EXPENSE TRACKER

A PROJECT REPORT

Submitted by

S. Reshma [RA2211032010008]

Harsini J.P. [RA2211032010007]

Hrishika Raj [RA2211032010004]

Under the Guidance of

Dr V. Nallarasan

Assistant professor, Department of Networking and Communications

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in INTERNET OF THINGS

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

MAY 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this DBMS Project Report titled **"EXPENSE TRACKER"** in the Course – **21CSC205P Database Management Systems,** is the bonafide work done by **S. Reshma [RA2211032010008], Harsini J.P. [RA2211032010007] and Hrishika Raj [RA2211032010004]** who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

03/05/24

Dr. V. Nallarasan
Assistant Professor
Department of Networking
and Communications
SRMIST – KTR

Dept. of Networking and Communications
Kattankulathur
603 203
SRM IST

Dr. Annapurani. K
Head of the Department
Department of Networking and
Communications
SRMIST - KTR

Date: 03/05/24

Department of Networking and Communications
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Own Work Declaration Form

Degree/ Course: B. Tech - Database Management Systems (DBMS)

Student Name: S. Reshma, Harsini J.P, Hrishika Raj

Registration Number: RA2211032010008, RA2211032010007, RA2211032010004

Title of Work: Expense Tracker

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly referenced / listed all sources as appropriate.
- Referenced and put in inverted commas all quoted text (from books, web, etc).
- Given the sources of all pictures, data etc. that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that we have received from others (e.g. fellow students, technicians, statisticians, external sources).
- Compiled with any other plagiarism criteria specified in the Course handbook / University website.

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:		
We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except were indicated by referring, and that we have followed the good academic practices noted above.		
HRaj	Harsini	S.Reshma
RA2211032010004	RA2211032010007	RA2211032010008

ACKNOWLEDGEMENT

We express our humble gratitude to Dr C. Muthamizhchelvan, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, Dr T.V.Gopal for his invaluable support. We wish to thank Dr Revathi Venkataraman, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, Dr. Annapurani K, Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor and Course Faculty, Dr. V Nallarasan, Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course, and for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Networking and Communications Department, staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

S Reshma (RA2211032010008) Harsini JP (RA2211032010007) Hrishika Raj (RA2211032010004)

ABSTRACT

This report presents the design and implementation of a Expense Tracker System utilizing a Database Management System (DBMS) for efficient data storage, management, and analysis. The FTS aims to provide users with a comprehensive platform to track their fitness activities, monitor progress, and achieve personal health goals. The proposed system employs a relational database model to organize and store diverse fitness data, including exercise routines, nutrition intake, biometric measurements, and goal setting. The schema of the FTS encompasses entities such as Users, Activities, Workouts, Nutrition Plans, Metrics, and Goals, interconnected through well-defined relationships. The implementation leverages SQL for database schema design, data manipulation, and retrieval operations, ensuring compatibility with various DBMS platforms. Additionally, a user-friendly interface is developed, accessible through web or mobile applications, to interact with the Fitness Tracker System seamlessly. The Fitness Tracker System in the DBMS serves as a valuable tool for individuals, fitness enthusiasts, athletes, and healthcare professionals to monitor and improve fitness levels, promote healthy lifestyle choices, and enhance overall well-being. Future enhancements may include integration with wearable devices for real-time data synchronization, advanced analytics for predictive insights, and social features for community engagement and support.

TABLE OF CONTENTS

CHAPTER 1

PROBLEM UNDERSTANDING, IDENTIFICATION OF ENTITY AND RELATIONSHIPS, CONSTRUCTION OF DB USING ER MODEL AND EMPATHY MAP FOR THE PROJECT

Problem Statement:

Managing personal finances is a crucial aspect of an individual's life, but many people struggle with keeping track of their expenses. In today's fast-paced world, individuals often find it challenging to maintain a detailed and organized record of their expenditures, leading to financial stress and uncertainty. To address this issue, there is a need for a user-friendly and efficient personal expense tracker that empowers users to monitor, analyze, and optimize their spending habits.

Identification Of Entity and Relationships:

1. User Profile: Stores information about users including their name, contact number, and email address.

Relationship: Connected to the `account` table via the `user_id` foreign key to associate users with their accounts.

2. Account Type: Contains different types of accounts available in the system.

Relationship: Linked to the `account` table through the `type_id` foreign key to specify the type of account associated with each account.

3. Account: Holds information about user accounts including account number, opening date, and status.

Relationships: Connected to `user_profile` table via `user_id` and to `acc_type` table via `type_id` for user and account type associations respectively.

4. Transaction Category: Defines various categories for transactions like income, expenses, etc.

Relationship: Linked to the `transaction` table via the `cat_id` foreign key to categorize transactions.

5. Transaction: Records individual transactions made within user accounts, including details like amount, type, and description.

Relationships: Connected to `account` table via `acc_no` and to `trn_cat` table via `cat_id` for account and category associations respectively.

6. Loan: Manages information related to loans taken by users such as loan amount, tenure, and start date.

Relationships: Associated with `account` table via `acc_no` to link loans with respective accounts.

7. Debt: Tracks debts associated with accounts including amount, interest, and due dates.

Relationships: Linked to the `account` table via `account_id` foreign key to specify which account the debt belongs to.

8. Insurance: Stores details about insurance policies like type, policy number, and premium amount.

No explicit relationship defined in the provided schema, but could potentially be linked to users or accounts.

9. Reminder: Contains reminders with details like date, frequency, name, and category.

No explicit relationship defined, but could be associated with users or accounts for personalized reminders.

10. Tax: Records tax-related information such as date, type, rate, and amount.

No explicit relationship mentioned, but could possibly be connected to users or accounts for tax tracking.

11. Expense: Logs expenses with details like date, amount, and notes.

No explicit relationship specified, but likely associated with users or accounts for tracking expenses.

12. Income: Stores details of income sources including type and amount.

No explicit relationship mentioned, but could be linked to users or accounts to track income sources.

13. Place of Expense: Records locations where expenses occur.

No explicit relationship defined, but could potentially be associated with expenses or users for expense tracking.

14. Saving: Manages savings amounts and types.

No explicit relationship mentioned, but could be linked to users or accounts to track savings.

15. Budget: Tracks budget allocations with details like duration, amount, and category.

No explicit relationship specified, but could be associated with users or accounts for budget management.

16. Investment: Stores investment details such as amount, name, and duration.

No explicit relationship mentioned, but could be linked to users or accounts for investment tracking.

17. Customer Support: Records customer support issues with details like ticket ID, date, and category.

No explicit relationship defined, but could be linked to users or accounts for issue resolution tracking.

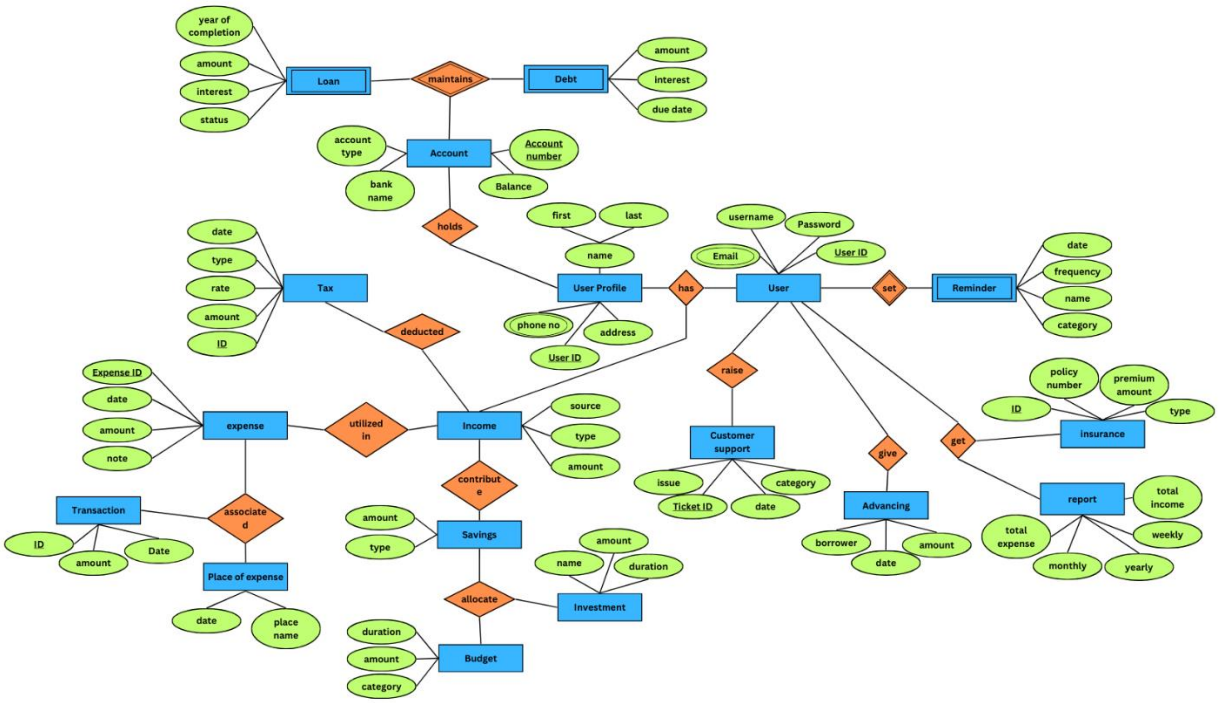
18. Advancing: Manages advances made to borrowers with details like borrower name, date, and amount.

No explicit relationship specified, but could be associated with users or accounts for advancing tracking.

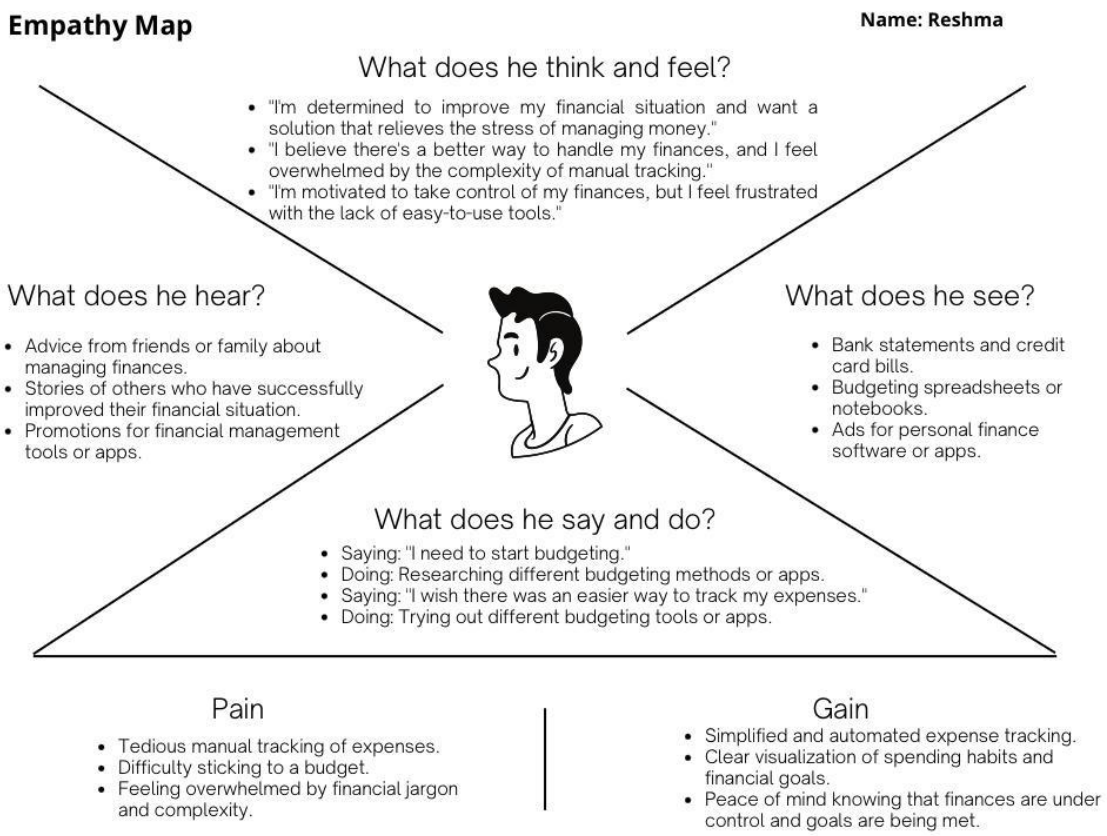
19. Report: Stores summarized financial report data including total income, expenses, and their breakdown.

