SuiteRates

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

System Design Specification and Planning Document

Draft 1
25 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	First draft.	04/25/07

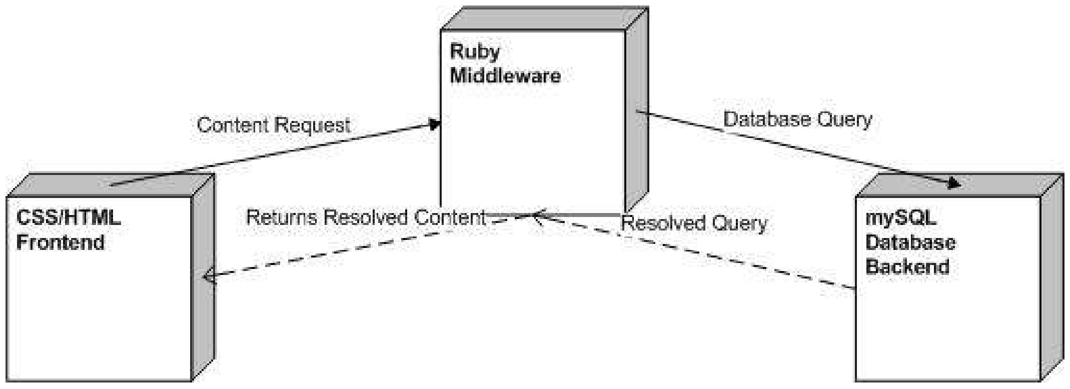
SuiteRates
Sung Tat Kwok, Brian Stone, Vadim Tkachev, Chris To, Giles Westerfield, Tim Wong

System Architecture

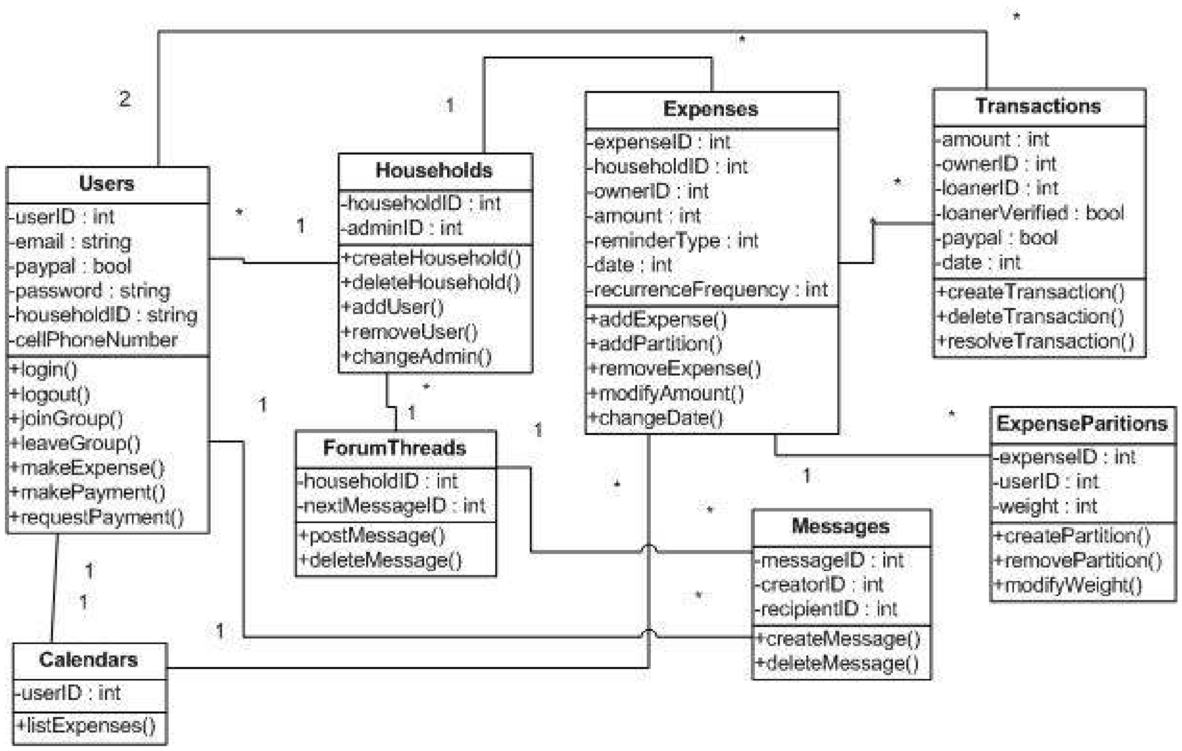
1. Introduction

The SuiteRates system architecture is composed of three components: the Ruby on Rails web service, the MySQL database, and the client front-end. Clients can connect to the SuiteRates service via a web-browser, which will interact with the Ruby on Rails web framework located on the server. This web framework will then interact with the MySQL database, storing or retrieving information as necessary.

