*Daniel Swain*

*PropertyDome | Perth, Western Australia*

*Final Project*

Assessment Four

Table of Contents

[Introduction 3](#_Toc453348855)

[Project Plan 3](#_Toc453348856)

[Collaboration 3](#_Toc453348857)

[*Collaboration Stats:* 3](#_Toc453348858)

[Coding Standards: 6](#_Toc453348859)

[Testing: 10](#_Toc453348860)

[*Test 1: User Login* 10](#_Toc453348861)

[*Test 2: Suburb/Postcode invalidation on Sell Page* 15](#_Toc453348862)

[Conclusion: 20](#_Toc453348863)

# **Introduction**

The following document outlines the final project project completion. It details the testing criteria as well as demonstrating compliance with our collaboration tools and coding standards.

# **Project Plan**

Please see the included Final Project Plan for an update of the project status, including final milestones, design decisions and functional requirements. Since the last project update on 21/04/2016 we have completed 2 additional sprints and have finished the development and delivery of the website in accordance with the functional requirements as listed in the Project Plan.

# **Collaboration**

For collaboration we are still using Trello, Slack and GitHub. We have found Slack to be the most used tool for discussion, with GitHub being useful for highlighting issues, bugs and feature requests. In the later sprints we started to use GitHub issues instead of Trello cards for tracking tasks for sprints as this integrated better with GitHub. For example, a user could reference their story/issue in their commit to track/tie the commit to that issue, making it easier to track.

## *Collaboration Stats:*

Commits to Project: 1034.

Git Issues: 1 Open (v2 feature list), 46 Closed (testing/bug reporting).

Pull Requests: 0 Open, 228 Closed.

Commits to Master: 632

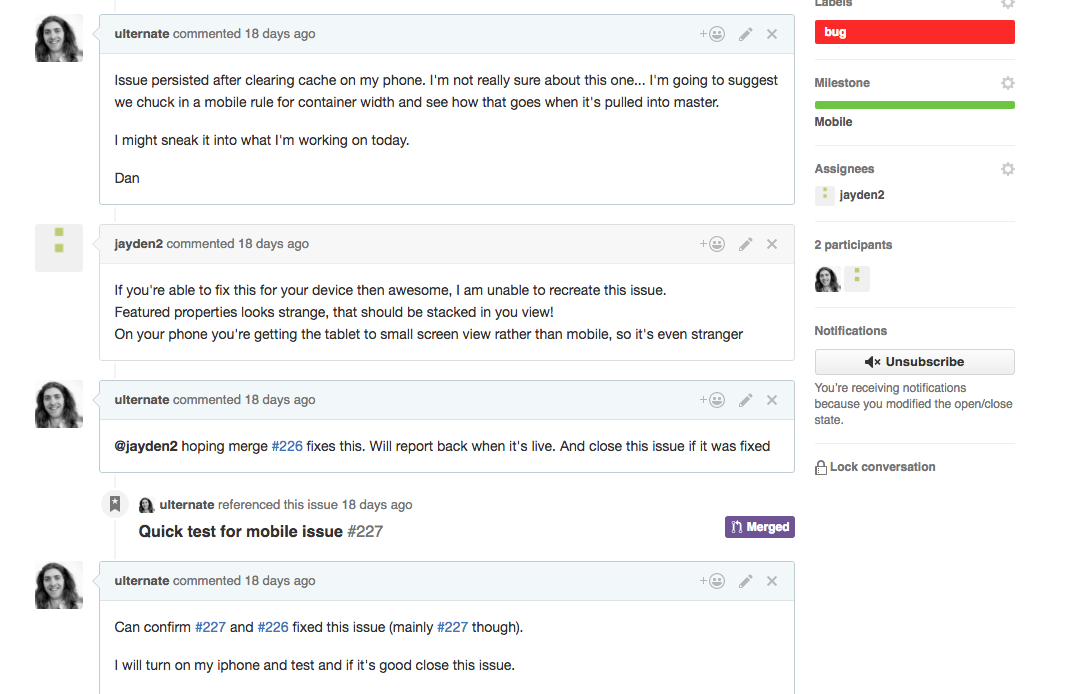
Team Breakdown (commits to master, doesn’t include pull requests or issues):