No explicit relationship mentioned, but could be generated based on data from other tables for financial analysis and reporting.

ER Diagram:


year of completion
amount
interest
status
Loan
maintains
Debt
amount
interest
due date
account type
Account
Account number
bank name
Balance
holds
first
last
name
User Profile
has
User
username
Password
Email
User ID
set
Reminder
date
frequency
name
category
date
type
rate
amount
ID
Tax
deducted
phone no
address
User ID
Expense ID
date
amount
note
expense
utilized in
Income
source
type
amount
raise
Customer support
issue
Ticket ID
date
category
give
Advancing
borrower
date
amount
get
Insurance
policy number
premium amount
ID
type
report
total income
total expense
weekly
monthly
yearly
Transaction
ID
amount
Date
associated
Place of expense
date
place name
contribute
amount
type
Savings
allocate
amount
name
duration
Investment
duration
amount
category
Budget

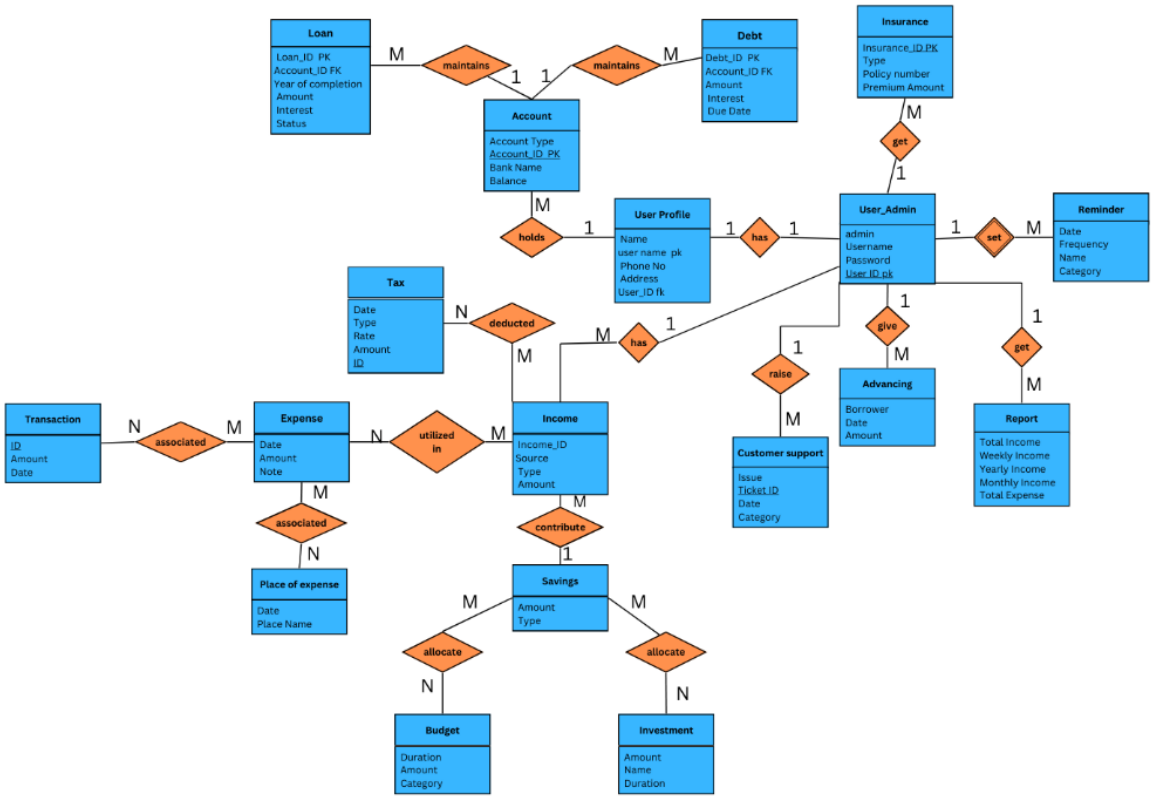
Empathy Map:


Empathy Map
Name: Reshma
What does he think and feel?
"I'm determined to improve my financial situation and want a solution that relieves the stress of managing money."
"I believe there's a better way to handle my finances, and I feel overwhelmed by the complexity of manual tracking."
"I'm motivated to take control of my finances, but I feel frustrated with the lack of easy-to-use tools."
What does he hear?
Advice from friends or family about managing finances.
Stories of others who have successfully improved their financial situation.
Promotions for financial management tools or apps.
What does he see?
Bank statements and credit card bills.
Budgeting spreadsheets or notebooks.
Ads for personal finance software or apps.
What does he say and do?
Saying: "I need to start budgeting."
Doing: Researching different budgeting methods or apps.
Saying: "I wish there was an easier way to track my expenses."
Doing: Trying out different budgeting tools or apps.
Pain
Tedious manual tracking of expenses.
Difficulty sticking to a budget.
Feeling overwhelmed by financial jargon and complexity.
Gain
Simplified and automated expense tracking.
Clear visualization of spending habits and financial goals.
Peace of mind knowing that finances are under control and goals are being met.

CHAPTER 2

DESIGN OF RELATIONAL SCHEMAS, CREATION OF DATABASE TABLES FOR THE PROJECT

Relational Schema:



Creation of Tables in Database:

- **USER PROFILE**

```
CREATE TABLE user_profile (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    user_name VARCHAR(45) NOT NULL,
    user_number INT NOT NULL,
    user_email VARCHAR(45) NOT NULL
);
```

- **ACCOUNT TYPE**

```
CREATE TABLE acc_type (
    type_id INT PRIMARY KEY AUTO_INCREMENT,
    acc_type VARCHAR(45) NOT NULL
);
```

- **ACCOUNT**

```
CREATE TABLE account (
    acc_no INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES user_profile(user_id),
    type_id INT NOT NULL,
    FOREIGN KEY (type_id) REFERENCES acc_type(type_id),
    acc_opendate DATE NOT NULL,
    acc_status VARCHAR(12) NOT NULL
);
```

- **TRANSACTION CATEGORY**

```
CREATE TABLE trn_cat (
    cat_id INT PRIMARY KEY AUTO_INCREMENT,
    cat_type VARCHAR(45) NOT NULL
);
```

- **TRANSACTION**

```
CREATE TABLE transaction (
    trn_id INT PRIMARY KEY AUTO_INCREMENT,
    acc_no INT NOT NULL,
```

```
    FOREIGN KEY (acc_no) REFERENCES account(acc_no),
    cat_id INT NOT NULL,
    FOREIGN KEY (cat_id) REFERENCES trn_cat(cat_id),
    trn_med VARCHAR(30) NOT NULL,
    trn_amt INT NOT NULL,
    trn_type VARCHAR(20) NOT NULL,
    trn_desc VARCHAR(200),
    trn_date DATE NOT NULL
);
```

- **LOAN**

```
CREATE TABLE loan (
    l_id INT PRIMARY KEY AUTO_INCREMENT,
    acc_no INT NOT NULL,
    FOREIGN KEY (acc_no) REFERENCES account(acc_no),
    l_amt INT NOT NULL,
    l_tenure INT NOT NULL,
    dwn_pay INT NOT NULL,
    due_date DATE NOT NULL,
    emi_amt INT NOT NULL,
    int_rate FLOAT NOT NULL,
    lsanction_date DATE NOT NULL,
    lstart_date DATE NOT NULL
);
```

- **DEBT**

```
CREATE TABLE debt (
    debt_id INT PRIMARY KEY AUTO_INCREMENT,
    account_id INT NOT NULL,
    amount INT NOT NULL,
    interest INT NOT NULL,
    due_date DATE NOT NULL,
    FOREIGN KEY (account_id) REFERENCES account(acc_no)
);
```

- **INSURANCE**

```
CREATE TABLE Insurance (
    Insurance_ID INT PRIMARY KEY AUTO_INCREMENT,
    Type VARCHAR(20) NOT NULL,
    Policy_Number VARCHAR(20) NOT NULL,
    Premium_Amount DECIMAL(10, 2) NOT NULL
);
```

- **REMINDER**

```
CREATE TABLE Reminder (
    Reminder_Date DATE,
    Frequency VARCHAR(10),
    Name VARCHAR(50),
    Category VARCHAR(20)
);
```

- **TAX**

```
CREATE TABLE Tax (
    Tax_Date DATE,
    Type VARCHAR(20),
    Rate DECIMAL(10, 2),
    Amount DECIMAL(10, 2),
    Tax_ID INT PRIMARY KEY AUTO_INCREMENT
);
```

- **EXPENSE**

```
CREATE TABLE Expense (
    Expense_Date DATE,
    Amount DECIMAL(10, 2),
    Note VARCHAR(255)
);
```

- **INCOME**

```
CREATE TABLE Income (
    Income_ID INT PRIMARY KEY AUTO_INCREMENT,
    Source VARCHAR(100),
    Type VARCHAR(50),
    Amount DECIMAL(10, 2)
);
```

- **PLACE OF EXPENSE**

```
CREATE TABLE Place_of_Expense (
    Expense_Date DATE,
```

```
    Place_Name VARCHAR(255)
);
```

- **SAVINGS**

```
CREATE TABLE Savings (
    Amount DECIMAL(10, 2),
    Type VARCHAR(20)
);
```

- **BUDGET**

```
CREATE TABLE Budget (
    Duration VARCHAR(20),
    Amount DECIMAL(10, 2),
    Category VARCHAR(20)
);
```

- **INVESTMENT**

```
CREATE TABLE Investment (
    Amount DECIMAL(10, 2),
    Name VARCHAR(100),
    Duration VARCHAR(20)
);
```

- **CUSTOMER SUPPORT**

```
CREATE TABLE Customer_Support (
    Issue VARCHAR(255),
    Ticket_ID INT PRIMARY KEY AUTO_INCREMENT,
    Ticket_Date DATE,
```

```
    Category VARCHAR(20)
);
```

- **ADVANCING**

```
CREATE TABLE Advancing (
    Borrower VARCHAR(100),
    Adv_Date DATE,
    Amount DECIMAL(10, 2)
);
```

- **REPORT**

```
CREATE TABLE Report (
    Total_Income DECIMAL(10, 2),
    Weekly_Income DECIMAL(10, 2),
    Yearly_Income DECIMAL(10, 2),
    Monthly_Income DECIMAL(10, 2),
    Total_Expense DECIMAL(10, 2)
);
```

Insertion of Values in Tables

- **Inserting Values in Table USER PROFILE**

```
INSERT INTO user_profile (user_name, user_number, user_email)
VALUES
('John Doe', 123456789, 'johndoe@example.com'),
('Jane Smith', 987654321, 'janesmith@example.com'),
('Jim Brown', 456789123, 'jimbrown@example.com'),
```

('Jake White', 789123456, 'jakewhite@example.com'),

('Jill Black', 135792468, 'jillblack@example.com'),

('Jessica Green', 246813579, 'jessicagreen@example.com'),

('James Yellow', 579246813, 'jamesyellow@example.com'),

('Joy Blue', 813579246, 'joyblue@example.com'),

('Jack Red', 924681357, 'jackred@example.com'),

('Janie Purple', 246813579, 'janiepurple@example.com'),

('Jared Orange', 357924681, 'jaredorange@example.com'),

('Jasmine Pink', 468135792, 'jasminepink@example.com');

- **Inserting Values in Table ACCOUNT TYPE**

INSERT INTO acc_type (acc_type)

VALUES

('Checking'),

('Savings'),

('Credit Card'),

('Investment'),

('Retirement'),

('Loan'),

('Mortgage'),

('Business'),

('Student'),

('Health Savings Account'),

('Travel'),

('Insurance');

- **Inserting Values in Table ACCOUNT**

INSERT INTO account (user_id, type_id, acc_opendate, acc_status)

VALUES

(1, 1, '2022-01-01', 'Open'),

(2, 2, '2022-01-05', 'Active'),

(3, 3, '2022-02-10', 'Closed'),

(4, 4, '2022-03-15', 'Open'),

(5, 5, '2022-04-20', 'Active'),

(6, 6, '2022-05-25', 'Open'),

(7, 7, '2022-06-30', 'Active'),

(8, 8, '2022-07-05', 'Closed'),

(9, 9, '2022-08-10', 'Open'),

(10, 10, '2022-09-15', 'Active'),

(11, 11, '2022-10-20', 'Open'),

(12, 12, '2022-11-25', 'Active');

