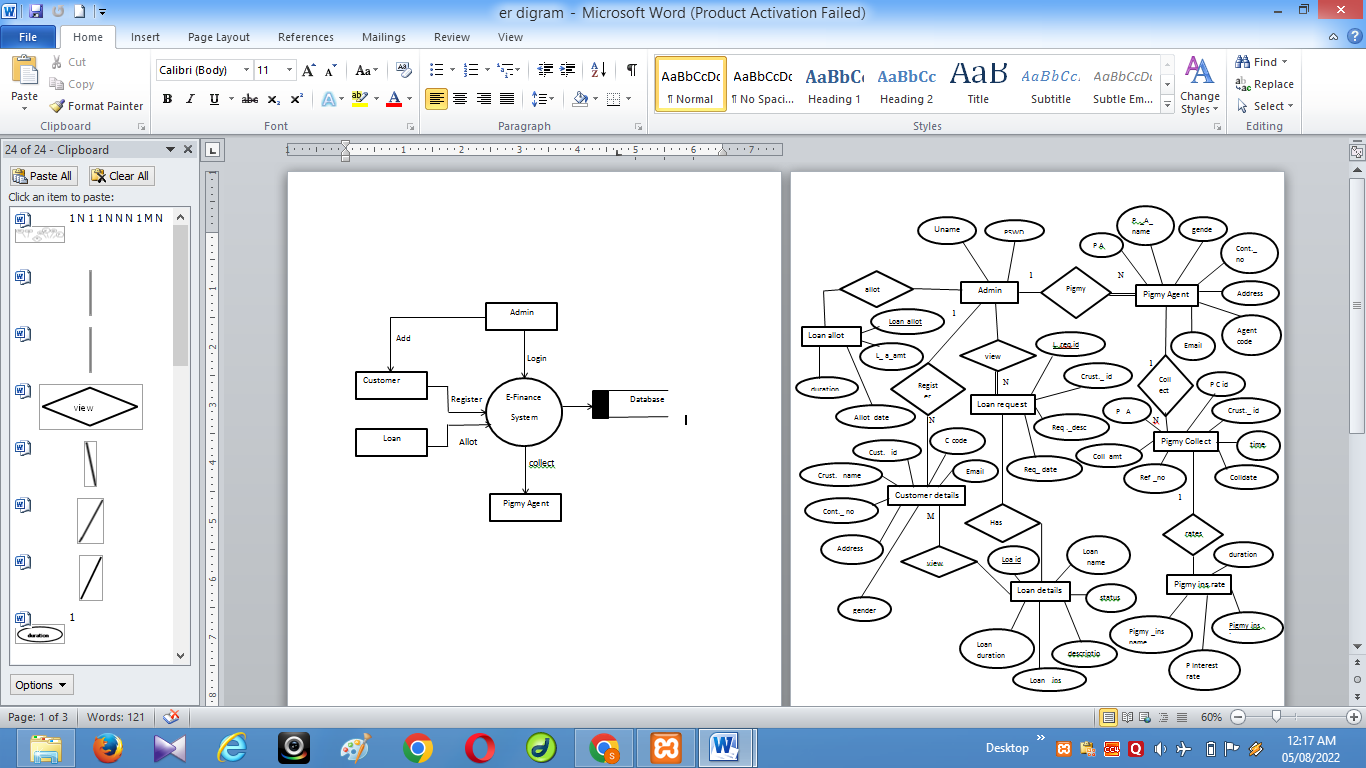
Total participation is represented by a double line.



**Cardinality:**

Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinarily is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinarily describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinarily specifies the absolute minimum number of relationships.

**E-R diagram:**

****

**7.DATABASE TABLES**

**Login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| Username | varchar | 100 | Not null |  |
| Password | varchar | 100 | Not null |  |
| Type | varchar | 50 | Not null |  |
| hintq | varchar | 100 | Not null |  |
| hinta | varchar | 200 | Not null |  |

**Customer Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Customer id |  |  |  |  |  |  |  | | varchar | 10 | Not null | Primary key |
| Customer full name | varchar | 20 | Not null |  |
| Gender | varchar | 50 | Not null |  |
| Occupation | varchar | 20 | Not null |  |
| City | varchar | 20 | Not null |  |
| Email id | varchar | 15 | Not null |  |
| Contact no | bigint | 15 | Not null |  |
| Customer Code | varchar | 10 | Not null |  |

**Loan Allot**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Loan allot id |  |  |  |  |  |  |  | | int | 10 | Not null | Primary key |
| Loan id | varchar | 20 | Not null |  |
| Customer id | varchar | 20 | Not null |  |
| Date | date |  | Not null |  |
| Allot amount | int | 10 | Not null |  |
| Duration(month) | int | 10 | Not null |  |

**Loan Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Loan id |  |  |  |  |  |  |  | | int | 20 | Not null | Primary key |
| Loan name | varchar | 50 | Not null |  |
| Loan description | int | 11 | Not null |  |
| Loan ins | varchar | 25 | Not null |  |
| Status | varchar | 50 | Not null |  |

**Loan Request**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Loan request id |  |  |  |  |  |  |  | | int | 11 | Not null | Primary key |
| Customer id | varchar | 11 | Not null |  |
| Customer id | varchar | 20 | Not null |  |
| Loan id | int | 11 | Not null |  |
| Request description | varchar | 200 | Not null |  |
| date | date |  | Not null |  |

**Pigmy Agent**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Pigmy agent id |  |  |  |  |  |  |  | | int | 11 | Not null | Primary key |
| Pigmy agent name | varchar | 10 | Not null |  |
| Gender | varchar | 15 | Not null |  |
| Address | varchar | 15 | Not null |  |
| Email id | varchar | 15 | Not null |  |
| Contact no | bigint | 20 | Not null |  |
| Agent code | varchar | 20 | Not null |  |

**Pigmy Collect**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Pigmy collect id |  |  |  |  |  |  |  | | int | 25 | Not null | Primary key |
| Customer id | varchar | 30 | Not null |  |
| Pigmy agent name | int | 11 | Not null |  |
| Collect amount | bigint | 10 | Not null |  |
| Collect date | date |  | Not null |  |
| time | varchar | 6 | Not null |  |
| Reference no | int | 20 | Not null |  |

**Pigmy interest rates**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraints** | **Key** |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Pigmy interest id |  |  |  |  |  |  |  | | int | 11 | Not null | Primary key |
| Pigmy interest name | varchar | 50 | Not null |  |
| Pigmy agent name | int | 11 | Not null |  |
| Interest rate | int | 11 | Not null |  |
| Duration(month) | varchar | 11 | Not null |  |

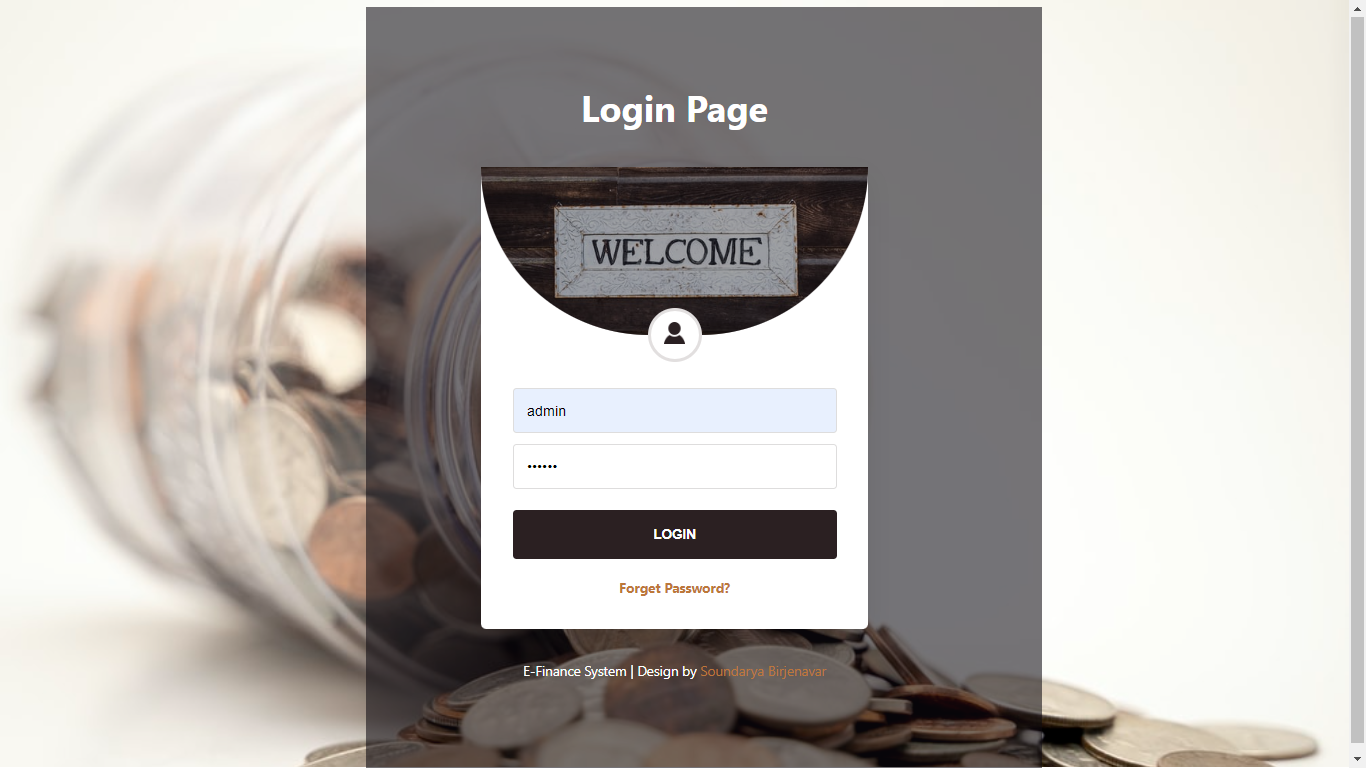
**8. SCREEN SHOTS**

**User Home Page**



The above screen shows the Home page of this project.

