Team Dinosaurs

Final Submission

Oct 26 2012

Milestone 4
Who Owes What to Whom

Table of Contents

Executive Summary

The **Who Owes What to Whom** system is a web-based system (to be implemented using Ruby on Rails) that keeps track of household expenses for roommates, flat-mates, apartment-makes, or just groups of friends. The system keeps track of an individual's purchases in relation to the group, and reports on how much they have spent, how much they are owed, group averages, and other diagnostic data. Although no money transferring software will be involved, this system aims to eliminate the use of paper/pen or spreadsheet methods of keeping track of group expenses.

Introduction

This document is the fourth of five milestones, and outlines the project's coding standards, change control, and finally, test cases for each of the Use Cases. The coding standards outlined in this document subscribe to general Ruby coding standards. Further, although other documents produced during this project have not had strict version control, this document (in addition to all future documents) will be under strict version control (as outlined in the section on Change Control). Finally, the document concludes with a suite of test cases for each of the project's use cases. These test cases are not comprehensive, but they give a survey of relevant test cases that need to pass (or fail) for each Use Case.

This document need not be read in conjunction with other documents. Reading this document in conjunction with Milestone 2 will provide a background for the Use Cases that are being tested in this document (although a Use Case to Feature Mapping table is provided) and will allow the reader to understand the conditions associated with the test cases. Further, reading this document in conjunction with Milestone 1 addresses the coding standards that were chosen.

Project Background

Features

Fund Tracking		
#	**User Benefit**	**Supporting Features**
F1	Tracking lending between users allows the users to see what has been lent and for how long.	User can track how much money they lend to other users via app.
F2	Tracking debts allows users to easily see exactly how much money someone is owed.	User can track how much they payed for an item and consequently how much they are owed.
F3	Users living with multiple people can track who has recently bought items and track those who haven't contributed recently.	App lists who recently purchased which items and for what cost.
F4	Users living with multiple people can track average expenditures per person of the group.	App calculates and displays group average of what is owed.
F5	Users can easily determine who in their group is spending above or below the group average.	The app calculates the group average and compares it to individual averages.

Feature	Status	Priority	Effort	Risk	Stability	Target Release	Reason
F1	Proposed	Critical	3 weeks	Low	High	V1.0	User can keep track of their lend
F2	Proposed	Critical	1 week	Medium	High	V1.0	User can keep track of their house purchases
F3	Proposed	Important	2 weeks	Low	Medium	V1.0	User can keep track of their purchases, and others can keep updated about purchases
F4	Proposed	Important	2 weeks	Low	Medium	V2.0	User convenience
F5	Proposed	Useful	2 weeks	Low	Medium	V2.0	User convenience

Use Case Identification

Account Creation (UC1)

Secure Login (UC2)

Group Creation (UC3)

Owed for a Single Item (UC4)

Owed for Multiple Items (UC5)

View Current Balance (UC6)

Indicating Payments (UC7)

View Group Averages (UC8)

Edit Account Information (UC9)

Edit Group Information (UC10)

Use Case Feature Mapping

	Use Case	Feature
UC1	Account Creation	
UC2	Secure Login	
UC3	Group Creation	
UC4	Single Item Entry	F2
UC5	Multiple Item Entry	F2, F3
UC6	View Current Balance	F1, F2
UC7	Indicate Payments	F2
UC8	View Group Averages	F4, F5
UC9	Edit Account Info	
UC10	Edit Group Info	

Coding Standards

The goals of well-styled code are correctness and simplicity, in that order.

Ruby code should be indented two spaces.

Statements that extend past 80 characters or so should be broken up and indented on the following line.

Ruby allows you to leave out parenthesis; in general we will keep them in.

Use if statements as expressions.

Only use trailing if statements in guard clauses.

Multiple return statements are fine.

Adapted from http://pathfindersoftware.com/2008/10/elements-of-ruby-style

Change Control

How do you receive requests? What information do you expect?