Daniel: 230 Jayden: 175 Simon: 99 Michael: 90 Erdal: 35 Tylden: 3

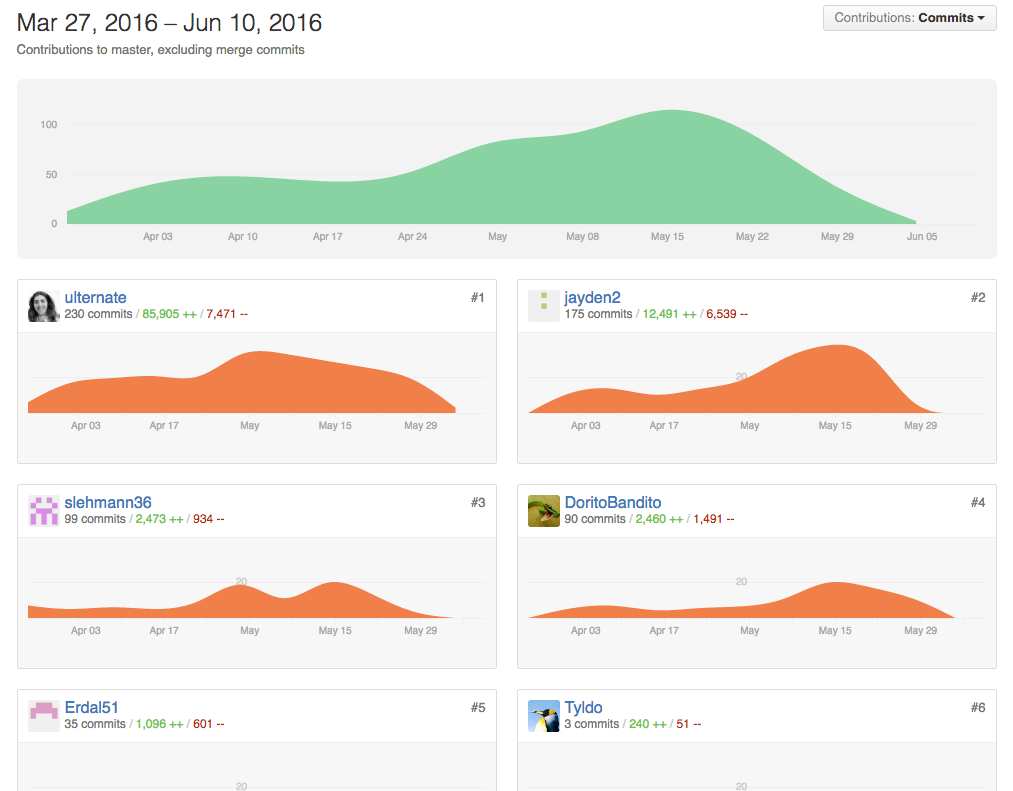
Interesting tidbit: we managed to get a commit to the project during one of all 24 hours of the day, with our biggest times being 1pm, 3pm, 11pm and 1am.