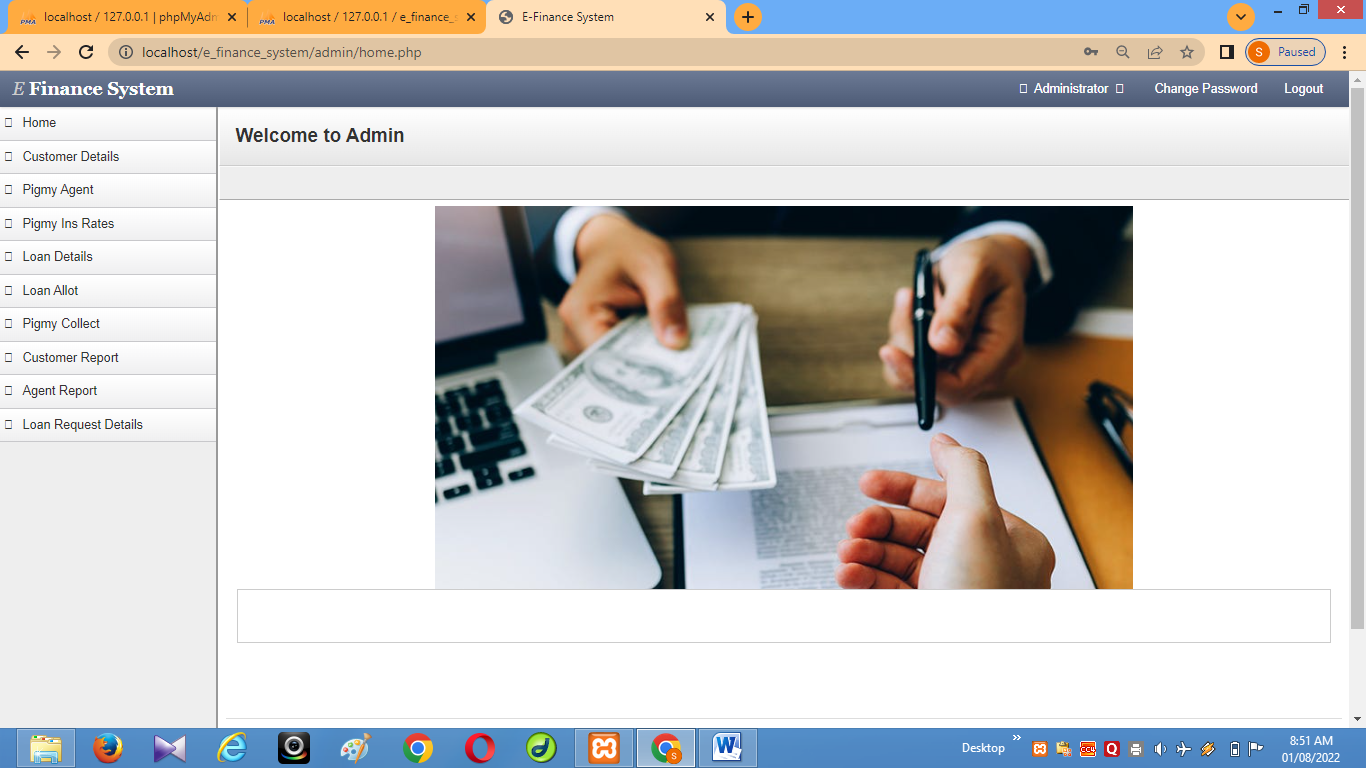
**Login Page**



The above screen shows the login of this project. The username & password is inserted.

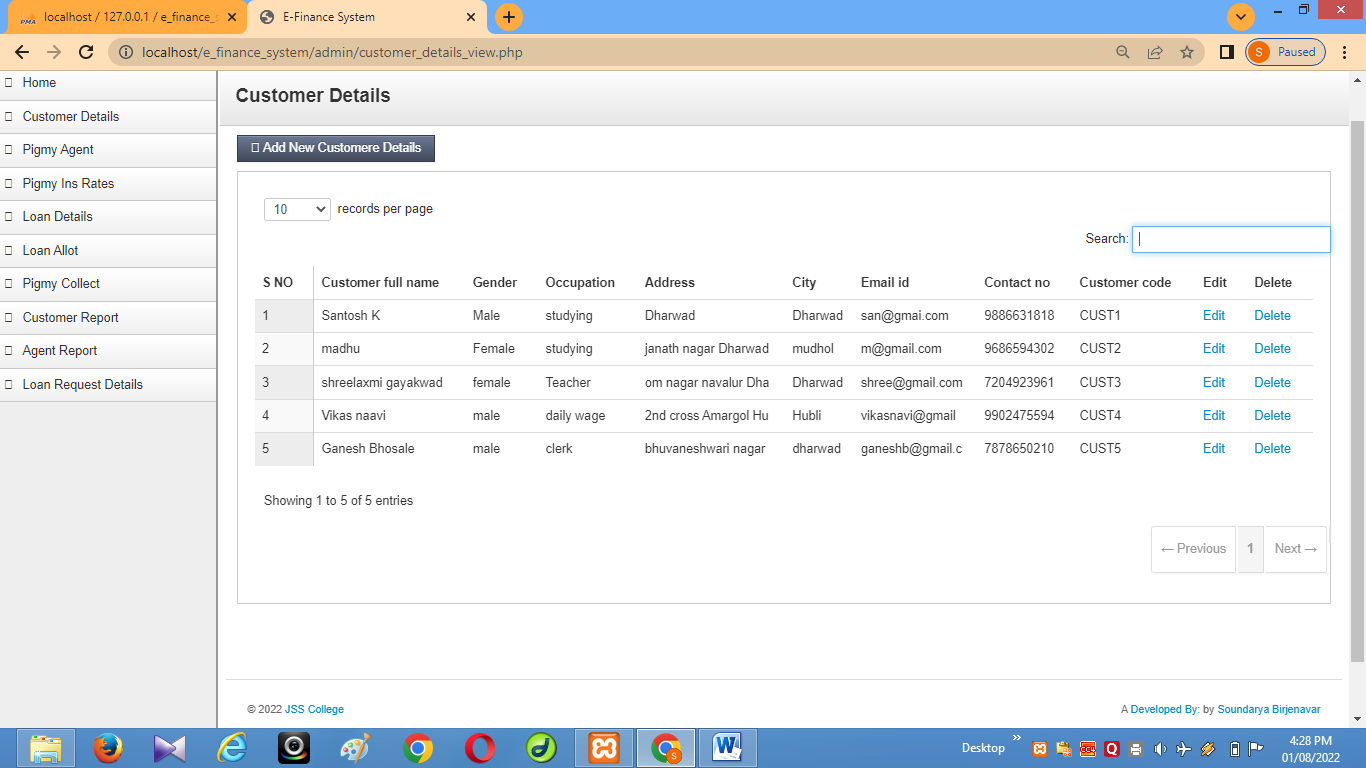
**ADMIN MODULE**

**Home Page**

****

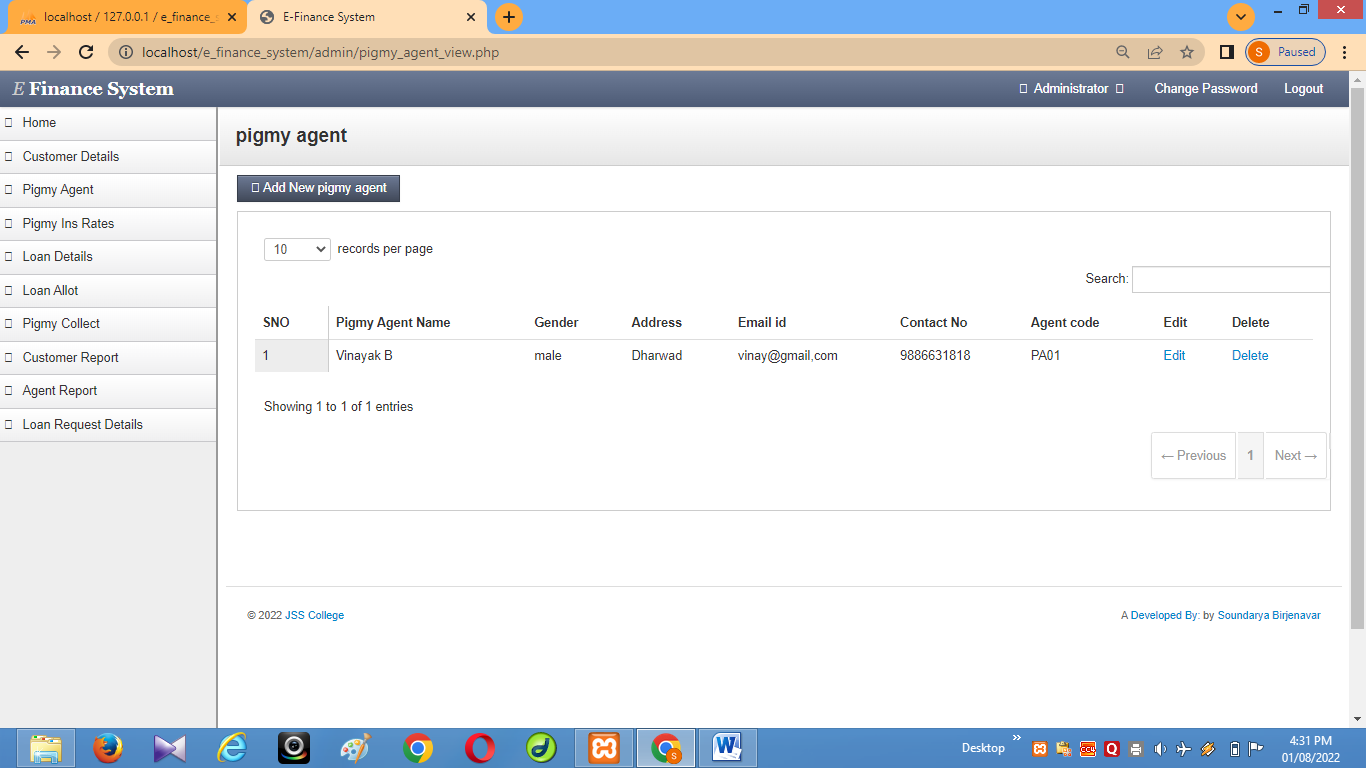
The above screen shows the Home page of the Admin.

