SuiteRates

Sung Tat Kwok
Brian Stone
Vadim Tkachev
Christopher To
Giles Westerfield
Tim Wong

Software Requirements Specification

Draft 1
Tuesday 17 April 2007

CSE 403 - CSRocks Inc.

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1	Sung Tat Kwok, Brian Stone, Vadim Tkachev, Christopher To, Giles Westerfield, Tim Wong	Overall Description, Scope, Use Cases, Feature List, UI Prototype	04/17/07

Overall Description

Description

SuiteRates is an online service for roommates who wish to organize and consolidate shared expenses easily and quickly. Anyone who has lived with one or more roommates will have experienced the tediousness required in coordinating all house or apartment expenses each month. SuiteRates allows roommates to securely manage expenses such as rent, utility bills, groceries, and home improvement items in one central location, without the worries of making tiresome and mistake-prone calculations. Users can define weekly or monthly recurring bills, and can opt to receive email alerts when those bills are due.

SuiteRates is designed to be flexible, allowing either a single roommate to be designated the task of making final payments, or allowing for the possibility for payments to be handled by PayPal. Furthermore, many transactions between roommates can flattened down to just one final payment, eliminating the need to keep track of payments and having to write many checks. All transaction history will be available for roommates to peruse, and if necessary, dispute.

Considering the breadth of its target consumer, SuiteRates will be capable of supporting a high volume of users. Users will register for a free account on our website which gives them the ability to create a “household”. All the other roommates can now register and join the household in order to get started. This system allows for as many customers to use our service as our hardware can support.

Scope

Depending on PayPal’s services, integration with PayPal may be beyond the scope of the system. At the very least, however, SuiteRates will be a service designed to manage and keep track of all shared expenses within a household. SuiteRates is designed to take the burden off roommates, by allowing users to add a payment and leaving the rest of the calculations to the system.

The SuiteRates service will run as a Ruby on Rails web service, with MySQL as its database system. This will require that the web server can handle Ruby applications and MySQL connection requests. On the client side, users of SuiteRates will require only a modern web browser that supports HTML, CSS and JavaScript. Interaction with the website will include both the classic and AJAX user interaction model.

Ruby on Rails was chosen as the primary scripting language due to its rapid development paradigm and modular Model-View-Controller architecture. The intuitive naming conventions in Rails greatly reduce the need for explicitly

defining database model objects (Rails can simply look at a database table and automatically create a corresponding object), and its rich API allows the developer to easily create dynamic form objects and make database queries without writing any HTML or SQL. Rails also comes with built-in CSS support, which will be essential in establishing a visual theme for our site. In general, the functionality that Rails provides is comparable to PHP and should be flexible enough for our project. The primary drawback to using Ruby on Rails is that it is generally slower than other languages like PHP because it tends to sacrifice performance speed in favor of more concise and simple code. However, this difference should be hardly noticeable, and will likely only begin to manifest itself if we are supporting tens of thousands of users. Should this become the case, there are still many ways to maximize performance under extreme conditions.