- **Inserting Values in Table TRANSACTION CATEGORY**

INSERT INTO trn_cat (cat_type)

VALUES

('Groceries'),

('Dining Out'),

('Entertainment'),

('Bills'),

('Travel'),

('Shopping'),

('Healthcare'),

('Gifts'),

('Miscellaneous'),

('Salary'),

('Rent'),

('Utilities');

- **Inserting Values in Table TRANSACTION**

INSERT INTO transaction (acc_no, cat_id, trn_med, trn_amt, trn_type, trn_desc, trn_date)

VALUES

(1, 1, 'Supermarket', 50, 'Debit', 'Weekly groceries', '2022-01-05'),

(2, 2, 'Restaurant', 30, 'Debit', 'Dinner with friends', '2022-01-10'),

(3, 3, 'Movie Theater', 20, 'Debit', 'Movie night', '2022-01-15'),

(4, 4, 'Electric Company', 100, 'Debit', 'Monthly electricity bill', '2022-01-20'),

(5, 5, 'Airline', 200, 'Debit', 'Flight ticket', '2022-01-25'),

(6, 6, 'Online Store', 50, 'Debit', 'Shopping spree', '2022-01-30'),

(7, 7, 'Pharmacy', 15, 'Debit', 'Medication', '2022-02-05'),

(8, 8, 'Gift Shop', 25, 'Debit', 'Birthday present', '2022-02-10'),

(9, 9, 'Miscellaneous', 10, 'Debit', 'Miscellaneous expense', '2022-02-15'),

(10, 10, 'Salary Deposit', 5000, 'Credit', 'Monthly salary', '2022-02-20'),

(11, 11, 'Rent Payment', 1000, 'Debit', 'Monthly rent', '2022-02-25'),

(12, 12, 'Utilities Company', 150, 'Debit', 'Utility bill', '2022-03-01');

- **Inserting Values in Table LOAN**

INSERT INTO loan (acc_no, l_amt, l_tenure, dwn_pay, due_date, emi_amt, int_rate, lsanction_date, lstart_date)

VALUES

(1, 5000, 12, 1000, '2022-01-05', 500, 6.5, '2022-01-05', '2022-01-10'),

(2, 8000, 24, 1500, '2022-01-10', 400, 7.2, '2022-01-10', '2022-01-15'),

(3, 10000, 36, 2000, '2022-01-15', 300, 8.0, '2022-01-15', '2022-01-20'),

(4, 12000, 48, 2500, '2022-01-20', 600, 8.5, '2022-01-20', '2022-01-25'),

(5, 15000, 60, 3000, '2022-01-25', 700, 9.0, '2022-01-25', '2022-01-30'),

(6, 18000, 72, 3500, '2022-01-30', 800, 9.5, '2022-01-30', '2022-02-05'),

(7, 20000, 84, 4000, '2022-02-05', 900, 10.0, '2022-02-05', '2022-02-10'),

(8, 22000, 96, 4500, '2022-02-10', 1000, 10.5, '2022-02-10', '2022-02-15'),

(9, 25000, 108, 5000, '2022-02-15', 1100, 11.0, '2022-02-15', '2022-02-20'),

(10, 28000, 120, 5500, '2022-02-20', 1200, 11.5, '2022-02-20', '2022-02-25'),

(11, 30000, 132, 6000, '2022-02-25', 1300, 12.0, '2022-02-25', '2022-03-01'),

(12, 35000, 144, 6500, '2022-03-01', 1400, 12.5, '2022-03-01', '2022-03-05');

- **Inserting Values in Table DEBT**

INSERT INTO debt (account_id, amount, interest, due_date)

VALUES

(1, 500, 50, '2022-01-05'),

(2, 800, 80, '2022-01-10'),

(3, 1000, 100, '2022-01-15'),

(4, 1200, 120, '2022-01-20'),

(5, 1500, 150, '2022-01-25'),

(6, 1800, 180, '2022-01-30'),

(7, 2000, 200, '2022-02-05'),

(8, 2200, 220, '2022-02-10'),

(9, 2500, 250, '2022-02-15'),

(10, 2800, 280, '2022-02-20'),

(11, 3000, 300, '2022-02-25'),

(12, 3500, 350, '2022-03-01');

- **Inserting Values in Table INSURANCE**

INSERT INTO Insurance (Type, Policy_Number, Premium_Amount)

VALUES

('Life', 'POL001', 500.00),

('Health', 'POL002', 800.00),

('Home', 'POL003', 1200.00),

('Auto', 'POL004', 600.00),

('Travel', 'POL005', 300.00),

('Pet', 'POL006', 200.00),

('Renter', 'POL007', 400.00),

('Disability', 'POL008', 1000.00),

('Business', 'POL009', 1500.00),

('Accidental', 'POL010', 700.00),

('Critical Illness', 'POL011', 900.00),

('Flood', 'POL012', 1100.00);

- **Inserting Values in Table REMINDER**

INSERT INTO Reminder (Reminder_Date, Frequency, Name, Category)

VALUES

('2022-01-01', 'Daily', 'Pay bills', 'Finance'),

('2022-01-02', 'Weekly', 'Grocery shopping', 'Personal'),

('2022-01-03', 'Monthly', 'Rent payment', 'Housing'),

('2022-01-04', 'Yearly', 'Insurance renewal', 'Finance'),

('2022-01-05', 'Daily', 'Exercise', 'Health'),

('2022-01-06', 'Weekly', 'Laundry', 'Personal'),

('2022-01-07', 'Monthly', 'Utilities payment', 'Housing'),

('2022-01-08', 'Yearly', 'Health checkup', 'Health'),

('2022-01-09', 'Daily', 'Read', 'Personal'),

('2022-01-10', 'Weekly', 'Cleaning', 'Personal'),

('2022-01-11', 'Monthly', 'Car servicing', 'Vehicle'),

('2022-01-12', 'Yearly', 'Tax filing', 'Finance');

- **Inserting Values in Table TAX**

INSERT INTO Tax (Tax_Date, Type, Rate, Amount)

VALUES

('2022-01-01', 'Income Tax', 10.5, 500),

('2022-01-02', 'Property Tax', 5.2, 300),

('2022-01-03', 'Sales Tax', 7.8, 200),

('2022-01-04', 'Corporate Tax', 15.3, 800),

('2022-01-05', 'Capital Gains Tax', 12.6, 1000),

('2022-01-06', 'Excise Tax', 8.4, 400),

('2022-01-07', 'Inheritance Tax', 20.1, 600),

('2022-01-08', 'Vehicle Tax', 6.7, 700),

('2022-01-09', 'Customs Duty', 9.8, 900),

('2022-01-10', 'VAT', 11.2, 1200),

('2022-01-11', 'Luxury Tax', 18.9, 1100),

('2022-01-12', 'Sin Tax', 14.3, 1500);

- **Inserting Values in Table EXPENSE**

INSERT INTO Expense (Expense_Date, Amount, Note)

VALUES

('2022-01-01', 50.00, 'Groceries'),

('2022-01-02', 30.00, 'Dining out'),

('2022-01-03', 20.00, 'Movie tickets'),

('2022-01-04', 100.00, 'Electricity bill'),

('2022-01-05', 200.00, 'Flight ticket'),

('2022-01-06', 50.00, 'Online shopping'),

('2022-01-07', 15.00, 'Medication'),

('2022-01-08', 25.00, 'Gift'),

('2022-01-09', 10.00, 'Miscellaneous'),

('2022-01-10', 5000.00, 'Monthly salary'),

('2022-01-11', 1000.00, 'Rent payment'),

('2022-01-12', 150.00, 'Utility bill');

- **Inserting Values in Table INCOME**

INSERT INTO Income (Source, Type, Amount)

VALUES

('Job', 'Salary', 3000.00),

('Freelance Work', 'Income', 500.00),

('Investment', 'Income', 1000.00),

('Business', 'Profit', 2000.00),

('Rent', 'Income', 1000.00),

('Gift', 'Income', 50.00),

('Bonus', 'Income', 300.00),

('Interest', 'Income', 80.00),

('Dividends', 'Income', 150.00),

('Royalties', 'Income', 200.00),

('Refunds', 'Income', 70.00),

('Other', 'Income', 120.00);

- **Inserting Values in Table PLACE OF EXPENSE**

INSERT INTO Place_of_Expense (Expense_Date, Place_Name)

VALUES

('2022-01-01', 'Supermarket'),

('2022-01-02', 'Restaurant'),

('2022-01-03', 'Movie Theater'),

('2022-01-04', 'Electric Company'),

('2022-01-05', 'Airline'),

('2022-01-06', 'Online Store'),

('2022-01-07', 'Pharmacy'),

('2022-01-08', 'Gift Shop'),

('2022-01-09', 'Miscellaneous Store'),

('2022-01-10', 'Employer'),

('2022-01-11', 'Landlord'),

('2022-01-12', 'Utility Company');

- **Inserting Values in Table SAVINGS**

INSERT INTO Savings (Amount, Type)

VALUES

(500.00, 'Emergency Fund'),

(800.00, 'Vacation Fund'),

(1200.00, 'Retirement Fund'),

(600.00, 'Education Fund'),

(300.00, 'Home Fund'),

(200.00, 'Car Fund'),

(400.00, 'Health Fund'),

(1000.00, 'Investment Fund'),

(1500.00, 'Business Fund'),

(700.00, 'Miscellaneous Fund'),

(900.00, 'Savings for Taxes'),

(1100.00, 'Special Occasion');

- **Inserting Values in Table BUDGET**

INSERT INTO Budget (Duration, Amount, Category)

VALUES

('Monthly', 500.00, 'Groceries'),

('Monthly', 300.00, 'Dining out'),

('Monthly', 200.00, 'Entertainment'),

('Monthly', 100.00, 'Utilities'),

('Monthly', 400.00, 'Transportation'),

('Monthly', 150.00, 'Miscellaneous'),

('Monthly', 50.00, 'Gifts'),

('Monthly', 80.00, 'Healthcare'),

('Monthly', 120.00, 'Education'),

('Monthly', 70.00, 'Savings'),

('Monthly', 200.00, 'Travel'),

('Monthly', 100.00, 'Investment');

- **Inserting Values in Table INVESTMENT**

INSERT INTO Investment (Amount, Name, Duration)

VALUES

(100.00, 'Stocks', 'Short-term'),

(200.00, 'Bonds', 'Short-term'),

(300.00, 'Mutual Funds', 'Short-term'),

(400.00, 'Real Estate', 'Long-term'),

(500.00, 'IRA', 'Long-term'),

(600.00, '401(k)', 'Long-term'),

(700.00, 'Cryptocurrency', 'Long-term'),

(800.00, 'ETFs', 'Short-term'),

(900.00, 'Commodities', 'Short-term'),

(1000.00, 'Savings Account', 'Short-term'),

(1100.00, 'Peer-to-Peer Lending', 'Short-term'),

(1200.00, 'Retirement Account', 'Long-term');

- **Inserting Values in Table CUSTOMER SUPPORT**

INSERT INTO Customer_Support (Issue, Ticket_Date, Category)

VALUES

('Billing inquiry', '2022-01-01', 'Billing'),

('Technical issue', '2022-01-02', 'Technical'),

('Product inquiry', '2022-01-03', 'General'),

('Account access problem', '2022-01-04', 'Account'),

('Refund request', '2022-01-05', 'Billing'),

('Service complaint', '2022-01-06', 'Service'),

('Cancellation request', '2022-01-07', 'Account'),

('Feature request', '2022-01-08', 'General'),

('Order status inquiry', '2022-01-09', 'Order'),

('Return authorization', '2022-01-10', 'Order'),

('Product defect report', '2022-01-11', 'Service'),

('Feedback submission', '2022-01-12', 'General');

- **Inserting Values in Table ADVANCING**

INSERT INTO Advancing (Borrower, Adv_Date, Amount)

VALUES

('John Smith', '2022-01-01', 200.00),

('Emily Johnson', '2022-01-02', 300.00),

('Michael Williams', '2022-01-03', 400.00),

('Emma Jones', '2022-01-04', 500.00),

('James Brown', '2022-01-05', 600.00),

('Olivia Davis', '2022-01-06', 700.00),

('William Miller', '2022-01-07', 800.00),

('Sophia Wilson', '2022-01-08', 900.00),

('Alexander Taylor', '2022-01-09', 1000.00),

('Grace Anderson', '2022-01-10', 1100.00),

('Daniel Thomas', '2022-01-11', 1200.00),

('Mia Moore', '2022-01-12', 1300.00);

