## VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELAGAVI-590018



**A PROJECT REPORT ON**

## “FAMILY EXPENSE MANAGEMENT SYSTEM”

BY

### PRAKHYATH K S SHREEJESH J BALLAL

**(4SF20CS088) (4SF20CS130)**

In the partial fulfillment of the requirement for V Sem. B. E. (CSE)

## DBMS LABORATORY WITH MINI PROJECT

Under the guidance of

### Mrs. Vidya V V

Assistant Professor, Dept. of CS&E



**Department of Computer Science & Engineering**

# SAHYADRI

## COLLEGE OF ENGINEERING & MANAGEMENT

**An Autonomous Institution Adyar, Mangaluru-575007 2022-2023**

# COLLEGE OF ENGINEERING & MANAGEMENT

**An Autonomous Institution Adyar, Mangaluru – 07**

#### DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled **“FAMILY EXPENSE MANAGEMENT SYSTEM”** is submitted in partial fulfillment for the requirement of V Sem, B.E. (Computer Science & Engineering), **“DBMS LABORATORY WITH MINI PROJECT”** during the year 2022 – 2023 is result of bona fide work carried out by

PRAKHYATH K S 4SF20CS088

SHREEJESH J BALLAL 4SF20CS130

………………………… …………………………

Mrs. VIDYA V V Dr. NAGESH H R

Ass. Prof. Dept. of CS&E Hod & Assoc. Prof. Dept. of CS&E

SCEM, Mangaluru SCEM, Mangaluru

**Signature of the Examiners**

1. …………………………..
2. ………………………….

A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool used to organize (create, retrieve, update and manage) data in a database. The main aim of a DBMS is to supply away to store up and retrieve database information that is both convenient and efficient.

**Family Expense Management System** is developed for managing and storing all the data related to Family Expenses. The purpose of developing this database is to computerize the traditional way of storing or entering the data. This software manages related problems of the Expense System and avoids problems that might occur when data carried out manually. It allows to access information about all the expense details in various sector. Family administrator will manage all expense details. Whenever an administrator enters the expense system it shows the basic sectors information i.e., Members, Expense, Loans, Goals, Income, Savings. Then he will be able to manage registration, update and delete operation to all the tables of particular sector. The total control is under the administrator and the member may query the expense details.

1

It is with great satisfaction and euphoria that we are submitting the Mini Project Report on **“Family Expense Management System”** We have completed it as a part of the V semester DBMS Laboratory with Mini Project (18CSL58) of Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belagavi.

We are profoundly indebted to our guides, **Mrs. Vidya V V,** Assistant Professor, Department of Computer Science & Engineering for innumerable acts of timely advice, encouragement and We sincerely express our gratitude.

We express our sincere gratitude to **Dr. Nagesh H R,** Head & Associate Professor, Department of Computer Science & Engineering for her invaluable support and guidance.

We sincerely thank **Dr. Rajesha. S**, Principal, Sahyadri College of Engineering & Management and **Dr. D. L. Prabhakara** , Director, Sahyadri Educational Institutions, who have always been a great source of inspiration.

Finally, yet importantly, we express our heartfelt thanks to our family & friends for their wishes and encouragement throughout the work.

|  |  |
| --- | --- |
| PRAKHYATH K S 4SF20CS088  V Sem, B.E., CSE  SCEM, Mangaluru | SHREEJESH J BALLAL 4SF20CS130  V Sem. B.E., CSE  SCEM, Mangaluru |

## INDEX

1. Introduction
   1. Introduction to Database Management System. 1
   2. [Background of the Project. 1](#_bookmark0)
   3. Necessity of the Project 2
   4. [Applications and Advantages. 2](#_bookmark1)
   5. [Implementation. 2](#_bookmark2)
   6. [Oracle. 2](#_bookmark3)
   7. [My SQL 3](#_bookmark4)
   8. [JAVA. 3](#_TOC_250000)
   9. [Net Beans. 3](#_bookmark5)
2. Requirement Specifications
   1. [Hardware requirements. 4](#_bookmark6)
   2. [Software requirements. 4](#_bookmark7)
3. Design
   1. [ER Diagram. 6](#_bookmark8)
   2. [Relational Schema. 7](#_bookmark9)
   3. [Schema Diagram. 1](#_bookmark10)0
4. Normalization 11
5. Implementations
   1. Table Structure 13
   2. Jdbc driver for connecting Java and MySQL 15
   3. Insert/update/delete used as Functionalities 16
   4. Triggers and stored procedure 19
6. Results 21
7. Conclusion & References 26

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Figure Name** | **Page Number** |
| 3.1.1  3.1.2  3.2.2  3.2.3  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9  6.10 | ER Notation ER Diagram Mapping of Regular Entity Mapping of 1: N  Diagram Schema Diagram  Shows Landing Page Login page  Admin Page Managing members  Viewing added family member list Viewing member tables  User total info Page Income managing page Manage Savings Page Manage Expense Page | 5  6  9  10  21  21  22  22  23  23  24  24  25  25 |

# CHAPTER 1

### INTRODUCTION

* 1. **INTRODUCTION TO DBMS**

DBMS Stands for "Database Management System." In short, a DBMS is a database program. Technically speaking, it is a software system that uses a standard method of cataloging, retrieving, and running queries on data. The DBMS manages incoming data, organizes it, and provides ways for the data to be modified or extracted by users or other programs.

A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible.

DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity. The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data (storage and hardware).

As long as programs use the application programming interface (API) for the database that is provided by the DBMS, developers won't have to modify programs just because changes have been made to the database. With relational DBMSs (RDBMSs), this API is SQL, a standard programming.

### BACKGROUND OF THE PROJECT

Family Expense Management System is developed for managing and storing all the data related to Expenses. The purpose of developing this database is to computerize the traditional way of storing or entering the data. This software manages related problems of the Expense System and avoids problems that might occur when data carried out manually. It allows to access information about all the Expense details in various sector. The purpose of this project is to make a system to carry out various operations that is needed to be performed on the table. The

database is designed in such a way that updating of various information become easy and faster access.

### NECESSITY OF PROJECT

Family Expense Management System used to store all the data about the expense. It has made work faster and easier in the family, that people can get fair response about expense. Family administrator will manage all expense details. Whenever an administrator enters the expense system it shows the basic sectors information i.e., Members, Expense, Loans, Goals, Income, Savings.

