CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

CASE STUDY REPORT

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

Submitted by:

Name: Atul Singh

UID: 23BCA10701

Section: 23BCA 4-B

Submitted to:

Name: Arvinder Singh

Designation:

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Expense Tracking System

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ABSTRACT

- **Introduction:**
- **Technique:**
- **System Configuration:**
- **INPUT:**
- **ER DIAGRAM:**
- **TABLE REALTION:**
- **TABULAR FORMAT:**
- **TABLE CREATION:**
- **SQL QUERIES WITH OUTPUT (at least 10 to 15):**
- **SUMMARY:**
- **CONCLUSION:**

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ABSTRACT

Managing personal and professional finances has become a necessity in today's fast-paced and unpredictable world. The Expense Tracking System is a robust, user-centric, database-driven application that facilitates real-time recording, monitoring, and analysis of income and expenses. It helps individuals and small businesses maintain budgets, analyze trends, and make informed financial decisions. The system offers features like category-based sorting, budget setting, and report generation. Developed using MySQL and integrated with modern front-end technologies like HTML, CSS, and JavaScript, it showcases the practical implementation of DBMS concepts such as ER modeling, normalization, foreign key constraints, and SQL optimization. The backend is powered by PHP or Node.js, allowing for secure data operations and scalability. Through intuitive design and robust architecture, the Expense Tracker ensures a seamless experience, enabling users to stay on top of their financial health at all times.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Introduction

In today's fast-paced world, managing personal and organizational finances efficiently is crucial for maintaining financial stability. An **Expense Tracker System** developed using a **Database Management System (DBMS)** provides a structured and automated approach to recording, categorizing, and analyzing expenses. This system helps individuals and businesses track their spending patterns, generate insightful reports, and make informed financial decisions.

Traditional expense tracking methods, such as manual record-keeping or spreadsheet-based systems, are often time-consuming, error-prone, and lack real-time analysis capabilities. A **DBMS-powered Expense Tracker** overcomes these limitations by offering a centralized, secure, and scalable solution. With features like user authentication, expense categorization, budget tracking, and automated report generation, this system ensures accuracy, accessibility, and efficiency in financial management.

This project aims to design and implement a robust **Expense Tracker System** using relational database principles, ensuring data integrity, security, and ease of use. By leveraging DBMS functionalities such as **SQL queries, transactions, and user access control**, the system provides a reliable platform for monitoring expenditures. Whether for personal budgeting or business accounting, this application demonstrates the practical benefits of DBMS in developing real-world financial solutions.

TECHNIQUE

1. Database Management System:

DBMS Used: MySQL 8.0+

Features: ACID compliance, indexing, stored procedures, views, triggers, foreign key constraints

Normalization Level: Up to 3NF to eliminate data redundancy and anomalies.

Backup and Recovery: Regular backups via mysqldump, replication for disaster recovery

Data Security: Access controls through GRANT/REVOKE, hashed passwords, encryption at rest and in transit

Performance Optimization: Query optimization using EXPLAIN, indexing of frequently accessed columns, partitioning of large tables

Concurrency Control: MySQL's InnoDB engine for transactional consistency and row-level locking

Data Integrity: Enforced via constraints (NOT NULL, UNIQUE, CHECK, FK)

2. Frontend Development:

Technologies: HTML5, CSS3, JavaScript ES6+, Bootstrap

Interactive Components: Dynamic modals, category dropdowns, date pickers

Optional UI Frameworks: React.js or Vue.js for modular components

3. Backend Development:

Languages: PHP (Laravel), Node.js (Express)

API Communication: RESTful APIs with CRUD operations

Security Protocols: JWT authentication, rate limiting, SQL injection protection

4. Tools and Platforms Used:

Tool	Use
MySQL Workbench	Schema visualization, query testing
Visual Studio Code	Code editing with extensions for formatting and linting
Postman	REST API testing
GitHub	Source control and project documentation
Draw.io	ER Diagram and system architecture
Chart.js	Frontend data visualization for reports

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SYSTEM CONFIGURATION

Hardware Requirements:

Server Side:

Processor: Intel i5 or AMD Ryzen 5 and above

RAM: 8 GB minimum

Storage: 256 GB SSD or above

Network: Stable 20+ Mbps broadband

Client Side:

Minimum: Dual-core CPU, 2GB RAM, Modern web browser (Chrome, Firefox)

Recommended: Quad-core CPU, 4GB RAM

Software Requirements:

Component	Details
OS	Windows 10/11, Ubuntu 20.04+, macOS Ventura
Web Server	Apache (for PHP), NGINX (for Node.js)
DBMS	MySQL 8.0+
IDE	VS Code / PhpStorm
Browser	Chrome v90+, Firefox v85+

INPUT MODULES

An Expense Tracking System relies on several well-defined input modules to manage user interaction and data entry effectively. These modules are designed to ensure a smooth, secure, and organized flow of financial information into the system. Each input module has a specific function, complete with validation rules and integrity checks to ensure accurate and meaningful data.

1. User Registration and Authentication

This module allows new users to create accounts and existing users to log in securely. During registration, the user must provide details such as full name, email address, and a password. The system checks for email uniqueness to prevent duplicate accounts and enforces a strong password policy for security. Depending on the system design, users can also be assigned roles such as 'User' or 'Admin'. Upon successful registration, users receive secure credentials to log in. Authentication mechanisms include session tokens or JSON Web Tokens (JWT), which help manage secure user sessions. This module forms the foundation of user-specific data segregation across other modules.

2. Expense Management Module

Users log their daily or occasional expenses using this module. Inputs include the amount spent, the expense category (e.g., food, transport, health), the date of transaction, payment method (cash, card, UPI), and an optional description. Validation ensures that the amount is non-negative and that all required fields are filled. This module is critical for capturing real-time expenses and linking them to the respective users. Expenses are also timestamped for accurate historical tracking, which feeds into reports and analytics.

3. Income Management Module

This module captures all income-related entries from users. Each entry requires a source (salary, freelance, interest, etc.), the amount received, and the date. The income values are validated to ensure they are positive, and like expenses, are associated with the corresponding user through a foreign key relationship. This module enables accurate financial comparisons and helps generate monthly savings reports by calculating net balances.

4. Budget Planning Module

Budgeting helps users allocate funds wisely across various categories. This module allows users to set budget limits on categories such as food, rent, or travel, for a specified duration (monthly or yearly). Input validation ensures that the budget amount is numeric and the category exists in the system. The system can trigger alerts when users exceed a predefined threshold, such as 80% or 100% of their budget, helping them maintain financial discipline.

5. Reporting and Analytics Module

Users can generate reports based on filters like date range, transaction type (income or expense), and category. Input parameters help customize the report format—daily,

monthly, or yearly. Reports are presented in both tabular and visual formats such as pie charts and bar graphs, making them easier to interpret. Data can also be exported to formats like PDF, CSV, or Excel.

6. Admin Dashboard (Optional)

If the system includes administrative controls, the admin dashboard accepts inputs related to user management, including creation, update, suspension, and deletion of accounts. It also handles inputs for audit logs, where admin actions are recorded. This module helps maintain overall system integrity and enforce rules for financial compliance.

These modules, when combined, provide a comprehensive, user-friendly interface that supports efficient financial tracking, data integrity, and insightful analytics.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ER DIAGRAM AND RELATIONSHIPS

Core Entities:

User: Stores login info, role, and authentication status

Expense: Links to user, contains category, amount, method, and date

Income: Linked to user, records income source, date, and amount

Budget: Planned limits per category per period

Audit_Log: Optional, for security and compliance tracking

Relationships:

One user → Many incomes, expenses, and budgets

Expense → User (M:1)

Income → User (M:1)

Budget → User (M:1)

Diagram Description:

Users table is central

All financial records reference user_id

Budget and audit logs include time-based triggers for alerts

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

TABLE RELATIONSHIP STRUCTURE

A well-structured relational database lies at the heart of the Expense Tracking System. It defines how different entities relate to each other through primary and foreign keys. The aim is to ensure data integrity, reduce redundancy, and make querying the system efficient.

Key Relationships

Users → Expenses: One-to-Many (1:M)

Each user can have multiple expense records.

Expenses.user_id is a foreign key referencing Users.user_id.

Users → Incomes: One-to-Many (1:M)

Each user can log multiple sources of income.