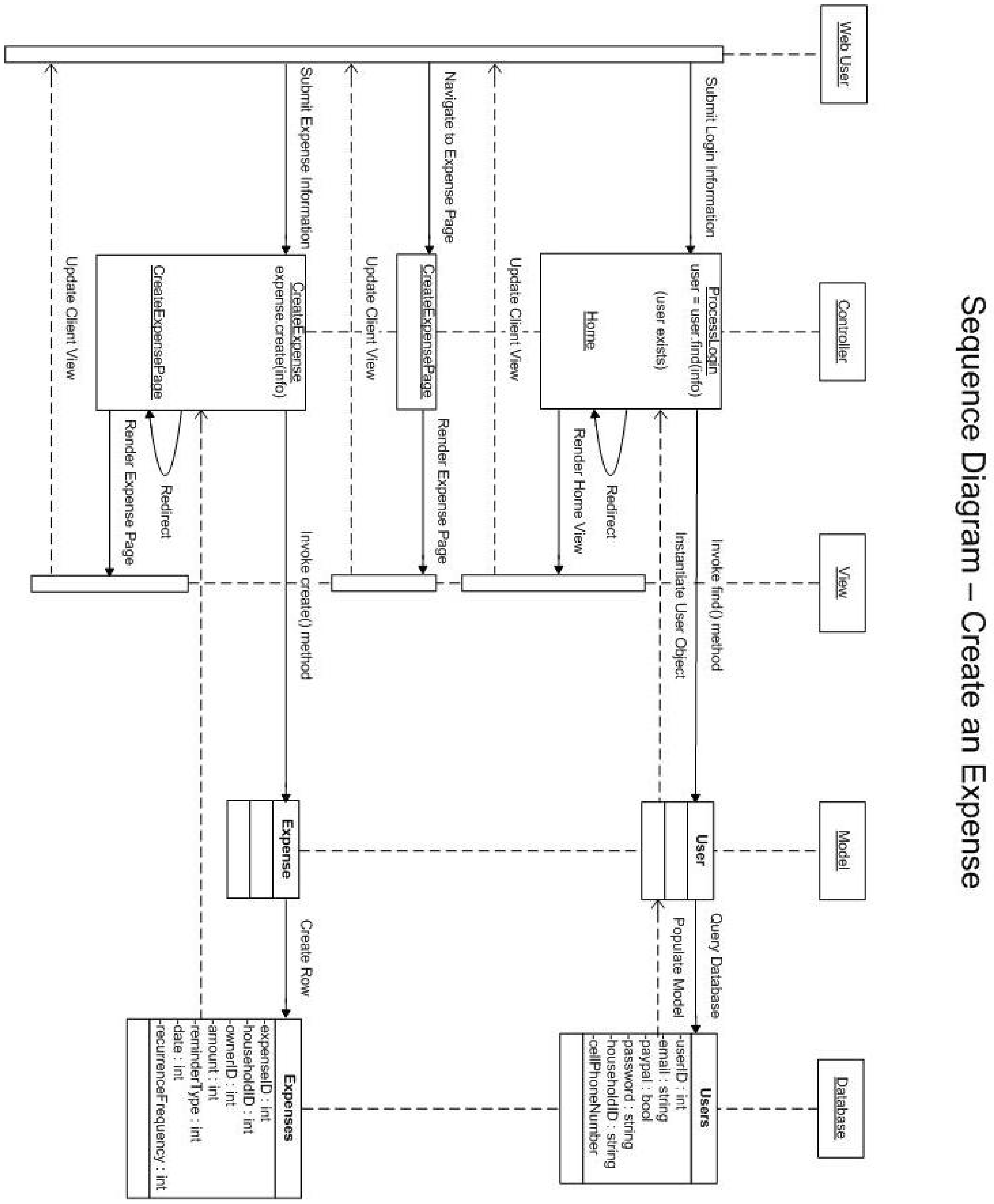
1. Design & System Architecture View



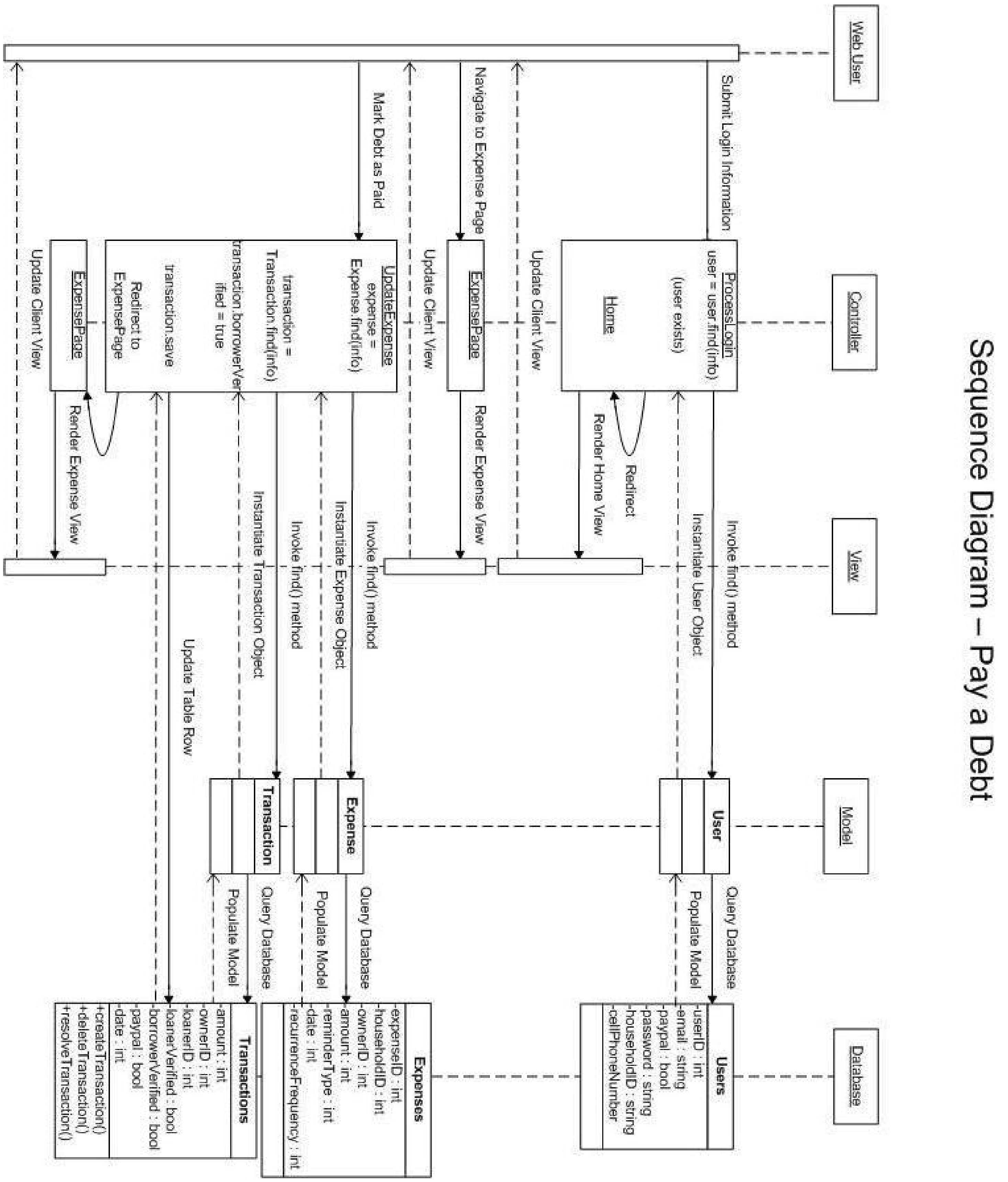
SuiteRates
Sung Tat Kwok, Brian Stone, Vadim Tkachev, Chris To, Giles Westerfield, Tim Wong



2. Process view - Use Case Sequence Diagrams



SuiteRates
Sung Tat Kwok, Brian Stone, Vadim Tkachev, Chris To, Giles Westerfield, Tim Wong



1. Database Schema

See class diagram. The database tables will be identical.

Development Plan

1. Team Structure

Our team structure is displayed below in the table. All team members will be assigned a specific role that they are responsible to oversee and manage. However, since our team is small and we only have a limited amount of time to develop the SuiteRates service, all team members will be expected to contribute with both the development and testing of the code and user interface.

