

**CEN 4010 - Software Engineering**

**Team 2**

**int elligence;**

**Clinton Benton, David Hughes, Elizabeth Moreno,**

**Michael O’Donnell, Tommy Owens, Aaron Ulmer**

Table of Contents

[**Overview** 3](#_Toc479427346)

[**Project Team** 3](#_Toc479427347)

[**Unit Test Approach** 4](#_Toc479427348)

[**Integration Test Approach** 5](#_Toc479427349)

[**System Test Cases** 6](#_Toc479427350)

[**Test Case 01 - Sign Up** 6](#_Toc479427351)

[**Test Case 02 - Login** 6](#_Toc479427352)

[**Test Case 03 - Logout** 7](#_Toc479427353)

[**Test Case 04 - Toggle between List and Card Property View** 7](#_Toc479427354)

[**Test Case 05 - Search Properties** 8](#_Toc479427355)

[**Test Case 06 - Filter Properties** 8](#_Toc479427356)

[**Test Case 07 - Add a Property** 9](#_Toc479427357)

[**Test Case 08 - Edit a Property** 10](#_Toc479427358)

[**Test Case 09 - Create a 3-Day Notice Document** 10](#_Toc479427359)

[**Test Case 10 - Create a 15-Day Notice Document** 11](#_Toc479427360)

[**Test Case 11 - Create a Lease Document** 12](#_Toc479427361)

[**Test Case 12 - Add a Charge** 12](#_Toc479427362)

[**Test Case 13 - Apply to a Property** 13](#_Toc479427363)

[**Test Case 14 - Make a Payment** 14](#_Toc479427364)

[**Test Case 15 - Send a Message** 14](#_Toc479427365)

[**Assumptions** 15](#_Toc479427366)

[**Risks** 15](#_Toc479427367)

# **Overview**

The goal of the project is to build an internet based program to handle property rentals and transactions between landlords and tenants, so that they could better handle the system without being so hands on or in person. This is done to simplify and quickly pass the transactions between the two. Through this, searching for properties, leasing, management, and payments can all be condensed to one web-based app where either party can handle their own responsibilities of the process.

The program needs to be able to handle basic tenant and landlord interactions. This includes searching for, applying to, leasing a property from a landlord, and payment management and communication; along with the landlord to be able to upload their properties and relevant documents, lease them and manage their fees and other matters. The program will contain a process for potential tenants to search and view available property within the system and message the landlord of that property for potential application proceedings. The applications process is sorted out by the landlord with the reviewing and signing of the legal documentation by the tenant. The landlord either keeps the documents on file within this system or external.

The addition functionality of the system needs to include the basics of a rating system for both the tenant and landlord. The purpose of the rating system is to create a validation system for landlords on tenants to avoid individuals who have a history of breaking contracts and for preventing slumlords.

The implied task of the system includes a communication system consisting of a basic instant messaging protocols, including notifications through the system and email. The customer is strongly requesting that it also be SMS compatible, aside from being through the system and notifications.

The program will contain a payment notification system and an annotation method for the tenant and landlord to validate that the payment has been made in the appropriate amount for that month. The validation system is due to tenants paying part in cash and part in the system or to check if proper amounts. The payment system will be capable of being updated to handle the current system alongside additional features to the payment system at a later date. This process needs to be intuitive so that any user can gain a quick visual status of their property’s upcoming fees.