Incomes.user_id is a foreign key referencing Users.user_id.

Users → Budgets: One-to-Many (1:M)

Users can set multiple budgets across different categories and periods.

Budgets.user_id is a foreign key referencing Users.user_id.

Users → Audit Logs: One-to-Many (1:M)

Every system action by a user is stored in audit logs for accountability.

Audit_Log.user_id maintains this reference.

Budget → Expense (via Category): One-to-Many (Logical Association)

Although not directly enforced by foreign key, the system matches Budgets.category and Expenses.category to evaluate budget status.

Summary Table of Relationships

Parent Table	Child Table	Foreign Key Used	Relationship Type
Users	Expenses	Expenses.user_id	One-to-Many
Users	Incomes	Incomes.user_id	One-to-Many
Users	Budgets	Budgets.user_id	One-to-Many
Users	Audit_Log	Audit_Log.user_id	One-to-Many
Budgets	Expenses (via category)	category (logical match)	One-to-Many (soft link)

Design Considerations

All child tables (Expenses, Incomes, Budgets, Audit_Log) must ensure referential integrity with Users.

Cascading updates are not enabled to avoid accidental data loss. Manual updates ensure control.

Expense and Income tables share a similar structure, making financial aggregation queries easier.

The logical association between Budget and Expense allows budget evaluation per category without overcomplicating foreign key structures.

Use in Queries

These relationships allow us to:

Join user financial data for reporting.

Filter transactions based on user-specific categories.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

TABLE STRUCTURE

✧ **Users Table**

Purpose: Stores user account details.

Key Fields:

- user_id (Primary Key)
- name, email, password
- role (User/Admin)

✧ **Expenses Table**

Purpose: Records individual spending by users.

Key Fields:

- expense_id (Primary Key)
- user_id (Foreign Key to Users)
- amount, category, method, description, date

✧ **Incomes Table**

Purpose: Stores all income entries.

Key Fields:

- income_id (Primary Key)
- user_id (Foreign Key to Users)
- amount, source, date

✧ Budgets Table

Purpose: Holds budget limits per user per category.

Key Fields:

- budget_id (Primary Key)
- user_id (Foreign Key to Users)
- category, limit_amount, period (Monthly/Yearly)

✧ Audit_Log Table

Purpose: Logs user actions (especially for Admins).

Key Fields:

- log_id (Primary Key)
- user_id (Foreign Key to Users)
- action, timestamp

Relationships Summary

- One User can have **many** Expenses, Incomes, Budgets, and Audit_Log entries.
- Budgets and Expenses are **logically linked** via category (but not a direct foreign key).

```
CREATE TABLE Users (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100),
  email VARCHAR(100) UNIQUE,
  password VARCHAR(255),
  role ENUM('User', 'Admin') DEFAULT 'User'
);
```

```sql
CREATE TABLE Expenses (
  expense_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT,
  amount DECIMAL(10,2),
  category VARCHAR(50),
  method VARCHAR(50),
  description TEXT,
  date DATE,
  FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

```sql
CREATE TABLE Incomes (
  income_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT,
  amount DECIMAL(10,2),
  source VARCHAR(100),
  date DATE,
  FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE Budgets (
  budget_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT,
  category VARCHAR(50),
  limit_amount DECIMAL(10,2),
  period ENUM('Monthly', 'Yearly'),
  FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

```
CREATE TABLE Audit_Log (
  log_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT,
  action VARCHAR(100),
  timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SAMPLE DATA ENTRIES

```
INSERT INTO Users (name, email, password) VALUES
('Ethan Walker', 'ethan@example.com', 'hashed123'),
('Olivia Brooks', 'olivia@example.com', 'hashed456'),
('Liam Carter', 'liam@example.com', 'hashed789'),
('Emily Stone', 'emily@example.com', 'hashed101'),
('Noah Bennett', 'noah@example.com', 'hashed202');