Use Cases

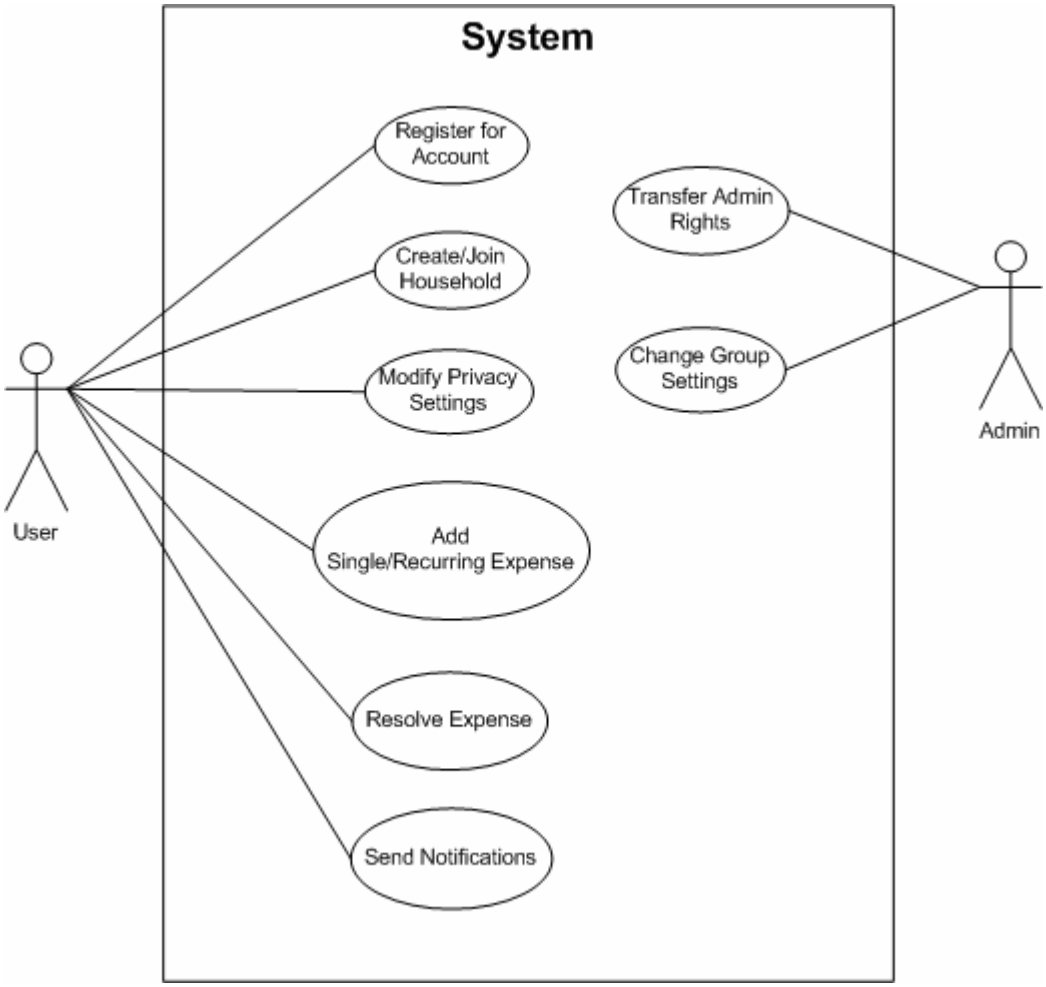
Formal Use Case 1

Goal	A user wishes to resolve the debt of another roommate
Level	User
Primary Actor	User
Precondition	User is at the login screen of the website; user has a registered account; user is a member of a household; user is owed money from another user
Success end condition	The debit is resolved
Failure end condition	The debt is not resolved
Trigger	User logs into the website
Main success scenario	1. User enters login information 2. Login information is authenticated and the user is signed on 3. User navigates to the *User* tab 4. User selects the *Resolve Balance* button for the coordinating expense 5. Expense is pending resolve until roommate verifies 6. Debt is resolved and the system updates the balances accordingly
Extensions	2a. User enters login information 2a 1 System notifies user of incorrect login info 2a 2 User attempts to re-authenticate or backs out 5a. Roommate declines the verification 5a 1 User is notified that the balance was not resolved
Variations	5. User is given the option to resolve only a portion of the balance 6. Roommate is also notified of his resolved balance

Formal Use Case 2

Goal	A user wishes to add a recurring expense that will be added each payment period
Level	User
Primary Actor	User
Precondition	User is at the login screen of the website; user has a registered account; user is a member of a household
Success end condition	A recurring expense is established
Failure end condition	The expense is never established and/or the expense is not recurring
Trigger	User logs into the website
Main success scenario	7. User enters login information 8. Login information is authenticated and the user is signed on 9. User navigates to the *Expenses* tab 10. User selects the *Add an Expense* button 11. Expense amount, payment period, and roommates are entered 12. Recurring expense is submitted and updated in the system
Extensions	2a. Login information is incorrect 2a 1 System notifies user of incorrect login info 2a 2 User attempts to re-authenticate or backs out 5a. Roommate does not exist 5a 1 System notifies user 5a 2 User given the option to proceed with or without said roommate 5a 3 User is given the option to invite the roommate
Variations	6. Affected roommates are also notified of new expense

Use Case Summary Diagram



Feature List

Feature	Type
Separate accounts to track users independently	Beta
Users can create or join "households" which have shared expenses	Beta
Verification process to determine whether a given user is allowed to join a household (other roommates must accept newcomer)	Beta
Add an expense to be paid by certain roommates, with a deadline and individual weights for each roommate	Beta
Add recurring household expenses that are shared between certain roommates	Beta
Household message board for public communication	Beta
Send private messages between users	Beta
Optional email reminders for upcoming expense deadlines	Beta
Two-way debt payment verification (both parties must agree that money was exchanged properly)	Beta
Transaction history for individual users and households	Beta
User profiles with personal information	Beta
Minimal GUI	Beta
Calendar view of coming expense deadlines for individual roommates and the household as a whole	Final
Contact mechanism to send email to site developers/tech support	Final
Polished GUI	Final
Final, reliable product—customer tested	Final
Documentation/Help/FAQ pages for users and developers	Final

