

1. Database (tables)

```
users (  
    user_id INTEGER PRIMARY KEY  
    username CHAR(30),  
    password CHAR(16)  
)
```

```
transactions (  
    transaction_id INTEGER PRIMARY KEY,  
    transaction_type TEXT,  
    transaction_category TEXT,  
    transaction_date TEXT,  
    transaction_amount REAL,  
    transaction_description TEXT,  
    user_id INTEGER,  
    FOREIGN KEY (user_id) REFERENCES users(user_id),  
)
```

2. Backend

***knows** - *association* (line)

***uses** - *dependency* (line w/ single solid arrow head)

***extends** - *inheritance* (line /w single hollow arrow head)

***has** - *aggregation* (line w/ single hollow diamond head)

***owns** - *composition* (line w/ single solid diamond head)

```
enum TransactionTypes {  
    "expense",  
    "savings",  
    "investment",  
    "income"  
}
```

```
enum ExpenseCategories {  
    "Bills",  
    "Education",  
    "Entertainment",  
    "Food & Drinks",  
    "Grocery",  
    "Healthcare",  
    "House",  
    "Shopping",  
    "Transportation",  
    "Wellness",  
    "Other"  
}
```

```
enum SavingsCategories {  
    "Monthly Allowance",  
    "Change",  
    "Miscellaneous"  
}
```

```
enum InvestmentCategories {  
    "Stocks",  
    "Crypto",  
    "Bonds",  
    "Real Estate"  
}
```

```
enum IncomeCategories {  
    "Salary",  
    "Bonus",  
    "Side-hustles",  
    "Tips"  
}
```

```
class UserRepository {  
    + addUser(username: str, password:str): int  
    + deleteUser(username: str, password:str): int  
}
```

```
class UserManager owns UserRepository {  
    - username: str  
    - password: str  
    - current_user_id: int  
  
    + signUp(): void  
    + login(): bool  
    + logout(): void  
}
```

```
dataclass Transaction {  
    - t_id: int  
    - t_date: str  
    - t_type: str  
    - t_category: str  
    - t_amount: float  
    - t_description: str  
}
```

```
class TransactionRepository uses Transaction {  
    + getAllTransactions(current_user_id: int): List[Transaction]  
    + getTransactionsByType(current_user_id: int, t_type: str): List[Transaction]  
    + getTransactionsByCategory(current_user_id: int, t_category: str):  
    List[Transaction]  
    + addTransaction(current_user_id: int, transaction: Transaction): void  
    + modifyTransaction(current_user_id: int, t_id: int, transaction: Transaction):  
    void  
    + deleteTransaction(current_user_id: int, t_id: int): void  
}
```

```
dataclass Finance {  
    - total_income: float  
    - total_expenses: float  
    - total_savings: float  
    - total_investment: float  
}
```

```
class TransactionManager owns TransactionRepository, and uses  
Transaction and Finance {  
    + calculateOverallFinance(): Finance  
    + calculateOverallBalance(overall_finance: Finance): float  
    + calculateMonthlyFinances(): Dict[str, Finance]  
    + calculateQuarterlyFinances(): Dict[str, Finance]
```

```
+ createMonthlyGraph(monthly_finances: Dict[str, Finance]): matplotlib.Figure
+ createQuarterlyGraph(quarterly_finances: Dict[str, Finance]):
  matplotlib.Figure
}
```