### APPLICATIONS AND ADVANTAGES

The software is used to store the data most securely and in proper arrangement way. This software make work easier and faster. This is even help to store all the relevant data about expense that are expended by each person**.** It allows to access information about all the expense details in various sector. Family administrator will manage all family expense details. Whenever an administrator enters the expense system it shows the basic sectors information i.e., Members, Expense, Loans, Goals, Income, Savings. To Improve financial stability and security,

### IMPLEMENTATION

The Family Expense Management System is implemented using home page module, registration/login module, store module, user access module, with insert, update, delete and trigger operations. Also includes multiple well-defined constraints providing optimized analysis of data and better accessibility throughout the use of the standalone application. MySQL which holds the database and java for the front end which displays the provided modules.

### ORACLE

Oracle Database (commonly referred to as Oracle RDBMS or simply as Oracle) is a multi- model database management system produced and marketed by Oracle Corporation. It is a database commonly used for running online transaction processing (OLTP), data warehousing (DW), etc. Larry Ellison and his two friends and former co-workers, Bob Miner and Ed Oates

started a consultancy called Software Development and Laboratories (SDL) in 1977. SDL developed the original version of oracle.

### MY SQL

Family administrator will manage all expense details. Whenever an administrator enters the expense system it shows the basic sectors information i.e., Members, Expense, Loans, Goals, Income, Savings. Then admin will be able to manage registration, update & delete operation to all the tables of particular sector. The total control is under the admin & the member may query the expense details. MySQL server is used for data operations like querying, sorting, filtering, grouping, modifying and joining the tables.

### JAVA

Java is a popular programming language. It is owned by Oracle, and more devices run Java. Java works on different platforms (Windows, Mac, Linux). It is open-source secure, fast, free and powerful. As Java is close to C++ and C#, it makes it easy for programmers to switch to Java. Java was developed in the mid-1990s by James A. Gosling, a former computer scientist with Sun Microsystems.

Java is an Object-Oriented programming language developed by James Gosling in the early 1990s. The team initiated this project to develop a language for digital devices such as set-top boxes, television, etc. Java is used in all kinds of applications like Mobile Applications (Android is Java-based), desktop applications, web applications, client-server applications, enterprise applications, and many more.

### NETBEANS

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. The NetBeans Platform is a framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications. The underlying NetBeans platform supports creation of new applications and further development of existing applications using modular software components. As an application running on the NetBeans Platform, the NetBeans IDE itself is extensible and can be extended to support new languages.

# CHAPTER 2

### REQUIREMENT SPECIFICATION

* 1. **Hardware Requirements**
     + Processor: Any processor above 500 MHz
     + RAM: Min 4GB

•Hard Disk: Min 256GB

* 1. **Software Requirements**
     + Backend: MYSQL
     + Programming Language: Java
     + IDE: NetBeans 16

# CHAPTER 3

### 3.1 ER-diagram

An entity–relationship model describes interrelated things of interest in a specific domain knowledge. The ER Diagram of our project is shown in the **figure:3.1.1**