# **Project Team**

|  |  |
| --- | --- |
| Clinton Benton | Software Engineer/Designer. Test Cases, Table of Contents |
| David Hughes | Database Management/Software Engineer. Risks and Assumptions |
| Elizabeth Moreno | Lead Designer/Software Engineer. Integration Test Approach |
| Michael O’Donnell | Team Leader/Software Engineer. Unit Test Approach, Test Cases |
| Tommy Owens | Software Engineer/Designer. Test Cases |
| Aaron Ulmer | Software Engineer/Designer. Test Cases |

# **Unit Test Approach**

Our project will be implementing both visual code reviews (code inspections and walkthroughs) and formal testing of the code via statement and branch testing to maximize the number of faults discovered while working on said code or after development for a piece has been completed. All the code will firstly be tested by the original developer to debug the code as it is being written and once done, to flesh out any problems they may see or had at the beginning of the process. This will involve statement and branch testing in order to test every decision (branch testing checks the performance of every path of a decision in the code, both TRUE and FALSE statements and the preceding code) and line (statement testing checks only the statements that are TRUE but also ensures that every line of code is executed and tested) in the code to ensure that all parts of the code function properly and to effectively find any faults. Once completed, the code will be brought to the group’s attention for further inspection. These inspections will have the focus of having more faults discovered with the idea that more people present could result in one of them catching something the original developer did not the first time. After which, the piece of code will go through group tests before and during integration.

When pieces of the project are being brought up for group inspection, the original developer of the piece will be explaining what the goal of the code is, allowing for the group to prepare and have a basic understanding of what is going. It is then that the original developer will be going through in detail what each line/segment of code does. During this process, other members of the group will be able to ask questions and get an in depth look at what is happening, hopefully allowing for faults to be discovered by someone else, or perhaps discover a better/more efficient way to get something done.

Note: Since all members of the group are already familiar with the product and goals of tasks (due to constant meetings), effectiveness of this procedure is increased.

HTML and Design - Tests and Inspections: The Design of the application will be done individually so it is in unison throughout the application, and then brought to the other members of the group in order that the look and semi-functionality of the design is brought to their knowledge, approved, and so that they know in general what they will be working with. The wireframing and HTML will be designed and set up next based off the original application design sketch. This will once again be brought to the group to be approved before being passed on to other members for the other parts of the code.

PHP/Database - Tests and Inspections: While writing and finishing the backend portion of the project, which includes both the PHP functionality and database queries and managements, the individual members working on it will be testing it throughout the writing process and again once finishing the code itself. Once done, it will be brought to the rest of the group for further intensive inspection (as mentioned previously, additional people could likely increase the effectiveness of discovering additional faults).

# **Integration Test Approach**

Integration testing occurs at each stage of the development process using the modified sandwich approach as discussed in class. This process will be done with separate portions of the project being done and tested alone, and then tested again once integrated together with other high-level pieces that have also been tested and then merged together. Through this process, the best amount of possible faults that can occur will be discovered and handled moving forward to ensure proper execution at any given time of the development process.

This process is facilitated through functional tests performed by the members of the group before and after development of pieces of code, and again after being merged with other subsystems.

* The web based portion of the tests will be tracked through a shared directory /Integration/WebBasedTests, which will contain a list of expected outputs of screens of the application on a browser and its correct content through functionality.
* Within this directory, each test case will revolve around each screen and specific functionality belonging to it. The test case will contain a description of what is being tested, any input, expected output, and tested output.
* These tests use a white box approach to test the Web systems at a high-level. This typically involves GUI input and complex web-based output.
* Each test of the screen and its functionality and content will be done by the group as to get full approval of the functions being performed through the screen input/output.
* For the tests that are not entirely web-based output, they will be kept in a subdirectory /Integration/WebBasedTests/BackendTests/, to keep the portions of tests that rely more on the backend of the project separate and organized.
* These tests will undergo the same treatment as /WebBasedTests, so test case information is the same. Approval will be done by the original developer and documented as to not stop production for every piece of functionality added. Only /WebBasedTests will undergo group approval tests to ensure full project functionality is correct when merged.
* In some cases, errors will need to be forced to check proper handling, this will be also documented inside test case information, detailing what was done to support the test