Name	Role	Responsibilities
Sung Tat Kwok	Database Architect Lead Tester	- will oversee the design and development of the database schema - will manage the database throughout the project to make sure everything is running smoothly - will develop test plans and test cases to test both the server code and the client user interface on the web-browser
Brian Stone	UI Architect	- will design and develop a fluid and intuitive user interface - will design and conduct usability tests ranging from surveys to observing customer behavior
Vadim Tkachev	UI Architect	- will design and develop a fluid and intuitive user interface - will design and conduct usability tests ranging from surveys to observing customer behavior
Chris To	Graphics Designer	- will develop the graphical features of the user interface, such as icons, logos, and color schemes
Giles Westerfield	Lead Developer	- will oversee and manage the development of the code - will make sure that the team is sticking to the designed system architecture - will make sure each developer knows their responsibilties
Tim Wong	Program Manager	- will organize and coordinate weekly meetings and coding sessions - will act as a communication proxy between our team, our customers, and CSRocks - will make sure the team is on track with upcoming internal and external deadlines - will provide support for team members to bring up any issues

2. Project Schedule

SuiteRates has both internal and external deadlines. Internal deadlines are deadlines within our team that act as checkoff points to continually keep the team on track so that we won't fall behind. External deadlines are those imposed by CSRocks. The following is a list of the combined internal and external deadlines for this project.

Alpha Release (internal) - May 5
Beta Release (external) - May 10
Gamma Release (internal) - May 20-22
Final Release (external) - May 30

Our feature list is presented below, along with the corresponding release time that we plan to have completed that particular feature. It is difficult to assign a specific feature to a team member who would be responsible for that feature, since many of our features depend on the functionality of other features. Thus, we have attempted to resolve this issue by scheduling weekly coding sessions, where the team can get together and code. While we don't necessarily need to code as a group, and will most likely be working independently or in pairs, coming together for weekly coding sessions will allow the team to quickly and easily communicate amongst each other. We feel that by providing this open communication support framework, all team members will understand their own responsibilities and can quickly benefit from the support and knowledge of their peers as problems arise. The Program Manager and Lead Developer will work together to manage and delegate the responsibilities of developing particular features amongst the team.

Feature	Release
Separate accounts to track users independently	Alpha
Users can create or join "households" which have shared expenses	Alpha
Verification process to determine whether a given user is allowed to join a household (other roommates must accept newcomer)	Alpha
Add an expense to be paid by certain roommates, with a deadline and individual weights for each roommate	Alpha
Add recurring household expenses that are shared between certain roommates	Alpha

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	Gamma
Contact mechanism to send email to site developers/tech support	Gamma
Polished GUI	Gamma
Final, reliable product—customer tested	Gamma
Documentation/Help/FAQ pages for users and developers	Final
“Smart Balancing” – If user A owes user B $10, and user B owes user C $10, the site updates the debt to show that user A owes user C $10, effectively turning two transactions into one	Final
Different privacy settings depending on user preference	Final

PayPal support for online transactions	Stretch
SMS reminder	Stretch
Scan-able recipe (pdf)	Stretch
Export personal transaction history to excel	Stretch

3. Risk Assessment

Our team's risk assessment is displayed in the table presented below. We feel that we have covered a broad set of risks that will allow us to understand the project as a whole, and drive our team to success.

Risk	**Chance of occurring (High, Med, Low)**	**Impact if it occurs (H,M,L)**	**Steps taking to increase chance it won't occur**	**Mitigation plan should it occur**
Most members have little to no experience with Ruby on Rails, which could hold back development time	Low	High	All team members reading and researching RoR and playing on own cubist account	All should be able to pick it up quickly. Giles has past experience with RoR, so is our team's Ruby guru.
Unexpected scheduling conflicts, no team organization and communication	conflicts: med organization + communication: low	Med	Keep good communication amongst group members, constant update on google groups and wiki of team status, centralized svn repository, PM to coordinate and manage team members to work well together	identify areas that team member was responsible for (based on wiki/team status + google groups), load balance the work among remaining available members

Insufficient design and planning, jumping straight into development	Low	Med-High	Spending time brainstorming, designing, and planning fundamental architecture before any coding is started. Documenting all design processes before we begin so everyone understands what to do. Keep good internal and external communication to make sure we're all on the right track.	Team members should not make design and architecture assumptions that could potentially throw off other members. Consult PM for final decisions, who can mediate between other members of the team. If we find ourselves jumping into development without sufficient planning, take steps back to reevaluate as a group.
Attempting to implement too many features for final release (being overambitious)	Med	High	Stick to SRS feature list to stay focussed on the essential features that need to be done. Any new features can be added as a stretch, to be implemented in the future after final release, but a significant time should not be spent on the design and planning of these.	Go back to the SRS feature list, and see where we stand. Realize that this project can be forever ongoing, and that it's important to first implement several features well, then continue