**Customer Details**



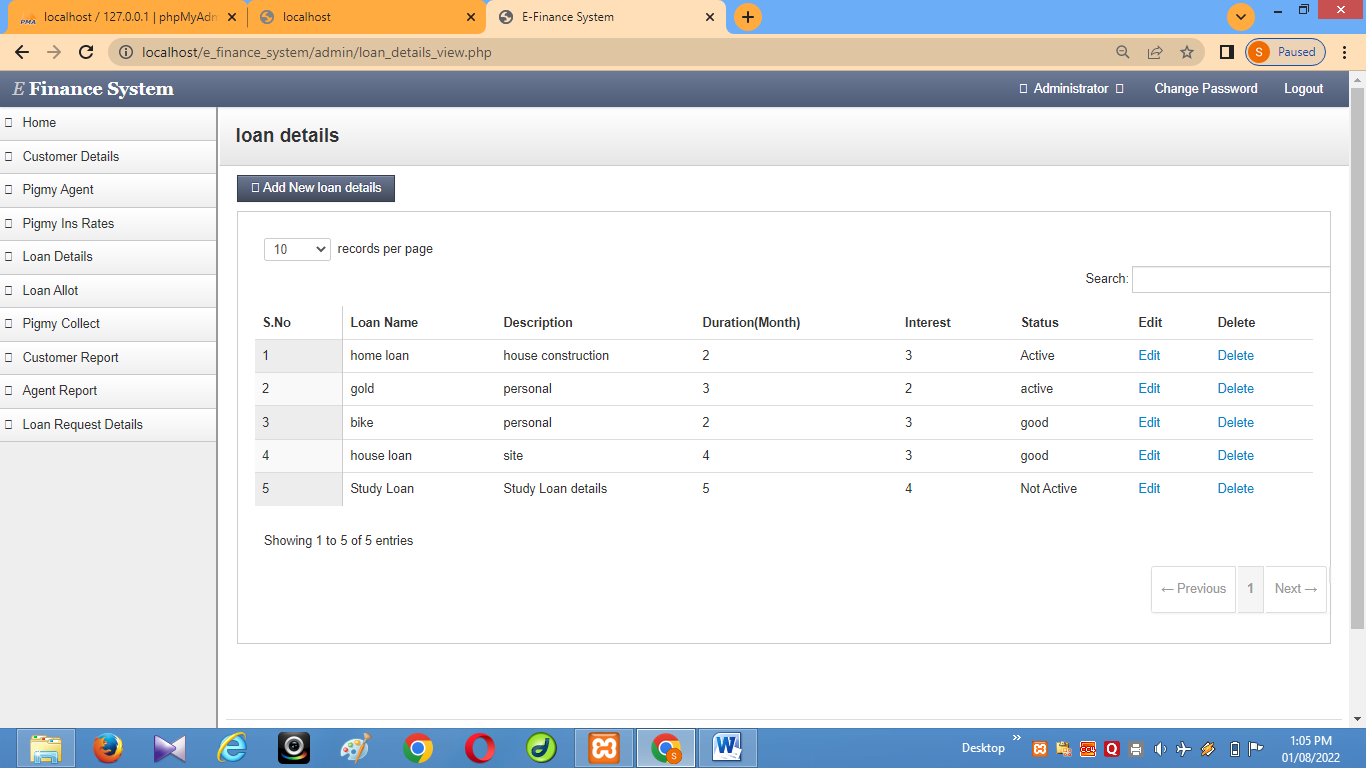
The above screen shows the Customer details view which has the available tasks to be performed.

**Pigmy Agent**

****

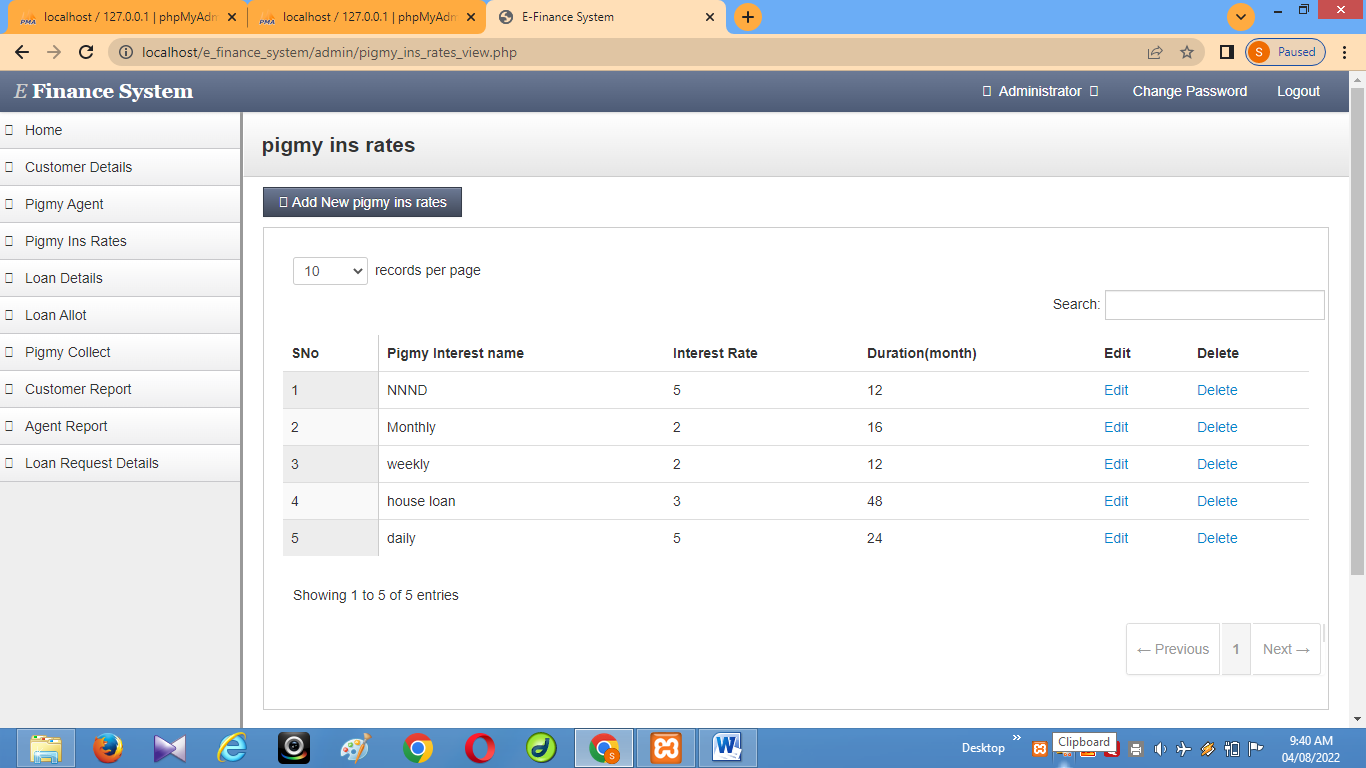
The above screen shows the Pigmy Agent view which has the available tasks to be performed.

**Loan Details**

****

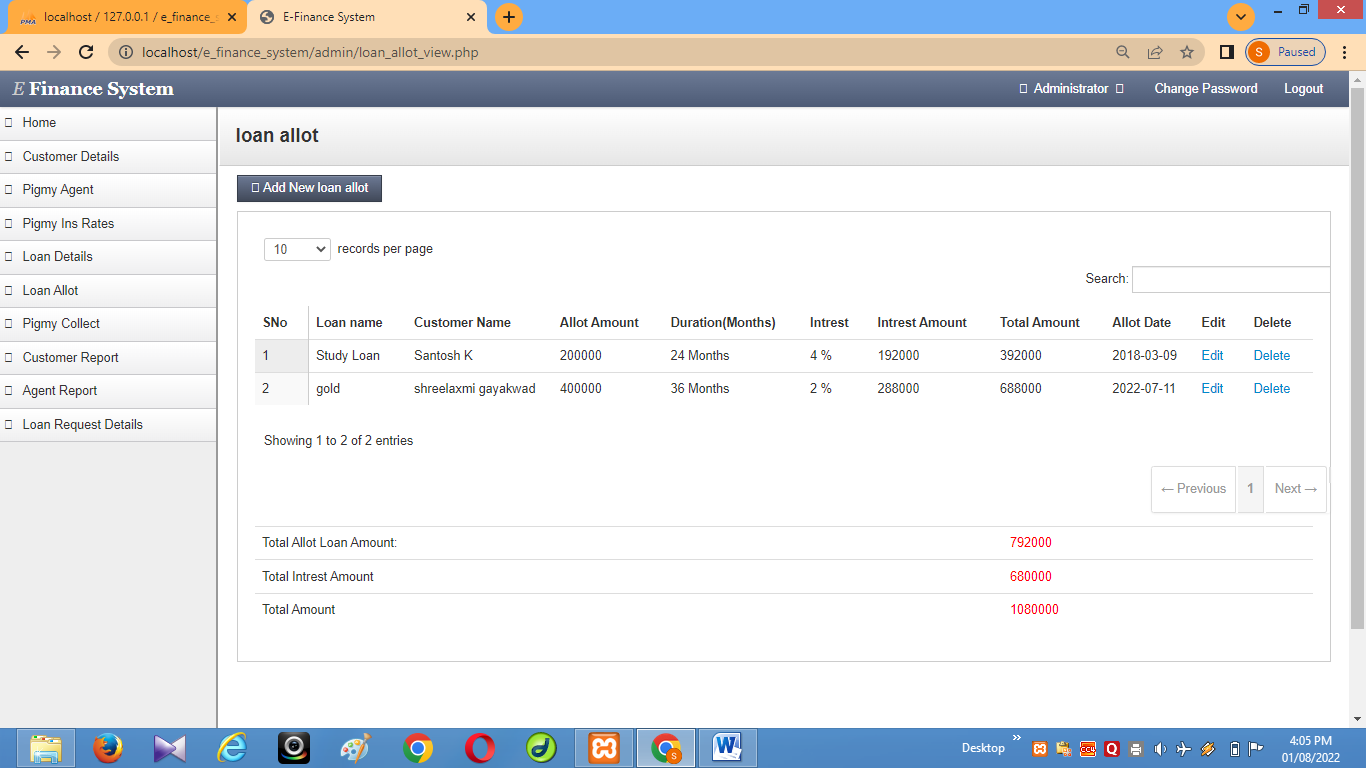
The above screen shows the Loan details view which has the available tasks to be performed.