We expect change to come from four places: 1) Fellow team-members, 2) Our Client, Preston Sego, 3) Our Project Manager, Susi Cisneros, and 4) Dr. Chenoweth. From each of these entities, we expect different information. From our team members, we expect information about the skeleton and the meat of the document in technical (although possibly colloquial) terms. Change requests from team members can be brought to light during meetings, or through emails, and may or may not be officially recorded. From Preston, we expect information pertaining to the content of the document in retrospect. Preston is less concerned with the creation of the document and more with revision of existing materials such that they satisfy his needs. These requests are a result of any meetings that are held, or any email correspondences over the life of the project. From Susi, we expect revision information about current documentation such that it satisfies Dr. Chenoweth and the format for the class. Although she is also interested in making sure that Preston is satisfied, it is our responsibility that we are on the same page with our Client, so we do not expect major content changes from her. We receive change requests from Susi during our weekly meetings with her and the feedback that she gives us about each of our PM submissions. Finally, Dr. Chenoweth is the final say for documentation feedback, although he (like Susi) is not necessarily as concerned with Client satisfaction during the submission of every milestone. Dr. Chenoweth provides change requests during his final assessment of the milestone using the grading rubric. At this point in the cycle, however, the feedback needs to be crucial to be incorporated into the document “postmortem”.

How do you manage change requests?

The decision of whether or not a change will be incorporated is influenced by 2 things

1. The Source of the Request
2. The Severity of the Change

Changes that our client suggests are inherently more crucial to make to the project than changes that are requested by other sources. Similarly, our Client is more interested in the “meat” (content) of the milestones, the documentation, and the system than the “skeleton” (format, layout, etc), and as such, these changes are considerably more severe. Changes submitted by our PM, on the other hand, typically fall somewhere between content and form, as she acts a liaison between client and classroom needs.

At the end of the day, the documents are living documents, so any changes requested by any party (through the avenues addressed above) are considered by the team.

How do you manage changes to project artifacts?

Changes to documentation have not been handled in any standard way thus far. Revisions to existing documentation are done on a “need-to-do” basis, where every team member contributor makes changes to relevant sections in the “living” document that is available to all team members via Dropbox. After submission to the PM (also via Dropbox), any feedback that is given is presented to the group, and the team members again make their changes to the version two document. Finally, the document is submitted via an Angel dropbox to Dr. Chenoweth.

Upon the creation of this document, and the realization that the project team needs to consider a “Change Log”, one has been added to the Dropbox. Because we did not consider keeping track of recorded changes to previous documents, they have not been included. However, changes made to this document and future documents will be documented in the Change Log.

Unit-Test Case Suite

UC1. Account Creation

ID	Scenario	Description	Input	Expected
1.1	Invalid Password	User fails to provide a valid password for account creation.	A password that either contains invalid characters or is of invalid length.	User prompted to re-enter username and password.
1.2	Invalid Email	User fails to provide a valid email for account creation.	An email that does not exist or contains invalid characters.	User prompted to re-enter email address.
1.3	Valid Information entered	User a valid email and password for account creation.	A valid email and a valid password.	User receives confirmation email for account registration.

UC2. Secure Login

ID	Scenario	Description	Input	Expected
2.1	Invalid log-in information	User fails to provide correct email or password for log-in.	User enters either invalid email/ password or their email and password do not match.	User prompted to re-enter email and password.
2.2	User forgets log-in information	User requests an email with their log-in information to be sent.	User presses “forgot password” button and enters their email.	User receives correct log-in information in an email.
2.3	User repeatedly enters incorrect information.	User fails to provide correct information for the same email 5+ times in a short period.	User enters the wrong password 5+ times for a single email.	Account, if exists, is temporarily locked for 1 hour.
2.4	Valid Information entered	User enters correct information.	User enters correct email and password.	User is able to log-in correctly.

UC3. Group Creation

ID	Scenario	Description	Input	Expected
3.1	Invalid group name entered	User fails to provide a valid name for group creation.	A group name that either contains invalid characters or is of invalid length.	User prompted to re-enter group name.
3.2	Unable to create group	User is unable to create groups because they do not have permission or belong to the maximum amount of groups.	User selects “create group” option.	Group is not created and user is informed that they are unable to create groups at this time and why.
3.3	Valid Information entered	User enters correct information.	User creates group with a valid name.	Group is created and user is informed that group has been created.