- **Inserting Values in Table REPORT**

INSERT INTO Report (Total_Income, Weekly_Income, Yearly_Income, Monthly_Income, Total_Expense)

VALUES

(5000.00, 1000.00, 60000.00, 5000.00, 3500.00),

(6000.00, 1200.00, 72000.00, 6000.00, 4000.00),

(7000.00, 1400.00, 84000.00, 7000.00, 4500.00),

(8000.00, 1600.00, 96000.00, 8000.00, 5000.00),

(9000.00, 1800.00, 108000.00, 9000.00, 5500.00),

(10000.00, 2000.00, 120000.00, 10000.00, 6000.00),

(11000.00, 2200.00, 132000.00, 11000.00, 6500.00),

(12000.00, 2400.00, 144000.00, 12000.00, 7000.00),

(13000.00, 2600.00, 156000.00, 13000.00, 7500.00),

(14000.00, 2800.00, 168000.00, 14000.00, 8000.00),

(15000.00, 3000.00, 180000.00, 15000.00, 8500.00),

(16000.00, 3200.00, 192000.00, 16000.00, 9000.00);

CHAPTER 3

COMPLEX QUERIES BASED ON THE CONCEPTS OF CONSTRAINTS, SETS, JOINS, VIEWS, TRIGGERS AND CURSORS

Sets:

```
SELECT user_name FROM user_profile
UNION
SELECT acc_type FROM acc_type;

SELECT user_name FROM user_profile
UNION ALL
SELECT acc_type FROM acc_type;

SELECT user_name FROM user_profile
INTERSECT
SELECT user_email FROM user_profile;

SELECT user_name FROM user_profile
MINUS
SELECT user_email FROM user_profile;
```

Joins:

```
SELECT a.acc_no, a.acc_status, u.user_name
FROM account a
INNER JOIN user_profile u ON a.user_id = u.user_id;

SELECT a.acc_no, a.acc_status, u.user_name
FROM account a
LEFT JOIN user_profile u ON a.user_id = u.user_id;

SELECT a.acc_no, a.acc_status, u.user_name
```

```
FROM account a
RIGHT JOIN user_profile u ON a.user_id = u.user_id;
```

Views:

```
CREATE VIEW user_account_info AS
SELECT u.user_name, a.acc_no, at.acc_type
FROM user_profile u
JOIN account a ON u.user_id = a.user_id
JOIN acc_type at ON a.type_id = at.type_id;

CREATE VIEW account_transaction_categories AS
SELECT a.acc_no, a.acc_status, tc.cat_type
FROM account a
JOIN transaction t ON a.acc_no = t.acc_no
JOIN trn_cat tc ON t.cat_id = tc.cat_id;

CREATE VIEW loan_debt_info AS
SELECT l.l_id, l.l_amt, l.l_tenure, l.dwn_pay, l.due_date AS loan_due_date, l.emi_amt,
l.int_rate, l.lsanction_date, l.lstart_date, d.amount, d.interest, d.due_date AS debt_due_date
FROM loan l
LEFT JOIN debt d ON l.acc_no = d.account_id;

CREATE VIEW income_savings_info AS
SELECT i.Income_ID, i.Source, i.Type AS income_type, i.Amount AS income_amount,
s.Amount AS savings_amount, s.Type AS savings_type
FROM Income i
LEFT JOIN Savings s ON i.Type = s.Type;

CREATE VIEW customer_support_categories AS
SELECT cs.Issue, cs.Ticket_ID, cs.Ticket_Date, cs.Category
FROM Customer_Support cs;
```

Triggers and Cursors:

```
DELIMITER //

CREATE TRIGGER update_summary AFTER INSERT ON transaction
FOR EACH ROW
BEGIN
   DECLARE acc_status_var VARCHAR(12);

   -- Retrieve the account status associated with the inserted transaction
   SELECT acc_status INTO acc_status_var
   FROM account
   WHERE acc_no = NEW.acc_no;

   -- Update summary_table based on the account status
   IF acc_status_var = 'Open' THEN
      UPDATE summary_table
      SET total_open_transactions = total_open_transactions + 1
      WHERE summary_condition_column = NEW.acc_no;
   ELSE
      UPDATE summary_table
      SET total_closed_transactions = total_closed_transactions + 1
      WHERE summary_condition_column = NEW.acc_no;
   END IF;
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER update_loan_status AFTER INSERT ON loan
FOR EACH ROW
BEGIN
   IF NEW.l_amt > 5000 THEN
```

```
        UPDATE loan_status_table
        SET high_loan_count = high_loan_count + 1
        WHERE condition_column = NEW.l_id;
    ELSE
        UPDATE loan_status_table
        SET low_loan_count = low_loan_count + 1
        WHERE condition_column = NEW.l_id;
    END IF;
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER update_debt_status AFTER INSERT ON debt
FOR EACH ROW
BEGIN
    IF NEW.amount > 10000 THEN
        UPDATE debt_status_table
        SET high_debt_count = high_debt_count + 1
        WHERE condition_column = NEW.debt_id;
    ELSE
        UPDATE debt_status_table
        SET low_debt_count = low_debt_count + 1
        WHERE condition_column = NEW.debt_id;
    END IF;
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER update_income_summary AFTER INSERT ON Income
FOR EACH ROW
```

```
BEGIN
   DECLARE income_category VARCHAR(50);

   -- Retrieve the income category associated with the inserted record
   SELECT Type INTO income_category
   FROM Income
   WHERE Income_ID = NEW.Income_ID;

   -- Update income_summary_table based on the income category
   IF income_category = 'Salary' THEN
      UPDATE income_summary_table
      SET total_salary = total_salary + NEW.Amount
      WHERE condition_column = NEW.Income_ID;
   ELSE
      UPDATE income_summary_table
      SET total_other_income = total_other_income + NEW.Amount
      WHERE condition_column = NEW.Income_ID;
   END IF;
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER update_expense_summary AFTER INSERT ON Expense
FOR EACH ROW
BEGIN
   DECLARE expense_category VARCHAR(255);

   -- Retrieve the expense category associated with the inserted record
   SELECT Note INTO expense_category
   FROM Expense
   WHERE Expense_Date = NEW.Expense_Date;
```

```
    -- Update expense_summary_table based on the expense category
    IF expense_category = 'Groceries' THEN
        UPDATE expense_summary_table
        SET total_groceries = total_groceries + NEW.Amount
        WHERE condition_column = NEW.Expense_Date;
    ELSE
        UPDATE expense_summary_table
        SET total_other_expenses = total_other_expenses + NEW.Amount
        WHERE condition_column = NEW.Expense_Date;
    END IF;
END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE process_transactions()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE acc_no_var INT;
    DECLARE cat_type_var VARCHAR(45);
    DECLARE trn_amt_var INT;

    -- Declare a cursor for selecting data from the transaction table
    DECLARE transaction_cursor CURSOR FOR
    SELECT acc_no, cat_type, trn_amt
    FROM transaction;

    -- Declare handler for the end of the cursor
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Open the cursor
    OPEN transaction_cursor;
```

```
    -- Start fetching rows from the cursor
    transaction_loop: LOOP
        -- Fetch the next row from the cursor into variables
        FETCH transaction_cursor INTO acc_no_var, cat_type_var, trn_amt_var;

        -- Check if there are no more rows to fetch
        IF done THEN
            LEAVE transaction_loop;
        END IF;

        -- Process the fetched row (perform operations based on the data)
        -- For example, you can print the values or perform calculations
        -- Here, we're just printing the fetched values
        SELECT acc_no_var, cat_type_var, trn_amt_var;
    END LOOP;

    -- Close the cursor
    CLOSE transaction_cursor;
END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE process_loans()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE loan_id_var INT;
    DECLARE loan_amount_var INT;
    DECLARE loan_tenure_var INT;

    -- Declare a cursor for selecting data from the loan table
    DECLARE loan_cursor CURSOR FOR
    SELECT l_id, l_amt, l_tenure
```

```
    FROM loan;

    -- Declare handler for the end of the cursor
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Open the cursor
    OPEN loan_cursor;

    -- Start fetching rows from the cursor
    loan_loop: LOOP
        -- Fetch the next row from the cursor into variables
        FETCH loan_cursor INTO loan_id_var, loan_amount_var, loan_tenure_var;

        -- Check if there are no more rows to fetch
        IF done THEN
            LEAVE loan_loop;
        END IF;

        -- Process the fetched row (perform operations based on the data)
        -- For example, you can print the values or perform calculations
        -- Here, we're just printing the fetched values
        SELECT loan_id_var, loan_amount_var, loan_tenure_var;
    END LOOP;

    -- Close the cursor
    CLOSE loan_cursor;
END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE process_debts()
BEGIN
```

```
DECLARE done INT DEFAULT FALSE;
DECLARE debt_id_var INT;
DECLARE account_id_var INT;
DECLARE amount_var INT;
DECLARE interest_var INT;

-- Declare a cursor for selecting data from the debt table
DECLARE debt_cursor CURSOR FOR
SELECT debt_id, account_id, amount, interest
FROM debt;

-- Declare handler for the end of the cursor
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

-- Open the cursor
OPEN debt_cursor;

-- Start fetching rows from the cursor
debt_loop: LOOP
    -- Fetch the next row from the cursor into variables
    FETCH debt_cursor INTO debt_id_var, account_id_var, amount_var, interest_var;

    -- Check if there are no more rows to fetch
    IF done THEN
        LEAVE debt_loop;
    END IF;

    -- Process the fetched row (perform operations based on the data)
    -- For example, you can print the values or perform calculations
    -- Here, we're just printing the fetched values
    SELECT debt_id_var, account_id_var, amount_var, interest_var;
END LOOP;

-- Close the cursor
```

```
    CLOSE debt_cursor;
END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE process_expenses()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE expense_date_var DATE;
    DECLARE amount_var DECIMAL(10, 2);
    DECLARE note_var VARCHAR(255);

    -- Declare a cursor for selecting data from the expense table
    DECLARE expense_cursor CURSOR FOR
    SELECT Expense_Date, Amount, Note
    FROM Expense;

    -- Declare handler for the end of the cursor
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Open the cursor
    OPEN expense_cursor;

    -- Start fetching rows from the cursor
    expense_loop: LOOP
        -- Fetch the next row from the cursor into variables
        FETCH expense_cursor INTO expense_date_var, amount_var, note_var;

        -- Check if there are no more rows to fetch
        IF done THEN
            LEAVE expense_loop;
        END IF;
```

```
        -- Process the fetched row (perform operations based on the data)
        -- For example, you can print the values or perform calculations
        -- Here, we're just printing the fetched values
        SELECT expense_date_var, amount_var, note_var;
    END LOOP;

    -- Close the cursor
    CLOSE expense_cursor;
END //

DELIMITER ;
```

CHAPTER 4

ANALYSING THE PITFALLS, IDENTIFYING THE DEPENDENCIES, AND APPLYING NORMALIZATIONS

Pitfalls in our Tables:

User_Profile Table:

- **Data Redundancy**: If multiple users share the same **user_name** or **user_email**, this would result in redundant data. For example, if two users have the same email address, it would be duplicated in multiple rows, leading to redundancy.
- **Update Anomalies**: If a user wants to update their email address, it would require modifying multiple rows if they have multiple accounts. This increases the likelihood of errors and inconsistencies.
- **Insertion Anomalies**: If a new user wants to register but doesn't provide all the information (e.g., email address), it may not be possible to insert the record due to NOT NULL constraints, leading to insertion anomalies.

Transaction Table:

- **Data Redundancy:** If multiple transactions have the same **trn_med** or **trn_desc**, this would result in redundant data. For example, if multiple transactions involve the same medium (e.g., "Online Store"), the medium name would be duplicated in multiple rows.
- **Update Anomalies:** If there's a typo in the **trn_med** or **trn_desc** of a transaction, updating it would require modifying multiple rows if the same value is repeated across transactions, increasing the risk of errors.
- **Insertion Anomalies:** If a new transaction record doesn't contain all the necessary information (e.g., description), it may not be possible to insert the record due to NOT NULL constraints, leading to insertion anomalies.