`

**Attribute**

**Entity**

**Multivalued Attribute**

**Weak Relationship**

**Relationship**

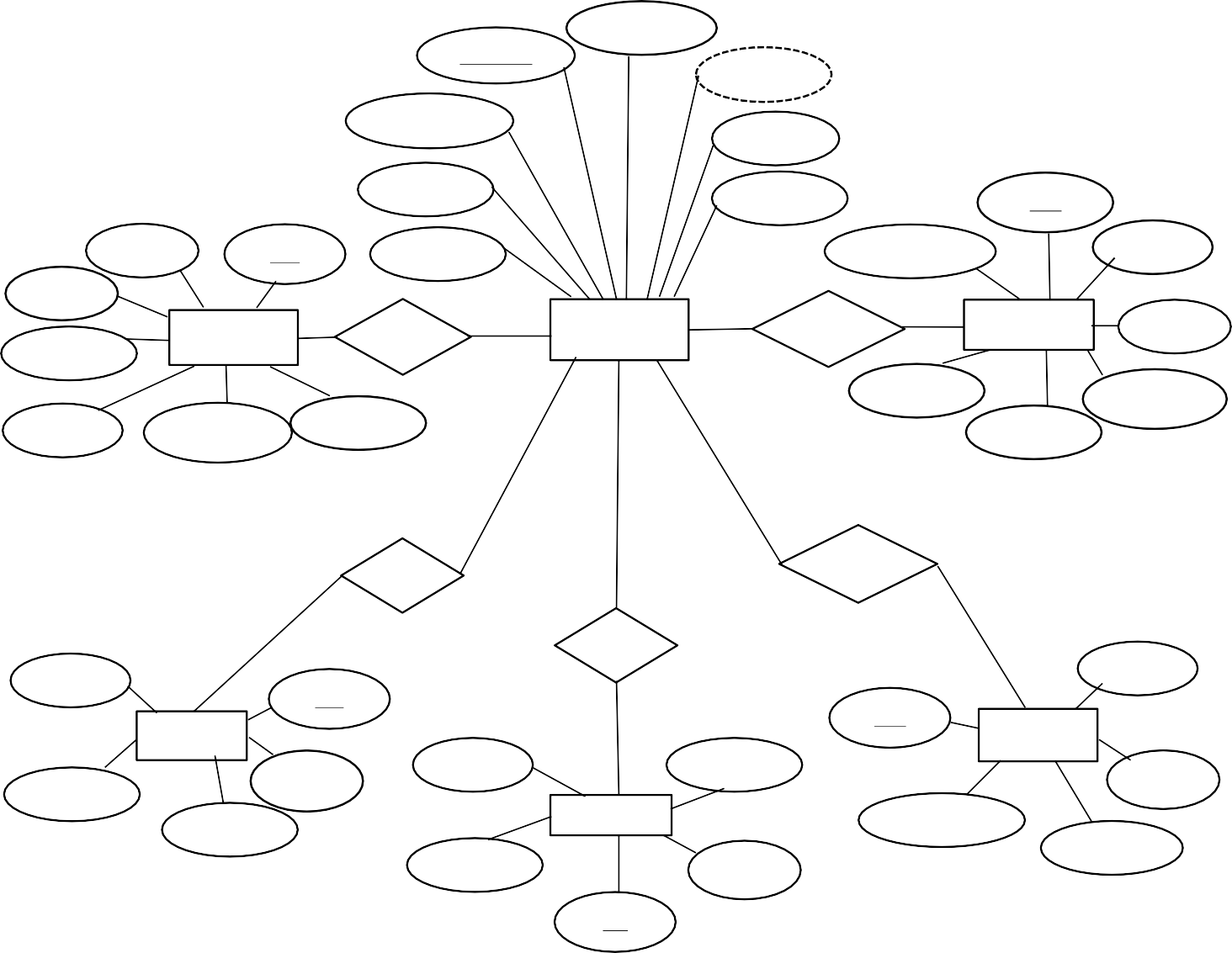
**Derived Attribute**

**Weak Entity**

**Key Attribute**

Key Attribute

**Figure 3.1.1 ER NOTATION**



Password

Username

Age

Occupation

DOB

Name

Gender

UID

Name

EID

Relation

LoanPeriod

User

Mode

EXPENSE

has

MEMBER

takes

LOAN

Name

Date

Date

Category

User

Category

Amount

Amount

has

starts

has

User

User

SID

GID

SAVING

GOAL

User Date

Date

Type

Goal

Amount

INCOME

StartDate

EndDate

Amount

Type

IID

**Figure 3.1.2** ER Diagram of Family Expense Management system

### RELATIONAL SCHEMA

* + 1. **Mapping From ER Diagram to Schema Diagram**
       1. Mapping of regular entities:-This step involves mapping all the regular entity types to tabular format by identifying their primary keys.
       2. Mapping of 1:1 Relation:-In this step foreign keys are assigned using foreign key approach. The primary key of the participating relation R or S is added as primary key to second entity types by looking at the participating constraints.
       3. Mapping of 1:N Relation:-Foreign key approach is used to add one sided primary key to the n sided entity at foreign key.
       4. Mapping of M:N Relation :-Here we use the cross reference approach where the relationship is converted to a new relation within attributes on primary keys of both participating relation.
       5. Mapping of Weak Entity :-When mapping weak entity types along with other attributes the partial key and primary key of parent entity together will form their primary key of the new relation.
       6. Mapping of N-ary Relation:-For mapping N ary relationship we create a new relation with a relationship name in its attribute and primary keys of all participating entity types.
       7. Mapping of Multivalued Relation :-For multivalued attributes a separate relation has to be created along with primary key of parent relation. A relational schema for a database is an outline of how data is organized.

### STEP 1: Mapping of regular entity types.

The regular entity types of our project are shown in figure

**MEMBERS**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Username | Name | DOB | Age | Gender | Occupation | Relation | Password |

**SAVINGS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SID | User | Type | Amount | Date |

**EXPENSES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EID | User | Name | Amount | Category | Date | Mode |

**LOANS**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LID | User | Name | Category | Amount | Date | Interest | LoanPeriod |

**INCOMES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IID | User | Type | Amount | Date |

**GOALS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GID | User | Goal | StartDate | EndDate |

**Figure 3.2.1**

**STEP 2: Mapping of weak entity types**

The ERD of our project doesn’t contain weak entity

**STEP 3: Mapping of binary 1:1 relation types**

The ERD of our project does not contain any 1:1 relation types.

**STEP 4: Mapping of 1: N relation types**

For every 1:N relation types identify the entity which is in the N-side. Make primary key of entity which is participating in 1 side as foreign key of entity which is N-side. If there are any attributes for the relationship add to the N-side.

The ERD of our project 1:N relation type is shown below

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EID | User | Name | Amount | Category | Date | Mode |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LID | User | Name | Category | Amount | Date | Interest | LoanPeriod |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SID | User | Type | Amount | Date |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GID | User | Goal | StartDate | EndDate |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IID | User | Type | Amount | Date |

**figure 3.2.2** 1:N Mapping

**STEP 5: Mapping of M: N relation types**

The ERD of our project does not contain any m:n relation type.

**STEP 6: Mapping of multivalued attributes**

The ERD of our project does not contain any Multivalued Attributes.

**STEP 7: Mapping of n-ary relation types**

The ERD of our project does not contain any n-ary relations.

### SCHEMA DIAGRAM

**MEMBERS**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Username | Name | DOB | Age | Gender | Occupation | Relation | Password |

**SAVINGS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | SID | User | | Type | Amount | Date |
|  | | |  | | | |

**EXPENSES**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | EID | User | | Name | Amount | Category | Date | Mode |
|  | | |  | | | | | |

**LOANS**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | LID | User | | Name | Category | Amount | Date | Interest | LoanPeriod |
|  | | |  | | | | | | |

**INCOMES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | IID | User | | Type | Amount | Date |
|  | | |  | | | |

**GOALS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | GID | User | | Goal | StartDate | EndDate |
|  | | |  | | | |

**Figure 3.2.4** Schema Diagra

# CHAPTER 4

### NORMALIZATION

**SAVINGS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SID | | User | Type | Amount | Date | |
|  |  | | | | |  |

1NF: The relations are in 1NF since there are no multivalued attributes or nested relations.

**EXPENSES**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EID | | User | Name | Amount | Category | Date | Mode | |
|  |  | | | | | | |  |

1NF: It is 1NF because there is no multivalued attributes in the relational schema

**LOANS**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LID | | User | Name | Category | Amount | Date | Interest | LoanPeriod | |
|  |  | | | | | | | |  |

1NF: The relations are in 1NF since there are no multivalued attributes or nested relations.

**INCOMES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IID | | User | Type | Amount | Date | |
|  |  | | | | |  |

1NF: The relations are in 1NF since there are no multivalued attributes or nested relations.

**GOALS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GID | | User | Goal | StartDate | EndDate | |
|  |  | | | | |  |

1NF: The relations are in 1NF since there are no multivalued attributes or nested relations.

**MEMBERS**



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Username | Name | DOB | Age | Gender | Occupation | Relation | Password |

FD1 : {Username} => { Name, DOB ,Age ,Gender , Occupation, Relation , Password} FD2 : {DOB} => { Age}

The above relation table is in 1NF and 2NF form. The FD violates 3NF as its LHS is not a super key (and RHS is set of non key attributes)

### MEMBERS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Username | Name | DOB | Gender | Occupation | Relation | Password |

**AGE**

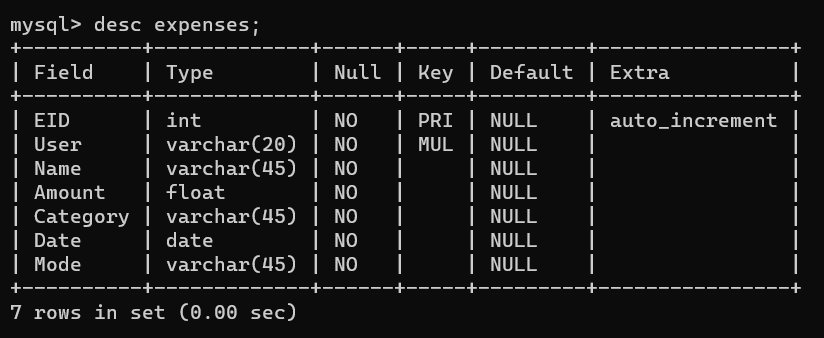
|  |  |
| --- | --- |
| DOB | Age |

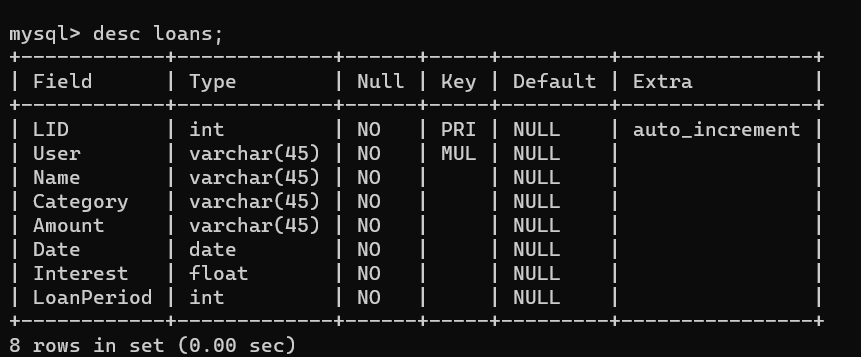
Now the relation is 3NF form.