# **System Test Cases**

## **Test Case 01 - Sign Up**

|  |  |
| --- | --- |
| Preconditions/Setup | Tester must be on the landing page |
| Requirement(s) Tested | Sign Up |
| Action 1 | The tester clicks on the sign up button |
| Verify 1 | The tester confirms that the sign-up module appears on the content pane |
| Action 2 | Tester fills out the sign-up form |
| Verify 2 | The Tester confirms that the sign-up form accepts all information entered correctly |
| Action 3 | The tester clicks the sign-up button at the bottom of the module |
| Verify 3 | The tester confirms they were taken back to the landing page with a message informing they were signed up |

## **Test Case 02 - Login**

|  |  |
| --- | --- |
| Preconditions/Setup | Tester must be signed up and be on the landing page |
| Requirement(s) Tested | Login/Logout |
| Action 1 | The tester clicks on the login button |
| Verify 1 | The tester confirms the login modal appears on the content pane |
| Action 2 | The tester fills in the form with email and password |
| Verify 2 | The tester confirms that the information is entered and password is hidden |
| Action 3 | The tester clicks the login button at the bottom of the modal |
| Verify 3 | The tester confirms they are now logged into their account and it appears on the content pane |

## **Test Case 03 - Logout**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester must be signed up, and be logged into their account |
| Requirement(s) Tested | Login/Logout |
| Action 1 | The tester navigates to and clicks the logout button |
| Verify 1 | The tester confirms through the content pane they were logged out and taken to the landing page |

## **Test Case 04 - Toggle between List and Card Property View**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester must be logged in. If they are logged in as a Landlord, they must be on the homepage. If they are logged in as a tenant they must be on the property search page |
| Requirement(s) Tested | Toggle between ‘List’ and ‘Card’ view of properties |
| Action 1 | The tester navigates to the toggle buttons and clicks the button that is not currently active. |
| Verify 1 | The tester confirms the toggle button pressed is active. The display view of properties then re-renders according to the option selected |
| Action 2 | The tester repeats the first action with the button that is now not currently active |
| Verify 2 | The tester confirms the toggle button pressed is now active. The properties should re-render according to the initial display view |

## **Test Case 05 - Search Properties**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester must be logged in as a tenant and on the homepage |
| Requirement(s) Tested | Search Properties |
| Action 1 | The tester navigates from the homepage to the property search page through the tile button at the top |
| Verify 1 | The tester confirms the property search page has loaded onto the content pane |
| Action 2 | The tester fills in the search bar with their search criteria |
| Verify 2 | The tester confirms that the searched material displayed on the content pane relates to the search criteria |
| Action 3 | The tester fills in the search bar again with different criteria (that will cause no properties to be found) |
| Verify 3 | The tester confirms that no properties were displayed on the content pane relating to the searched criteria, only the message “No Properties Found” is displayed |

## **Test Case 06 - Filter Properties**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester must be logged in. If they are logged in as a Landlord, they must be on the homepage. If they are logged in as a tenant they must be on the property search page |
| Requirement(s) Tested | Filter properties |
| Action 1 | The tester navigates to the ‘Options’ line and clicks the ‘+’ button |
| Verify 1 | The tester confirms a form modal pops up with a drop-down list of filter options |
| Action 2 | The tester selects a filter option from the drop-down list |
| Verify 2 | The tester confirms a second input field appears that provides the appropriate input for the filter selected |
| Action 3 | The tester inputs an appropriate value for the filter selected and clicks the submit button |
| Verify 3 | The tester confirms a display view of properties re-renders and only includes those properties matching the condition specified by the filter criteria. |

## **Test Case 07 - Add a Property**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester must be signed in as a landlord and on the homepage |
| Requirement(s) Tested | Add a Property |
| Action 1 | The tester navigates to and clicks the ‘Add Property’ tile button |
| Verify 1 | The tester confirms a form modal appears on the content pane and displays input fields to enter property related information |
| Action 2 | The tester fills out the form appropriately and presses the submit button |
| Verify 2 | The tester confirms a confirmation message appears and the tester is then redirected to the property’s detail page that correctly displays the information they entered in the form |
| Action 3 | The tester returns to the form model (Setup and Action 1) and fills out the form modal incorrectly and presses the submit button |
| Verify 3 | The tester confirms a message appears informing that the form was incorrectly filled out and returned to the modal |