Account Table:

- **Data Redundancy:** If multiple accounts have the same **acc_status**, this would result in redundant data. For example, if multiple accounts have the status "Active," the status would be duplicated in multiple rows.
- **Update Anomalies:** If there's a need to update the **acc_status** of an account, it would require modifying multiple rows if the same status is repeated across accounts, increasing the risk of errors.
- **Insertion Anomalies:** If a new account record doesn't contain all the necessary information (e.g., status), it may not be possible to insert the record due to NOT NULL constraints, leading to insertion anomalies.

Applying Normalisation:

1)Loan:

1NF:

```
CREATE TABLE Loan (
   L_ID INT NOT NULL PRIMARY KEY,
   A_Number INT NOT NULL,
   L_amount INT NOT NULL,
   L_tenure INT NOT NULL,
   Down_Payment INT,
   Due_Date DATE NOT NULL,
   EMI_amount INT NOT NULL,
   Interest_Rate FLOAT NOT NULL,
   L_Sanction_Date DATE NOT NULL,
   L_Start_Date DATE NOT NULL,
   FOREIGN KEY (A_Number) REFERENCES Account(A_Number)
```

```
);
```

2NF:

```
CREATE TABLE Loan_Payment_Details(
    Due_Date DATE NOT NULL,
    L_ID INT NOT NULL PRIMARY KEY,
    FOREIGN KEY (L_ID) REFERENCES Loan(L_ID)
);
```

3NF:

```
CREATE TABLE Loan_Repayment_Details(
        EMI_amount INT NOT NULL,
        L_ID INT NOT NULL PRIMARY KEY,
    FOREIGN KEY (L_ID) REFERENCES Loan(L_ID)
);
```

Normalized Table:

```
SQL> desc Loan
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 L_ID                                      NOT NULL NUMBER(38)
 A_NUMBER                                  NOT NULL NUMBER(38)
 L_AMOUNT                                  NOT NULL NUMBER(38)
 L_TENURE                                  NOT NULL NUMBER(38)
 DOWN_PAYMENT                                       NUMBER(38)
 DUE_DATE                                  NOT NULL DATE
 EMI_AMOUNT                                NOT NULL NUMBER(38)
 INTEREST_RATE                             NOT NULL FLOAT(126)
 L_SANCTION_DATE                           NOT NULL DATE
 L_START_DATE                              NOT NULL DATE
```

```
SQL> desc Loan_Payment_Details
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DUE_DATE                                  NOT NULL DATE
 L_ID                                      NOT NULL NUMBER(38)
```

```
SQL> desc Loan_Repayment_Details
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMI_AMOUNT                                NOT NULL NUMBER(38)
 L_ID                                      NOT NULL NUMBER(38)
```

2)Transaction:

1NF:

CREATE TABLE Transaction (

T_ID INT NOT NULL PRIMARY KEY,

A_Number INT NOT NULL,

T_date DATE NOT NULL,

T_Medium VARCHAR(15) NOT NULL,

T_Amount INT NOT NULL,

T_Type VARCHAR(15) NOT NULL,

T_Description VARCHAR(200),

Category_ID INT NOT NULL,

FOREIGN KEY (A_Number) REFERENCES Account(A_Number),

FOREIGN KEY (Category_ID) REFERENCES T_Category(Category_ID)

);

2NF:

CREATE TABLE Transaction_Details (

```
    T_ID INT NOT NULL PRIMARY KEY,
    A_Number INT NOT NULL,
    T_date DATE NOT NULL,
    T_Amount INT NOT NULL,
    Category_ID INT NOT NULL,
    FOREIGN KEY (T_ID) REFERENCES Transaction(T_ID),
    FOREIGN KEY (Category_ID) REFERENCES T_Category(Category_ID)
);
```

3NF:

```
CREATE TABLE Transaction_Type(
    T_ID INT NOT NULL PRIMARY KEY,
    T_Medium VARCHAR(15) NOT NULL,
    T_Type VARCHAR(15) NOT NULL,
    FOREIGN KEY (T_ID) REFERENCES Transaction(T_ID)
);
```

Original table:

```
SQL> desc Transaction
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 T_ID                                      NOT NULL NUMBER(38)
 A_NUMBER                                  NOT NULL NUMBER(38)
 T_DATE                                    NOT NULL DATE
 T_MEDIUM                                  NOT NULL VARCHAR2(15)
 T_AMOUNT                                  NOT NULL NUMBER(38)
 T_TYPE                                    NOT NULL VARCHAR2(15)
 T_DESCRIPTION                                      VARCHAR2(200)
 CATEGORY_ID                               NOT NULL NUMBER(38)
```

Normalized table:

```
SQL> desc Transaction
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 T_ID                                      NOT NULL NUMBER(38)
 A_NUMBER                                  NOT NULL NUMBER(38)
 T_DATE                                    NOT NULL DATE
 T_MEDIUM                                  NOT NULL VARCHAR2(15)
 T_AMOUNT                                  NOT NULL NUMBER(38)
 T_TYPE                                    NOT NULL VARCHAR2(15)
 T_DESCRIPTION                                      VARCHAR2(200)
 CATEGORY_ID                               NOT NULL NUMBER(38)
```

```
SQL> desc Transaction_Details
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 T_ID                                      NOT NULL NUMBER(38)
 A_NUMBER                                  NOT NULL NUMBER(38)
 T_DATE                                    NOT NULL DATE
 T_AMOUNT                                  NOT NULL NUMBER(38)
 CATEGORY_ID                               NOT NULL NUMBER(38)
```

```
SQL> desc Transaction_Type
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 T_ID                                      NOT NULL NUMBER(38)
 T_MEDIUM                                  NOT NULL VARCHAR2(15)
 T_TYPE                                    NOT NULL VARCHAR2(15)
```

3)Accounts:

1NF:

```
CREATE TABLE Account_1 (
   A_Number INT NOT NULL PRIMARY KEY,
   U_ID INT NOT NULL,
   Type_ID INT NOT NULL,
   A_OPENdate DATE NOT NULL,
   A_Status VARCHAR(6) NOT NULL,
   FOREIGN KEY (U_ID) REFERENCES User_Profile(U_ID)
);
```

2NF:

```
CREATE TABLE Account_2 (
   A_Number INT NOT NULL PRIMARY KEY,
   U_ID INT NOT NULL,
   Type_ID INT NOT NULL,
   A_OPENdate DATE NOT NULL,
   A_Status VARCHAR(6) NOT NULL,
   FOREIGN KEY (U_ID) REFERENCES User_Profile(U_ID),
   FOREIGN KEY (Type_ID) REFERENCES Account_Type(Type_ID)
);
```

3NF:

```
CREATE TABLE Account_Details (
   A_Number INT NOT NULL PRIMARY KEY,
   U_ID INT NOT NULL,
   Type_ID INT NOT NULL,
   A_Type VARCHAR(15) NOT NULL,
   A_OPENdate DATE NOT NULL,
   A_Status VARCHAR(6) NOT NULL,
   FOREIGN KEY (U_ID) REFERENCES User_Profile(U_ID),
   FOREIGN KEY (Type_ID) REFERENCES Account_Type(Type_ID)
);
```

Original Table:

```
SQL> select * from Account;

  A_NUMBER       U_ID    TYPE_ID A_OPENDAT A_STAT  A_BALANCE
---------- ---------- ---------- --------- ------ ----------
     50001          1          1 01-MAR-24 Idle        14230
     50002          1          2 02-MAR-24 Idle        -3050
     50003          2          1 01-MAR-24 ACTIVE          0
     50004          2          2 02-MAR-24 ACTIVE          0
     50005          3          1 01-MAR-24 ACTIVE          0
     50006          3          2 02-MAR-24 ACTIVE          0
```

Normalized Table:

```
SQL> desc Account
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A_NUMBER                                  NOT NULL NUMBER(38)
 U_ID                                      NOT NULL NUMBER(38)
 TYPE_ID                                   NOT NULL NUMBER(38)
 A_OPENDATE                                NOT NULL DATE
 A_STATUS                                  NOT NULL VARCHAR2(6)
 A_BALANCE                                          NUMBER(38)
 NEW_A_NUMBER                                       NUMBER(38)
```

```
SQL> desc Account_1
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------------------
 A_NUMBER                                  NOT NULL NUMBER(38)
 U_ID                                      NOT NULL NUMBER(38)
 TYPE_ID                                   NOT NULL NUMBER(38)
 A_TYPE                                    NOT NULL VARCHAR2(15)
 A_OPENDATE                                NOT NULL DATE
 A_STATUS                                  NOT NULL VARCHAR2(6)

SQL> desc Account_2
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------------------
 A_NUMBER                                  NOT NULL NUMBER(38)
 U_ID                                      NOT NULL NUMBER(38)
 TYPE_ID                                   NOT NULL NUMBER(38)
 A_OPENDATE                                NOT NULL DATE
 A_STATUS                                  NOT NULL VARCHAR2(6)
```

```
SQL> desc Account_Type
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TYPE_ID                                   NOT NULL NUMBER(38)
 A_TYPE                                    NOT NULL VARCHAR2(15)
```

CHAPTER 5

IMPLEMENTATION OF CONCURRENCY CONTROL AND RECOVERY MECHANISMS, FRONT END APPLICATION

Concurrency in the context of your XAMPP app, implemented using PHP and hosted on XAMPP, refers to the system's ability to manage multiple users accessing and interacting with the application simultaneously, especially during peak usage times such as exam periods. XAMPP, which stands for Cross-Platform, Apache, MySQL, PHP, and Perl, provides a local development environment for building and testing web applications like your app.

In the case of the app hosted on XAMPP, concurrency becomes crucial for ensuring smooth user experience and maintaining data integrity. XAMPP utilizes Apache as the web server, which handles incoming HTTP requests and serves PHP scripts to users' web browsers. PHP scripts executed by Apache may access data stored in the MySQL database, which is also included in the XAMPP stack.

To effectively manage concurrency in the exam app hosted on XAMPP, similar strategies as mentioned earlier can be employed:

1. **Session Management:** Utilize PHP's built-in session management mechanisms to handle user sessions, ensuring that each user's session data is isolated and maintained correctly, even during concurrent access.
2. **Database Locking:** Implement database locking mechanisms in MySQL to prevent data inconsistencies and conflicts caused by concurrent access. Using locks such as row-level or table-level locks can ensure that only one user can modify specific records at a time.
3. **Concurrency Control:** Implement concurrency control mechanisms within the PHP application logic to manage access to shared resources and prevent race conditions. Using techniques like mutexes or semaphores can ensure that critical sections of code are executed by only one user at a time.
4. **Optimized Resource Usage:** Ensure efficient resource management by optimizing database queries, minimizing file I/O operations, and caching frequently accessed data to improve performance under concurrent usage scenarios.

By implementing these concurrency management strategies within the PHP-based exam app hosted on XAMPP, you can ensure that the application can handle multiple users accessing and interacting with the system concurrently, maintaining data integrity, consistency, and performance, even when hosted locally on a development environment like XAMPP.

Recovery Mechanisms:

In addition to managing concurrency, implementing robust recovery mechanisms is vital to ensure the reliability and availability of our project hosted on XAMPP. In the event of unexpected failures or crashes, Apache and MySQL offer various features to aid in system recovery. Apache's error handling mechanisms can gracefully manage system failures, ensuring minimal disruption to other users accessing the application. Moreover, MySQL provides essential features such as transaction logging and automatic crash recovery, which help restore the database to a consistent state after a failure. Additionally, regular backups of the application files and database are essential as a fail-safe measure. These backups allow for quick restoration of the system to a previous state, minimizing data loss and ensuring continuity of service. By incorporating these recovery mechanisms into our project, we can mitigate the impact of failures and ensure the reliability and integrity of our application, providing users with a seamless experience even in the face of unforeseen events.