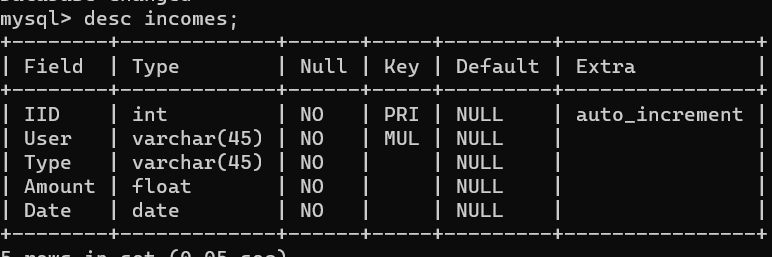
# CHAPTER 5

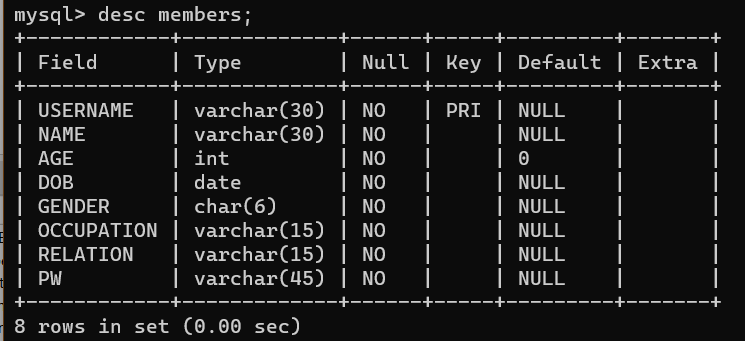
**Implementation**

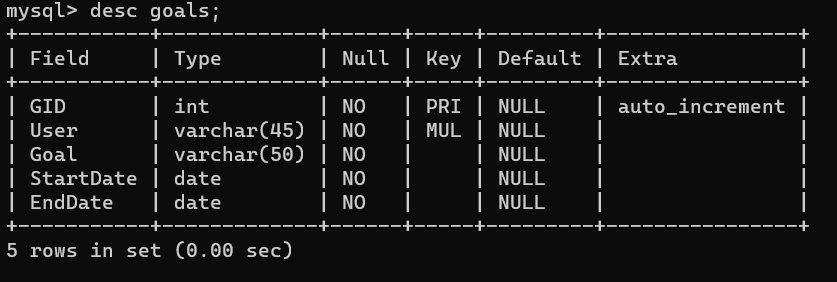
* 1. Create Table <table name> <description>

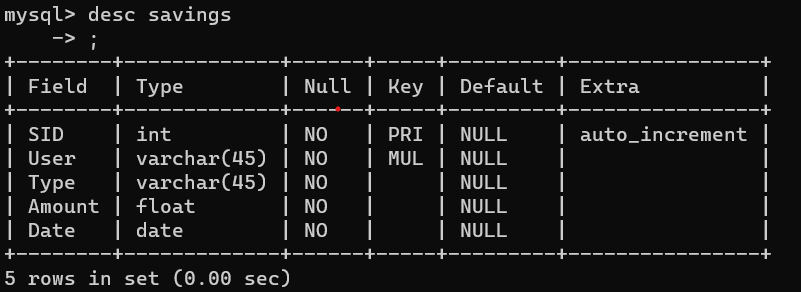


.









#### JDBC DRIVER IS USED TO CONNECT JAVA AND MY SQL CODE:

package Backend;

import java.sql.Connection; import java.sql.DriverManager; import java.sql.SQLException;

public class ConnectSQL { static Connection con;

public static Connection createC()

{

try {

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost/familyexp ensemanagement", "root", "root");

System.out.println("connected ");

} catch (ClassNotFoundException | SQLException e) {

}

return con;

}

}

* 1. Insert/update/delete option has been implemented as follows: UPDATE

String name, dob, gender, occupation, relation, pw; String uname = txtUsername.getText();

name = txtName.getText();

dob = txtDOB.getText(); gender = txtGender.getText();

occupation = txtOccupation.getText(); relation = txtRelation.getText();

pw = txtPw.getText(); try {

if (!uname.trim().isEmpty()) {

if (!name.trim().isEmpty() || !dob.trim().isEmpty() || !gender.trim().isEmpty() ||

!occupation.trim().isEmpty() || !relation.trim().isEmpty() || !pw.trim().isEmpty()) {

pst = con.prepareStatement("select \* from members where username=?");

pst.setString(1, uname); rs = pst.executeQuery();

if (rs.next()) {

pst = con.prepareStatement("update members set name=?,dob=?,gender=?,occupation=?,relation=?,pw=? where username=?"); pst.setString(1, name);

pst.setString(2, dob); pst.setString(3, gender); pst.setString(4, occupation); pst.setString(5, relation); pst.setString(6, pw); pst.setString(7, uname); pst.executeUpdate();

JOptionPane.showMessageDialog(this, "Successfully updated the record!"); txtName.setText("");

txtDOB.setText(""); txtGender.setText(""); txtOccupation.setText(""); txtRelation.setText(""); txtUsername.setText(""); txtPw.setText(""); txtUsername.requestFocus();

} else {

JOptionPane.showMessageDialog(this, "No user found with username=" + uname);

}

} else {

JOptionPane.showMessageDialog(this, "Please fill in all the details!");

}

} else {

JOptionPane.showMessageDialog(this, "Please fill the username!");

}

} catch (SQLException ex) { JOptionPane.showMessageDialog(this, ex); ex.printStackTrace();