UC4. Single Item Entry

ID	Scenario	Description	Inputs		Expected
			Cost Entered	**Selected Category**	
4.1	Valid item entry	User enters a valid entry	Positive cost	Category Exists	The item is accepted and added to the running total
4.2	Valid item entry	User enters a valid entry in a new category	Positive cost	“Add Category” selected	System first prompts user for new category name
4.3	Invalid item entry	User enters an invalid entry	Negative cost	Category Exists	System highlights the cost field and notifies user of negative amount

ID	Scenario	Description	Inputs		Expected
			Cost Entered	Selected Category	
4.4	Invalid item entry	User enters an invalid entry in a new category	Negative cost	"Add Category" selected	User if prompted for new category name, then notified of invalid amount once submit is pressed.

UC5. Multiple Item Entry

ID	Scenario	Description	Inputs		Expected
			# of Values > $0	# of Values < $0	
5.1	Valid item entry	User enters valid items	All entries positive	No negative values	The items are accepted and added to the running total
5.2	Invalid item entry	User enters an invalid entry	Most entries positive	One negative value	System highlights the invalid negative cost and notifies the user of it. No changes made
5.3	Invalid item entry	User enters multiple invalid entries	Any number of positive entries	Multiple negative values	System highlights all invalid cost fields and notifies user of negative amounts

UC6. View Current Total Owed/Borrowed

ID	Scenario	Description	Inputs		Expected
			Entries exist?	Sum of Entries	
6.1	Group with debts	User selects a group that owes them money	Entries recorded	Negative debt sum	The system displays that the user is owed money, and shows a list of items that factor into that total
6.2	Group with debts	User selects a group that they owe money	Entries recorded	Positive debt sum	The system tells the user that they owe this group money and shows a list of the items factored into this total
6.3	Group without debts	User selects a group that they do not owe money to and that doesn't owe them anything	Entries recorded	Zero debt sum	System tells them that there are no debts in this group, but can display the items that lead to this balanced total
6.4	Group without entries	User selects a group that doesn't yet have any items entered	No entries	N/A	System notifies the user that there are no debts because there are no items yet. It doesn't show a list of items

UC7. Indicate Payments

ID	Scenario	Description	Input	Expected
7.1	Failed Login	User fails login verification.	Invalid email or password	User prompted to log in again.
7.2	Cancelled	User cancels the process at any time.	“Cancel” Button	User taken to main menu.
7.3	No Debts	User has no debts to be paid.	(none)	User notified that they have no debts, and then they are returned to the main screen.
7.4	Payment Indicated	User indicates that they have paid one or more debts.	“Pay Debt” Button	User has one or more debts paid off.

UC8. View Group Averages

ID	Scenario	Description	Input	Expected
8.1	Failed Login	User fails login verification.	Invalid email or password	User prompted to log in again.
8.2	Cancelled	User cancels the process at any time.	“Cancel” Button	User taken to main menu.
8.3	Averages	User views the group averages.	“View Group Averages” Button	User has seen the averages for the group.
8.4	Bar Graph	User views the bar graph for the group.	“View Bar Graph” Button	User has seen the bar graph representing group averages.
8.5	Pie Chart	User views the pie chart for the group.	“View Pie Chart” Button	User has seen the pie chart representing group averages.

UC9. Edit Account Information

ID	Scenario	Description	Input	Expected
9.1	Failed Login	User fails login verification.	Invalid email or password	User prompted to log in again.
9.2	Cancelled	User cancels the process at any time.	“Cancel” Button	User taken to main menu.
9.3	Invalid Email	User attempts to change their email.	Invalid email or taken email	User prompted for another email.
9.4	Valid Email	User attempts to change their email.	Valid and unique email	User’s email updated.
9.5	Mismatched Passwords	User attempts to change their password.	New password and new password confirmation do not match	User prompted for another password.

ID	Scenario	Description	Input	Expected
9.6	Password Changed	User attempts to change their password.	New password and new password confirmation match	User's password updated.

UC10. Edit Group Information

ID	Scenario	Description	Input	Expected
10.1	Failed Login	User fails login verification.	Invalid email or password	User prompted to log in again.
10.2	Cancelled	User cancels the process at any time.	"Cancel" Button	User taken to main menu.
10.3	Added User	User adds another user to group.	Valid email and "Add User" Button	Other user is added to the group.
10.4	Removed User	User removes another user from group.	Valid email and "Add User" Button	Other user is removed from group.
10.5	Mismatched Group Codes	User attempts to change group code.	New group code and group code confirmation do not match	User prompted for another password.
10.6	Group Code Changed	User attempts to change group code.	New group code and group code confirmation match	Group password updated.

