INDEX:

Introduction :

Managing expenses effectively is essential for both individuals and organisations to maintain financial health. An **Expense Management System** (**EMS**) is a tool designed to help users track, categorise, and analyse their financial transactions in an efficient manner. This project aims to develop a web-based EMS that simplifies the process of managing expenses and income, offering users a structured way to track their financial activities.

The goal of this system is to provide a user-friendly interface for users to record their daily expenses, categorise them, and track their financial goals. This solution is aimed at individuals and small businesses who need an easy-to-use platform to manage their finances, monitor spending patterns, and generate reports.

Purpose of the Project :

The primary objective of this project is to create an **Expense Management System** using **Spring Boot** for the backend and a **React framework** for the frontend. The system will interact with a **MySQL** or **MongoDB** database for efficient data storage and retrieval. The key functionalities will include:

- **Expense and Income Tracking**: Users can record and categorise their expenses and income.
- **Financial Goal Management**: Users can set financial goals and track progress towards them.
- **Report Generation**: Users can generate reports to visualise spending habits and financial status.
- **Dashboard**: A summary dashboard providing insights into financial data.

Problem Statement :

Tracking expenses manually or using traditional methods like spreadsheets often leads to inaccuracies and inefficiency. Many users lack an effective tool for managing their financial data in real-time, leading to difficulties in understanding their financial situation. An automated, easy-to-use system is required to streamline expense tracking and reporting.

Objectives of the Project :

1. **Simplify Financial Tracking**: Provide a platform to easily input and track daily expenses and income.
2. **Generate Financial Reports**: Automatically generate reports and summaries to help users analyse their financial activities.
3. **Enhance Financial Discipline**: Encourage users to track their expenses and stick to a budget by setting spending limits.
4. **Provide Financial Insights**: Offer users insights into their spending patterns, helping them make informed financial decisions.

Technology Stack :

- **Backend**: **Spring Boot** will be used for the backend, providing a reliable and scalable framework.
- **Frontend**: **React** will be used to build a dynamic and responsive user interface.
- **Database**: **MySQL** or **MongoDB** will be used for efficient data storage.
- **Security**: User authentication and authorization will be implemented to ensure data security.

Significance of the Project :

This **Expense Management System** will offer users an efficient way to track and manage their finances. By automating the process and generating real-time reports, users will gain better control over their financial activities, leading to informed decision-making and improved financial discipline.

Functional Requirements :

1. User Registration and Authentication :

- Users should be able to register with a username, email, and password.
- Provide login/logout features for users to access the system securely.

2. Expense tracking :

- Users can add, edit, or delete expense records.
- Each expense should include details like date, category (e.g., food, travel), amount, and a brief description.

3. Income Tracking :

- Users can input their income sources, with details such as date, source, and amount.
- Allow users to edit and delete income entries.

4. Category Management :

- Users can create and manage custom categories for expenses and income.
- Categories should be editable and deletable.

5. Report Generation :

- Provide monthly and yearly expense reports.
- Display insights, like total expenditure, income, and a summary by category.

6. Budgeting Feature:

- Users can set a monthly budget, and the system will track expenses against it.
- Notify users if they are close to exceeding their budget.

7. Search and Filter:

- Enable users to search and filter expenses by date, category, and amount.

8. Data Export:

- Allow users to export their data as a CSV or PDF for external use.

Non-Functional Requirements :

1. Usability:

- The system should have a user-friendly interface, easy navigation, and clear labels.
- Support accessibility features for a wider range of users.

2. Performance:

- The application should respond quickly, with a target load time under 2 seconds.
- Handle concurrent user requests without significant lag.

3. Security:

- Implement strong password encryption and secure authentication mechanisms.
- Use HTTPS to secure data in transit.

4. Scalability:

- The system should handle an increasing number of users and large volumes of data without performance issues.

5. Reliability:

- The system should have high uptime (99.9%) and be capable of handling server errors gracefully.
- Regular backups to prevent data loss.

6. Compatibility:

- The application should be accessible on mobile and desktop devices.
- Ensure cross-browser compatibility for web access.

7. Maintainability:

- The codebase should be organised and modular to facilitate future updates.
- Provide clear documentation for developers.

8. Data Integrity:

- Ensure accurate tracking and storage of expense and income data.
- Validate user inputs to prevent data entry errors.

