Splitwise is a popular expense-sharing application that simplifies the process of tracking and managing shared expenses among friends, family members, or roommates.
It's widely used to keep a record of who owes money to whom, making it easier to settle debts and avoid conflicts.
Here are some key details about the Splitwise application:

Main Features:

Expense Tracking: Users can create and record shared expenses, including who paid for each expense, the amount, and a description.

Multiple Expense Types: Splitwise supports different types of expenses, including:

Equal: Expenses are split equally among all participants.

Exact: Users can specify the exact amounts owed by each participant.

Percent: Expenses are split based on the percentage each user owes.

Group Management: Users can organise expenses by creating groups for specific events, trips, or shared households. This simplifies the tracking of multiple expenses within a particular context.

User Authentication: Secure user registration and authentication to ensure privacy and data protection.

Balance Simplification: Splitwise provides an option to simplify balances, which reduces the number of transactions needed to settle debts.
For example, if User A owes User B and User B owes User C, the application can simplify this into a direct transaction from User A to User C.

User-Friendly Interface: The application is designed for ease of use, with a clean and intuitive user interface.

Explanation of Components:

User Interface (UI):

Represents the front-end of the application, which can be a web interface.
Interacts with the application server through HTTP requests.

Application Server:

The central component that handles business logic, user requests, and communicates with the database.
Manages user authentication, expense creation, group management, and notifications.

User Authentication:

Manages user registration, login, and authentication.
Provides access control and session management for authenticated users.

Expense Management:

Responsible for creating, updating, and deleting expenses.
Manages various types of expenses, including Equal, Exact, and Percent.

Group Management:

Allows users to create and manage groups for sharing expenses.
Associates groups with individual expenses.

Transaction Tracking:

Tracks individual transactions within expenses.
Manages lenders, borrowers, amounts, and settlement statuses.

Database Server:

Stores application data, including user balances, expenses, groups, transactions, and simplified balances.
Typically used a relational database like MySQL.

Simplified Expenses Service:

A background service that periodically simplifies balances for users who enable this feature.
Recalculates balances and updates expense records accordingly.

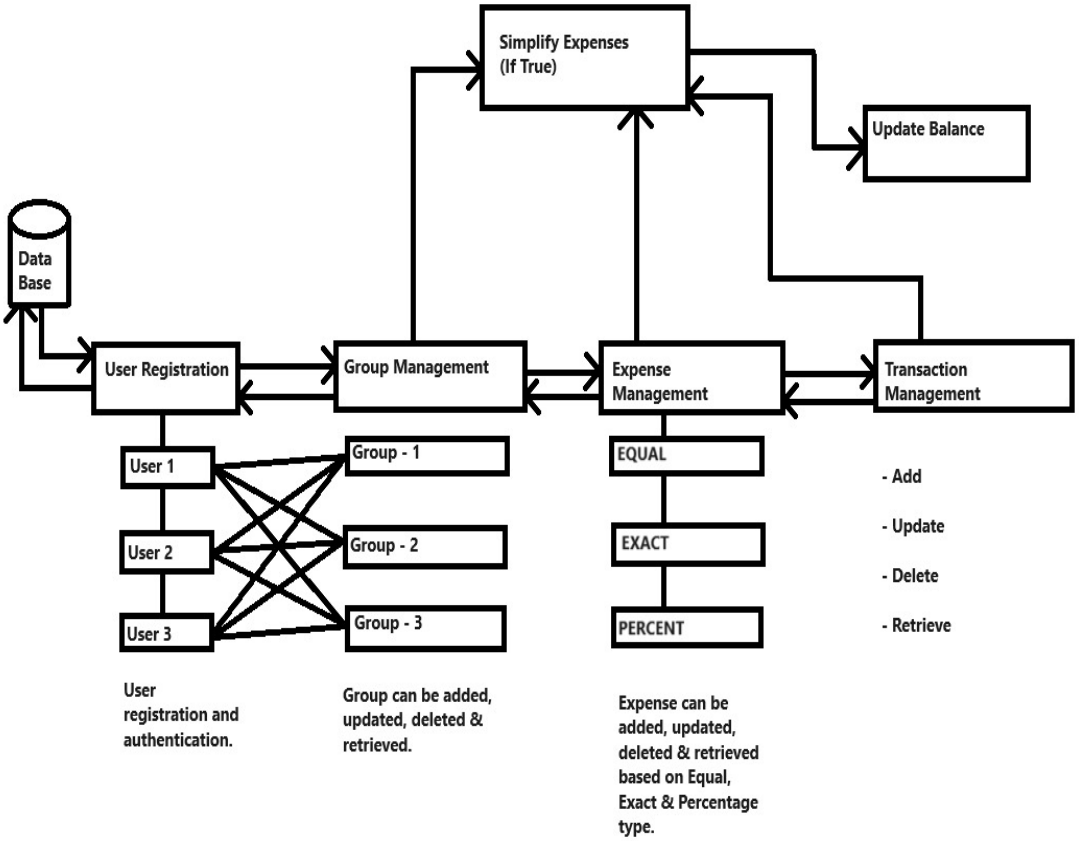
Email Notifications:

Sends email notifications to users when they are added to an expense.
Provides details about amounts owed, simplifications, and other information.

