**Personal Finance Management App**

A screenshot of a mobile application

Description automatically generated

**Group members :**

Samreen Abbasi- 6396941 Amrutha balumuri- 6334423  
 Gabriel Alberro - 6209261  
 Bhuvan Loknath - 6455938  
 Lewis Mesa - 6189964 Deepak Polasi Sai -6425586

**Tables:**

1. **Users Table**:
   * This table stores information about users of the personal finance management app.
   * Columns:
     + UserID: Unique identifier for each user (Primary Key).
     + Name: Name of the user.
     + Email: Email address of the user (Unique).
     + Username: Username chosen by the user (Unique).
     + PasswordHash: Hashed password of the user for security.
     + FinancialGoals: Text field to store the financial goals of the user.
2. **Accounts Table**:
   * This table stores information about the financial accounts owned by users.
   * Columns:
     + AccountID: Unique identifier for each account (Primary Key).
     + UserID: Foreign key referencing the UserID in the Users table.
     + Name: Name of the account.
     + Type: Type of the account (e.g., checking, savings, credit card).
     + Balance: Current balance of the account.
     + Institution: Name of the institution associated with the account.
3. **Categories Table**:
   * This table stores different categories that can be assigned to transactions and budgets.
   * Columns:
     + CategoryID: Unique identifier for each category (Primary Key).
     + Name: Name of the category (Unique).
4. **Budgets Table**:
   * This table stores information about budgets set by users for different spending categories.
   * Columns:
     + BudgetID: Unique identifier for each budget (Primary Key).
     + AccountID: Foreign key referencing the AccountID in the Accounts table.
     + CategoryID: Foreign key referencing the CategoryID in the Categories table.
     + MonthlyLimit: Monthly spending limit set for the category.
     + CurrentSpending: Current spending in the category for the budget period.
5. **Transactions Table**:
   * This table stores information about financial transactions recorded by users.
   * Columns:
     + TransactionID: Unique identifier for each transaction (Primary Key).
     + AccountID: Foreign key referencing the AccountID in the Accounts table.
     + Date: Date of the transaction.
     + Amount: Amount of the transaction.
     + CategoryID: Foreign key referencing the CategoryID in the Categories table.
     + Payee: Payee involved in the transaction.
     + Description: Description of the transaction.
6. **Investments Table**:
   * This table stores information about investments made by users.
   * Columns:
     + InvestmentID: Unique identifier for each investment (Primary Key).
     + UserID: Foreign key referencing the UserID in the Users table.
     + Name: Name of the investment.
     + Type: Type of the investment (e.g., stock, bond).
     + Quantity: Quantity or amount of the investment.
     + Value: Value of the investment.
     + PurchaseDate: Date when the investment was purchased.
7. **Bills Table**:
   * This table stores information about bills to be paid by users.
   * Columns:
     + BillID: Unique identifier for each bill (Primary Key).
     + UserID: Foreign key referencing the UserID in the Users table.
     + Name: Name or description of the bill.
     + DueDate: Due date of the bill.
     + Amount: Amount of the bill.
     + Payee: Payee of the bill.
     + Description: Description of the bill.
     + Status: Status of the bill (Pending or Paid).
8. **FinancialProjections Table**:
   * This table stores financial projections made by users.
   * Columns:
     + ProjectionID: Unique identifier for each projection (Primary Key).
     + UserID: Foreign key referencing the UserID in the Users table.
     + Date: Date of the projection.
     + Income: Projected income for the period.
     + Expenses: Projected expenses for the period.
     + NetWorth: Projected net worth for the period.
9. **Goals Table**:
   * This table stores financial goals set by users.
   * Columns:
     + GoalID: Unique identifier for each goal (Primary Key).
     + UserID: Foreign key referencing the UserID in the Users table.
     + Name: Name or description of the goal.
     + TargetAmount: Target amount set for the goal.
     + CurrentAmount: Current amount achieved towards the goal.
     + Deadline: Deadline for achieving the goal.
     + Status: Status of the goal (InProgress or Completed).

**Relationships:**

1. **Users Table**:
   * One user can have multiple accounts.
   * One user can have multiple investments.
   * One user can have multiple bills.
   * One user can have multiple financial projections.
   * One user can have multiple goals.
2. **Accounts Table**:
   * Each account belongs to one user.
   * Each account can have multiple transactions.
   * Each account is associated with one budget.
3. **Categories Table**:
   * Categories are referenced by transactions and budgets.
4. **Budgets Table**:
   * Each budget is associated with one account.
   * Each budget is associated with one category.
5. **Transactions Table**:
   * Each transaction belongs to one account.
   * Each transaction is associated with one category.
6. **Investments Table**:
   * Each investment belongs to one user.
7. **Bills Table**:
   * Each bill belongs to one user.
8. **FinancialProjections Table**:
   * Each financial projection belongs to one user.
9. **Goals Table**:
   * Each goal belongs to one user.