## **Test Case 08 - Edit a Property**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in as a landlord and be on the detail page for a given property |
| Requirement(s) Tested | Edit a Property |
| Action 1 | The tester navigates to the detail tile and clicks on the pencil icon |
| Verify 1 | The tester confirms a form modal pops up and displays input fields that are pre-populated with the information related to the current property |
| Action 2 | The tester makes one or more changes to the input fields and clicks the submit button |
| Verify 2 | The tester confirms the property detail page reloads and the tile information reflecting the changes made by the tester |

## **Test Case 09 - Create a 3-Day Notice Document**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in as a landlord and be on the detail page for a given property |
| Requirement(s) Tested | Create a 3-Day Notice Document |
| Action 1 | The tester navigates to the ‘Documents’ section and clicks the button labeled ‘Create Documents’ |
| Verify 1 | The tester confirms a modal pops up displaying the possible documents to create |
| Action 2 | The tester clicks on the option selected ‘3-Day Notice’ |
| Verify 2 | The tester confirms a form appearing displaying input fields corresponding to the information needed to generate a ‘3-Day Notice’ document. |
| Action 3 | The tester fills out the form appropriately and clicks the submit button |
| Verify 3 | The tester confirms a new tab window opening displaying a 3-Day Notice PDF document with the information entered in the form |
| Action 4 | The tester returns to the form modal (Setup and Action 1) and fills out the form incorrectly and clicks the submit button |
| Verify 4 | The tester confirms a message appears on the content pane informing the form was incorrectly filled out and is redirected back to the modal |

## **Test Case 10 - Create a 15-Day Notice Document**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in as a landlord and be on the detail page for a given property |
| Requirement(s) Tested | Create a 15-Day Notice Document |
| Action 1 | The tester navigates to the ‘Documents’ section and clicks the button labeled ‘Create Documents’ |
| Verify 1 | The tester confirms a modal popping up displaying the possible documents to create |
| Action 2 | The tester clicks on the option selected ‘15-Day Notice’ |
| Verify 2 | The tester confirms a form appearing displaying input fields corresponding to the information needed to generate a ‘15-Day Notice’ document. |
| Action 3 | The tester fills out the form appropriately and clicks the submit button |
| Verify 3 | The tester confirms a new tab window should open displaying a 15-Day Notice PDF document with the information entered in the form |
| Action 4 | The tester returns to the form modal (Setup and Action 1) and fills out the form incorrectly and clicks the submit button |
| Verify 4 | The tester confirms a message appears on the content pane informing the form was incorrectly filled out and is redirected back to the modal |

## **Test Case 11 - Create a Lease Document**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in as a landlord and be on the detail page for a given property |
| Requirement(s) Tested | Create a Lease Document |
| Action 1 | The tester navigates to the ‘Documents’ section and clicks the button labeled ‘Create Documents’ |
| Verify 1 | The tester confirms a modal popping up displaying the possible documents to create |
| Action 2 | The tester clicks on the option selected ‘Lease’ |
| Verify 2 | The tester confirms a form appearing displaying input fields corresponding to the information needed to generate a ‘Lease’ document. |
| Action 3 | The tester fills out the form appropriately and clicks the submit button |
| Verify 3 | The tester confirms a new tab window opening displaying a Lease PDF document with the information entered in the form |
| Action 4 | The tester returns to the form modal (Setup and Action 1) and fills out the form incorrectly and clicks the submit button |
| Verify 4 | The tester confirms a message appears on the content pane informing the form was incorrectly filled out and is redirected back to the modal |