**Pigmy Interest Rates**



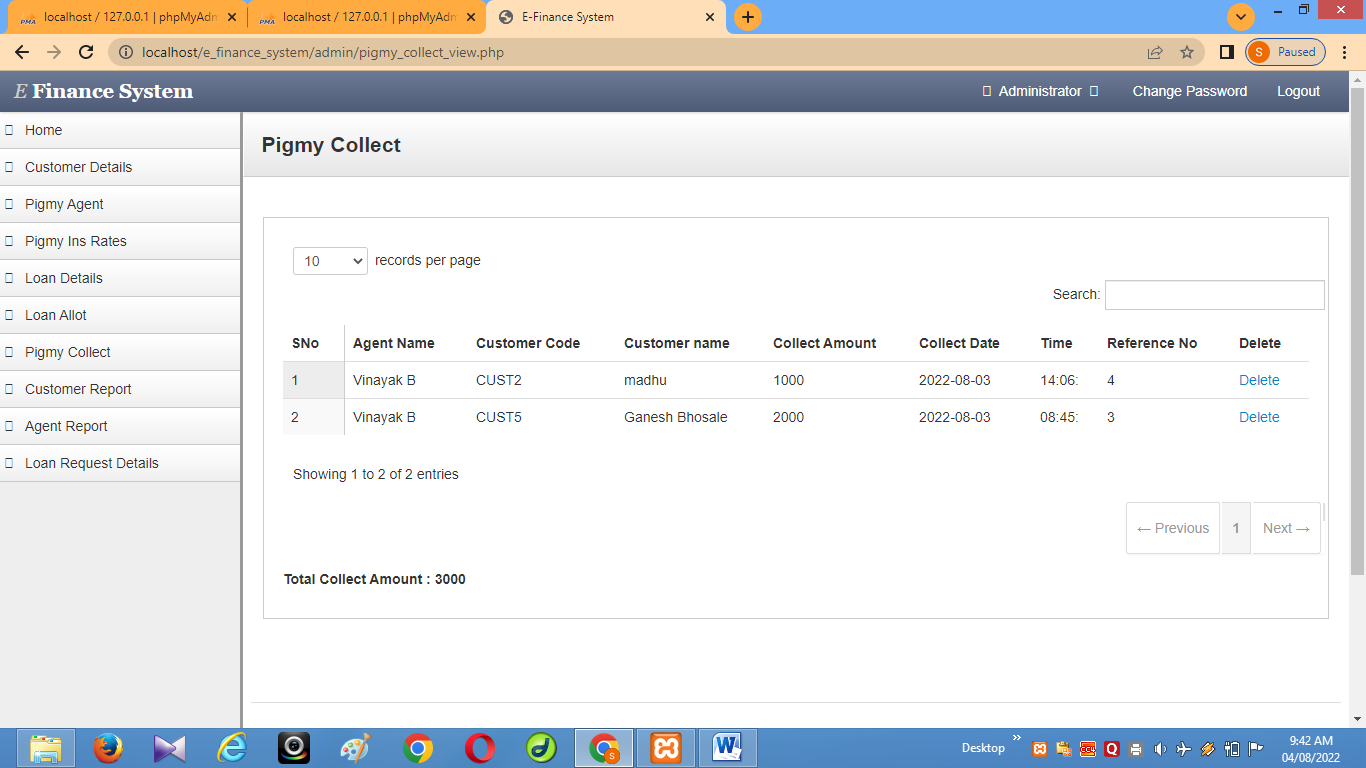
The above screen shows the Pigmy interest rates view which has the available tasks to be performed.

**Loan Allot**



The above screen shows the Loan allot details view which has the available tasks to be performed.

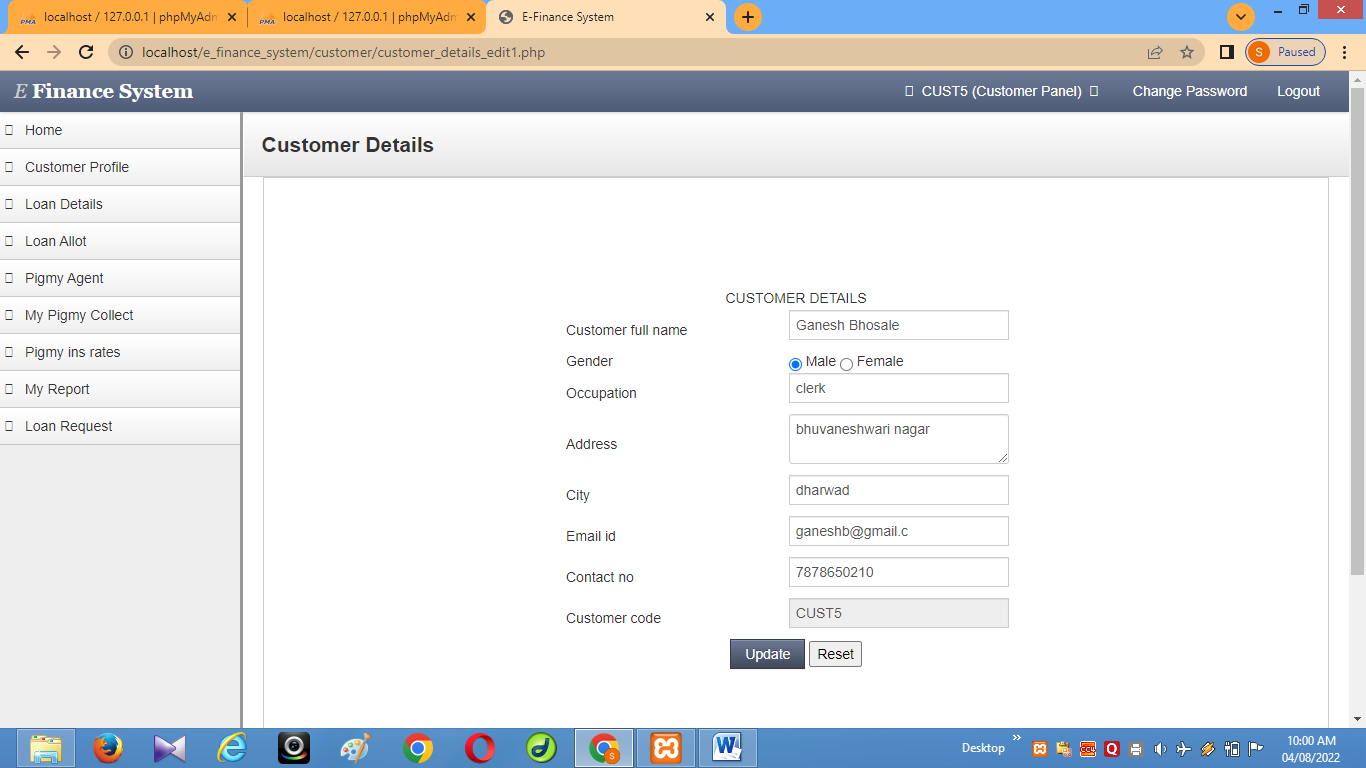
**Pigmy Collect**

****

The above screen shows the Pigmy Collect details view which has the available tasks to be performed.