**Data Integrity:**

● The use of primary keys and foreign keys to ensure data integrity.

● The use of “Restrictions”, “Uniques” on columns to ensure the correct data format on specific columns.

● Hashing of Passwords to ensure data security for each user

**Efficiency:**

● Tables are normalized up to Boyce-Codd Normal Form to reduce redundancy in the table data.

● Create appropriate indexes on frequently used columns for faster queries.

**Justifications:**

1. **Normalization**: The database is normalized to reduce redundancy and minimize data duplication. This reduces storage space requirements and ensures data consistency, as updates only need to be made in one place.
2. **Indexes**: Appropriate indexes are created on frequently used columns such as UserID, AccountID, CategoryID, etc. Indexes improve query performance by allowing the database to quickly locate relevant rows, especially in large tables.
3. **Primary Keys and Foreign Keys**: Primary keys are defined for each table to enforce uniqueness and facilitate efficient data retrieval. Foreign keys establish relationships between tables, ensuring data integrity and efficient joins.
4. **Data Types**: Suitable data types are chosen for each column to optimize storage space and query performance. For example, using DECIMAL for monetary values ensures precision without wasting storage.
5. **Query Optimization**: The schema is designed with query performance in mind. By properly indexing columns used in WHERE clauses and JOIN conditions, the database engine can execute queries more efficiently.
6. **Denormalization**: While normalization is crucial for data integrity, selective denormalization may be applied where necessary to improve performance. This could involve duplicating some data to avoid costly joins in frequently executed queries.
7. **Foreign Key Constraints**: Foreign key constraints are used to maintain referential integrity, preventing orphaned records and ensuring that related data remains consistent. This contributes to efficient data management and avoids data corruption issues.
8. **Transaction Management**: The use of transactions ensures data consistency and reliability. By grouping related database operations into transactions, the database engine can optimize resource utilization and minimize locking contention.
9. **Optimized Joins**: Tables are joined efficiently based on their relationships to minimize the computational overhead associated with complex queries. Proper indexing and query optimization techniques further enhance join performance.

**Bonus:**

* Consider how you would adapt the schema to accommodate future needs, such as bill pay integration or cryptocurrency tracking.

1. **Bill Pay Integration**:
   * We can create a new table for bills, similar to the existing Bills table, but with additional columns such as BillType (e.g., utility, rent), DueDate, Amount, Payee, Description, and Status.
   * Include foreign key references to the Users table to associate bills with specific users.
   * Consider adding additional attributes related to bill payment status, payment method, and recurring billing information.
2. **Cryptocurrency Tracking**:
   * We can also Introduce a new table named Cryptocurrencies to track users' cryptocurrency holdings.
   * Define columns such as CryptoID, UserID (foreign key), Name (e.g., Bitcoin, Ethereum), Quantity, Value, PurchaseDate, and any other relevant attributes.
   * Establish foreign key relationships to the Users table to associate cryptocurrency data with specific users.
   * Include indexes on frequently queried columns for better performance, especially on UserID and CryptoID.

* implement data visualization and reporting features.

1. **Data Visualization Libraries**:
   * We can integrate data visualization libraries such as Chart.js, D3.js, or Plotly.js into your application frontend.
   * Utilize these libraries to create interactive charts, graphs, and dashboards that visualize users' financial data.
   * Visualize income vs. expenses, budget comparisons, investment performance, and net worth trends over time.

2. **Reporting Features**:

* + We can implement reporting features to generate customizable reports for users.
  + Allow users to generate reports on their income, expenses, investments, and financial goals.
  + Provide options to filter reports based on specific timeframes, categories, or accounts.
  + Include export functionality to save reports in various formats such as PDF or Excel for offline viewing.

1. **Data Aggregation and Analysis**:
   * We will aggregate and analyze users' financial data to derive meaningful insights.
   * Implement algorithms to identify spending patterns, trends, and anomalies.
   * Provide personalized recommendations and insights based on users' financial behavior and goals.
2. **User Dashboard**:
   * We will create a user dashboard where users can view summarized financial information at a glance.
   * Include widgets or cards displaying key metrics like total assets, liabilities, net worth, and progress towards financial goals.
   * Allow users to customize their dashboard layout and choose which metrics to display.