CHAPTER 6
CODE FOR THE PROJECT

Css Code:

```
body {
background-color: #efefef;
}
.feather {
width: 20px;
height: 20px;
stroke: #4d4d4d;
stroke-width: 2;
stroke-linecap: round;
stroke-linejoin: round;
fill: none;
vertical-align: text-bottom;
}
.list-group {
background-color: #ffffff;
}
.list-group-item {
border: none;
}

.user {
text-align: center;
border-bottom: 1px solid #ddd;

}
.user img {
padding: 10px;
```

```
}
.toggler {
color: #000;
background-color: #fff;
border: none;
outline: none;
}
.sidebar-active {
color: #47a04b;
font-weight: 500;
}
.sidebar-active .feather {
stroke: #47a04b;
font-weight: 500;
}

#wrapper {
overflow-x: hidden;
background-color: #fff;
}

#sidebar-wrapper {
min-height: 100vh;
margin-left: -15rem;
-webkit-transition: margin 0.25s ease-out;
-moz-transition: margin 0.25s ease-out;
-o-transition: margin 0.25s ease-out;
transition: margin 0.25s ease-out;
}

#sidebar-wrapper .sidebar-heading {
padding: 0.875rem 1.25rem;
```

```
font-size: 14px;
font-weight: bold;
text-transform: uppercase;
color: #999;
}

#sidebar-wrapper .list-group {
width: 15rem;
}

#page-content-wrapper {
min-width: 100vw;
}

#wrapper.toggled #sidebar-wrapper {
margin-left: 0;
}

@media (min-width: 768px) {
#sidebar-wrapper {
margin-left: 0;
}

#page-content-wrapper {
min-width: 0;
width: 100%;
}

#wrapper.toggled #sidebar-wrapper {
margin-left: -15rem;
}
}
```

```
/* Add Rounded Border to Card */
.card {
margin-bottom: 10px;
}
.container-fluid {
margin-right: 20px;
}
/* Custom Gradients */
.card-gradient-1 {
border: none;
color: #ffffff;
background: #ed213a; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#93291e,
#ed213a
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#93291e,
#ed213a
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-2 {
border: none;
color: #ffffff;
background: #e44d26; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#e44d26,
```

```
#f16529
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#e44d26,
#f16529
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-3 {
border: none;
color: #ffffff;
background: #cc2b5e; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#753a88,
#cc2b5e
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#753a88,
#cc2b5e
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-4 {
border: none;
color: #ffffff;
background: #00b4db; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#0083b0,
```

```
#00b4db
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#0083b0,
#00b4db
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-5 {
border: none;
color: #ffffff;
background: #136a8a; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#267871,
#136a8a
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#267871,
#136a8a
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-6 {
border: none;
color: #ffffff;
background: #2b5876; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#4e4376,
```

```
#2b5876
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#4e4376,
#2b5876
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-7 {
border: none;
color: #ffffff;
background: #6a3093; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#a044ff,
#6a3093
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#a044ff,
#6a3093
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-8 {
border: none;
color: #ffffff;
background: #b24592; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#f15f79,
```

```
#b24592
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#f15f79,
#b24592
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-9 {
border: none;
color: #ffffff;
background: #c94b4b; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#4b134f,
#c94b4b
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#4b134f,
#c94b4b
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-10 {
border: none;
color: #ffffff;
background: #c33764; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#1d2671,
```

```
#c33764
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#1d2671,
#c33764
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}

.card-gradient-11 {
border: none;
color: #ffffff;
background: #eb3349; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#f45c43,
#eb3349
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#f45c43,
#eb3349
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}
.card-gradient-12 {
border: none;
color: #ffffff;
background: #283048; /* fallback for old browsers */
background: -webkit-linear-gradient(
to right,
#859398,
#283048
```

```
); /* Chrome 10-25, Safari 5.1-6 */
background: linear-gradient(
to right,
#859398,
#283048
); /* W3C, IE 10+/ Edge, Firefox 16+, Chrome 26+, Opera 12+, Safari 7+ */
}
```

Html and Php Code:

```
<?php
include("session.php");
$one_month_ago = date("Y-m-d", strtotime("-1 month"));
$exp_category_dc = mysqli_query($con, "SELECT expensecategory FROM expenses
WHERE
user_id = '$userid' AND expensedate >= '$one_month_ago' GROUP BY expensecategory");
$exp_amt_dc = mysqli_query($con, "SELECT SUM(expense) FROM expenses WHERE
 user_id = '$userid' AND expensedate >= '$one_month_ago' GROUP BY expensecategory");

$one_week_ago = date("Y-m-d", strtotime("-1 week"));
$exp_date_line = mysqli_query($con, "SELECT DATE_FORMAT(expensedate, '%b %d')
AS
 day_month FROM expenses WHERE user_id = '$userid' AND expensedate >=
'$one_week_ago'
GROUP BY expensedate");
$exp_amt_line = mysqli_query($con, "SELECT SUM(expense) FROM expenses WHERE
user_id = '$userid' AND expensedate >= '$one_week_ago' GROUP BY expensedate");

$yearly_expenses_query = "SELECT YEAR(expensedate) AS year, SUM(expense) AS
total_expense
FROM expenses
WHERE user_id = '$userid'
GROUP BY YEAR(expensedate)
ORDER BY YEAR(expensedate)";
$yearly_expenses_result = mysqli_query($con, $yearly_expenses_query);
```

```
$year_labels = [];
$yearly_expense_data = [];
while ($row = mysqli_fetch_assoc($yearly_expenses_result)) {
$year_labels[] = $row['year'];
$yearly_expense_data[] = $row['total_expense'];
}

$monthly_expenses_query = "SELECT DATE_FORMAT(expensedate, '%Y-%m') AS
month_year,
SUM(expense) AS total_expense
FROM expenses
WHERE user_id = '$userid'
AND expensedate >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY DATE_FORMAT(expensedate, '%Y-%m')
ORDER BY expensedate";
$monthly_expenses_result = mysqli_query($con, $monthly_expenses_query);
$monthly_labels = [];
$monthly_expense_data = [];
while ($row = mysqli_fetch_assoc($monthly_expenses_result)) {
$monthly_labels[] = $row['month_year'];
$monthly_expense_data[] = $row['total_expense'];
}

$today_expense = mysqli_query($con, "SELECT SUM(expense) FROM expenses WHERE
user_id = '$userid' AND expensedate = CURDATE()");
$yesterday_expense = mysqli_query($con, "SELECT SUM(expense) FROM expenses
WHERE
user_id = '$userid' AND expensedate = DATE_SUB(CURDATE(), INTERVAL 1 DAY)");
$this_week_expense = mysqli_query($con, "SELECT SUM(expense) FROM expenses
WHERE
 user_id = '$userid' AND expensedate >= DATE_SUB(CURDATE(), INTERVAL 1
WEEK)");
$this_month_expense = mysqli_query($con, "SELECT SUM(expense) FROM expenses
WHERE
```

```
user_id = '$userid' AND expensedate >= DATE_SUB(CURDATE(), INTERVAL 1
MONTH)");

$this_year_expense = mysqli_query($con, "SELECT SUM(expense) FROM expenses
WHERE

 user_id = '$userid' AND expensedate >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)");

$total_expense = mysqli_query($con, "SELECT SUM(expense) FROM expenses WHERE
user_id = '$userid'");


$today_expense_amount = '0' + mysqli_fetch_assoc($today_expense)['SUM(expense)'];

$yesterday_expense_amount ='0' +
mysqli_fetch_assoc($yesterday_expense)['SUM(expense)'];

$this_week_expense_amount = '0' +
mysqli_fetch_assoc($this_week_expense)['SUM(expense)'];

$this_month_expense_amount = '0' +
mysqli_fetch_assoc($this_month_expense)['SUM(expense)'];

$this_year_expense_amount = '0' +
mysqli_fetch_assoc($this_year_expense)['SUM(expense)'];

$total_expense_amount = '0' + mysqli_fetch_assoc($total_expense)['SUM(expense)'];


?>

<!DOCTYPE html>

<html lang="en">


<head>


<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">


<title>Expense Manager - Dashboard</title>


<!-- Bootstrap core CSS -->

<link href="css/bootstrap.css" rel="stylesheet">
```

```
<!-- Custom styles for this template -->
<link href="css/style.css" rel="stylesheet">

<!-- Feather JS for Icons -->
<script src="js/feather.min.js"></script>
<style>
.card a {
color: #000;
font-weight: 500;
}

.card a:hover {
color: #28a745;
text-decoration: dotted;
}
.try {
font-size: 28px; /* Adjust the font size as needed */
color: #333;    /* Adjust the color as needed */
padding: 5px 0px 0px 0px;   /* Adjust the padding as needed */
}
.container {
padding:0px 20px 20px 20px;/* Add padding to the container */
}
.card.text-center {
border: 3px solid #ccc;
padding: 10px;
margin: 10px;
background-color: #f8f9fa;
border-radius: 5px;
}
```

```
.card-title {
font-size: 17.5px;
margin-bottom: 1px ;
color: #333;
}

.card-text {
font-size: 24px;
font-weight: bold;
color: #6c757d;
}

</style>

</head>

<body>

<div class="d-flex" id="wrapper">

<!-- Sidebar -->
<div class="border-right" id="sidebar-wrapper">
<div class="user">
<img class="img img-fluid rounded-circle" src="uploads\default_profile.png" width="120">
<h5><?php echo $username ?></h5>
<p><?php echo $useremail ?></p>
</div>
<div class="sidebar-heading">Management</div>
<div class="list-group list-group-flush">
<a href="index.php" class="list-group-item list-group-item-action sidebar-active"><span
data-feather="home"></span> Dashboard</a>
<a href="add_expense.php" class="list-group-item list-group-item-action "><span data-
feather="plus-square"></span> Add Expenses</a>
```

```
<a href="manage_expense.php" class="list-group-item list-group-item-action "><span data-feather=

"dollar-sign"></span> Manage Expenses</a>

<a href="expensereport.php" class="list-group-item list-group-item-action"><span data-feather="file-text"></span> Expense Report</a>


</div>

<div class="sidebar-heading">Settings </div>

<div class="list-group list-group-flush">

<a href="profile.php" class="list-group-item list-group-item-action "><span data-feather="user"></span> Profile</a>

<a href="logout.php" class="list-group-item list-group-item-action "><span data-feather="power"></span> Logout</a>

</div>

</div>

<!-- /#sidebar-wrapper -->


<!-- Page Content -->

<div id="page-content-wrapper">


<nav class="navbar navbar-expand-lg navbar-light  border-bottom">


<button class="toggler" type="button" id="menu-toggle" aria-expanded="false">

<span data-feather="menu"></span>

</button>

<div class="col-md-0 text-center">

<h3 class="try">Dashboard</h3>

</div>

</nav>

<div class="container-fluid">

<h4 class="mt-4">Full-Expense Report</h4>

<div class="row">

<div class="container mt-4">
```

```
<div class="row">
<div class="col-md-3">
<div class="card text-center">
<div class="card-body">
<h5 class="card-title">Today's Expense</h5>
<p class="card-text">₹<?php echo $today_expense_amount; ?></p>
</div>
</div>
</div>
<div class="col-md-3">
<div class="card text-center">
<div class="card-body">
<h5 class="card-title">Yesterday's Expense</h5>
<p class="card-text">₹<?php echo $yesterday_expense_amount; ?></p>
</div>
</div>
</div>
<div class="col-md-3">
<div class="card text-center">
<div class="card-body">
<h5 class="card-title">Last 7Day's Expense</h5>
<p class="card-text">₹<?php echo $this_week_expense_amount; ?></p>
</div>
</div>
</div>
<div class="col-md-3">
<div class="card text-center">
<div class="card-body">
<h5 class="card-title">Last 30Day's Expense</h5>
<p class="card-text">₹<?php echo $this_month_expense_amount; ?></p>
</div>
</div>
```

```
</div>
<div class="col-md-3">
<div class="card text-center">
<div class="card-body">
<h5 class="card-title">Current Year Expense</h5>
<p class="card-text">₹<?php echo $this_year_expense_amount; ?></p>
</div>
</div>
</div>
<div class="col-md-3">
<div class="card text-center">
<div class="card-body">
<h5 class="card-title">Total Expense</h5>
<p class="card-text">₹<?php echo $total_expense_amount; ?></p>
</div>
</div>
</div>
</div>
</div>
<!-- Daily Expenses Chart -->
<div class="col-md-6">
<div class="card">
<div class="card-header">
<h5 class="card-title text-center">Daily Expenses</h5>
</div>
<div class="card-body">
<canvas id="expense_line" height="200"></canvas>
</div>
</div>
</div>
<!-- Expense Category Chart -->
<div class="col-md-6">
```

```
<div class="card">
<div class="card-header">
<h5 class="card-title text-center">Expense Category</h5>
</div>
<div class="card-body">
<canvas id="expense_category_pie" height="200"></canvas>
</div>
</div>
</div>
<!-- Monthly Expenses Chart -->
<div class="col-md-6">
<div class="card">
<div class="card-header">
<h5 class="card-title text-center">Monthly Expenses</h5>
</div>
<div class="card-body">
<canvas id="monthly_expense_line" height="200"></canvas>
</div>
</div>
</div>
<!-- Yearly Expenses Chart -->
<div class="col-md-6">
<div class="card">
<div class="card-header">
<h5 class="card-title text-center">Yearly Expenses</h5>
</div>
<div class="card-body">
<canvas id="expense_yearly_line" height="200"></canvas>
</div>
</div>
</div>
</div>
```

```
</div>
</div>
<!-- /#page-content-wrapper -->

</div>
<!-- /#wrapper -->

<!-- Bootstrap core JavaScript -->
<script src="js/jquery.slim.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/Chart.min.js"></script>
<!-- Menu Toggle Script -->
<script>
$("#menu-toggle").click(function(e) {
e.preventDefault();
$("#wrapper").toggleClass("toggled");
});
</script>
<script>
feather.replace()
</script>
<script>
var ctx = document.getElementById('expense_category_pie').getContext('2d');
var categories = [<?php while ($a = mysqli_fetch_array($exp_category_dc)) {
echo '"' . $a['expensecategory'] . '",';
} ?>];
var expenses = [<?php while ($b = mysqli_fetch_array($exp_amt_dc)) {
echo '"' . $b['SUM(expense)'] . '",';
} ?>];
var colors = [
'#6f42c1',
'#dc3545',
```

```
'#28a745',
'#007bff',
'#ffc107',
'#20c997',
'#17a2b8',
'#fd7e14',
'#e83e8c',
'#6610f2'
];
var dataset = {
labels: categories,
datasets: [{
label: 'Expense by Category (Last Month)',
data: expenses,
backgroundColor: colors,
borderWidth: 1
}]
};
var options = {
scales: {
x: {
beginAtZero: true,
ticks: {
autoSkip: false,
maxRotation: 45,
minRotation: 45
}
},
y: {
beginAtZero: true
}
}
```

```
};
var myChart = new Chart(ctx, {
type: 'bar',
data: dataset,
options: options
});

var yearlyColors = [
'#dc3545',  // Red
'#28a745',  // Green
'#007bff',  // Blue
'#ffc107',  // Yellow
'#20c997',  // Teal
'#17a2b8',  // Cyan
'#fd7e14',  // Orange
'#e83e8c',  // Pink
'#6610f2'
];

var yearlyLine = document.getElementById('expense_yearly_line').getContext('2d');
var yearlyChartData = {
labels: [<?php echo '"' . implode('","', $year_labels) . '"'; ?>],
datasets: [{
label: 'Yearly Expense',
data: [<?php echo implode(',', $yearly_expense_data); ?>],
borderColor: yearlyColors,
backgroundColor: yearlyColors,
fill: false,
borderWidth: 2
}]
};
```

```
var yearlyExpenseChart = new Chart(yearlyLine, {
type: 'bar',
data: yearlyChartData,
options: {
scales: {
x: {
ticks: {
autoSkip: false,
maxRotation: 45,
minRotation: 45
}
}
}
}
});

var monthlyLine = document.getElementById('monthly_expense_line').getContext('2d');
var monthlyChartData = {
labels: [<?php echo "" . implode("","", $monthly_labels) . ""; ?>],
datasets: [{
label: 'Monthly Expense (Last Year)',
data: [<?php echo implode(',', $monthly_expense_data); ?>],
borderColor: [
'#fd7e14'
],
backgroundColor: [
'#fd7e14'
],
fill: false,
borderWidth: 2
}]
};
```