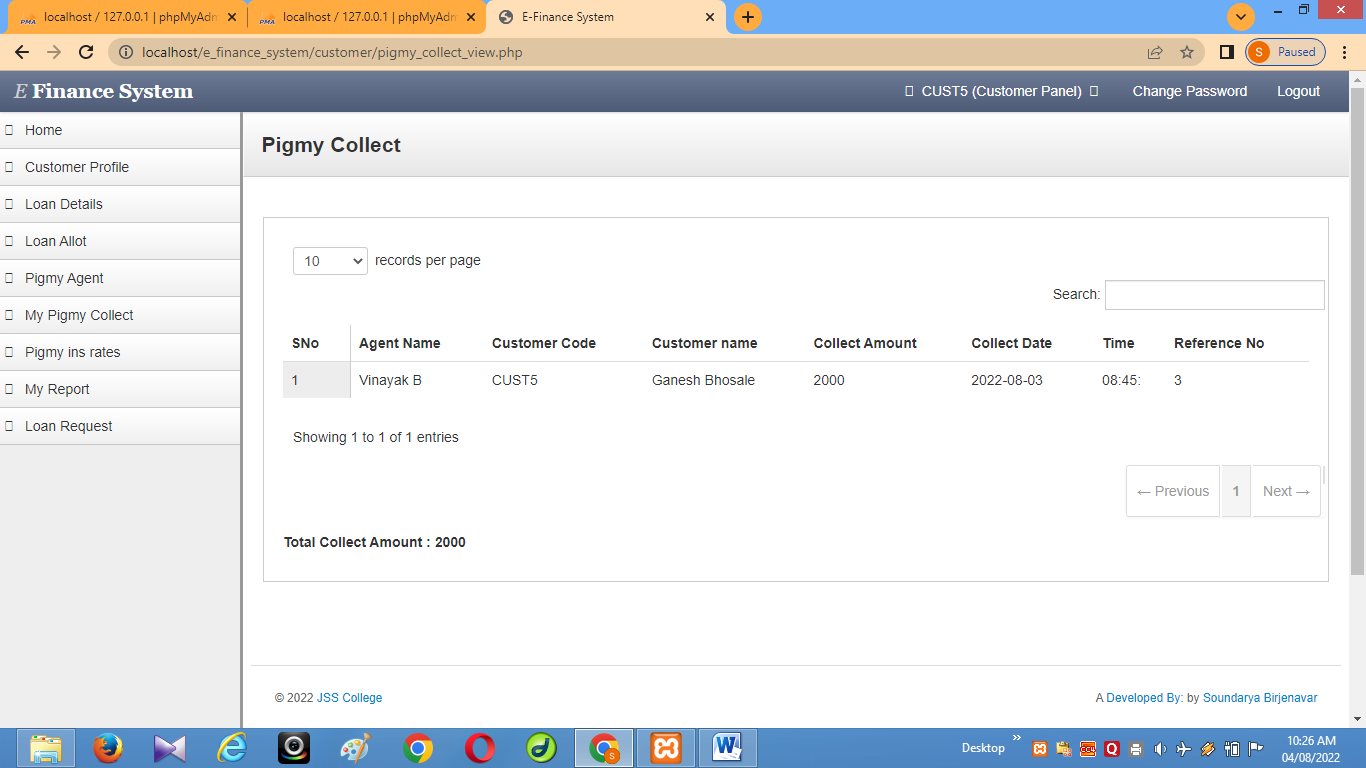
**CUSTOMER MODULE**

**Customer profile**



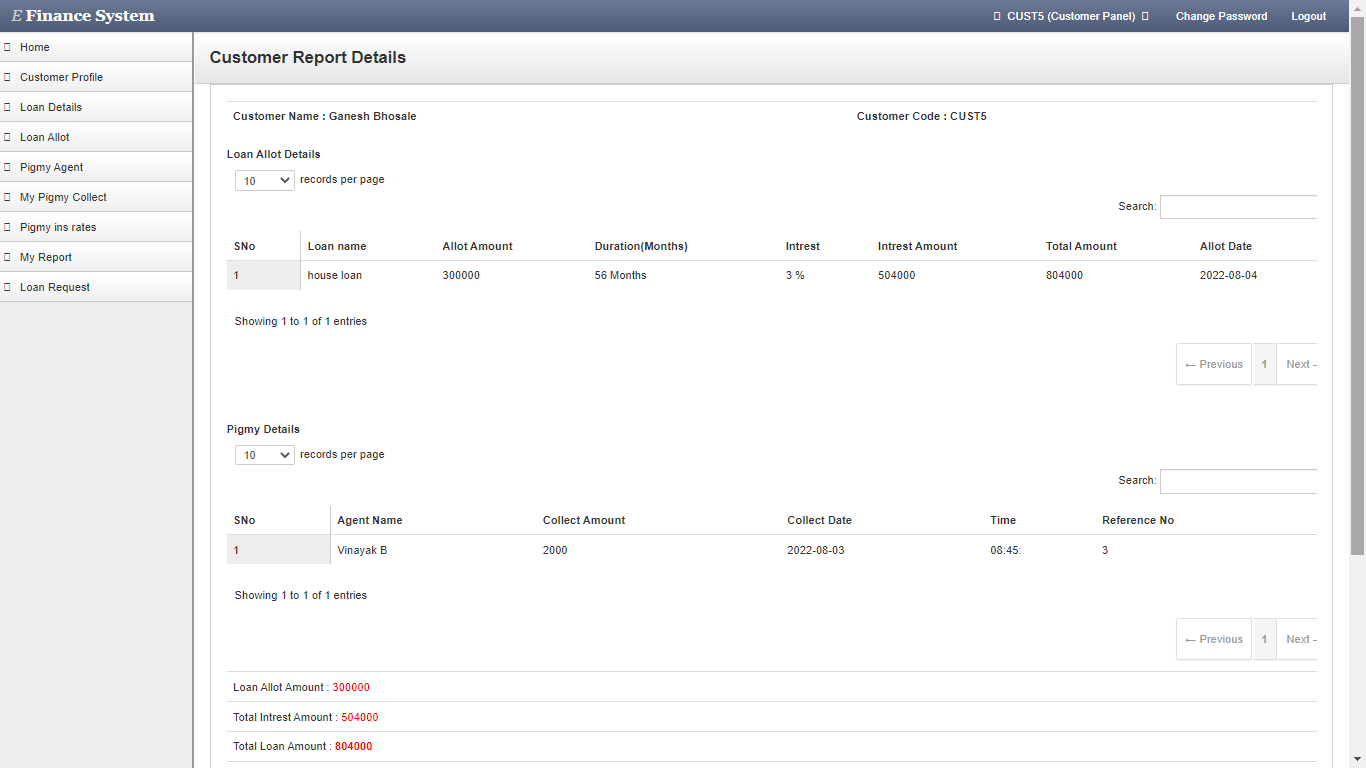
The above screen shows the Customer details view which has the available tasks to be performed.

**My Pigmy Collect**

****

The above screen shows the Pigmy collect view to the customer.

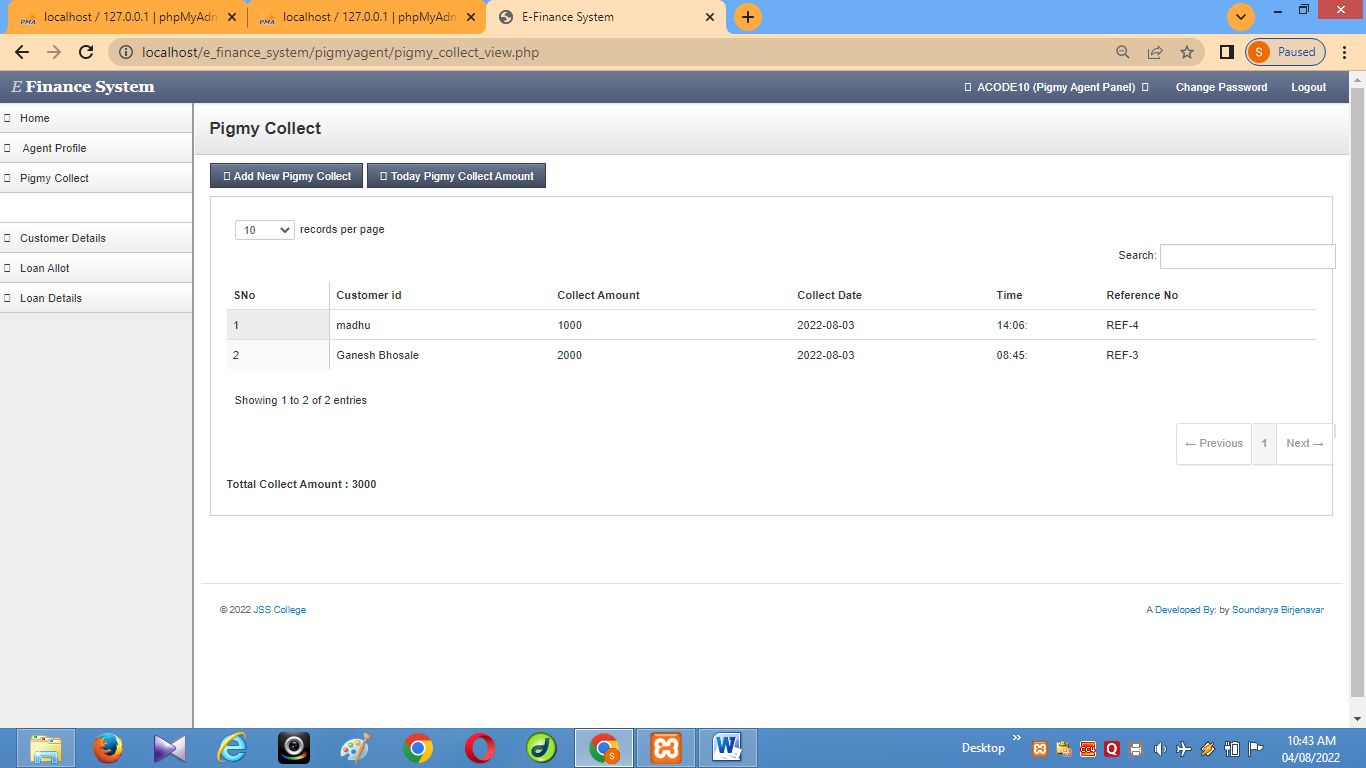
**My Report**



The above screen shows the report view to the customer.

