1. Database (tables) Schema

```
users (
        user_id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE,
        password TEXT,
        created_at DATETIME,
        updated_at DATETIME
)

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

2. Backend Structure

*__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"
}
```

dataclass Account {

- *username*: **str**
- *password*: **str**

}

class UserRepository {

- *connection*: **sqlite3.Connection**
- *cursor*: **sqlite3.Cursor**
+ initializeDatabase()
+ addAccount(*account*: **Account**): **bool**
+ getAccountID(*account*: **Account**): **int**

}

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(*user_id*: **int**): **List[Transaction]**
+ getTransactionsByType(*user_id*: **int**, *t_type*: **str**): **List[Transaction]**
+ getTransactionsByCategory(*user_id*: **int**, *t_category*: **str**): **List[Transaction]**
+ getRecentTransactions(*user_id*: **int**, *t_count*: **int**): **List[Transaction]**
+ addTransaction(*user_id*: **int**, *transaction*: **Transaction**): **void**
+ modifyTransaction(*user_id*: **int**, *t_id*: **int**, *transaction*: **Transaction**): **void**
+ deleteTransaction(*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(*user_id*: **int**): **Finance**
+ calculateOverallBalance(*user_id*: **int**, *overall_finance*: **Finance**): **float**
+ calculateMonthlyFinances(*user_id*: **int**): **Dict[str, Finance]**
+ calculateQuarterlyFinances(*user_id*: **int**): **Dict[str, Finance]**
+ createMonthlyGraphs(*user_id:* **int**, *width_in:* **float**, *height_in:* **float**, *title_size*: **int**, *label_size*: **int**): **Tuple[matplotlib.figure.Figure, matplotlib.figure.Figure]**

+ createQuarterlyGraph(*user_id:* **int***, width_in:* **float***, height_in:* **float**, *title_size*: **int**, *label_size*: **int**): **matplotlib.figure.Figure**

}