QA Driven Test Cases

Functional Requirements Test Cases

ID	Scenario	Description	Input	Expected
F.1	Maintain User Information	Through multiple logins, the system should keep user information.	Multiple logins and logouts	All user information maintained.
F.2	Maintain Group Information	Through multiple logins, the system should keep group information.	Multiple logins and logouts	All group information maintained.
F.3	Maintain Debt Record	Through multiple debt entries, the system should keep debt records.	Several months of single and multiple item entries.	All debt records maintained.
F.4	Synchronized Debt Information	Each group member should be able to see the items that they owe/are owed for.	User A adds an item owed to/by User B.	User B can see that User A added a new item.
F.5	Correctly Track Debts	Through multiple debt entries, the system should correctly track the debts.	Several months of single and multiple item entries.	Correct amount owed to/by for user.
F.6	Editing Other Users' Info	The system should not allow people to edit other users' information.	User A clicks on User B's profile.	User A not allowed to make changes to B's profile.
F.7.1	Group Edit by User	Normal user tries to edit group information.	User clicks on "Group Management"	User not allowed access to group management page.
F.7.2	Group Edit by Admin	Group administrator tries to edit group information.	Admin clicks on "Group Management"	Admin allowed to edit group information.

Reliability Requirements Test Cases

R1. Effective Handling of Database Errors

ID	Scenario	Description	Inputs		Expected
			Errors	Timeout Status	
R1.1	No error occurs	The database transaction is completed without an error	No errors	Timeout not reached	The system continues to the next task without any noticeable delay to the user
R1.2	Error Occurs	A single error occurs, but it is resolved on the next attempt	One error	Timeout not reached	The system continues to the next task without any noticeable delay and without notifying the user
R1.3	Errors Occur	Errors continue to occur, but not long enough to reach the timeout	Multiple errors	Timeout not reached	System continues on to next step after a short (less than the 5 second timeout) delay

ID	Scenario	Description	Inputs		Expected
			Errors	Timeout Status	
R1.4	Errors Occur	Errors persist long enough to cause the system to timeout (5 seconds)	Multiple errors	Timeout reached	System notifies the user that there has been a problem accessing the database, and requests that they try again later

R2. Page Load Error Handling

ID	Scenario	Description	Inputs		Expected
			Interruptions	Timeout Status	
R2.1	No interruption occurs	The webpage (or portion thereof) loads without any interruptions	No interruptions	Timeout not reached	The page is displayed to the user properly formatted without delay
R2.2	Interruption occurs	A single interruption takes place, but it is quickly resolved	One interruption	Timeout not reached	The system retries loading the portion of the page it was on, and then displays it correctly after no noticeable delay
R2.3	Interruptions occur	Multiple interruptions (or one prolonged one) occur, causing delays	Multiple/ continued interruptions	Timeout not reached	System displays portions of page already loaded, waiting for the remaining sections to load and displaying them as they do.
R2.4	Interruptions occur	Interruption(s) persist long enough to cause the system to timeout (5 seconds)	Multiple/ continued interruptions	Timeout reached	System displays properly loaded sections in correct format and gives a notification that the remaining sections failed to load due to connection problems

Supportability Requirements Test Cases

S1. System Upgrade Handling (No interruption to users during system updates)

ID	Scenario	Description	Inputs		Expected
			Change Type	User Data	
S1.1	No change	The pages the user is working on do not change while in use	No changes	N/A	The system continues to the next task without any problems and behaves as usual
S1.2	Changes made	A change is made to the page the user is currently viewing, but without data	Change to current page	No data	The system continues to work as expected, but will change if refreshed
S1.3	Changes made	A Change is made to a data-entry page the user is working on	Change to current page	Data entered reached	The system attempts to copy the data over to the new page and prompts for any new data needed

ID	Scenario	Description	Inputs		Expected
			Change Type	User Data	
S1.4	Page removed	A page that is linked to is removed	Page removed	Data entered	System notifies the user that the page has been removed, but saves the data and reloads the current page, which should have an updated link