Periodic Reminders:

A background task or service that sends weekly reminders to users with pending expenses.
Includes information about simplified balances.

Architectural Diagram:



Structure of classes:

Explanation of the class structure:

UserProfile:

Represents user profiles, extending the default Django User model.

Group:

Represents groups that users can create to organize and share expenses.
It includes a name field and a many-to-many relationship with the User.

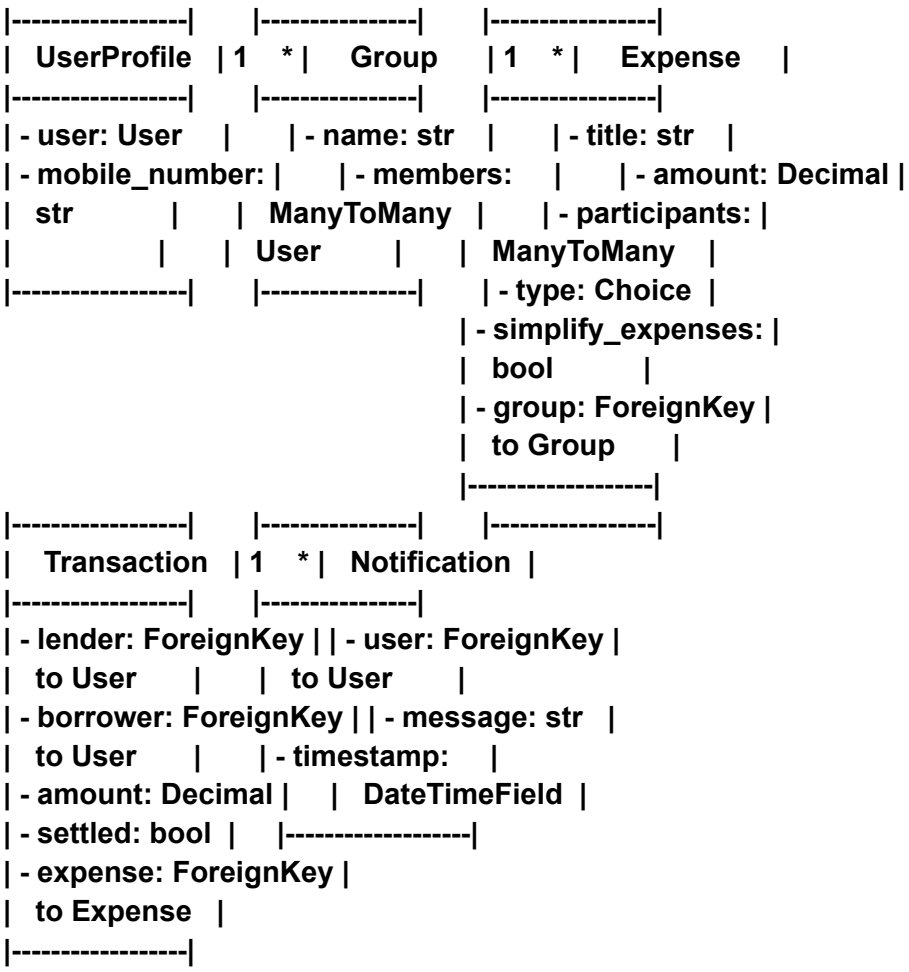
Expense:

Represents individual expenses within a group. It includes fields for the title, amount, participants, type (for equal, exact, or per cent expenses),
and a flag for simplify_expenses. It also has a foreign key relationship with the Group model.

Transaction:

Represents individual transactions within an expense, tracking who lent or borrowed money.
It includes fields for the lender, borrower, amount, and a flag for settled. It has a foreign key relationship with the Expense model.

Various methods and logic are added to handle expense calculations, balance updates, and other application-specific functionality as needed.
Additionally, user authentication, views, and forms for user interactions are implemented.



HTTP Endpoints:

User Registration and Authentication

POST /api/register/ : Register new user
POST /api/login/ : Log in and obtain an authentication token.
GET / api/token/refresh/ : Get token and refresh
POST /api/logout/ : Log out and invalidate the token

Group management

POST/api/groups/create/ : Create group
GET /api/groups/ : Get a list of user's groups

GET/api/groups/{group_id}/ : Get group details
PUT/api/groups/{group_id>}/update/ : Update group
DELETE/groups/{group_id}/delete/ : Delete group

Expense management

POST/api/expenses/create/ :Create expense
GET/api/expenses/ : Get a list of user's expenses
GET/api/expenses/{expense_id}/ : Get details of expenses
PUT/api/expenses/{expense_id/update/ : Update expenses
DELETE/api/expenses/{expense_id}/delete/ : Delete expenses

Transaction management

POST/api/transactions/create/ : Create transaction
GET/api/transactions/ : Transaction list
GET/api/transactions/{transaction_id}/ : Transaction detail
PUT/api/transactions/{transaction_id}/update/ : Update Transaction
DELETE/api/transactions/{transaction_id}/delete/ : Delete transaction

Update balance

PUT/api/balance/update/ : Update balance

Simplify Expenses

PUT/api/simplify/expenses/ : simplify_expenses

Technology Stack:

Backend:
Programming Language:Python
Framework:Django

Frontend:
Html

Database:
Mysql

Attached References:

1. Mysql dumb
2. ER diagram