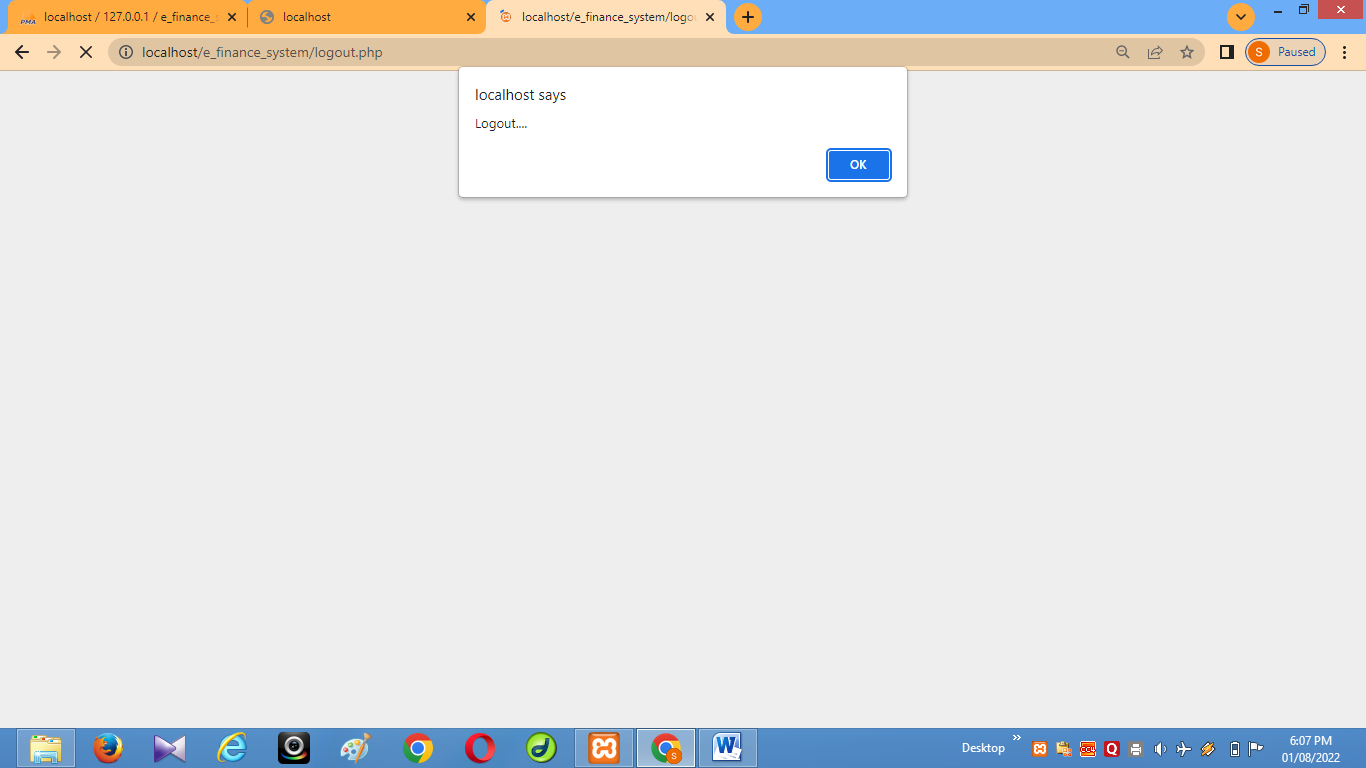
**PIMY AGENT MODULE:**

**Pigmy collects:**

****

The above screen shows the Pigmy collect view which has the available tasks to be performed.

**Logout**



The above screen shows the Logout page.

**9. SOURCE CODE**

Source Code:

CODE

**9.1.Database connection:**

<?php

$server="localhost";

$user="root";

$password="";

$database="e\_finance\_sysytem";

$conn=new mysqli($server,$user,$password,$database);

?>

**9.2. Form code:**

<? Php include('meta\_tags.php');?>

<? php include('menus.php');?>

<? php include('side\_menu.php');?>

<div class="content">

<div class="header">

<h1 class="page-title">Customer Details</h1>

</div>

<div class="container-fluid">

<div class="row-fluid">

<div class="well">

<div id="myTabContent" class="tab-content">

<div class="tab-pane active in" id="home">

<? php include('val.php');?>

<form name="form1" id="formID" method="post" action="customer\_details\_insert.php">

<p>&nbsp;</p>

<table width="462" border="0" align="center">

<tr>

<td colspan="2"><div align="center">CUSTOMER DETAILS </div></td>

</tr>

<tr>

<td width="215">Customer full name </td>

<td width="231"><input name="customer\_full\_name" type="text" class="validate[required,custom[onlyLetter]]" id="customer\_full\_name"></td>

</tr>

<tr>

<td>Gender </td>

<td><input name="gender" type="radio" value="radiobutton">

male

<input name="gender" type="radio" value="radiobutton">

female</td>

</tr>

<tr>

<td>Occupation </td>

<td><input name="occupation" type="text" class="validate[required,custom[onlyLetter]]" id="occupation"></td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="address" class="validate[required]" id="address"></textarea></td>

</tr>

<tr>

<td>City</td>

<td><input name="city" type="text" class="validate[required,custom[onlyLetter]]" id="city"></td>

</tr>

<tr>

<td>Email id </td>

<td><input name="email\_id" type="text" class="validate[required,custom[email]]" id="email\_id"></td>

</tr>

<tr>

<td>Contact no </td>

<td><input name="contact\_no" type="text" class="validate[required,custom[telephone]]" id="contact\_no"></td>

</tr>

<tr>

<td>Customer code </td>

<td><input name="customer\_code" type="text" class="validate[required]" id="customer\_code"></td>

</tr>

<tr>

<td colspan="2"><div align="center">

<input type="submit" name="Submit" value="Submit" class="btn btn-primary">

<input type="reset" name="Reset" value="Reset" class="btn btn-danger">

</div></td>

</tr>

</table>

<p>&nbsp;</p>

</form>

</div>

</div>

</div>

<div class="modal small hide fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h3 id="myModalLabel">Delete Confirmation</h3>

</div>

<div class="modal-body">

<p class="error-text"><i class="icon-warning-sign modal-icon"></i>Are you sure you want to delete the user?</p>

</div>

<div class="modal-footer">

<button class="btn" data-dismiss="modal" aria-hidden="true">Cancel</button>

<button class="btn btn-danger" data-dismiss="modal">Delete</button>

</div>

</div>

<?php include('footer.php');?>

**9.3. Insert code:**

<?php

$customer\_full\_name=$\_POST['customer\_full\_name'];

$gender=$\_POST['gender'];

$occupation=$\_POST['occupation'];

$address=$\_POST['address'];

$city=$\_POST['city'];

$email\_id=$\_POST['email\_id'];

$contact\_no=$\_POST['contact\_no'];

$customer\_code=$\_POST['customer\_code'];

include('dbconnect.php');

$sql="insert into customer\_details values(null,'$customer\_full\_name','$gender','$occupation','$address','$city','$email\_id','$contact\_no','$customer\_code')";

mysqli\_query($conn,$sql);

?>

<script>

alert("Inserted...");

document.location="customer\_details\_view.php";

</script>

**9.4. Delete code:**

<?php

$customer\_id=$\_REQUEST['customer\_id'];

include('dbconnect.php');

$sql="delete from customer\_details where customer\_id='$customer\_id'";

$res=mysqli\_query($conn,$sql);

?>

<script>

alert("Values Deleted");

document.location="customer\_details\_view.php";

</script>

**9.5. Update code:**

<?php

$customer\_id=$\_POST['customer\_id'];

$customer\_full\_name=$\_POST['customer\_full\_name'];

$gender=$\_POST['gender'];

$occupation=$\_POST['occupation'];

$address=$\_POST['address'];

$city=$\_POST['city'];

$email\_id=$\_POST['email\_id'];

$contact\_no=$\_POST['contact\_no'];

$customer\_code=$\_POST['customer\_code'];

include('dbconnect.php');

$sql="update customer\_details set customer\_full\_name='$customer\_full\_name',gender='$gender',occupation='$occupation',address='$address',city='$city',email\_id='$email\_id',contact\_no='$contact\_no',customer\_code='$customer\_code' where customer\_id='$customer\_id'";

mysqli\_query($conn,$sql);

?>

<script>

alert("updated...");

document.location="customer\_details\_edit1.php";

</script>

**9.6. Edit code:**

<?php include('meta\_tags.php');?>

<?php include('menus.php');?>

<?php include('side\_menu.php');?>

<div class="content">

<div class="header">

<h1 class="page-title">Customer Details</h1>

</div>

<div class="container-fluid">

<div class="row-fluid">

<div class="well">

<div id="myTabContent" class="tab-content">

<div class="tab-pane active in" id="home">

<?php include('val.php');?>

<?php

include('dbconnect.php');

$uname=$\_SESSION['uname'];

$sql="select \* from customer\_details where customer\_code='$uname'";

$res=mysqli\_query($conn,$sql);

$row=mysqli\_fetch\_array($res);

?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<title>Untitled Document</title>

</head>

<body>

<form name="form1"id="formID" method="post" action="customer\_details\_edit2.php">

<input type="hidden" value="<?php echo $row["customer\_id"]; ?>" name="customer\_id">

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<table width="462" border="0" align="center">

<tr>

<td colspan="2"><div align="center">CUSTOMER DETAILS </div></td>

</tr>

<tr>

<td width="215">Customer full name </td>

<td width="231"><input name="customer\_full\_name" type="text" class="validate[required,custom[onlyLetter]]" id="customer\_full\_name" value="<?php echo $row["customer\_full\_name"];?>"></td>

</tr>

<tr>

<td>Gender </td>

<td><input name="gender" type="radio" value="male" <?php if($row["gender"]=='male') { ?> checked <?php } ?>>

Male

<input name="gender" type="radio" value="female" <?php if($row["gender"]=='female') { ?> checked <?php } ?>>

Female</td>

</tr>

<tr>

<td>Occupation </td>

<td><input name="occupation" type="text" class="validate[required,custom[onlyLetter]]" id="occupation" value="<?php echo $row["occupation"];?>"></td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="address" class="validate[required]" id="address"><?php echo $row["address"];?></textarea></td>

</tr>

<tr>

<td>City</td>

<td><input name="city" type="text" class="validate[required,custom[onlyLetter]]" id="city" value="<?php echo $row["city"];?>"></td>

</tr>

<tr>

<td>Email id </td>

<td><input name="email\_id" type="text" class="validate[required,custom[email]]" id="email\_id" value="<?php echo $row["email\_id"];?>"></td>

</tr>

<tr>

<td>Contact no </td>

<td><input name="contact\_no" type="text" class="validate[required,custom[telephone]]" id="contact\_no" value="<?php echo $row["contact\_no"];?>"></td>

</tr>

<tr>

<td>Customer code </td>

<td><input name="customer\_code" type="text" class="validate[required]" readonly="" id="customer\_code" value="<?php echo $row["customer\_code"];?>"></td>

</tr>

<tr>

<td colspan="2"><div align="center">

<input type="submit" name="Submit" value="Update" class="btn btn-primary">

</div></td>

</tr>

</table>

<p>&nbsp;</p>

</form>

</div>

</div>

</div>

<div class="modal small hide fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h3 id="myModalLabel">Delete Confirmation</h3>

</div>

<div class="modal-body">

<p class="error-text"><i class="icon-warning-sign modal-icon"></i>Are you sure you want to delete the user?</p>

