Test Report for Quarters

Team 6
James Anthony (anthonjb)
Wenqiang Chen (chenw25)
Carolyn Chong (chongce)
Kevin Ly (lyk2)

March 20, 2016

Contents

List of Figures

List of Tables

Revision History

Date	Comments
March 20, 2016	Created first draft.

1 Introduction

This testing report shows the results of both system tests and non-functional tests on the Quarters application. The system tests are reported based on each individual module. Non-functional tests included tests on usability, performance and robustness.

2 Automated Testing

[Explain use of automated testing, or explain why it was not feasible for this project. —CC] Automatic testing is being used in this project to unit test various parts of the system. The project's components are broken up in to several parts: Client side javascript components, Server side access and security.

The client side javascript is tested every week once a week on an alternate web server using QUnit, a javascript unit-testing framework. Unit tests were written for each component to ensure every method does their intended action. The unit tests are rigorously tested to ensure all exceptions are handled.

Server Side access is tested via a python web crawler. To ensure that ever page is reachable. This is ran once a week similarly to the client side tests. The web crawler also crawls through all available link in a demonstration and checks for broken links. Security also tested within the web crawler by crawling with authentication and without authentication. A predetermined set of pages can only be accessible with out authentication such as the landing page, login and registration.

The unit tests will be updated and modified as more development continues. In practice these unit tests should ensure that updates to the code base and other changes to the server end do not break the system.

3 System Tests

[Specific system tests. All tests should be fully summarized in terms of initial state, input and expected output. Tests should be named. In cases where there are many similar tests just summarize the results. Provide enough info that someone could reproduce your tests. Provide traceability to test plan by referring to test case numbers or modules. —CC]

4 System Tests

[Specific system tests. All tests should be fully summarized in terms of initial state, input and expected output. Tests should be named. In cases where there are many similar tests just summarize the results. Provide enough info that someone could reproduce your tests.

Provide traceability to test plan by referring to test case numbers or modules. —CC] In this section the test cases carried out on each individual module are described. Trivial cases for some modules are not explicitly written out but instead described at a high level. Additional details are provided when necessary.

4.1 Calendar

No.	Test Case	Initial State	Input	Expected Output	Actual Output	Result
3.1-3.4	Add event to Calendar.	Calendar page.	User selects date to add new event, enters information, clicks save.	Modal opens with fields, and closes upon save. Event is updated correctly on Calendar. The same output results if user selects existing event to modify.	As expected.	PASS
3.5	Delete event from Calendar	Calendar page.	User selects event to delete, clicks delete.	Modal opens with fields. Upon clicking delete, the modal closes and the event is removed from the Calendar.	As expected.	PASS

4.2 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

4.1	Quarters Web Application.	Open maintenance system.	Maintenance system opens and shows new maintenance tickets with existing tickets in chronological order.
4.2	Maintenance System.	Click on maintenance ticket.	Inner dialog opens displaying all properties in a maintenance ticket.
4.3	Maintenance System.	Entering a search query or adding a filter.	Sort and filter maintenance tickets and reveal only successful tickets.
4.4	Maintenance Ticket Window.	Modifying properties of a ticket.	Save icon appears in dialog to confirm changes.
4.5	Maintenance Ticket Window.	Saving ticket properties	Window will close, and database will be updated to reflect changes.
4.6	Maintenance Ticket Window.	Deleting Ticket.	Confirmation window will appear. Upon deletion confirmation, close window and remove data from database.
4.7	Maintenance System	Click on create new request.	Opens a new ticket window.
4.8	New Maintenance ticket window.	Click on create empty fields.	Window will remain opening, prompt will display error message.
4.9	New Maintenance ticket window.	Click on create, required fields filled.	Window closes, database will be updated with new ticket.
4.10	New Maintenance ticket window.	Click on cancel with fields filled.	Window remains open, prompt will ask for confirmation on close.
4.11	Confirmation Prompt.	Click on OK.	Closes prompt and dialog.
4.12	Confirmation Prompt.	Click on cancel.	Closes prompt, dialog remains open.

4.3 Landing Page

No.	**Test Case**	**Initial State**	**Input**	**Expected Output**	**Actual Output**	**Result**

6.1,6.2	Access login or registration.	Not logged in.	Clicks on login.	Modal opens and email and password fields appear. The same output results if user clicks on register.	As expected.	PASS

4.4 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
7.1	No payments due.	User initiates PayPal transaction.	Error message indicating that no payments are due at the current time.
7.2	No payments due.	User posts payment request with some deadline.	New payment deadline added.
7.3	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.
7.4	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.
7.5	Payment due.	User who posted the original due payment removes the request.	Payment request is removed from the list of due payments.

4.5 Notifications

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
8.1	Main page. User(A) logged in.	User(B) sends money request.	User(A) sees notification of pending payment due.
8.2	Main page. User(A) logged in.	User(A) pays user(B).	User(B) sees notification of payment completed.
8.3	Main page. User(A) logged in.	User(A) has late payment.	User(A) sees notification of late payment.
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.
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.
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.
8.7	Main page. User(A)(landlord) logged in.	User(B) sends maintenance ticket(Minor).	User(A) sees notification of unresolved maintenance ticket.
8.8	Main page. User(A) logged in.	User(B)(Landlord) resolves a maintenance ticket.	User(A) sees notification of resolved maintenance ticket.
8.9	Main page. User(A) logged in.	User(B) sends user(A) a message.	User(A) sees notification of unread message.

8.10	Main page. User(A) logged in.	User(B) makes a post in discussion board.	User(A) sees notification of unread post.
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.
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.
8.13	Main page. User(A) logged in.	User(B) adds event to Calendar.	User(A) sees notification of added post.
8.14	Main page. User(A) logged in.	User(B) deletes event from Calendar.	User(A) sees notification of deleted event.
8.15	Main page. User(A) logged in.	User(A)has event happening on day.	User(A) sees notification of event.
8.16	Main page. User(A) logged in. Notification displayed.	User clicks on Notification icon.	Notification disappears.

4.6 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
10.1	No posts on bulletin board.	A post with 0 characters	Empty post is disgarded ["discarded" —DS] and not added to bulletin board.
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.

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.
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.
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.
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.

5 Non-Functional Tests

[Nonfunctional qualities are evaluated as appropriate. These qualities include usability,performance, and robustness. Quantify results. If these tests are not performed, there absence should be explicitly justified. —CC]

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 Robustness

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]

6 Summary of Changes

[Summarize changes made in response to testing. —CC]