|  |  |  |
| --- | --- | --- |
| **IOweU** | Prepared by : Group 12  Nishith Mehta Varun Patil Mayur Tare Chathuri Wimalasena | |
| IOweU enables a user to conveniently split the bills among fellow users and it also provides him the functionality of forming user groups. | |  |

**1. Introduction**

**1.1 Project Overview:**

IOweU (IOU) simplifies the task of managing shared expenses. Imagine that you are sharing an apartment with three friends. You will have expenses such as groceries, rent etc.; some of which may be shared by all of you while others only two of you may be sharing. On most occasions, bill will be paid via a single credit card. In such scenarios, IOweU will facilitate the settlement of shared bills.

**1.2 Purpose of this document**

In this document, we describe the conceptual features of IOweU application as well as its main design components. In particular, the document contains the system’s functional requirements, the data modeling and web interface design. The data modeling design contains both the ER modeling design and the design of the relational schema. The web interface design includes the skeleton of the web pages and the transition among them.

**1.3 Terminology**

* Private Users – Users added by registered users, which can only seen by them
* Settling Transaction – Transaction, which settles the debts
* Billing Transaction – These transactions represent the bills paid by the user who added the transaction.

**2. Feasibility Analysis and System Functions**

This section covers the functionality provided to user by IOweU.

**2.1 User Registration**

A new user can register for IOweU services with a valid Email Address and Username. User should create a unique Username and Password which will be used in further transactions in IOweU.

**2.2 User Authentication - Security**

Once a User is registered with IOweU, he/she can log in to IOweU using his credentials which will be authenticated at the time of logging in.

**2.3 Create a Group**

User can create a group of registered users. This will enable him/her to manage debts with these users easier. Bills can be filtered by Group name for user’s convenience.

**2.5 Add a Private User**

User can add Private users for his reference. These users will not be accessible to other registered users. This functionality is only to facilitate user to maintain a record of his debts with people who are not registered with IOweU.

**2.4 Add Bill**

A User can add a Bill which can be shared with the already registered users. The mandatory details required for adding a Bill are Description, Total Amount, and list of users with whom the bill is shared or pick a group to share the bill with and the percentage share for each user.

Users can disapprove the Bills which are shared with them.

A transaction will be considered valid if it is “Approved” by all users associated with it.

**2.5 Graphical Representation**

Available Data for the logged in user will be displayed using Pie-charts for better readability.

**2.6 Recent Activity**

A User can view recent activity associated with him, such as approval or disapproval of bills, settlement etc.

**2.7 Transaction Log**

Transaction log will be generated for all transaction in which logged in user is associated. This log can be filtered on various parameters such as Added by, Shared with, Group Name, Status etc.

**2.8 Settlement of the Debts**

User can select a group or a list of users to settle debts. He will be able to settle the debts only after generating a simplified report. The simplified report is generated such that there will be minimum number of transactions between the selected Users which will settle everyone’s debts.

**3. Data Modelling and Design**

**3.1 ER Diagram**



An abstract of ER diagram:

In this application, Users are segregated into two types Registered Users and Private Users. A registered user can add Private users which will be stored in “User” with a system generated Username and identified by the attribute ‘IsRegistered’. A registered User will be able to add a Bill which he can share with other users. “BillingTransaction” will store the Transaction details and “ShareBill” will store the sharing details. A transaction can have categorized information which will be stored in “CategorizedTransaction” and “Category” will contain categories available to the User. If a transaction is a recurring transaction then a copy is stored in “RecurringTransaction” for future reference. A user can create groups of users for ease of tracking his debts. Finally user can settle his debts with other users by generating a simplified report and final settling transactions will be stored in “SettlingTransaction”.

**3.3 Relational Schema**

USER (uname, name, IsRegistered)

REGISTEREDUSER (uname, email, pwdhash)

GROUP (gname)

GROUPCONSISTUSER (ForeignKey (gname) references Group.gname, ForeignKey (uname) references RegisteredUser.uname)

TRACKBT (ForeignKey (uname) references RegisteredUser.uname, ForeignKey (tid) references BillingTransaction.tid, desc)

TRACKGROUP (ForeignKey (uname) references RegisteredUser.uname, ForeignKey (gname) references Group.gname, desc)

TRACKST (ForeignKey (uname) references RegisteredUser.uname, ForeignKey (tid) references SettlingTransaction.tid, desc)

