Test Plan for Quarters

James Anthony (anthonjb)
Wenqiang Chen (chenw25)
Carolyn Chong (chongce)
Kevin Ly (lyk2)

January 5, 2016

Contents

Revision History

Date	Comments
October 21, 2015	Created first draft.

Template

This document makes use of the Validation, Verification and Testing Plan Template, and ”Integrated Stroke Test Plan” for all of its organization.

[A brief introduction or system description would be useful. —DS]

1 Acronyms and Definitions

Acronym	Description
PoC	Proof of Concept

2 Plans for Automated Testing

All tests that will be automated will be completed for the Final Demo April 1. We will automate testing for every feature, except financing, notification and chat. We will also test the integrity of the database using automated testing. We will also check if every page is reachable, to ensure there are no unhandled HTTP errors. We plan to run automated tests on a weekly basis, every Sunday night. Automated testing tools include Grunt, SinonJS and custom Python scripts.

3 Plans for Unit Testing

All unit tests will be automated. Unit tests will be created as features are developed. They will be separated by module. Since the majority of modules will be Javascript, we will be using QUnit. QUnit is a Javascript unit testing framework.

[You should start planning your unit tests now. You can determine the majority of them ahead of time. —DS]

4 System Tests

4.1 User Registration

Test Type: Functional, Dynamic, Automated.
Tools Used: Custom Scripts, Google reCAPTCHA.
Schedule: Begin testing November 8. Complete manual tests by PoC Demo November 16. Complete automated dynamic tests by Final Demo April 1.
Team Member Responsible: Carolyn Chong.
Methodology: The main objective of user registration is to create a user account to be used for login. Users must use a valid email address and pass a user identification procedure. This ensures the user is human and prevents spam and automated scripts from accessing the application and abusing its services. Testing is manual and automated. Manual testing involves people manually going through the registration process in real-time as a user. Automated testing involves systemically attempting SQL injections to test for valid and invalid registrations. Google reCAPTCHA validates that users are legitimate.

Test Case	Initial State	Input	Output
4.1.1	Landing page. Empty fields.	Email and password entered and passes reCAPTCHA test. Clicks register.	Verification email sent. Redirected to application main page.
4.1.2	Landing page. Empty fields.	Empty field(s). Clicks register.	Stays on the same page. Error message appears. Empty field is highlighted.
4.1.3	Landing page. Empty fields.	Email address already stored in database. Clicks register.	Stays on the same page. Error message appears. Email field is highlighted.
4.1.4	Landing page. Empty fields.	Fails reCAPTCHA test.	Stays on the same page. Error message appears. Test field is highlighted.

4.2 User Login

Test Type: Functional, Dynamic, Automated.
Tools Used: Custom Scripts.
Schedule: Begin testing November 8. Complete manual tests by PoC Demo November 16. Complete automated dynamic tests by Final Demo April 1.
Team Member Responsible: Carolyn Chong.
Methodology: The main objective of user login is to ensure a secure process where only valid users are allowed to enter the application. Testing involves authenticating users against an existing database to determine if they are valid users or not. Testing is automated. Automated testing involves systemically attempting SQL injections to test for valid and invalid logins.

Test Case	Initial State	Input	Output
4.2.1	Landing page. Empty username and password fields.	Valid username and password combination. Clicks login.	Redirected to application main page.
4.2.2	Landing page. Empty username and password fields.	Invalid username and password combination. Clicks login.	Stays on the same page. Error message appears. Fields are highlighted. After 5 unsuccessful attempts, user cannot login for 10 minutes.

4.2.3	Landing page. Empty username and password fields.	Empty username and/or password fields. Clicks login.	Stays on the same page. Error message appears. Fields are highlighted.
4.2.4	Application main page.	Clicks logout.	User is successfully logged out from system. Redirected to login page.
4.2.5	Landing page. Empty username and password fields. User attempting to login on another device while already logged in on a device.	Valid username and password combination. Clicks login.	Stays on the same page. Error message appears.

4.3 Calendar

Test Type: Functional, Dynamic, Automated.
Tools Used: Custom Scripts.
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: Carolyn Chong.
Methodology: The Calendar feature allows users to add/delete events and chores to a shared Calendar between members of a house. This shared Calendar can be synched with a user's personal Calendar. Testing is automated. Automated testing involves a script that will go through the process of adding/deleting an event or chore to the Calendar in real-time as a user, and then checking if those updates are properly synched with the user's personal Calendar.

Test Case	**Initial State**	**Input**	**Output**
4.3.1	Calendar page.	Select Calendar view (Year, Month, Week, or Day).	Calendar view is displayed.
4.3.2	Calendar page.	Select current or future date.	Form opens.
4.3.3	Calendar page.	User(A) selects event/chore that User(A) created.	Form opens.