}

#### INSERTING

try {

String name, dob, gender, occupation, relation, username, pw; name = txtName.getText();

dob = txtDOB.getText(); gender = txtGender.getText();

occupation = txtOccupation.getText(); relation = txtRelation.getText(); username = txtUsername.getText(); pw = txtPw.getText();

pst = con.prepareStatement("insert into members(name,dob,gender,occupation,relation,username,pw)values(?,?,?,?,?,?,?)");

pst.setString(1, name); pst.setString(2, dob); pst.setString(3, gender); pst.setString(4, occupation); pst.setString(5, relation); pst.setString(6, username); pst.setString(7, pw); pst.executeUpdate(); txtName.setText(""); txtDOB.setText(""); txtGender.setText(""); txtOccupation.setText(""); txtRelation.setText(""); txtUsername.setText(""); txtPw.setText(""); txtUsername.requestFocus();

JOptionPane.showMessageDialog(this, "Added Successfully\nUsername :" + username + "\nPassword:" + pw);

} catch (SQLException ex) { ex.printStackTrace(); JOptionPane.showMessageDialog(this, ex);

}

#### DELETING

try {

String username = txtUsername.getText(); if (username.trim().isEmpty()) {

JOptionPane.showMessageDialog(this, "Enter a valid username ");

} else {

pst = con.prepareStatement("select \* from department where username=?"); pst.setString(1, username);

rs = pst.executeQuery(); if (rs.next()) {

pst = con.prepareStatement("delete from members where username=?"); pst.setString(1, username);

pst.executeUpdate(); JOptionPane.showMessageDialog(this, "Record Deleted!"); txtName.setText("");

txtDOB.setText(""); txtGender.setText(""); txtOccupation.setText(""); txtRelation.setText(""); txtUsername.setText(""); txtPw.setText(""); txtUsername.requestFocus();

} else {

JOptionPane.showMessageDialog(this, "No user found with username=" +

username);

}

}

} catch (SQLException ex) { JOptionPane.showMessageDialog(this, ex); ex.printStackTrace();

}

#### SEARCHING

try {

String uname = txtUsername.getText();

pst = con.prepareStatement("select \* from members where username = ?"); pst.setString(1, uname);

rs = pst.executeQuery(); if (rs.next()) {

txtName.setText(rs.getString(2)); txtDOB.setText(rs.getString(4)); txtGender.setText(rs.getString(5)); txtOccupation.setText(rs.getString(6)); txtRelation.setText(rs.getString(7)); txtPw.setText(rs.getString(8)); JOptionPane.showMessageDialog(this, "Fetched Successfully");

} else {

JOptionPane.showMessageDialog(this, "User not found!"); txtName.setText("");

txtDOB.setText(""); txtGender.setText(""); txtOccupation.setText(""); txtRelation.setText(""); txtUsername.setText(""); txtPw.setText(""); txtUsername.requestFocus();

}

} catch (SQLException ex) { ex.printStackTrace(); JOptionPane.showMessageDialog(this, ex);

}

* 1. Triggers and Storing Procedure
     1. Trigger:

We have used the below trigger to derive age from the dob provided by the admin during sign up of family member. We have also used trigger to raise an exception when an invalid username and gender format is inserted into the fields (validation).

CREATE DEFINER=`root`@`localhost` TRIGGER `members\_BEFORE\_INSERT` BEFORE INSERT ON `members` FOR EACH ROW BEGIN

set new.age = timestampdiff(year,new.dob,curdate()); IF NOT NEW.username REGEXP '^[a-zA-Z]+[0-9]+'

THEN SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Username must with

letters followed by numbers!'; END IF;

IF NOT NEW.gender REGEXX'(Male|Female|M|F|FEMALE|MALE|female|male)' THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Gender must be in the format Male|Female|M|F|FEMALE|MALE|female|male';

END IF; END

* + 1. Stored Procedure:

We have used the below procedure in order to retrieve all details from member table.

CREATE DEFINER=`root`@`localhost` PROCEDURE `new\_procedure`(in uname varchar(20))

BEGIN

select \* from members;