INSERT INTO Expenses (user_id, amount, category, method, description, date) VALUES
(1, 2500, 'Rent', 'Bank Transfer', 'March Rent', '2025-03-01'),
(1, 500, 'Food', 'UPI', 'Groceries and snacks', '2025-03-03'),
(1, 800, 'Transport', 'Cash', 'Cab and metro', '2025-03-05'),
(2, 300, 'Health', 'Card', 'Medicine purchase', '2025-03-04'),
(3, 1200, 'Education', 'Net Banking', 'Online course fee', '2025-03-07'),
(2, 1000, 'Entertainment', 'UPI', 'Movie and dinner', '2025-03-10'),
(4, 450, 'Groceries', 'Card', 'Weekly groceries', '2025-03-09'),
(5, 700, 'Utilities', 'UPI', 'Electricity bill', '2025-03-11'),
(5, 150, 'Snacks', 'Cash', 'Evening snacks', '2025-03-12');

INSERT INTO Incomes (user_id, amount, source, date) VALUES
(1, 25000, 'Monthly Salary', '2025-03-01'),
(1, 5000, 'Freelancing', '2025-03-15'),
(2, 18000, 'Part-Time Job', '2025-03-01'),
(3, 22000, 'Internship', '2025-03-05'),
(4, 30000, 'Full-Time Job', '2025-03-01'),
(5, 4000, 'Tutoring', '2025-03-08');
```

```
INSERT INTO Budgets (user_id, category, limit_amount, period) VALUES
(1, 'Food', 4000, 'Monthly'),
(1, 'Transport', 2000, 'Monthly'),
(2, 'Health', 1500, 'Monthly'),
(3, 'Education', 5000, 'Monthly'),
(4, 'Groceries', 3500, 'Monthly'),
(5, 'Utilities', 2000, 'Monthly');

INSERT INTO Audit_Log (user_id, action) VALUES
(1, 'Logged in'),
(2, 'Added new expense'),
(3, 'Updated income source'),
(1, 'Generated monthly report'),
(4, 'Created budget for groceries'),
(5, 'Viewed income summary');
```

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SQL Queries with Outputs

1. Show total expenses by each user

```sql
-- 1. Show total expenses by each user
SELECT u.name, SUM(e.amount) AS total_spent
FROM Expenses e
JOIN Users u ON e.user_id = u.user_id
GROUP BY u.name;
```

Output:

name	total_spent
Emily Stone	450
Ethan Walker	3800
Liam Carter	1200
Noah Bennett	850
Olivia Brooks	1300

2. List all users with their total income

```sql
-- 2. List all users with their total income
SELECT u.name, SUM(i.amount) AS total_income
FROM Incomes i
JOIN Users u ON i.user_id = u.user_id
GROUP BY u.name;
```

Output:

name	total_income
Emily Stone	30000
Ethan Walker	30000
Liam Carter	22000
Noah Bennett	4000
Olivia Brooks	18000

3. Get all expenses in March 2025

```sql
-- 3. Get all expenses in March 2025
SELECT * FROM Expenses
WHERE date LIKE '2025-03%';
```

Output:

expense_id	user_id	amount	category	method	description	date
1	1	2500	Rent	Bank Transfer	March Rent	2025-03-01
2	1	500	Food	UPI	Groceries and snacks	2025-03-03
3	1	800	Transport	Cash	Cab and metro	2025-03-05
4	2	300	Health	Card	Medicine purchase	2025-03-04
5	3	1200	Education	Net Banking	Online course fee	2025-03-07
6	2	1000	Entertainment	UPI	Movie and dinner	2025-03-10
7	4	450	Groceries	Card	Weekly groceries	2025-03-09

4. Find users who spent more than 1000 in total

```sql
-- 4. Find users who spent more than 1000 in total
SELECT u.name, SUM(e.amount) AS total_spent
FROM Expenses e
JOIN Users u ON e.user_id = u.user_id
GROUP BY u.name
HAVING total_spent > 1000;
```

Output:

name	total_spent
Ethan Walker	3800
Liam Carter	1200
Olivia Brooks	1300

5. Compare each user's income and expense totals