</div>

<div class="modal-footer">

<button class="btn" data-dismiss="modal" aria-hidden="true">Cancel</button>

<button class="btn btn-danger" data-dismiss="modal">Delete</button>

</div>

</div>

<?php include('footer.php'); ?>

**9.7. View code:**

<?php include('meta\_tags.php'); ?>

<?php include('menus.php'); ?>

<?php include('side\_menu.php'); ?>

<div class="content">

<div class="header">

<h1 class="page-title">Customer Details</h1>

</div>

<div class="container-fluid">

<div class="row-fluid">

<div class="btn-toolbar">

<a href="customer\_details.php" class="btn btn-primary"><i class="icon-plus"></i> Add New Customere Details</a>

<div class="btn-group">

</div>

</div>

<div class="well">

<table class="table" id="demo-dtable-01">

<thead>

<tr>

<th>S NO </th>

<th>Customer full name </th>

<th>Gender</th>

<th>Occupation</th>

<th>Address</th>

<th>City</th>

<th>Email id </th>

<th>Contact no </th>

<th>Customer code </th>

<th>Edit</th>

<th>Delete</th>

</tr>

</thead>

<tbody>

<?php

include('dbconnect.php');

$sn=1;

$sql="select \* from customer\_details";

$res=mysqli\_query($conn,$sql);

while($row=mysqli\_fetch\_array($res))

{

?>

<tr>

<td><?php echo $sn++;?></td>

<td><?php echo $row["customer\_full\_name"];?></td>

<td><?php echo $row["gender"];?></td>

<td><?php echo $row["occupation"];?></td>

<td><?php echo $row["address"];?></td>

<td><?php echo $row["city"];?></td>

<td><?php echo $row["email\_id"];?></td>

<td><?php echo $row["contact\_no"];?></td>

<td><?php echo $row["customer\_code"];?></td>

<td><a href="customer\_details\_edit1.php?customer\_id=<?php echo $row['customer\_id'];?>" onClick="return confirm('are you sure want to delete this?')">Edit</a></td>

<td><a href="customer\_details\_delete.php?customer\_id=<?php echo $row['customer\_id'];?>" onClick="return confirm('are you sure want to delete this?')">Delete</a></td>

</tr>

<?php

}

?>

</tbody>

</table>

</div>

<div class="modal small hide fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h3 id="myModalLabel">Delete Confirmation</h3>

</div>

<div class="modal-body">

<p class="error-text"><i class="icon-warning-sign modal-icon"></i>Are you sure you want to delete the user?</p>

</div>

<div class="modal-footer">

<button class="btn" data-dismiss="modal" aria-hidden="true">Cancel</button>

<button class="btn btn-danger" data-dismiss="modal">Delete</button>

</div>

</div>

<?php include('footer.php'); ?>

**10.SYSTEM TESTING**

**Testing:-**

**10.1 Introduction:**

Software Testing is a process of executing program with an indent of finding error. Testing is vital to success of the system. Testing demonstrates that the software functions appear to be working according to the specifications and performance requirements appeared to have been met. If a test is conducted successfully, it will discover errors in the software.

Software Testing consists of all test life cycle activities like static and dynamic testing concerned with planning, preparation and evaluation of software products to determine that the software products satisfy customer’s requirements and are fit the customer use.

The various strategies that were used in testing this software are as follows:

• Unit Testing

• Integration Testing

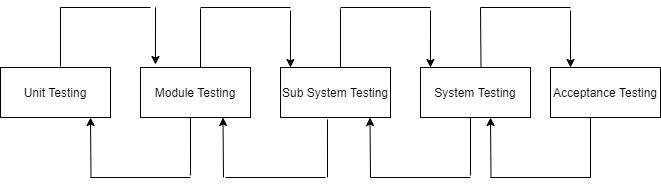
• System Testing

o Validation Testing

o Black Box Testing

o White Box Testing

• Acceptance Testing



**TYPES OF TESTS**

**10.2.1 SOFTWARE TESTING**

**UNIT TESTING**

Unit testing is done on individual modules as they are completed and become executable. This system was tested with the set of proper test data for each module and results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module.

This is also known as MODULE TESTING. This testing is carried out during phases, each module is founded to be working satisfactory as regards to the expected result from the module. Unit testing involves the design of the test cases that validate the internal program logic is functioning properly, and that program input produces valid output. All decision branches and internal code flow should be validated**.**

**INTEGRATION TESTING**

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied as shown by the Unit testing, the combination of the components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Integration Testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests to aggregates and delivers as its output. The Integration Testing verifies functional, performance, and reliability requirements placed on a major design items.

**FUNCTIONAL TESTING**

Functional tests provide a systematic demonstration of the functions tested that are available as specified by the business and technical requirements, system documentation, and user manuals. Functional Testing is centred on the following items: Valid Input : Identified classes of valid input must be accepted. Invalid Input: Identified classes of invalid input must be accepted. Functions: Identified functions must be exercised. Output : Identified classes of application outputs must be exercised. Systems/Procedures: Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows, data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of the current test is determined

**SYSTEM TESTING**

In this the entire software system is tested. The reference document for this process is the requirement document. Here the entire software is tested and the performance of the system was observed to see that it satisfies the requirement specification. System testing tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points. System testing involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user. The system meets all requirements of the client’s specifications.

**The following are the types of system tests that were carried out for the system:**

• **Validation Testing**

The system has been tested and implemented successfully and thus ensured that all requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

**• Black Box Testing**

This method focuses on the functional requirements of the software. This testing enables to derive set input conditions that will fully exercise all functional requirements of the program. Black Box Testing attempts to find errors in the following category.

• Incorrect or missing functions.

• Interface errors.

•Error in external database access.

• Performance errors.

• Initialization and Termination errors.

• **White Box Testing**

This is performed early in the testing process, while Black Box testing is applied during the last stage of testing. In this test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

It has been used to generate the test case in the following test cases:

• Guarantee that all independent paths have been executed.

• Execute all logical decisions from their True and False side.

• Execute all loops at their boundaries and within their operational bounds.

• Execute internal data structures to ensure their validity.

• Ensure whether all the possible validity checks and validity lookups have been

provided to validate data entry.

**➢ Performance Testing**

Performance Testing can serve different purpose. It can demonstrate that the system meets the performance criteria. It can compare two systems to find which performs better, or it can measure what parts of the system or workload cause the system to perform badly. In the diagnostic case, software engineers use tools such as profilers to measure what parts of a device or software contributes most to the poor performance. It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

**➢ Acceptance Testing**

It is a pre-delivery testing in which entire system is tested at client’s site on real world data to find errors. It deals with successful satisfaction of user needs. This project is approved and accepted by the clients. The process flow and execution is 99% working with respect to system testing procedure.

**10.3 Testing Strategies:**

There are two general strategies for testing software. There are follows

**Client Needs Acceptance Testing**

**Requirements System Testing**

**Design Integration Testing**

**Coding Unit Testing**

**10.3.1 Code Testing:**

This examines the logic of the program. To follow this test, cases are developed such that very path of the program is tested.

**10.3.2 Specification testing:**

Specification testing examines the specifications starting what the program should do and how it should perform under various conditions. Then test cases are developed for each conditions and combinations of conditions and to be submitted for processing.

**10.4 Testing Method Used:**

We have used Black Box Testing and Statistical testing .In Black Box Testing, we have given all possible types of inputs and seen for corresponding outputs and if not giving, code was corrected.

In **Statistical Testing**, we have checked for all variables whether they assigned values before using it, whether array bound correctly defined, whether looping statements terminating without going to infinite loop, whether function parameters are passed in order and about number of parameters etc. are checked successfully and found correct and everything working satisfactorily.

**Test Objectives**

• The system is tested with variety of inputs. The System is tested for accuracy and

correctness of the results obtained. Finally, the system is tested for inter-operability.

• All field entries must work properly.

• Pages must be activated from the identified link.

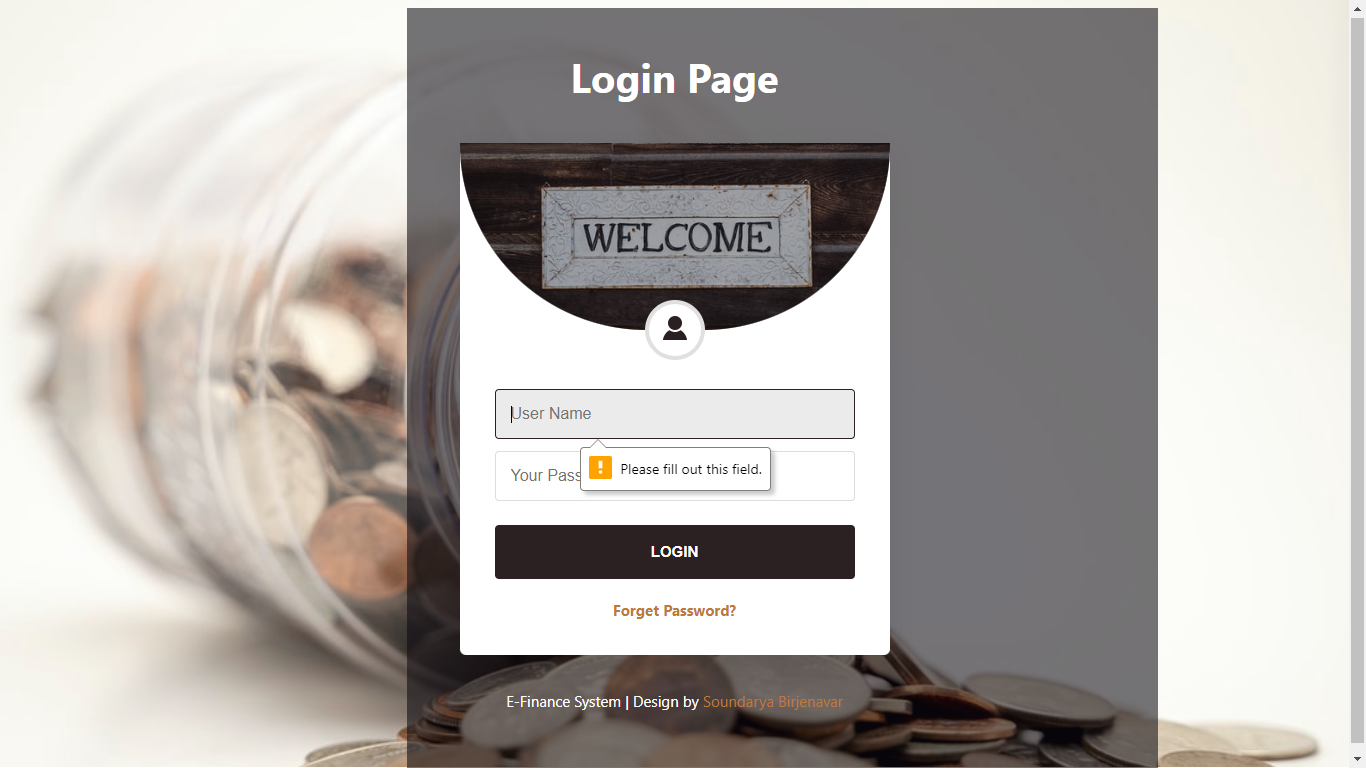
• The entry screen, messages and responses must not be delayed.