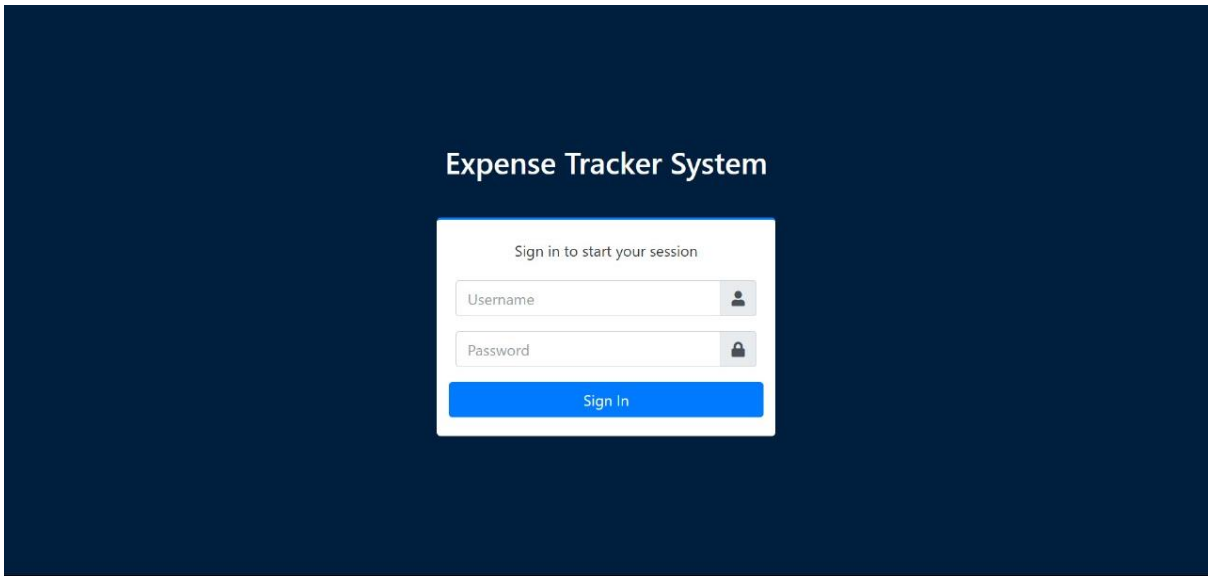
```
var monthlyExpenseChart = new Chart(monthlyLine, {
type: 'line',
data: monthlyChartData,
options: {
scales: {
x: {
ticks: {
autoSkip: false,
maxRotation: 45,
minRotation: 45
}
}
}
}
});
var line = document.getElementById('expense_line').getContext('2d');
var myChart = new Chart(line, {
type: 'line',
data: {
labels: [<?php while ($c = mysqli_fetch_array($exp_date_line)) {
echo '"' . $c['day_month'] . '",';
} ?>],
datasets: [{
label: 'Expense by Day (Last Week)',
data: [<?php while ($d = mysqli_fetch_array($exp_amt_line)) {
echo '"' . $d['SUM(expense)'] . '",';
} ?>],
borderColor: [
'#adb5bd'
],
backgroundColor: [
'#6f42c1',
```

```
'#dc3545',
'#28a745',
'#007bff',
'#ffc107',
'#20c997',
'#17a2b8',
'#fd7e14',
'#e83e8c',
'#6610f2'
],
fill: false,
borderWidth: 2
}]
}
});
</script>
</body>
</html>
```