BILLINGTRANSACTION (tid, date, ForeignKey (addedby) references RegisteredUser.uname, ForeignKey (group) references Group.gname, amt, desc)

RECURRINGTRANSACTION (tid, isweekly, day)

SHAREBILL (ForeignKey (tid) references BillingTransaction.tid, ForeignKey (uname) references User.uname, amt, status, msg)

CATEGORY (cname)

CATEGORIZEDTRANSACTION (ForeignKey (tid) references BillingTransaction.tid, ForeignKey (cname) references Category.cname, amt)

SETTLINGTRANSACTION (tid, date, desc)

SETTLEBETWEENUSER (ForeignKey (uname) references User.uname, ForeignKey (tid) references SettlingTransaction.tid)

SETTLE (ForeignKey (stid) references SettlingTransaction.tid, ForeignKey (btid) references BillingTransaction.tid)

**4. Data Flow and Business Logic**

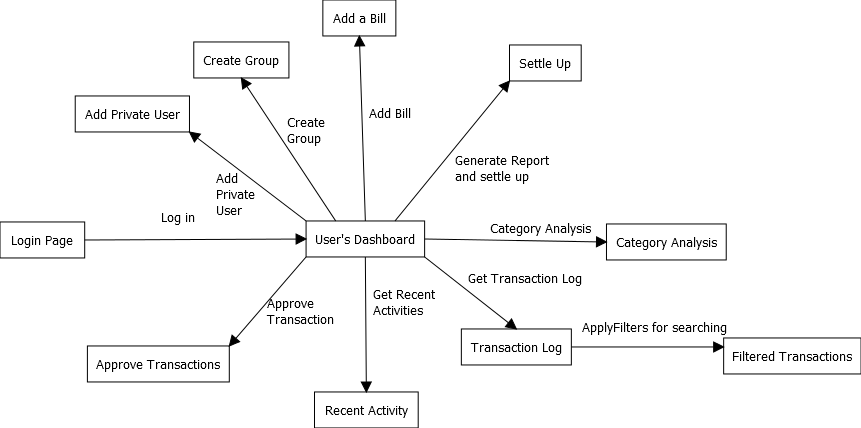
**4.1 Business Logic**

Take the function “Add Bill” as an example.

Once the user logged in, he will be able to add a bill. When adding a bill, user can provide the necessary information in the form such as, amount of the transaction, who are the users that transaction will be shared, what is the percentage that each user shares and other necessary information. Once the user clicks the Add Bill button, here are the steps that will happen at front end and back-end.

1. Front end level validation
2. Call the necessary controller (addTransaction method of the Dashboard controller)
3. addTransaction controller get the user filled data from the request and compute necessary objects in order to call the backend API method.
4. In the controller, several tasks are done before calling the backend method
   1. Since users can add both registered users as well as private users in the transaction, controller will check the username getting from the request is a registered user or a private user. Then instantiate objects accordingly.
   2. Calculating the amount that each user share according to the percentages given
   3. Assign rest of the transaction amount to the user who added the transaction
   4. Add category wise amounts
   5. Once all the information are processed, call the necessary backend method
5. In the backend, SQL insert statement will be executed in order to insert necessary information to the database. According to our database design, when a user adds a bill, we need to update two tables (BILLINGTRANSACTION, SHAREBILL)
6. Once the transaction executed successfully, backend will pass a success message to controller, which will be passed to the front end.
7. Once the transaction added successfully, user will notified saying “Transaction added successfully”. If the transaction fails, user will get understandable error message.

**4.2 Transition Diagram**



**5. Detailed Implementation**

**5.1 Tools and Platform Used**

IUGitHub was used as a project management and version control system for this project. Application is hosted at <https://156.56.179.104:8443/ioweu/home.jsp>

The actual implementation was developed using Spring Framework, MySQL on server side which were called from client side from jQuery and AJAX. For front end development we used Twitter Bootstrap, AngularJs and JQuery.

**6.** Limitations and future work

This beta version of the Application has following limitations,

1. Capability of Editing or deleting the added bills.
2. Email notifications

Future Scope,

The next version of this application can be released with additional functionalities such as Email notifications, capability of adding or editing bills.

Mobile version can be developed.

**7. Acknowledgement**

Thanks to Professor and AIs for all the suggestions and help they have provided for this project.