4.3.4	Calendar page.	User(B) selects event/chore that User(A) created.	Nothing.
4.3.5	Calendar page. Existing form created by User(A) is open.	User(A) modifies event details and saves changes.	Form closes and updated Calendar is displayed.
4.3.6	Calendar page.	Select past date.	Nothing.
4.3.7	Calendar page. Empty form.	Add event/chore. Text entered in fields and saves.	Form closes. Event/chore is updated on Calendar.
4.3.8	Calendar page. Empty form.	Add event/chore. Empty field(s). Clicks save.	Form remains open. Error message appears. Empty fields are highlighted.
4.3.9	Calendar page.	Click to delete event/chore.	Event/chore is no longer displayed on Calendar. [Who can delete events/chores? —DS]

4.4 Maintenance Tracking

Test Type: Functional, Dynamic, Static, Automated.
Tools Used: QUnit, Chron Scripts. [Chron scripts? —DS]
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: Kevin Ly.
Methodology: The maintenance tracking system allows tenants to create maintenance requests, where the landlord then responds and updates with further information. This portion of the system is restricted based on the user type; tenants cannot modify maintenance ticket properties. This component will be tested using unit tests for functionality, with automated testing to ensure the permissions are handled properly. Static database checkers will be used in conjunction with the automated test cases to check for proper database modifications.

Test Case	**Initial State**	**Input**	**Output**
4.4.1	Quarters Web Application.	Open maintenance system.	Maintenance system opens and shows new maintenance tickets with existing tickets in chronological order.
4.4.2	Maintenance System.	Click on maintenance ticket.	Inner dialog opens displaying all properties in a maintenance ticket.

4.4.3	Maintenance System.	Entering a search query or adding a filter.	Sort and filter maintenance tickets and reveal only successful tickets.
4.4.4	Maintenance Ticket Window.	Modifying properties of a ticket.	Save icon appears in dialog to confirm changes.
4.4.5	Maintenance Ticket Window.	Saving ticket properties	Window will close, and database will be updated to reflect changes.
4.4.6	Maintenance Ticket Window.	Deleting Ticket.	Confirmation window will appear. Upon deletion confirmation, close window and remove data from database.
4.4.7	Maintenance System	Click on create new request.	Opens a new ticket window.
4.4.8	New Maintenance ticket window.	Click on create empty fields.	Window will remain opening, prompt will display error message.
4.4.9	New Maintenance ticket window.	Click on create, required fields filled.	Window closes, database will be updated with new ticket.
4.4.10	New Maintenance ticket window.	Click on cancel with fields filled.	Window remains open, prompt will ask for confirmation on close.
4.4.11	Confirmation Prompt.	Click on OK.	Closes prompt and dialog.
4.4.12	Confirmation Prompt.	Click on cancel.	Closes prompt, dialog remains open.

4.5 House Management

Test Type: Functional, Dynamic, Automated.
Tools Used: QUnit.
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: Kevin Ly.
Methodology: The house management system allows users to create, delete, view, and modify information about houses. Unit tests will be created for each function in the feature which will be included in the automated testing sequence.

Test Case	**Initial State**	**Input**	**Output**
4.5.1	House Management, not admin.	Click modify information.	Nothing.
4.5.2	House Management, admin.	Click modify information.	Input fields become editable.
4.5.3	House Management, admin.	Modify information fields.	Save button opens, discard changes appears.
4.5.4	House Management, any user.	Click on View Documents.	Redirects to new page showing all uploaded documents in House.
4.5.5	House Documents, any user.	Clicks on a document.	Retrieves documents and initiates file transfer.
4.5.6	House Documents, admin.	Clicks on Add Documents.	Upload window opens for user upload, file will be transfer to server and information is updated in database.
4.5.7	House Documents, admin.	Clicks on delete document.	Prompt opens.
4.5.8	Deletion prompt, admin.	Clicks on yes.	Prompt closed, file is removed from display, database is updated.
4.5.9	Deletion prompt, admin.	Clicks on no.	Prompt closed.
4.5.10	House Management, any user.	Clicks on view members.	Shows all memebers of the house and their role.
4.5.11	House Management, admin, members list visible.	Clicks on add member.	Dialog will appear.
4.5.12	Member Dialog, admin, fields empty.	Clicks on ok.	Prompt opens, notifying missing fields.

4.5.13	Member Dialog, admin, fields complete.	Clicks on ok.	Window closes, new user is notified, database is updated, member status pending.
4.5.14	Member Dialog, admin.	Clicks on cancel.	Window closes.