END

# CHAPTER 6

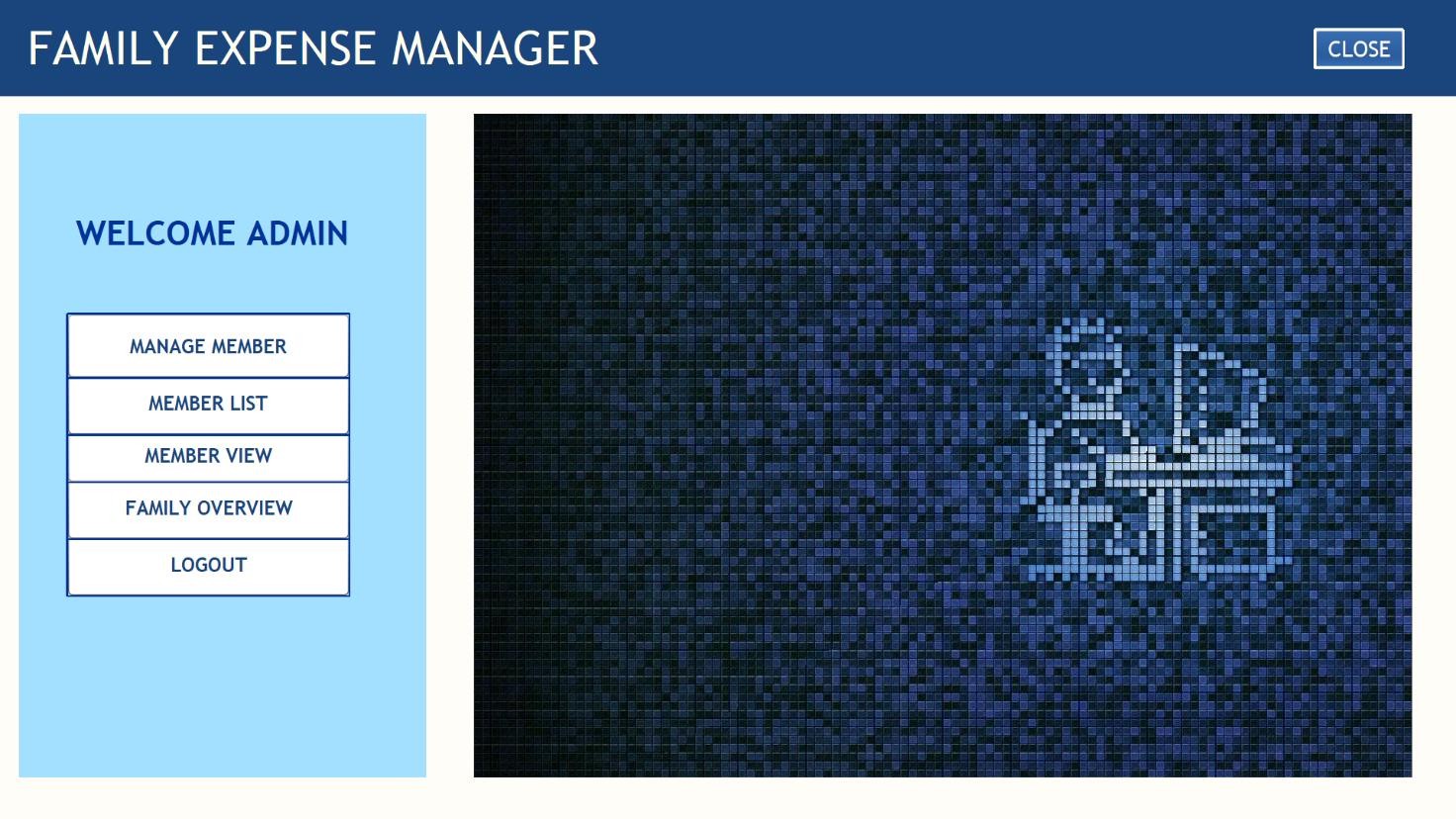
### RESULTS AND DISSCUSSION



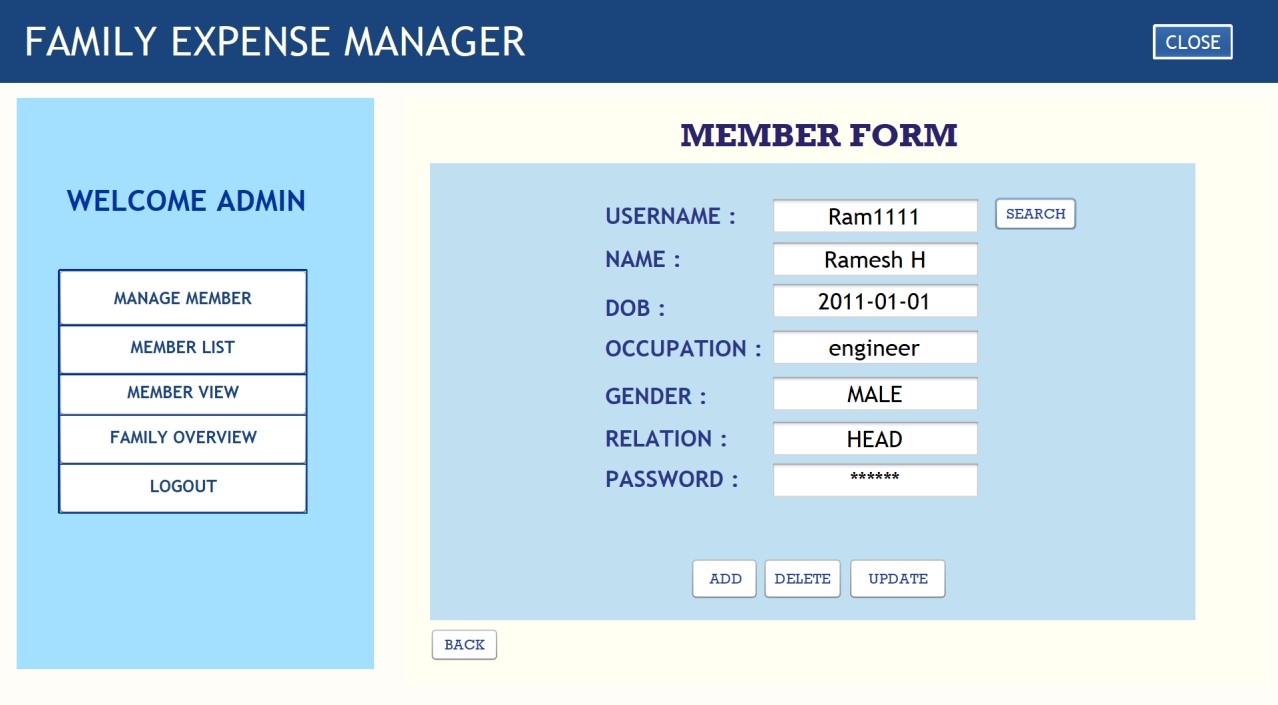
**Figure6**.**1** Shows Landing Page



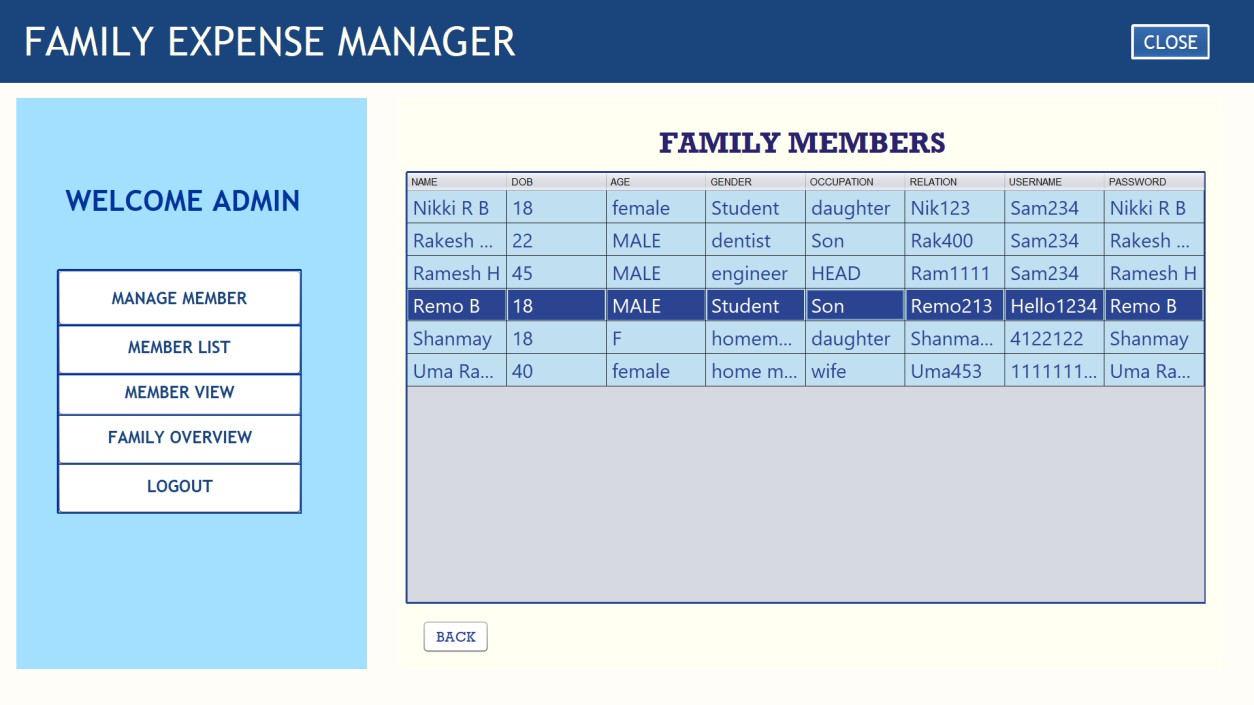
**Figure 6.2** Login page



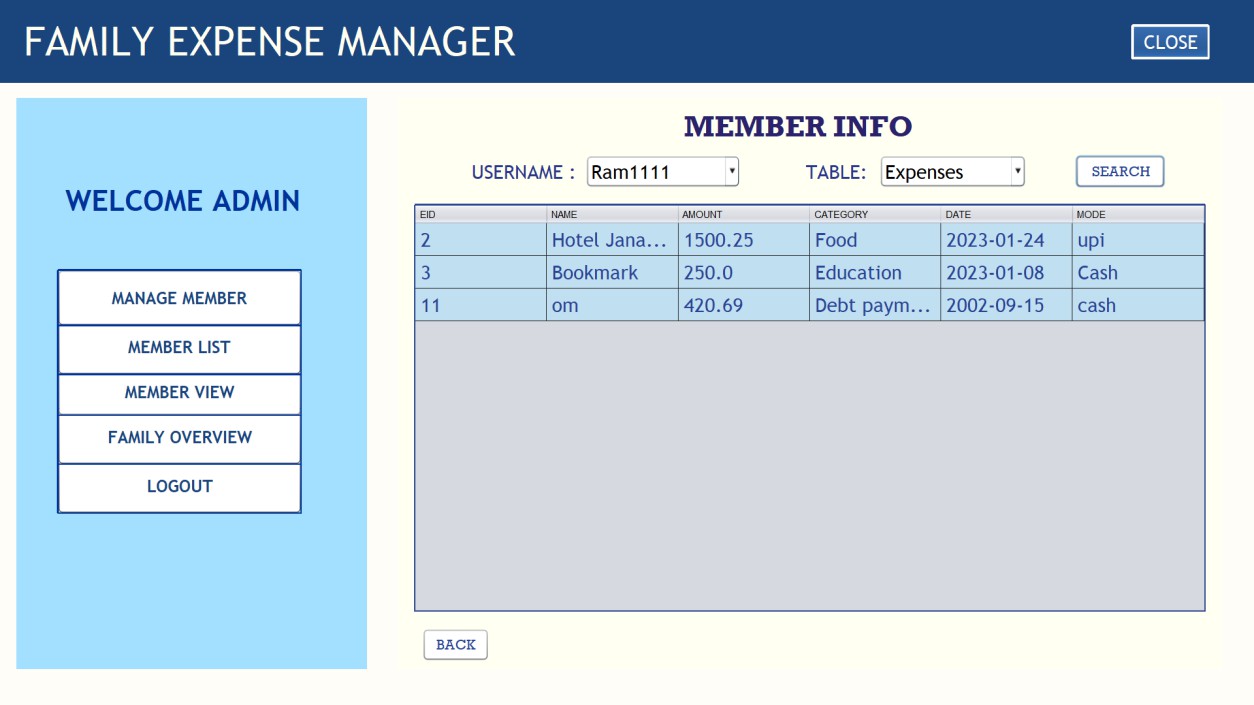
**Figure 6.3** Admin Page



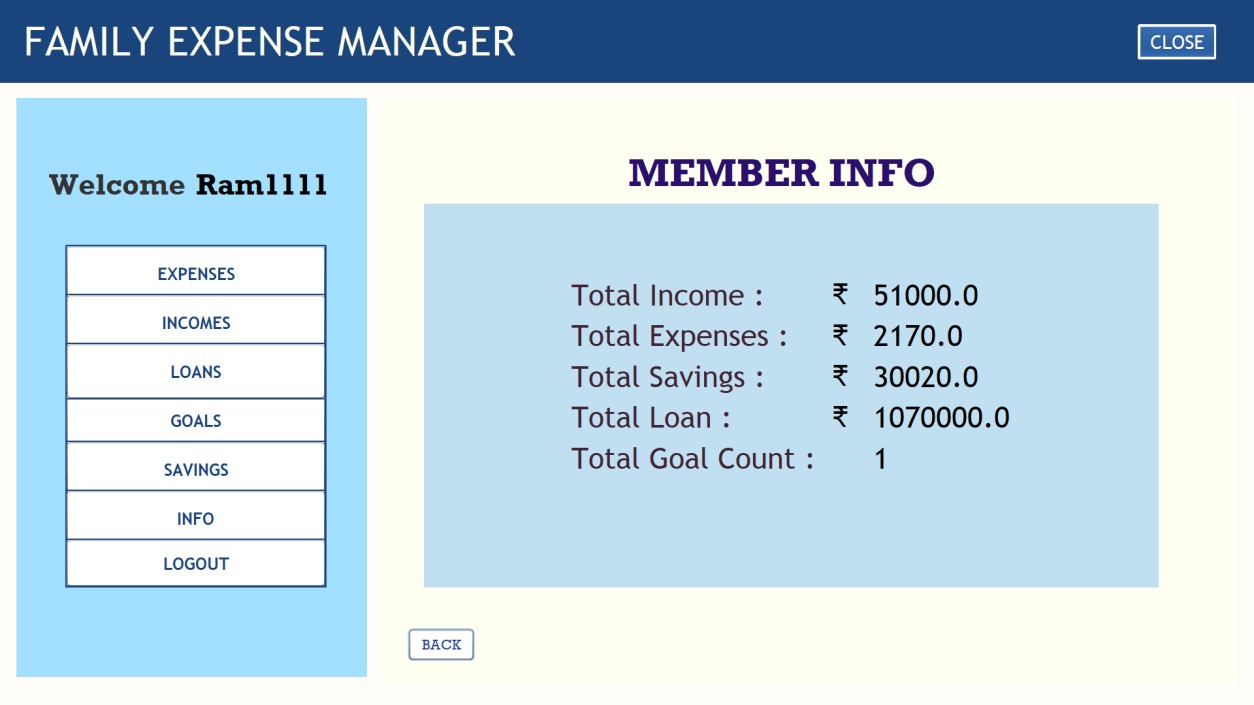
**Figure 6.4** Managing members



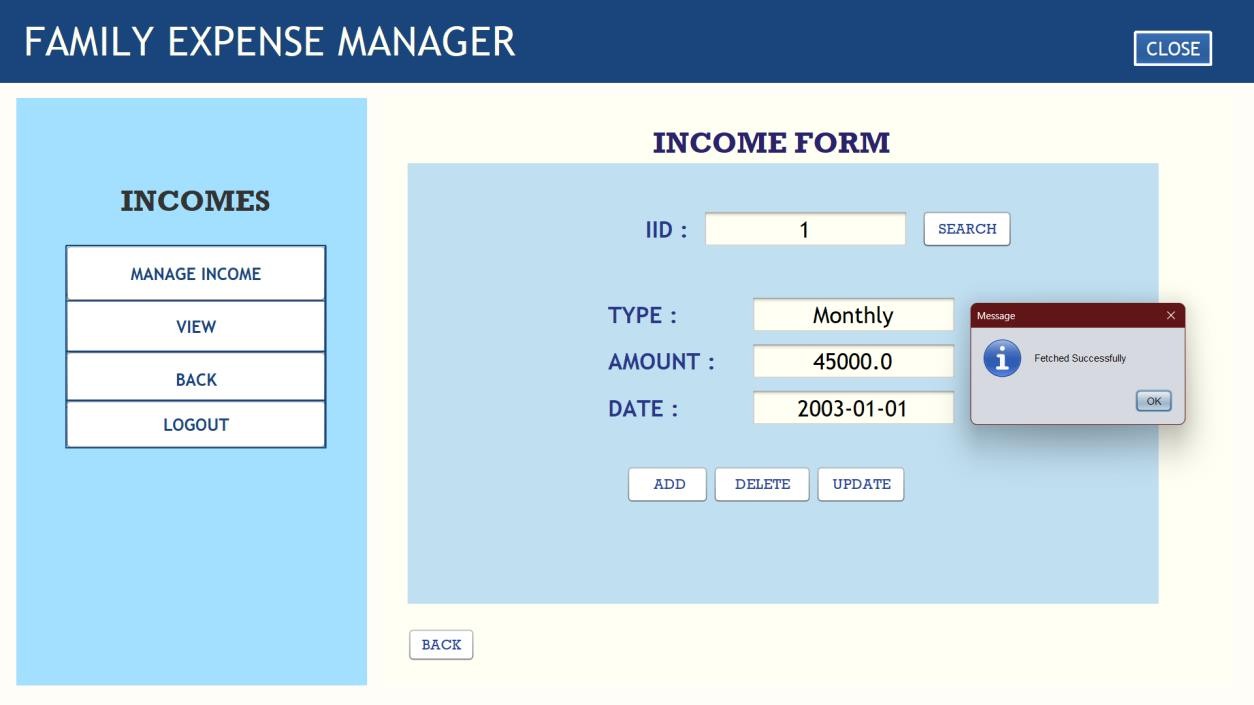
**Figure 6.5** Viewing added family member list



**Figure 6.6** Viewing member tables



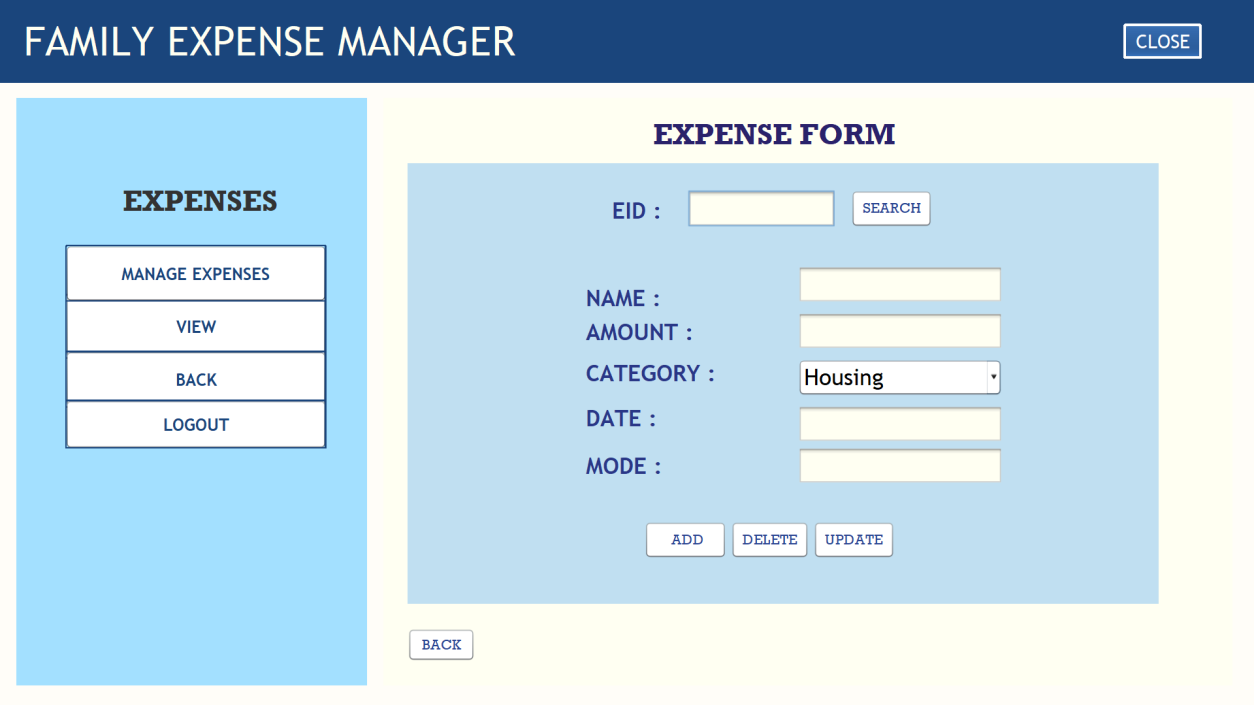