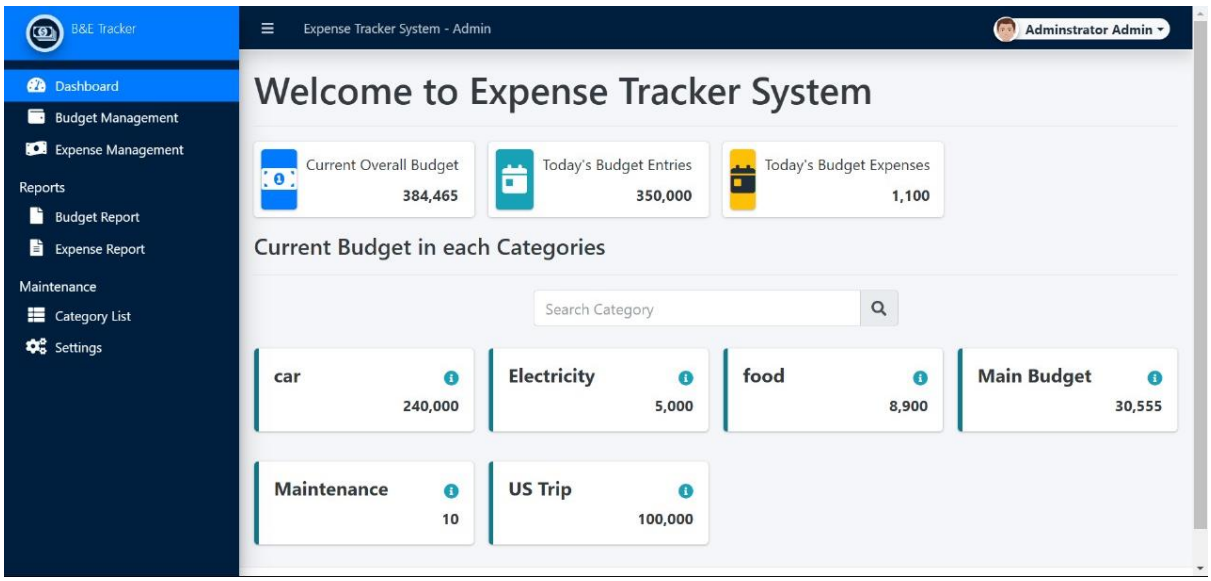
CHAPTER 7

RESULT AND DISCUSSION

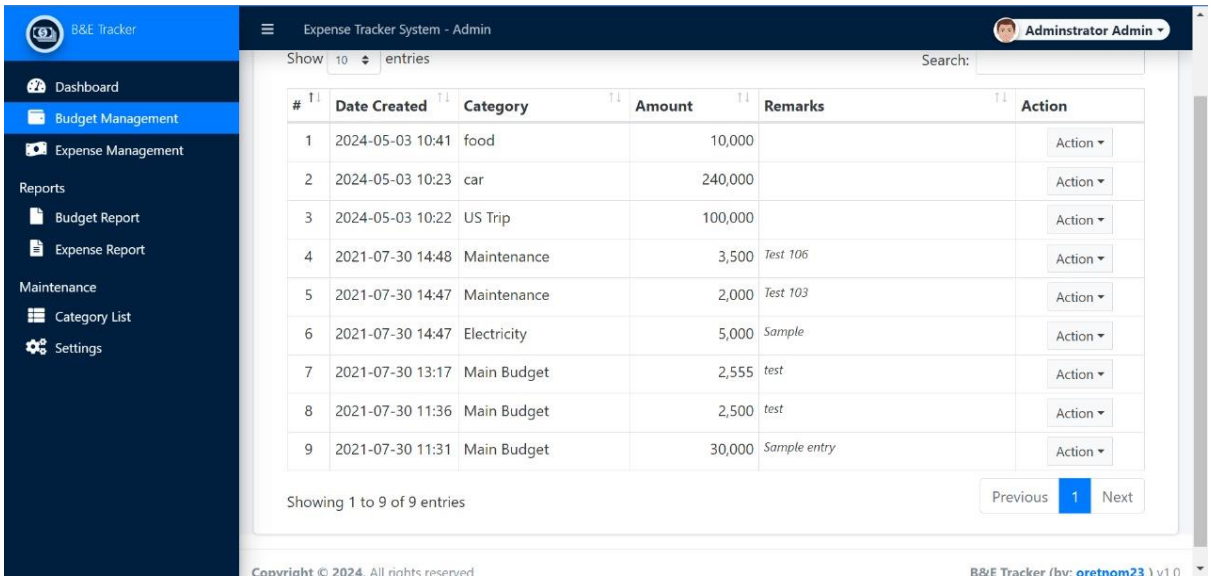
User Login



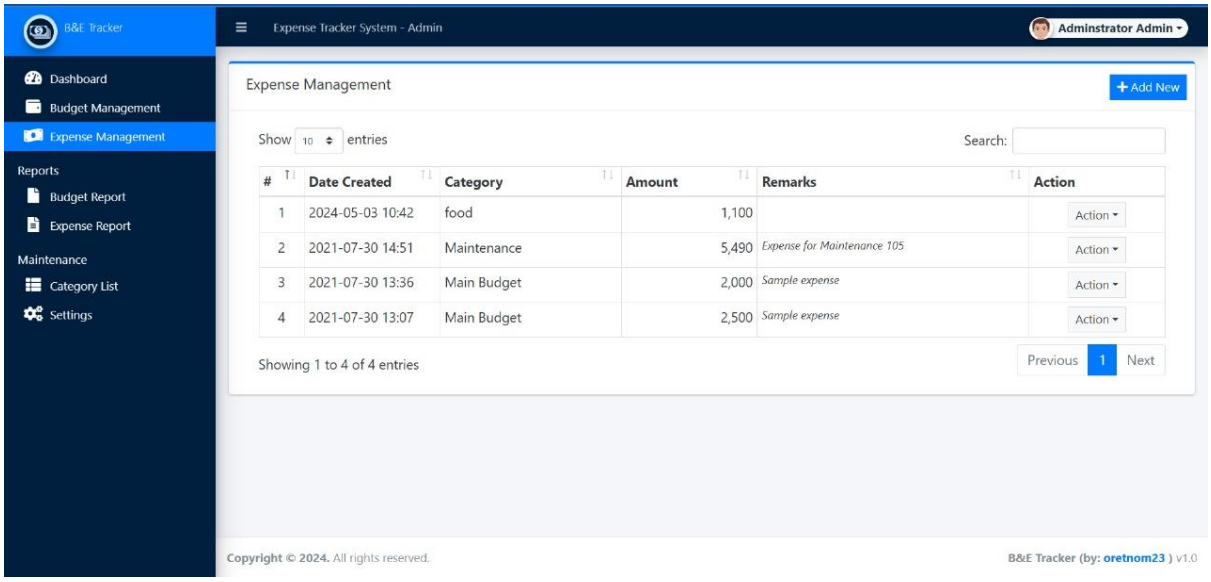
Dashboard



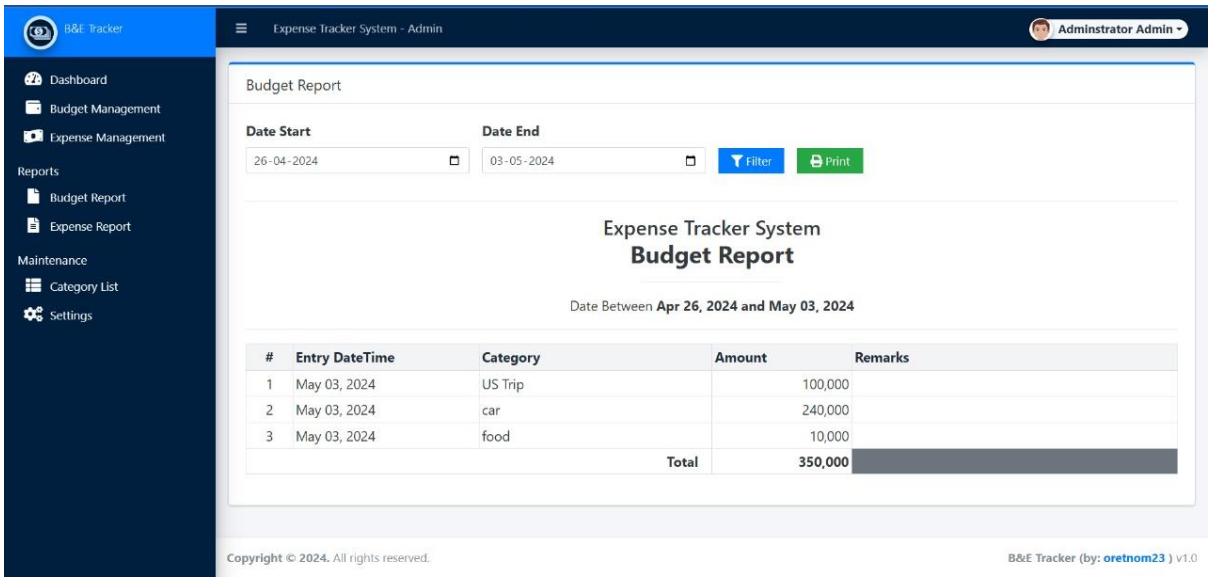
Budget Management



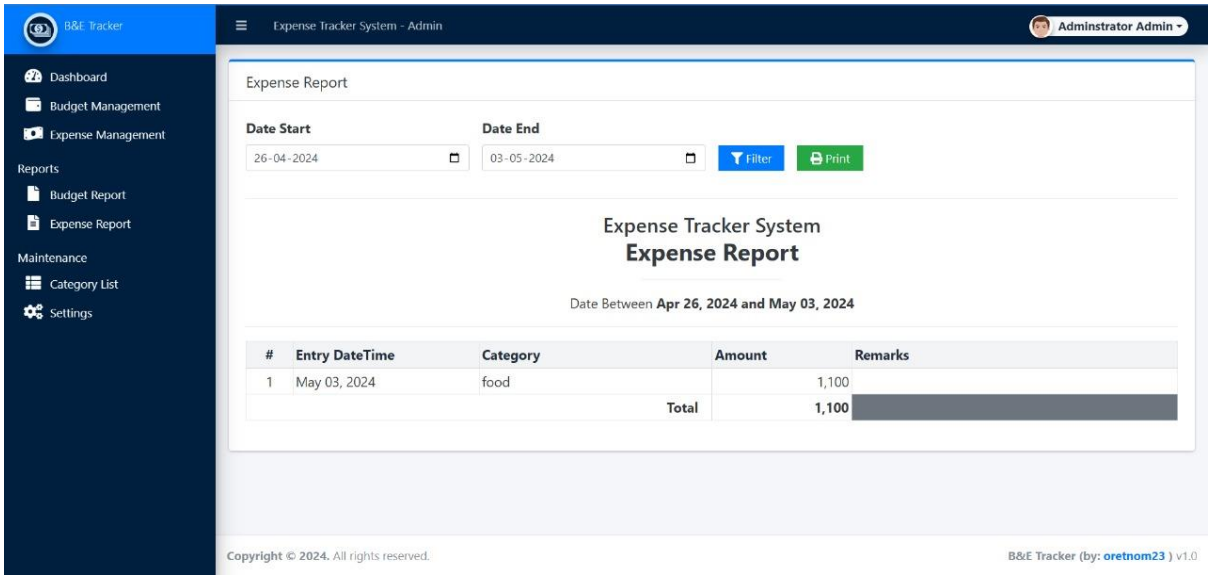
Expense Management



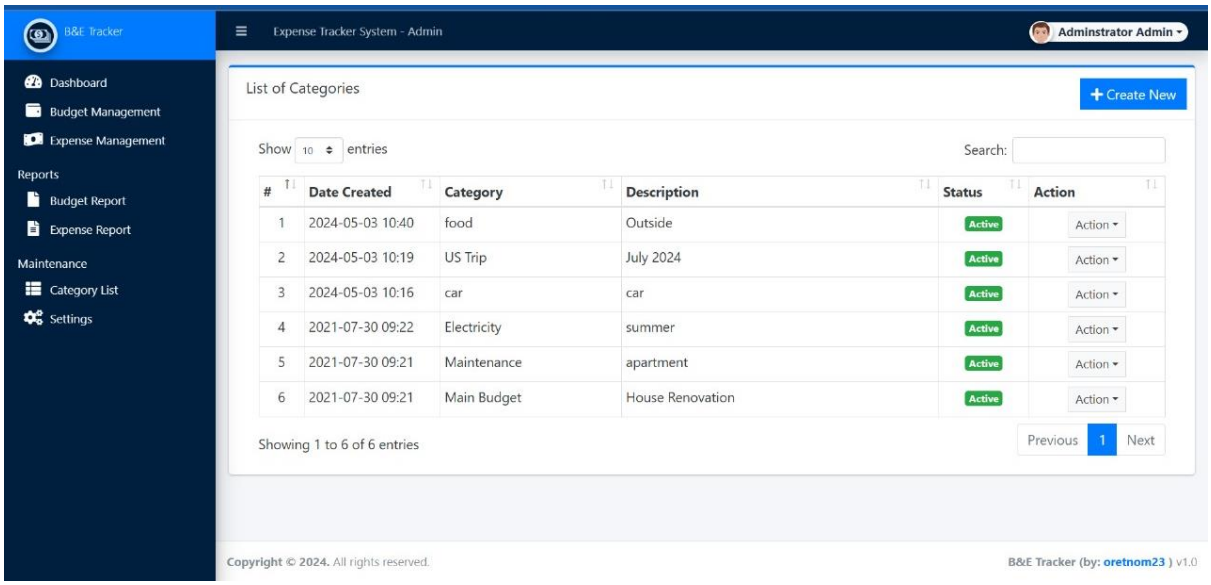
Budget Report



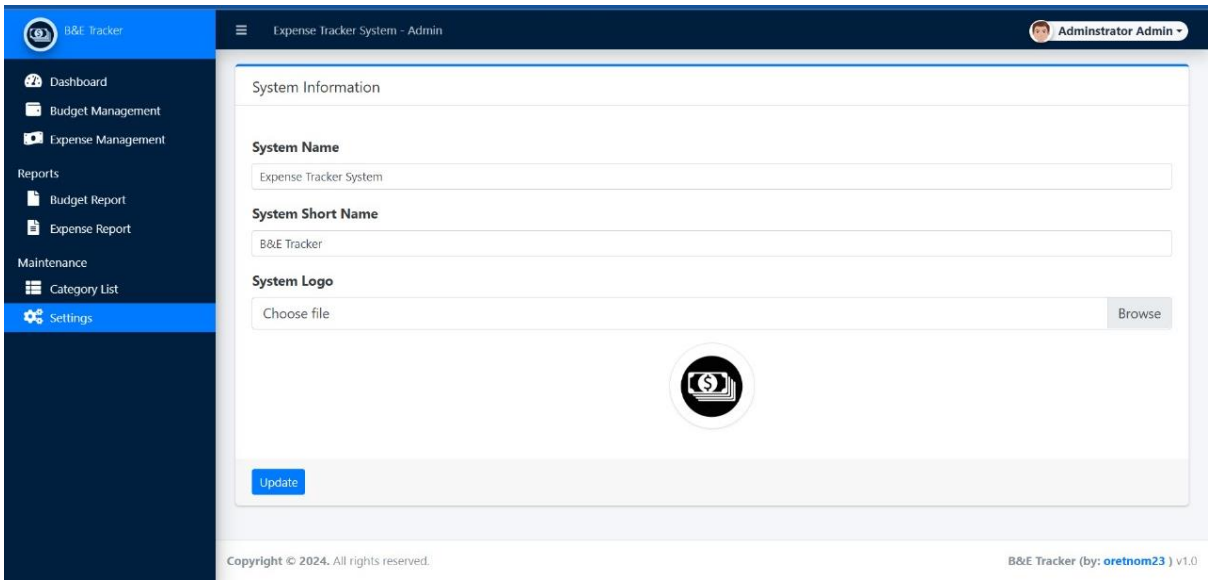
Expense Report



List of Categories



System Information



CHAPTER 8
CONCLUSION

In summary, the development and delivery of our financial research software represents a success in the use of technology and powerful data management systems to help users manage their finances through integration, cursors, normalization and concurrency." Control In our database architecture, PHP, which supports dynamic content creation and server-side operations, SQL and HTML / CSS, which supports the interaction of documents, are meaningful and visually adapt to the front-end interface. As we complete this phase of development, we recognize the nature of software development and are committed to continuous improvement based on user input and new technologies. Through constant change and innovation, our goal is to enhance our position as a trusted friend of personal finance and provide users with the tools they need to achieve their financial goals effectively and efficiently.

CHAPTER 9
FUTURE SCOPE

In today's dynamic business landscape, effective financial management is essential for the success and sustainability of organizations across diverse industries. The integration of digital technologies has revolutionized traditional financial practices, offering innovative solutions to streamline processes, enhance decision-making, and optimize resource allocation. Our finance tracking project aims to address these evolving needs by providing a comprehensive platform for managing financial data, analysing key performance metrics, and facilitating informed decision-making. Leveraging advanced technologies such as data analytics, automation, and integration, our project offers a transformative solution for optimizing financial management practices in various sectors. For instance, in the retail industry, our platform enables businesses to monitor sales data, manage inventory, and optimize pricing strategies in real-time. By automating transaction recording and providing insights into sales trends, retailers can improve operational efficiency, reduce costs, and maximize profitability in a competitive market environment.

CHAPTER 10

ONLINE COURSE CERTIFICATES

Hrishika Raj [RA2211032010004]

CERTIFICATE
OF EXCELLENCE
THIS CERTIFICATE IS AWARDED TO

SCALER Topics

HRISHIKA RAJ

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 08 April 2024

Anshuman Singh
Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER

Harsini J.P [RA2211032010007]

CERTIFICATE
OF EXCELLENCE
THIS CERTIFICATE IS AWARDED TO

SCALER Topics

HARSINI J P (RA2211032010007)

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 03 May 2024

Anshuman Singh
Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER

S. Reshma [RA2211032010008]

CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

S.RESHMA

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 03 May 2024

Anshuman Singh

Co-founder SCALER

CERTIFICATE OF EXCELLENCE BY SCALER