4.6 Landing Page

Test Type: Functional, Dynamic, Static, Automated.
Tools Used: Custom scripts.
Schedule: Begin testing November 8. Complete manual tests by PoC Demo November 16. Complete automated dynamic tests by Final Demo April 1.
Team Member Responsible: Kevin Ly.
Methodology: The landing page allows users to login and create new accounts. This page also displays information about the application for visitors. This feature will be manually tested since there are not many test cases.

Test Case	Initial State	Input	Output
4.6.1	Landing Page, not logged in.	Clicks on login.	Username field appears, password field appears.
4.6.2	Landing Page, not logged in.	Clicks on registration.	Email and password fields appear.

[So how does registration work? What happens if someone tries to register an existing email/username? —DS]

4.7 Live Chat

Test Type: Functional, Dynamic, Manual.
Tools Used: None.
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: Wenqiang Chen.
Methodology: The main objective of live chat is to allow another means of communication inside the house, in addition to the Discussion Board. The testing involves one user establishing live chat with another user. Testing will be manual. Manual testing involves one user(A) sending a message to another user(B) and ensuring user(B) receives the message without delay.

Test Case	Initial State	Input	Output

4.7.1	Main page. User logged in. House selected. House members displayed.	Click on house member to send message.	Chat window opens, with target user's name as window name.
4.7.2	Main page. User logged in. Chat window open.	Input text and click send.	Chat windows displays user's message. Time stamp is displayed.
4.7.3	Main page. User logged in. Chat windown ["window" —DS] open.	No text input and click send.	Chat windows does not change, nothing is sent.
4.7.4	Main page. User logged in. Chat notification displayed.	Click on notification.	Chat window opens with sender's message and time stamp displayed. Notification disappears.

4.8 Notification

Test Type: Functional, Dynamic, Manual.
Tools Used: None.
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: Wenqiang Chen.
Methodology: The main objective of notification is to remind user of events that has had happened; users should be notified immediate after the event has taken place. The testing involves one user completing different actions which generates notification and have another user related to this event receive notification.

Test Case	Initial State	Input	Output
4.8.1	Main page. User(A) logged in.	User(B) sends money request.	User(A) sees notification of pending payment due.
4.8.2	Main page. User(A) logged in.	User(A) pays user(B).	User(B) sees notification of payment completed.
4.8.3	Main page. User(A) logged in.	User(A) has late payment.	User(A) sees notification of late payment.

4.8.4	Main page. User(A) logged in.	User(A) joins a house.	Other users in that house sees notification that user(A) joined the house.
4.8.5	Main page. User(A)(landlord) logged in.	User(B) sends maintenance ticket(Critical).	User(A) sees notification of unresolved maintenance ticket, receives email, receives text message.
4.8.6	Main page. User(A)(landlord) logged in.	User(B) sends maintenance ticket(Major.)	User(A) sees notification of unresolved maintenance ticket, receives email.
4.8.7	Main page. User(A)(landlord) logged in.	User(B) sends maintenance ticket(Minor).	User(A) sees notification of unresolved maintenance ticket.
4.8.8	Main page. User(A) logged in.	User(B)(Landlord) resolves a maintenance ticket.	User(A) sees notification of resolved maintenance ticket.
4.8.9	Main page. User(A) logged in.	User(B) sends user(A) a message.	User(A) sees notification of unread message.
4.8.10	Main page. User(A) logged in.	User(B) makes a post in discussion board.	User(A) sees notification of unread post.
4.8.11	Main page. User(A) logged in.	User(B) replies to a post made by user(A).	User(A) sees notification of unread reply.
4.8.12	Main page. User(A) logged in.	User(A) leaves a house.	Other users in that house sees notification that user(A) left the house.
4.8.13	Main page. User(A) logged in.	User(B) adds event to Calendar.	User(A) sees notification of added post.
4.8.14	Main page. User(A) logged in.	User(B) deletes event from Calendar.	User(A) sees notification of deleted event.
4.8.15	Main page. User(A) logged in.	User(A)has event happening on day.	User(A) sees notification of event.

4.8.16	Main page. User(A) logged in. Notification displayed.	User clicks on Notification icon.	Notification disappears.

4.9 Administrative File Storage

Test Type: Functional, Dynamic, Automated.
Tools Used: Custom Scripts.
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: James Anthony.
Methodology: A script can be used to test the process of uploading and downloading multiple files of different types and sizes.