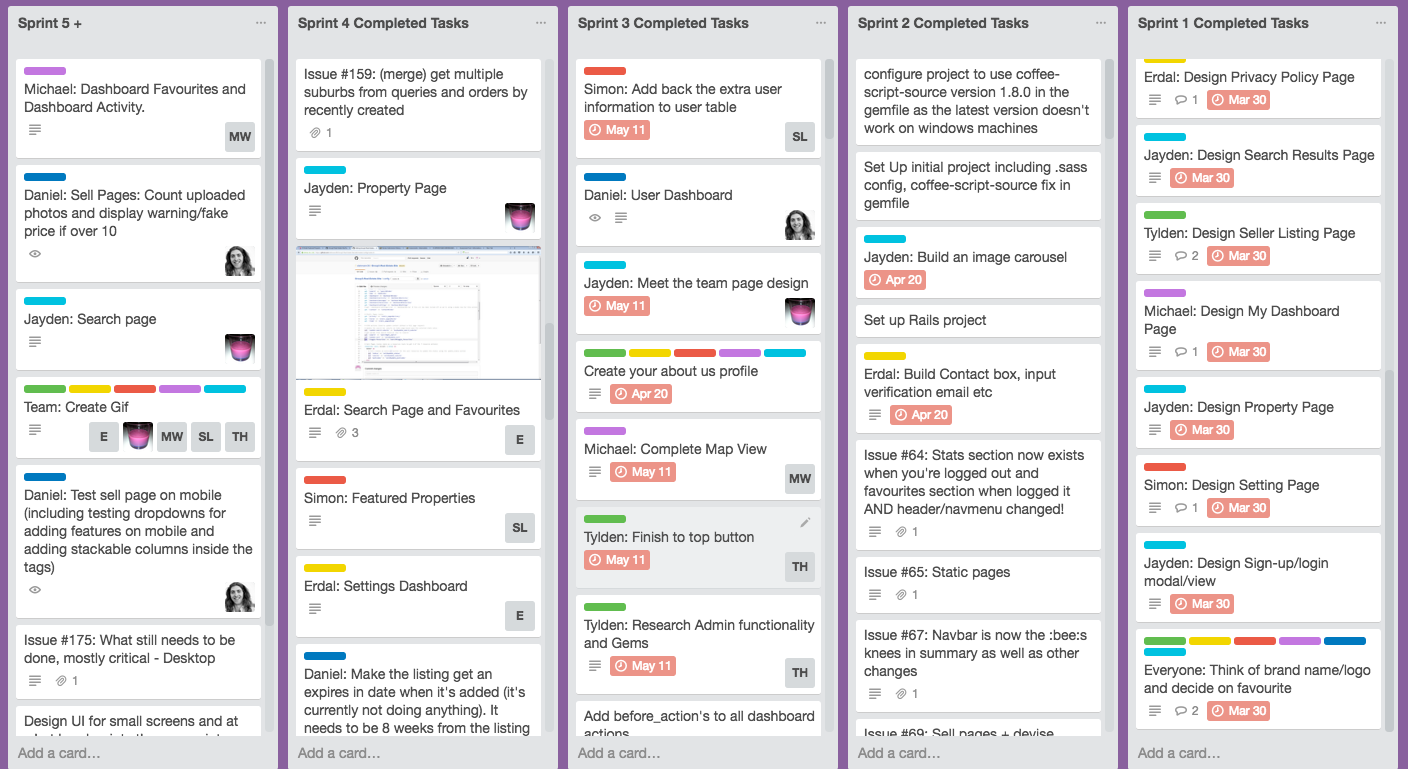
The following are examples of the collaboration tools being used:



GitHub Issues being used to pinpoint bugs/issues in site features.



Screenshot of the project on GitHub, showing commits to master from team members. Apart from a few outliers, contribution to the project was fairly even and consistent from all members.



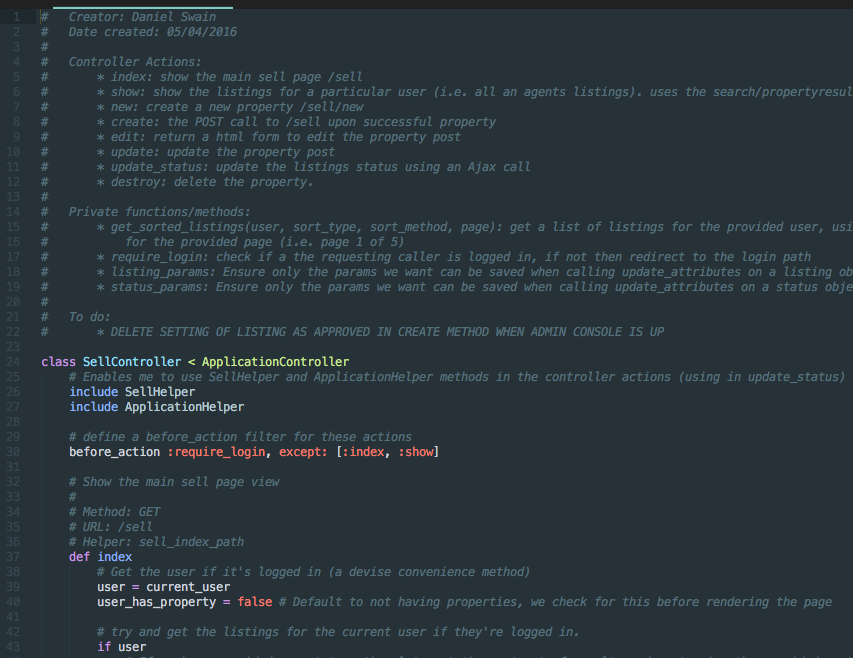
Screenshot showing Trello tracking completed sprint tasks and issues.