UI Prototype

Figure 1: General Expenses Screen

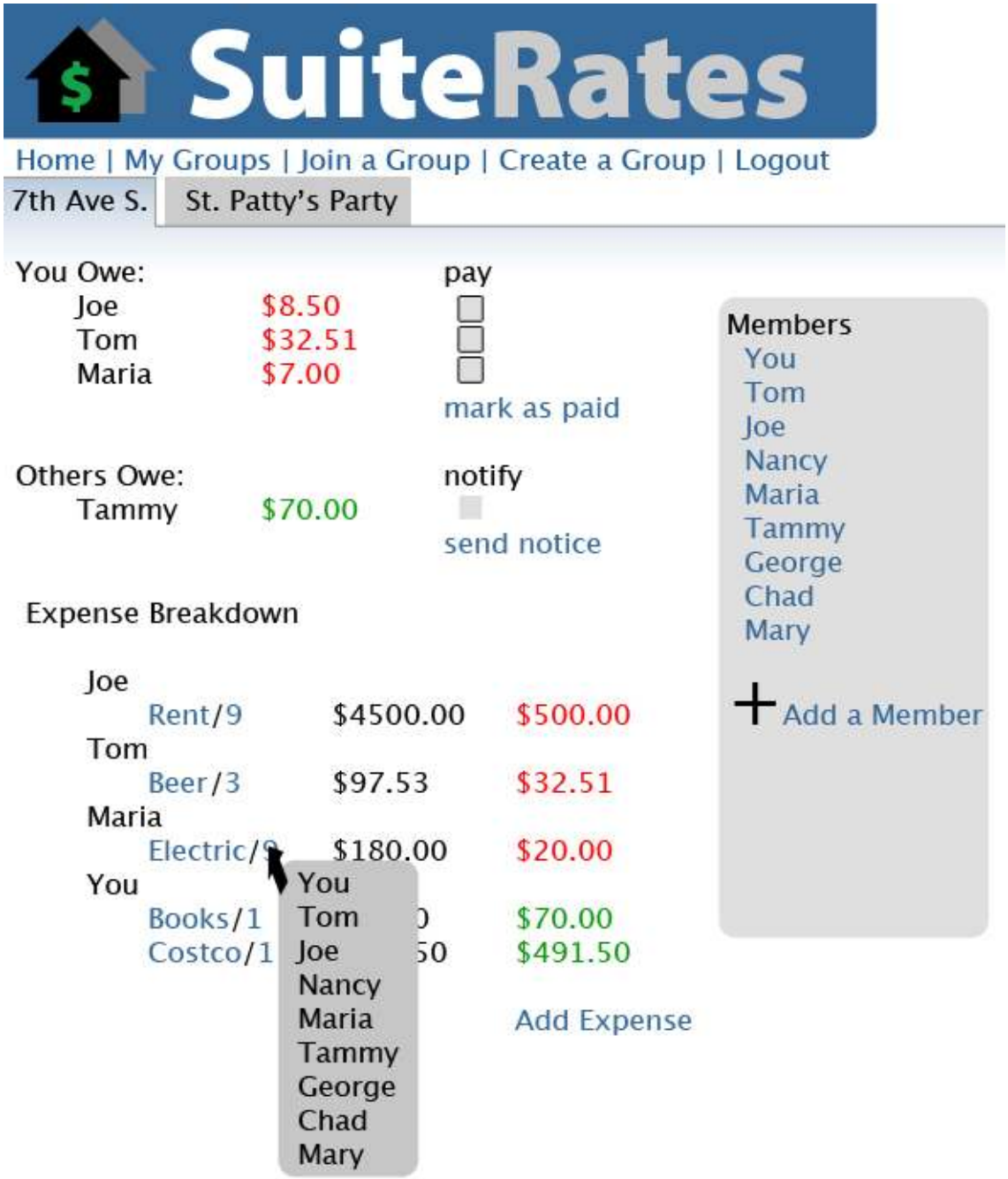


Figure 2: Adding an Expense