**10.5 Test Cases and Results:**

1. **Input:** Blank Username or Password

**Expected O/P:**Enter Username and Password

**Result:** Passed.



1. **Input:** Invalid Username or Password

**Expected O/P:** Invalid User.

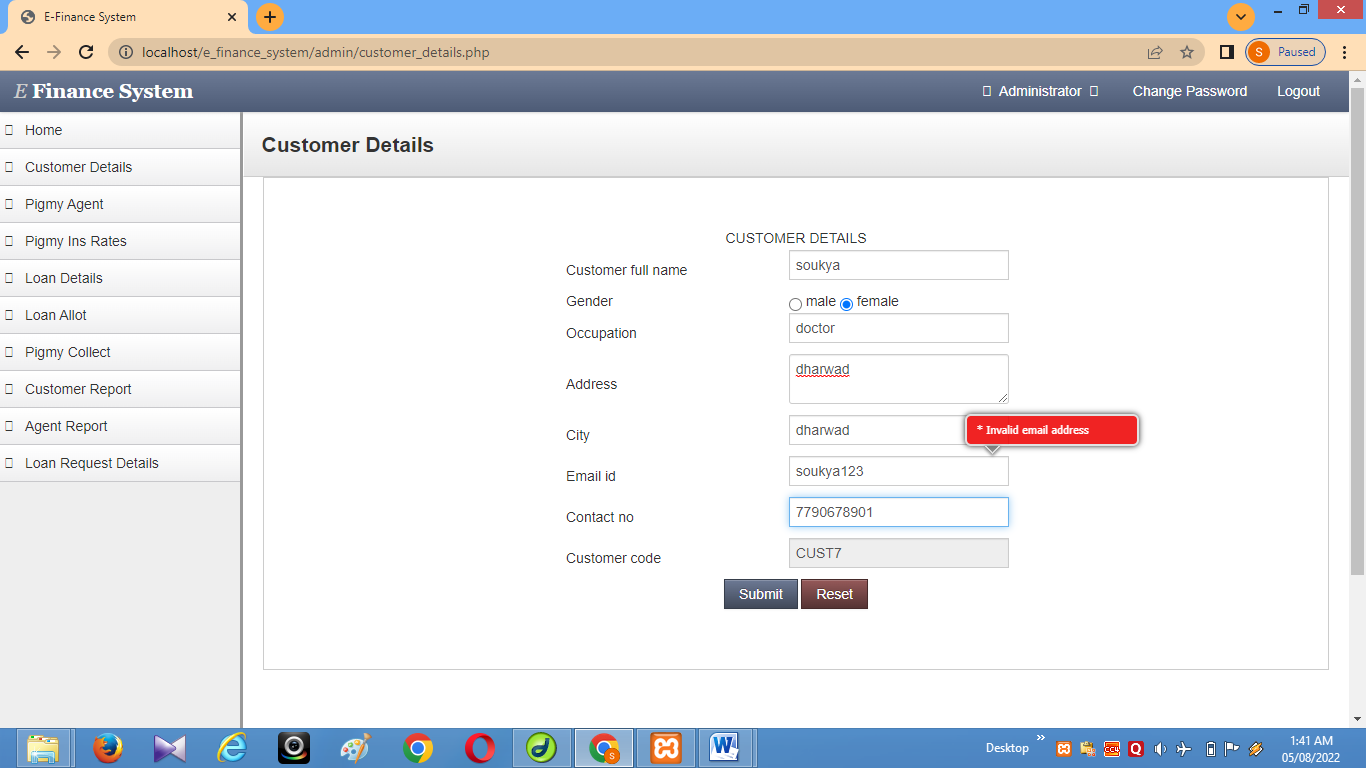
**Result:** Passed.



1. **Input:** In Email field if there no is @ symbol

**Expected O/P:**Invalid Email address

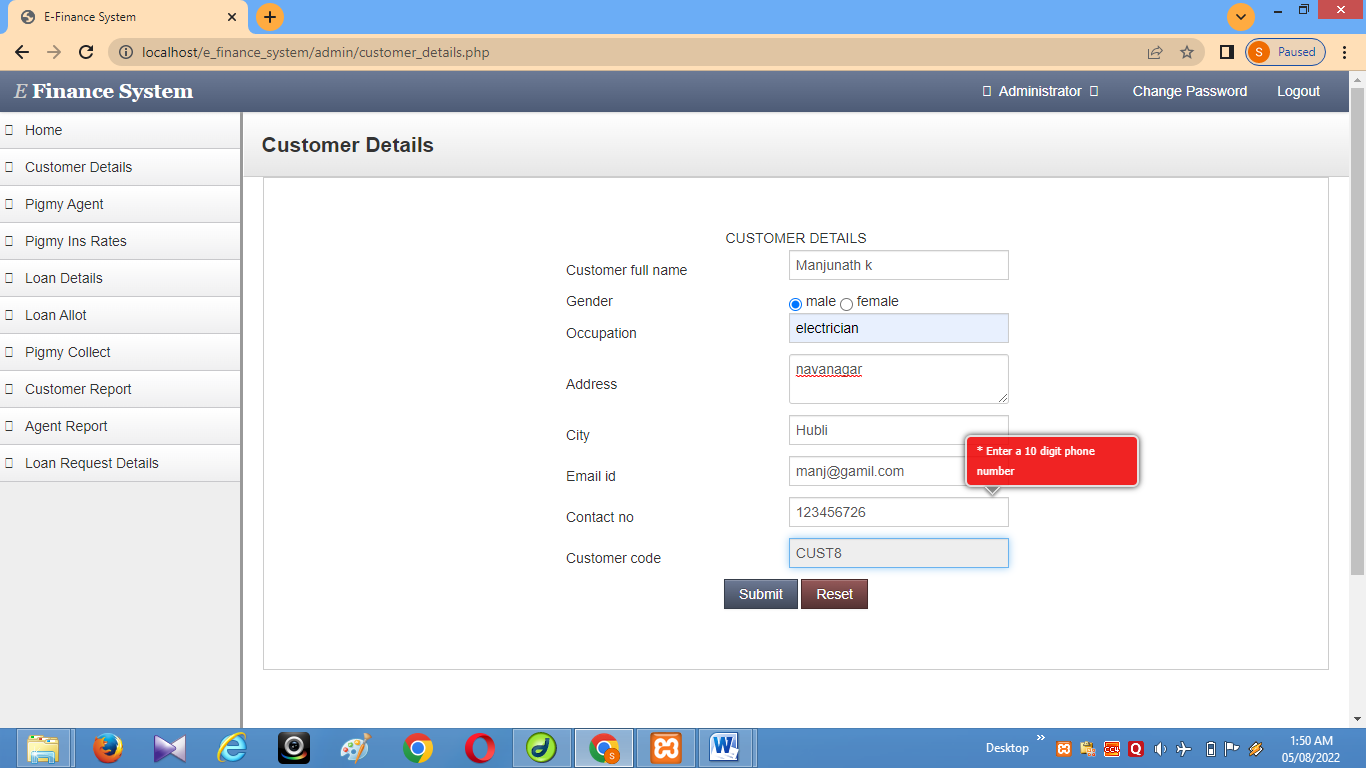
**Result:** Passed.



1. **Input:** In phone number if there if there less than 10 number

**Expected O/P:**Enter a 10 digit phone number

**Result:** Passed.



**Test cases and results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case** | **Input** | **Expected output** | **Actual output** | **Result** |
| 1 | For registered customer(Valid username/email and password) | Registered customer’s page which includes his details | Registered customer’s page | Passed |
| 2 | For registered customer (invalid Username and password.) | It should give appropriate error message saying “Enter valid details” | Error message displayed | Passed |
| 3 | Valid Username and Password | It should display respective page according to user type.(admin index or customer index) | Index page is Displayed | Passed |
| 4 | Invalid Username and Password | It should give appropriate error message saying “Enter valid User-Id and Password” | Error message Displayed | Passed |
| 5 | Enter less than 10 digits in mobile number | It should give appropriate error message saying “Enter valid mobile number” | Error message Displayed | Passed |
| 6 | Leave the customer detail blank | It should give appropriate error message saying “Enter customer details” | Error message Displayed | Passed |
| 7 | Validation | Validating each field according to usage | Validated | Passed |
| 8 | Change password | Changing The Admin password | Changed successfully | Passed |
| 9 | Submit button | On entering the valid values the submit button should be used to redirect. | Submit button redirecting | Passed |
| 10 | Logout | It should go to the Login page, and it should not go to the previous page. | Logout successful | Passed |

**11. BIBLIOGRAPHY**

**Books**

1. “Software Engineering”, by Ian Somerville, Sixth Edition, Pearson Education Ltd 2007.
2. “Web Programming”, by ‘Chris Bates’ Wiley Dreamtech India, 2nd Edition.
3. IEEE SRS Format.
4. Database Management Systems, by Navathe.

**WEBSITES**

1. [**http://www.google.com**](http://www.google.com)
2. [**http://www.wikipedia.com**](http://www.wikipedia.com)
3. [**http://w3schools.com**](http://w3schools.com)
4. [**http://www.stackoverflow.com**](http://www.stackoverflow.com)