Neglecting the importance of our customer	Med	High	PM to organize weekly meetings with customer to address UI and design issues, customer needs and requirements, and to retrieve any other feedback. Keep constant communication with customer, so that we can develop our product to fit the customer's needs.	Scheduling weekly meetings and keeping in touch via email will insure that we have constant communication between the development team and our customers. If this should fall through (due to scheduling conflicts), we can devise online usability feedback surveys that our customers can take on their own time.

Test and Documentation Plan

1. Test Plan

Overview

Testing will ensure the correctness, security, and quality of our service. The testing will be divided into different parts, namely: unit testing, integration testing, and usability testing.

a. Unit Testing

Unit testing is the foundation of all the other forms of testing in the testing plan. If the individual components in our system architecture do not work by themselves, they will not work well together. It is very important to test thoroughly how all the modules use and interact with each other throughout the system. Our unit tests will be written for all the methods and classes that have interaction in the new feature. The testers will write test methods that make certain assertions about code, working against a test fixture. A bunch of these test methods will be bundled up into a test suite that can be run easily by a developer when needed. The tools that we will use to develop our testing framework are:

– Ruby Unit Testing Framework

– "Web Application Testing in Ruby", or Watir for short: Watir is a free, open source functional testing tool for automating browser-based tests of web applications. The advantage is that Watir drives the browser the same way people do, automating link clicking, filling in forms, and pressing buttons. Watir also checks results, such as whether expected text appears on the page.

In addition, tests will be written for each newly discovered bug. This will ensure that new fixes and features do not have undesired effects on existing code and also isolate potentially recurring software defects.

b. Integration Testing

Integration testing determines whether the major subsystems that make up the project can work and play well with each other. It acts as an extension to unit testing.

General use cases will suffice for the smoke test. The test plan will be drawn from the specification document to ensure that the product successfully encompasses all the features of each component along with component integration. The ACID properties of our database transactions will also be tested. The database will be populated with arbitrary data to represent user information. This can be done by loading a script into MySQL to generate arbitrary data. The benefits of using sample data are:

– easy to observe whether user information are processed correctly by the server and client front-end

- ability see whether there are any dirty-reads raised by other users, or by faulty server code

c. Usability Test Strategy

Usability testing is different from the other types of testing discussed above. It is performed with real users, under real environmental conditions. The criteria to testing will be in terms of human factors. In this part, we will ask several ‘users’ to experience our system. After the testing, we will provide them with a survey to see whether the system meets our project goals and fit our customer's needs. It is important to know whether there are any misunderstands during requirements analysis that needs to be addressed and the software is user-friendly to keep our customers happy.

d. Bug tracking mechanism and plan of use

We will utilize the BugZilla bug tracking management software to track defects and features in our software project. This will allow us to keep track of every issue, assign issues to certain team mates, and determine what issues are unresolved and resolved.

2. Documentation Plan

User Guide

The user guide will be a hyperlink within the website. This will be a formal and complete documentation to guide the user in how to use this system to its fullest capabilities. This documentation will provide the specifications which are required to support the product, and will fully describe features of the product from the perspective of the users. Demonstrations are also essential in the user guide, allowing users to follow demonstrations via videos or screenshots on the webpage links.

Admin Guide

This will be documentation for internal use. It will include the following:

– Setting up and running the system

– Configuration of the system which details the software and hardware requirements

– Design view of the system

– Description of design components.

– Maintenance

Quick and Dirty Guide

This documentation sums up the main features of the system which allow users to use the product quickly and fairly effectively without reading the User Guide. In other words, it will be an outlined version of the User Guide. Users probably spend 5 -10 minutes to read through the document.

Help Pages

Help pages will be a frequently asked questions (FAQ) section to address common user issues. The team will collect common users' problems found during usability and other types of testing. Furthermore, this help section will be continually updated to address customer concerns, by allowing customers to provide feedback and questions.

Additional resources

The group can use the team Wiki page to track current progress. The Wiki page is open to the public so that other people are also able to know the flow of the development. In addition, this document will post the evaluation of the product and its progress, such as design decisions, administrative decisions, and recommendations for future releases.