# **Coding Standards:**

The coding standards we implemented can be found here: <https://drive.google.com/open?id=18F52kEOdb0eG1TtQPi_kskgJIjToppe2-23_J8VWmzw>

The following images demonstrate these coding standards being followed in various languages:

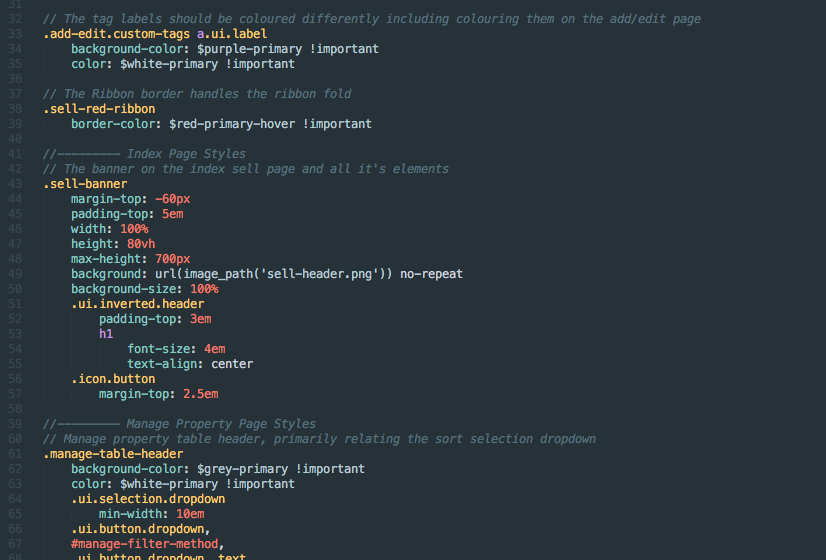
Image 1 – sell\_controller.rb:



Notes:

* Documented with creator, date, action explanation and to-do
* Aligned in accordance with coding standard
* Variables named in accordance with coding standard (snake\_case)
* NB (the database column names were created using CamelCase)

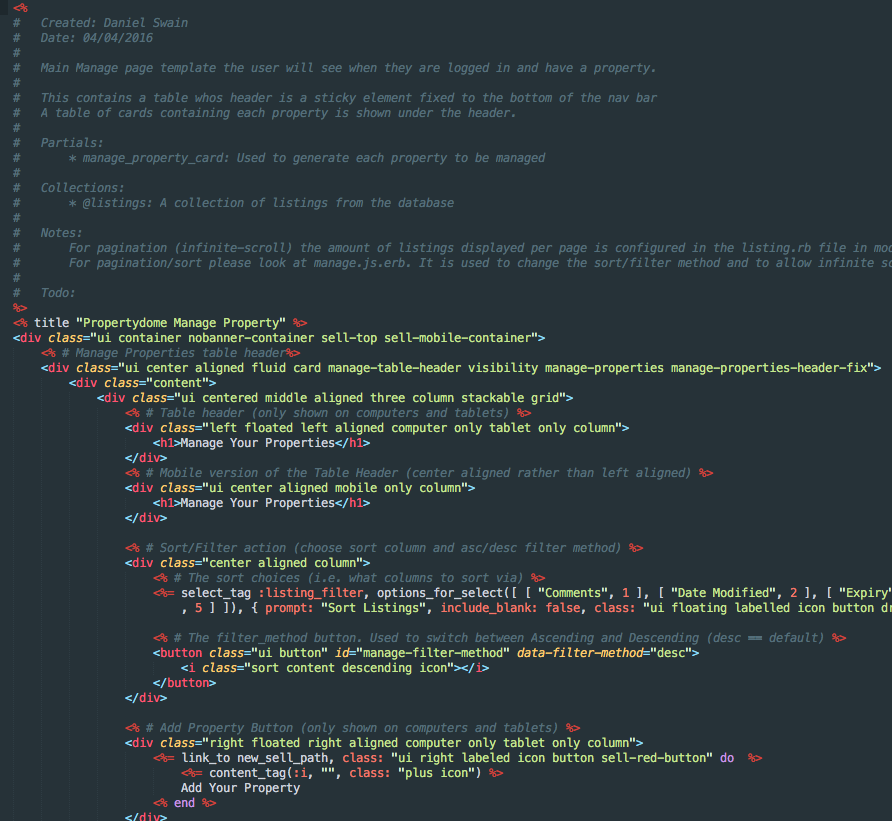
Image 2 – \_sell.sass:



Notes:

* Documented with explanations for style definitions
* Aligned in accordance with coding standard
* Using correct variables for global variables like colours (colour-shade-action)
* Using ‘em’s as much as possible (unavoidable for the banner height and top margin)

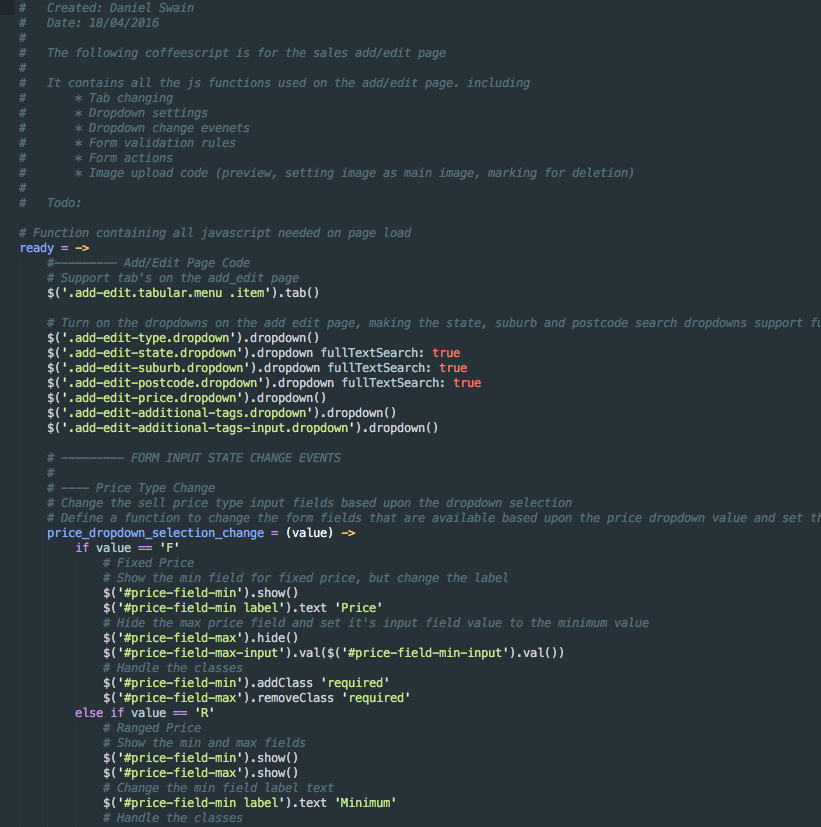
Image 3 – manage.html.erb:



Notes:

* Documented with creator, date, partials/dependencies, explanations and to-do
* Aligned in accordance with coding standard
* Using snake\_case for ruby variables and hyphen for html id’s/classes

Image 4 – add\_edit.coffee:



Notes:

* Documented with creator, date, explanations and to-do
* Aligned in accordance with coding standard
* Single quotes used for variable/JQuery selectors

# **Testing:**

Testing for the project was a continuous process as evidenced by our 46 closed issues on GitHub. These issues ranged from JavaScript functions firing twice on page load ([link](https://github.com/slehmann36/Group3-Real-Estate-Site/issues/156)), to browser specific issues for browsers such as Safari or Firefox ([Safari issue](https://github.com/slehmann36/Group3-Real-Estate-Site/issues/120), [Firefox issue](https://github.com/slehmann36/Group3-Real-Estate-Site/issues/138)).

In addition to the continuous testing completed throughout the development of the project we also completed discrete tests for individual functions and functionality. These tests helped validate our assumptions and confirm that the functionality being added was robust.

The following are two such tests I completed during development.

## *Test 1: User Login*

The user login functionality is completed using the Devise gem. This gem handles the authentication checking (validating the username and password with the database records) as well as producing error messages for the user.

I developed the user login modal that is accessed from the navigation bar and connected it to the Devise gem using the recommended Devise form. To confirm the user modal works as expected these are the following test cases I put it through, as well as the expected results.