```sql
-- 5. Compare each user's income and expense totals
SELECT u.name,
  (SELECT SUM(i.amount) FROM Incomes i WHERE i.user_id = u.user_id) AS total_income,
  (SELECT SUM(e.amount) FROM Expenses e WHERE e.user_id = u.user_id) AS total_expense,
  (SELECT SUM(i.amount) FROM Incomes i WHERE i.user_id = u.user_id) -
  (SELECT SUM(e.amount) FROM Expenses e WHERE e.user_id = u.user_id) AS net_savings
FROM Users u;
```

Output:

name	total_income	total_expense	net_savings
Ethan Walker	30000	3800	26200
Olivia Brooks	18000	1300	16700
Liam Carter	22000	1200	20800
Emily Stone	30000	450	29550
Noah Bennett	4000	850	3150

6 Show users who spent more than their budget in any category

```sql
-- 6. Show users who spent more than their budget in any category
SELECT u.name, b.category, b.limit_amount, SUM(e.amount) AS spent
FROM Budgets b
JOIN Expenses e ON b.user_id = e.user_id AND b.category = e.category
JOIN Users u ON b.user_id = u.user_id
GROUP BY u.name, b.category, b.limit_amount
HAVING spent > b.limit_amount;
```

Output:

name	action	timestamp
Noah Bennett	Viewed income summary	2025-03-06 16:30:00
Emily Stone	Created budget for groceries	2025-03-05 08:45:00
Ethan Walker	Generated monthly report	2025-03-04 18:20:00
Liam Carter	Updated income source	2025-03-03 09:00:00
Olivia Brooks	Added new expense	2025-03-02 12:15:00
Ethan Walker	Logged in	2025-03-01 10:00:00

7. List all audit logs ordered by most recent

```sql
-- 7. List all audit logs ordered by most recent
SELECT u.name, a.action, a.timestamp
FROM Audit_Log a
JOIN Users u ON a.user_id = u.user_id
ORDER BY a.timestamp DESC;
```

Output:

category	total
Food	500
Rent	2500
Transport	800

8. Get category-wise expense total for user 'Ethan Walker'

```sql
-- 8. Get category-wise expense total for user 'Ethan Walker'
SELECT category, SUM(amount) AS total
FROM Expenses
WHERE user_id = 1
GROUP BY category;
```

Output:

9. Find top 3 highest single expenses

```sql
-- 9. Find top 3 highest single expenses
SELECT u.name, e.category, e.amount, e.date
FROM Expenses e
JOIN Users u ON e.user_id = u.user_id
ORDER BY e.amount DESC
LIMIT 3;
```

Output:

name	category	amount	date
Ethan Walker	Rent	2500	2025-03-01
Liam Carter	Education	1200	2025-03-07
Olivia Brooks	Entertainment	1000	2025-03-10

10. Show all users with no recorded expenses

```sql
-- 10. Show all users with no recorded expenses
SELECT name FROM Users
WHERE user_id NOT IN (SELECT DISTINCT user_id FROM Expenses);
```

Output:

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Summary

The **Expense Tracking System** is a comprehensive application designed to help users efficiently manage their finances by tracking income, expenses, and budgets. This system uses a simple database structure to store user data, categorize expenses, and set budget limits, all while maintaining user-friendly access to crucial financial information. It leverages essential database management system (DBMS) principles such as normalization and foreign key relationships, ensuring data integrity and optimal performance.

With functionalities for tracking expenses across different categories, monitoring income, and generating audit logs, the system provides valuable insights for both personal and small business financial management. It is built using MySQL, making it easily adaptable for various use cases. The system also allows users to set up budgets for different categories, ensuring they stay within their financial limits.

Through careful application of SQL queries and efficient database design, the **Expense Tracking System** enables users to monitor their financial health and make informed decisions, helping them achieve their financial goals.

Conclusion

The **Expense Tracking System** effectively demonstrates the practical application of database management system concepts to solve everyday financial management challenges. This system allows users to track their income, expenses, and budgets while maintaining data integrity and security. By using simple table structures and relationships, it ensures efficient data management while being easy to implement and use. Through various functionalities such as expense categorization, income tracking, and budget monitoring, the system provides users with comprehensive financial insight.

The project successfully applies key DBMS principles such as normalization, foreign key relationships, and SQL queries to create a user-friendly tool for individuals and businesses alike. With scalability in mind, this system can be expanded to include more detailed financial features, such as report generation and analytics. Overall, the **Expense Tracking System** provides an invaluable tool for improving financial discipline and aiding users in their financial decision-making.