9. Compliance:

- Adhere to data privacy standards like GDPR (if required).
- Clearly communicate data usage policies to users.

User Stories :

1. User Registration and Authentication :

- As a user, I want to create an account so that I can securely access my financial information.
- As a user, I want to log in to my account so that I can view and manage my income, expenses, and savings.
- As a user, I want to log out from my account so that my financial data remains private.

2. Income Management :

- As a user, I want to add my monthly income so that I can set a baseline for tracking my expenses and savings.
- As a user, I want to edit my income if it changes so that my budget remains accurate.
- As a user, I want to view my income history so that I can track changes over time.

3. Expense Tracking :

- As a user, I want to add expenses with categories, amounts, and descriptions so that I can organise my spending.
- As a user, I want to view a summary of my expenses by category so that I can see where I spend the most.
- As a user, I want to edit or delete expenses so that I can correct any mistakes or remove unnecessary entries.

4. Savings Goals and Reminders :

- As a user, I want to set specific savings goals (e.g., medical fund) so that I have a certain amount set aside for future needs.
- As a user, I want to receive reminders for saving specific amounts each month so that I can stay on track with my savings goals.
- As a user, I want to track my progress toward savings goals so that I can see how close I am to achieving them.

5. Spending Limit Alerts :

- As a user, I want to set a spending limit based on my income so that I can avoid overspending.
- As a user, I want to receive alerts when I approach or exceed my spending limit so that I can adjust my expenses accordingly.

6. Reports and Analytics :

- As a user, I want to see weekly and monthly summaries of my income and expenses so that I can get a snapshot of my financial situation.
- As a user, I want to view spending trends over time so that I can identify areas to cut back on.

7. Data Backup and Restore :

- As a user, I want to back up my data periodically so that I don't lose my financial records.

Database Design :

1. Schema :
    a. expense_management: A schema to manage expense-related data for individual users.

```
1   CREATE SCHEMA `expense_management` ;
2
```

2. Tables and Columns :
    a. Users: Stores user information.
        i. User_ID: Unique identifier for each user.
        ii. Name: User's full name.
        iii. Email: User's email address, must be unique.
        iv. Password: User's password (hashed for security).
        v. Created_at: Timestamp of when the user was added.

```
1   CREATE TABLE `expense_management`.`users` (
2     `User_ID` INT NOT NULL AUTO_INCREMENT,
3     `Name` VARCHAR(100) NOT NULL,
4     `Email` VARCHAR(100) NOT NULL,
5     `Password` VARCHAR(255) NOT NULL,
6     `Created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
7     PRIMARY KEY (`User_ID`),
8     UNIQUE INDEX `Email_UNIQUE` (`Email` ASC) VISIBLE);
9
```

b. Categories: Contains categories for expense classification.

i. Category_ID: Unique identifier for each category.

ii. Name: Name of the category (e.g., "Food", "Transport").

iii. Description: Optional description of the category.

```
CREATE TABLE `expense_management`.`categories` (
  `Category_ID` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(50) NOT NULL,
  `Description` VARCHAR(255) NULL,
  PRIMARY KEY (`Category_ID`));

```

c. Expenses: Holds details of individual expenses.

i. Expense_ID: Unique identifier for each expense.

ii. User_ID: Foreign key linking to the user who made the expense.

iii. Category_ID: Foreign key linking to the category of the expense.

iv. Amount: Amount spent.

v. Description: Optional description of the expense.

```
CREATE TABLE `expense_management`.`expenses` (
  `Expense_ID` INT NOT NULL AUTO_INCREMENT,
  `User_ID` INT NOT NULL,
  `Category_ID` INT NULL,
  `Amount` DECIMAL(10,2) NOT NULL,
  `Description` VARCHAR(255) NULL,
  PRIMARY KEY (`Expense_ID`),
  INDEX `User_ID_idx` (`User_ID` ASC) VISIBLE,
  INDEX `Category_ID_idx` (`Category_ID` ASC) VISIBLE,
  CONSTRAINT `User_ID`
    FOREIGN KEY (`User_ID`)
    REFERENCES `expense_management`.`users` (`User_ID`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `Category_ID`
    FOREIGN KEY (`Category_ID`)
    REFERENCES `expense_management`.`categories` (`Category_ID`)
    ON DELETE NO ACTION
    ON UPDATE SET NULL);

```

d. Income : Tracks income details for users.