The account used for the following tests was. email: [nic2015@email.com](mailto:nic2015@email.com) and password: L3tsBu1ld

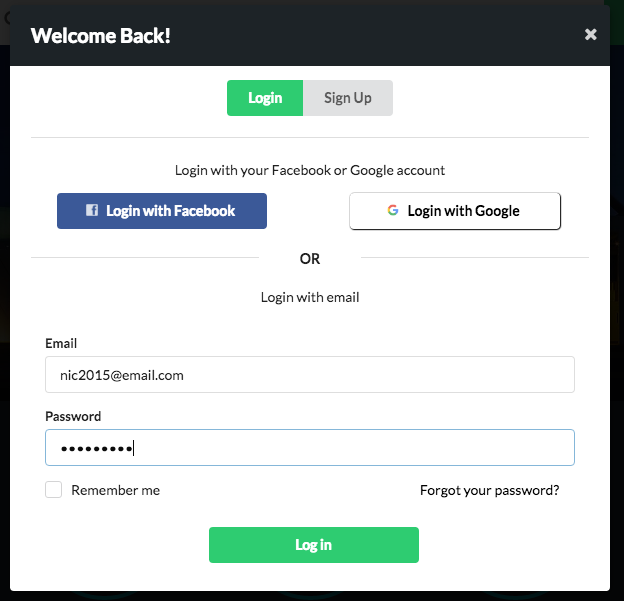
|  |  |  |  |
| --- | --- | --- | --- |
| # | Task | Expected Result | Successful |
| 1 | Log in using correct user details (email + password) | * This should log the user in; * Update the nav menu with their email address/username; * Show “My Dashboard” instead of “Contact Us” in the nav menu; and * Redirect the user to the previous page they were on when they clicked log in. | * Y * Y * Y * Y |
| 2 | Log in using incorrect email address (correct password) | * This should redirect user to root\_url/login; * Show the following error: “Incorrect email address or password.”; * The password field should be cleared; and * When successfully logged in it should redirect the user to root\_url/buy. | * Y * Y * Y * Y |
| 3 | Log in using incorrect password (correct email address) | * This test has the same expected outcomes as test 2. | * Y |
| 4 | Log in using incorrect email and password | * This test has the same expected outcomes as test 2. | * Y |

Test methods and passing criteria for the Login functionality

**Screenshots for Test 1:**



Nav menu before logging in showing “Contact Us” and the login/join buttons that launch the modal.

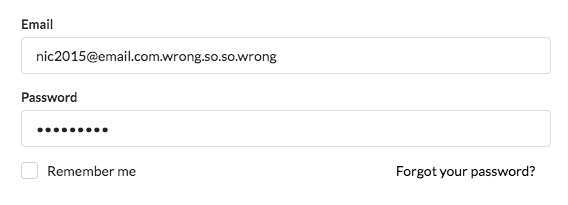


Modal with correct information prior to selecting Log in.

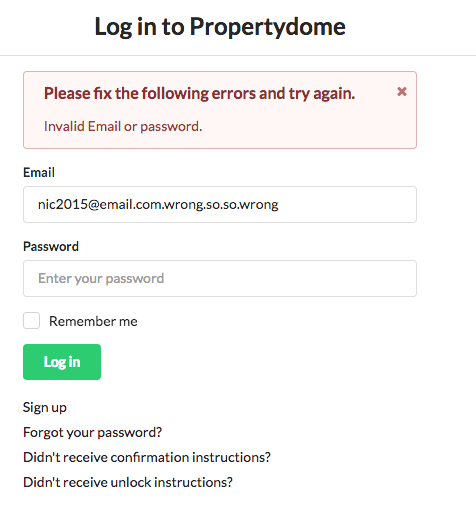


Nav menu showing MyActivity in the menu, and the user’s username showing

**Screenshots for Test 2:**

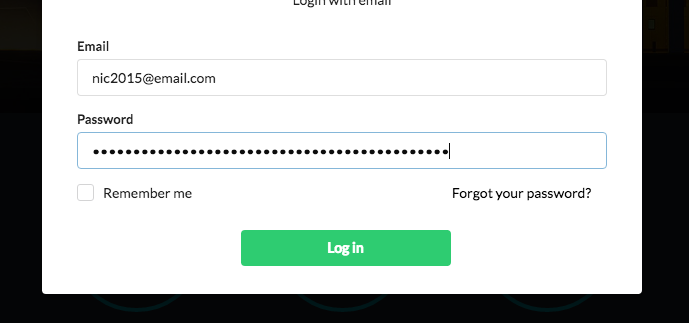


The incorrect email address used. This should redirect to the /login/ url with an error