Usability Requirements

ID	Scenario	Description	Input	Expected
U.1	User navigates to the website	The user has a modern internet browser and attempts to get to the web app	The user enters the URL into their browser.	The page should load with correct function and format.
U.2	User attempts to use various functions of the website	The user attempts to use various functionalities of the website without formal training.	The various functions of the website.	The user should be able to accomplish their task without formal training.
U.3	User attempts to lend to another user.	The user attempts to use the lending functionality between themselves and another user.	The user enters the user to lend to and what they're lending.	The two users should be able to view the lent items status and value.
U.4	User attempts to lend to a group.	The user attempts to use the lending functionality between themselves and a group.	The user enters the group to lend to and what they're lending.	The group members should be able to view the lent items status and value.
U.5	User sets a group admin.	The user that created a group attempts to set another user as group admin.	The user enters the group settings page, selects a user and gives them admin rights.	The user selected should have group admin rights.

Performance Requirements

ID	Scenario	Description	Input	Expected
P.1	The application must built to accommodate a small to medium-sized user base.	The application must support a user base sized 10-100.	10-100 Users are registered in the system.	The system will remain online with no performance degradation.
P.2	The application should be built to accommodate an approximate average of 8 users active at any time.	The application must support an average of 8 active users at any time.	8 Users are active on the system.	The system will remain online with no performance degradation.
P.3	The application should not have any processes/queries that take longer than 2.5 seconds.	The applications queries and processes must take no longer than 2.5 seconds	A process or query is run.	The system will perform the task in under 2.5 seconds.

ID	Scenario	Description	Input	Expected
P.4	The application should be built to have a non-noticeable variation in response times.	The difference in completion times of different user requests must be negligible.	A user requests two different responses.	The system will carry out both requests with no noticeable time difference between the two.

Prototype to Test Case Relations

Our prototype addresses the following test cases:

1.3 - Valid information for account creation. If the user enters valid information into the email/password box, the system notifies them that their account has been created. (Although not via email, the system does a pop-up notification).

5.1 - Valid item entry during multiple item entry. When the user enters valid items into the prototype, the system notifies the user of the total amount entered.

6.1 - The user selects a group that owes them money. The system displays that the user is owed money in a particular group, and shows a list of the items that factor into this total.

6.2 - The user selects a group to whom they owe money. The system displays that the user owes the group money, and shows a list of items that factor into their debt.

F.3 - Maintain Debt Record. The system maintains debt records for many months of (simulated) data.

F.6 - Editing Other Users' Info. The system does not allow any particular user to modify the info of other users.

S1.1 - No change to user pages when updates are pushed. The prototype server does not push changes to a users pages when the pages are in use.

S1.4 - A page that is linked to is removed in an update. The system displays an error message if the user requests a page that has been removed.

U.1 - The page can be visited by modern web browser.

U.2 - The page can be used without formal training. This will be analyzed in the next milestone.

P.4 - The prototype was built to have a non-noticeable variation in response times. Although the response time of the prototype isn't maximized, every page loads in approximately the same amount of time.

Appendix & Glossary

Breadcrumb Navigation	Allows the user to keep track of their locations relative to parent pages.
Effort	Estimate of the amount of time required for a particular feature.
Group	A household; one or more users associated with each other
Heroku	Heroku is a cloud platform supporting several programming languages, including Ruby, Java, Python, and PHP.
Modern Internet Browsers	Google Chrome, Mozilla Firefox, Safari
Preston Sego	Client
Priority	Ranks the relative priority or benefit to the end user.
Risk	Indicates a measure of the probability that the feature will cause undesirable events, such as cost overruns, schedule delays or cancellation.
Ruby On Rails	An open source, full-stack web application framework for the Ruby programming language.
Stability	Reflects a measure of the probability that the feature will change or that the team's understanding of the feature will change.
Status	Tracks progress during definition of the project baseline and subsequent development

Index

References

Class Slides

Leffingwell, Dean, and Don Widrig. *Managing Software Requirements: A Use Case Approach*. Boston: Addison-Wesley, 2003. Print.

Preston Sego, during Phone Interviews and email correspondences

Ruby on Rails Development Website: http://rubyonrails.org/

Change Log

Version	Date of Release	Changes
V 1.0	10/20/12	Initial draft of document - outline of use cases and tests
V 1.1	10/25/12	Added QA Test cases, Change Log
V 2.0	10/26/12	Final draft of document - revisions to existing test cases, index updated