**Figure 6.7** User total info Page



**Figure 6.8** Income managing page



**Figure 6.9** Manage Savings Page



**Figure 6.10** Manage Expense Page

# CHAPTER 7

### CONCLUSION

Family administrator will manage all expense details. Whenever an administrator enters the expense system it shows the basic sectors information i.e., Members, Expense, Loans, Goals, Income, Savings. Then admin will be able to manage registration, update and delete operation to all the tables of particular sector. The total control is under the administrator and the member may query the expense details.

**Its application**: Improved financial stability and security, Increased ability to save for future goals. Learnings From project are Applying and understanding the methods of using JDBC, MySQL and JAVA using NetBeans

### REFERENCES

**[1]. MYSQL:** [**https://dev.mysql.com/doc/refman/8.0/en/**](https://dev.mysql.com/doc/refman/8.0/en/)

**[2]. JAVA:** [**https://www.w3schools.com/java/default.asp**](https://www.w3schools.com/java/default.asp)

**[3]. JDBC :** [**https://www.geeksforgeeks.org/introduction-to-jdbc/**](https://www.geeksforgeeks.org/introduction-to-jdbc/) **[4]. YOUTUBE:** [**https://youtu.be/YyZoazwkbow**](https://youtu.be/YyZoazwkbow)

**[5]. NETBEANS:** [**https://netbeans.apache.org/kb/docs/java-se.html**](https://netbeans.apache.org/kb/docs/java-se.html)