Test Case	**Initial State**	**Input**	**Output**
4.9.1	0 files in storage.	User tries to upload a file of size s, where $s \leq$ max file size.	Successful file upload.
4.9.2	0 files in storage.	User tries to upload a file of size s, where $s >$ max file size.	Error message indicating file has not been uploaded.
4.9.3	n files in storage.	User tries to upload a file of size s, where $s \leq$ total remaining space.	Successful file upload.
4.9.4	n files in storage.	User tries to upload a file of size s, where $s >$ total remaining space.	Error message indicating file has not been uploaded.
4.9.5	n files in storage.	User tries to upload a file with an invalid type.	Error message indicating file has not been uploaded.
4.9.6	n files in storage.	User requests to download a file.	Successful file download.
4.9.7	n files in storage.	Connection interrupted while download is in progress.	Error message indicating file has not been downloaded.
4.9.8	n files in storage.	User tries to upload $n > 1$ files.	Error message indicating only one file can be uploaded at a time.
4.9.9	n files in storage.	User clicks delete file.	File removed.

4.10 Bulletin Board

Test Type: Functional, Dynamic, Automated.
Tools Used: Custom Scripts.
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: James Anthony.
Methodology: A script can be used to test the process of posting on the discussion board, and commenting on existing posts.

Test Case	**Initial State**	**Input**	**Output**
4.10.1	No posts on bulletin board.	A post with 0 characters	Empty post is disgarded [“discarded” —DS] and not added to bulletin board.
4.10.2	No posts on bulletin board.	A post with n characters, where $n > 0$.	Bulletin board is updated with the post of n characters.
4.10.3	p posts on bulletin board, where $p > 0$.	A post with 0 characters	Empty post is disgarded and not added to bulletin board.
4.10.4	p posts on bulletin board, where $p > 0$.	A post with n characters, where $n > 0$.	Bulletin board is updated with the post of n characters.
4.10.5	p posts on bulletin board, where $p > 0$.	A comment with 0 characters on an existing post p.	Empty comment is disgarded [discarded —DS] and not added to bulletin board.
4.10.6	p posts on bulletin board, where $p > 0$.	A comment with n characters where $n > 0$, on an existing post p_i.	Comment is added to the list of comments associated with post p_i.

4.11 Finance

Test Type: Functional, Dynamic, Manual.
Tools Used: None.
Schedule: Begin testing after the PoC Demo. Complete automated tests by Final Demo April 1.
Team Member Responsible: James Anthony.
Methodology: Tests can be performed by having one user add payment deadlines, and having another user send arbitrary amounts of money via PayPal. Speed and accuracy of transactions can be tracked. Transaction records can be manually evaluated for correctness.

Test Case	Initial State	Input	Output
4.11.1	No payments due.	User initiates PayPal transaction.	Error message indicating that no payments are due at the current time.
4.11.2	No payments due.	User posts payment request with some deadline.	New payment deadline added.
4.11.3	Payment due.	User initiates PayPal transaction.	Transaction is handled by PayPal. All users involved are notified of the completed payment. Deadline is removed from list of current payments due.
4.11.4	Payment due.	Payment has not been completed, and deadline has passed.	All users involved are notified that the deadline has passed. Deadline is marked as past due, and users will continue to be notified until either the payment has been completed, or the due payment is removed.
4.11.5	Payment due.	User who posted the original due payment removes the request.	Payment request is removed from the list of due payments.

5 Non-Functional Tests

5.1 Look and Feel

To test that the system is attractive and intuitive and appears professional and secure, we will survey ten users to rate the user interface on a scale of 1 to 10, where a 1 means ”ugly, unprofessional and would not return to the site”, and a 10 means ”captivating, professional, and would refer a friend”. We will also ask users to provide comments or suggestions for qualitative feedback. The testing schedule will include a test December 7 and February 22. The first test will be used as a baseline. We will do a second test before the Final Demo to see if we improved.

[If you are providing a survey, you should include a copy of it as an appendix. —DS]

5.2 Usability

To test that the system is intuitive to use and navigate, we will ask ten users to complete a set of tasks on the site. Five users will act as landlords, who will create an account, login, send an email invitation to invite users to a house, and then create a post on the discussion board. Five other users will act as tenants who will accept an email invitation, create an account, login, and then create a post on the discussion board. Users will rate their experience on a scale of 1 to 10, where a 1 is ”frustrated, could not complete task(s), would not recommend”, and a 10 means ”user friendly, easy to navigate, would recommend to a friend”. We will also ask users to provide comments or suggestions for qualitative feedback. The testing schedule will include a test December 7 and February 22. The first test will be used as a baseline. We will do a second test before the Final Demo to see if we improved.

[Can you more accurately describe what the users will be tasked with doing? —DS]

5.3 Performance

To test the server, we will do a load testing to make sure the server can handle 100 simultaneous requests.

[How? —DS]

5.4 Security

To test the security of the system, including file access, failed password attempts, SQL injections, and expired sessions, we will do manual testing.

[Be more descriptive. —DS]