## **Test Case 12 - Add a Charge**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in as a landlord and be on the detail page for a given property |
| Requirement(s) Tested | Add a Charge |
| Action 1 | The tester navigates to the ‘Finances’ section and clicks the button labeled ‘Add Entry’ |
| Verify 1 | The tester confirms a modal pops up displaying two radio buttons labeled ‘Payment’ and ‘Charge’ |
| Action 2 | The tester selects the button labeled ‘Charge’ |
| Verify 2 | The tester confirms a form appears under the radio buttons containing input fields corresponding to a charge entry |
| Action 3 | The tester returns to the form modal (Setup and Action 1) and fills out the form incorrectly and clicks the submit button |
| Verify 3 | The tester confirms the property detail page reloads and the table under the ‘Finances’ section should include a new row reflecting the new charge with the correct information entered in the form |

## **Test Case 13 - Apply to a Property**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in as a tenant and be on the property search page |
| Requirement(s) Tested | Apply to a Property |
| Action 1 | The tester clicks on the button labeled ‘Apply’ for a specific property |
| Verify 1 | The tester confirms a modal pops up displaying a form containing input fields for the property application |
| Action 2 | The tester fills out the form appropriately and clicks the submit button |
| Verify 2 | The tester confirms a confirmation message appears and that they are redirected back to the property search page |
| Action 3 | The tester returns to the form modal (Setup and Action 1) and fills out the form incorrectly and clicks the submit button |
| Verify 3 | The tester confirms a message appears on the content pane informing the form was incorrectly filled out and is redirected back to the modal |

## **Test Case 14 - Make a Payment**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in as a tenant and be on the homepage |
| Requirement(s) Tested | Make a Payment |
| Action 1 | The tester navigates to the ‘Finances’ section and clicks the button labeled ‘Make Payment’ |
| Verify 1 | The tester confirms a modal pops up displaying a form containing input fields to make a payment |
| Action 2 | The tester returns to the form modal (Setup and Action 1) and fills out the form incorrectly and clicks the submit button |
| Verify 2 | The tester confirms the property detail page reloads and the table under the ‘Finances’ section should include a new row reflecting the new payment with the correct information entered in the form |

## **Test Case 15 - Send a Message**

|  |  |
| --- | --- |
| Preconditions/Setup | The tester should be logged in and be on the notifications page |
| Requirement(s) Tested | Send a Message |
| Action 1 | The tester clicks the button labeled ‘New Message’ |
| Verify 1 | The tester confirms a modal pops up displaying a form containing input fields to create a message |
| Action 2 | The tester fills out the form by selecting a recipient, writing a message, and clicking the submit button |
| Verify 2 | The tester confirms a confirmation message appears informing them the message was sent and that they are redirected back to the notifications page |

# **Assumptions**

* We assume each client device will be accessing our application using a supported web browser. We plan on supporting the latest stable versions of Chrome, Safari, Internet Explorer, and Edge. We have chosen these specific browsers because they are the defaults for most mobile devices and PCs/laptops. Chrome is the default browser on Android-based phones/tablets and Chrome OS devices. Safari is the default browser on iPhones/iPads and Mac OS devices. Internet Explorer and Edge are the defaults on Windows PCs and Microsoft phones/tablets respectively. Therefore the application will be accessible to all users regardless of device being used.
* We assume each client will have internet access on their device to connect to our application.
* We assume the company hosting MySQL will not let the database go down.
* We assume that this product will be free to use due to the nature of the of the API’s used to support the application.

# **Risks**

* Due to two members of our team being in the process of learning PHP throughout the development and testing process, this can lead to a challenge in getting portions of the project done in time. Group meetings and other members of the team will be readily available to answer questions to any issues with PHP and overcome this risk in development.
* At this point in the project, the main obstacle to the completing the project is the amount of available time each member of the team. All members of the group have either part-time/full-time jobs with two courses or are full time students and can pose a risk to development and testing time.