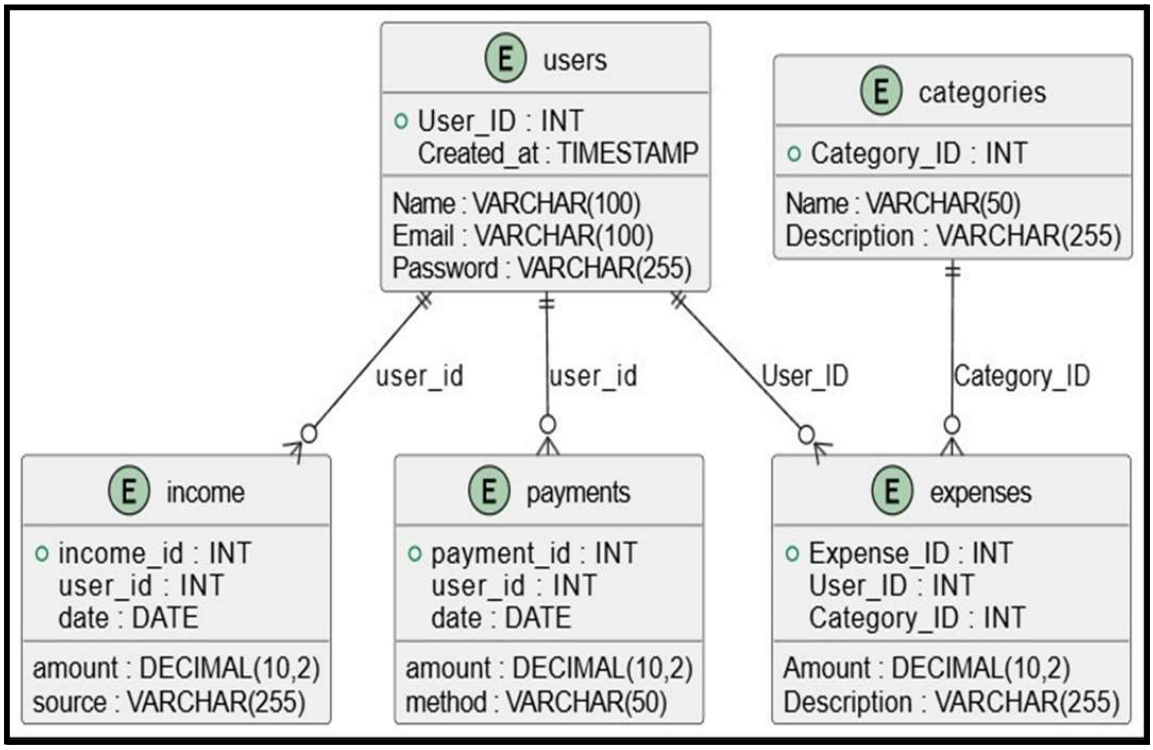
i. income_id: Unique identifier for each income record.
ii. user_id: Foreign key linking to the user.
iii. amount: Income amount.
iv. date: Date the income was received.
v. source: Source or description of the income.

```
CREATE TABLE Income (
    income_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    date DATE NOT NULL,
    source VARCHAR(255),
    FOREIGN KEY (User_ID) REFERENCES users(User_ID) ON DELETE CASCADE
);
```

e. Payments: Records user payment transactions.

i. payment_id: Unique identifier for each payment.
ii. user_id: Foreign key linking to the user.
iii. amount: Payment amount.
iv. date: Date of the payment.
v. method: Method of payment (e.g., cash, card).

```
CREATE TABLE Payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    date DATE NOT NULL,
    method VARCHAR(50),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```


users
User_ID : INT
Created_at : TIMESTAMP
Name : VARCHAR(100)
Email : VARCHAR(100)
Password : VARCHAR(255)
categories
Category_ID : INT
Name : VARCHAR(50)
Description : VARCHAR(255)
user_id
user_id
User_ID
Category_ID
income
income_id : INT
user_id : INT
date : DATE
amount : DECIMAL(10,2)
source : VARCHAR(255)
payments
payment_id : INT
user_id : INT
date : DATE
amount : DECIMAL(10,2)
method : VARCHAR(50)
expenses
Expense_ID : INT
User_ID : INT
Category_ID : INT
Amount : DECIMAL(10,2)
Description : VARCHAR(255)