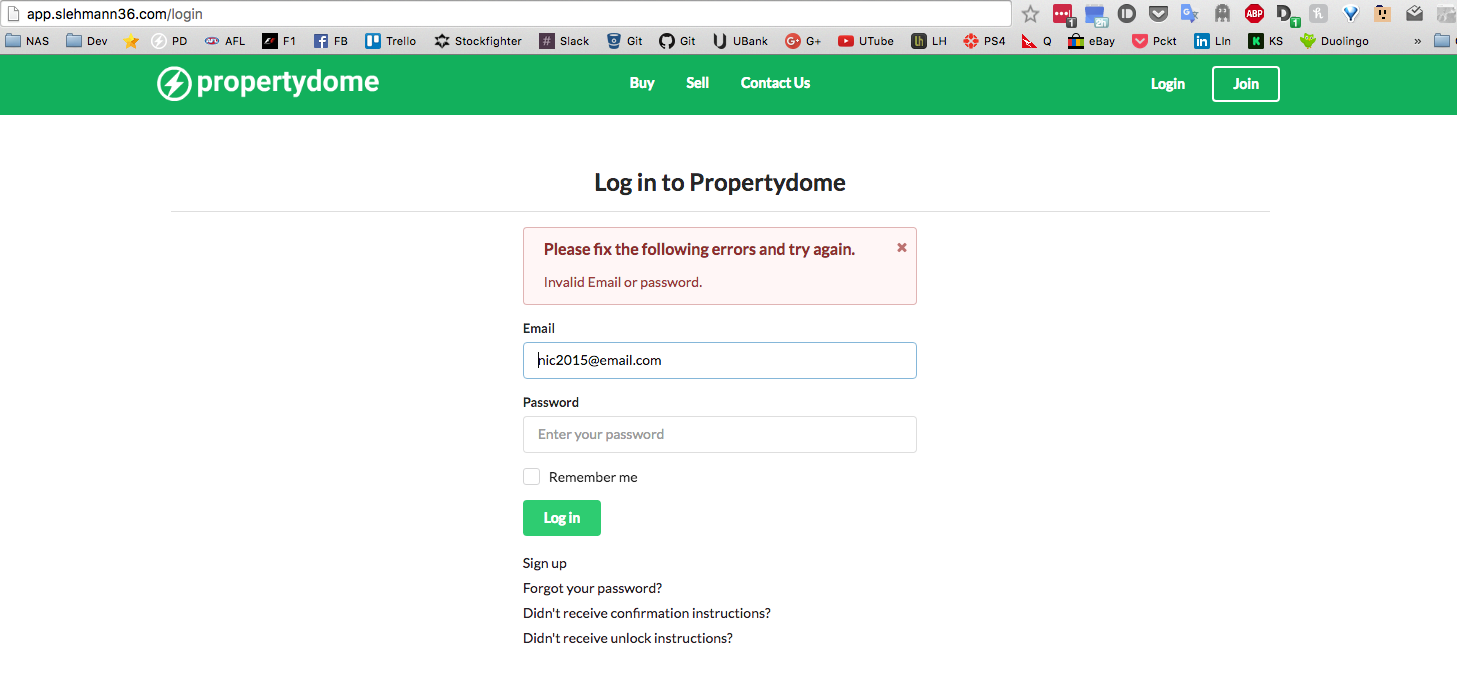


This is from the root\_url/login/ page and it has kept the old email and shown the error to the user. Entering the correct details redirected back to /buy as expected.

**Screenshots for Test 3:**

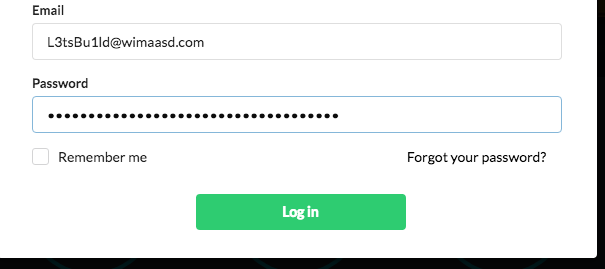


This is the incorrect password, but correct email test.

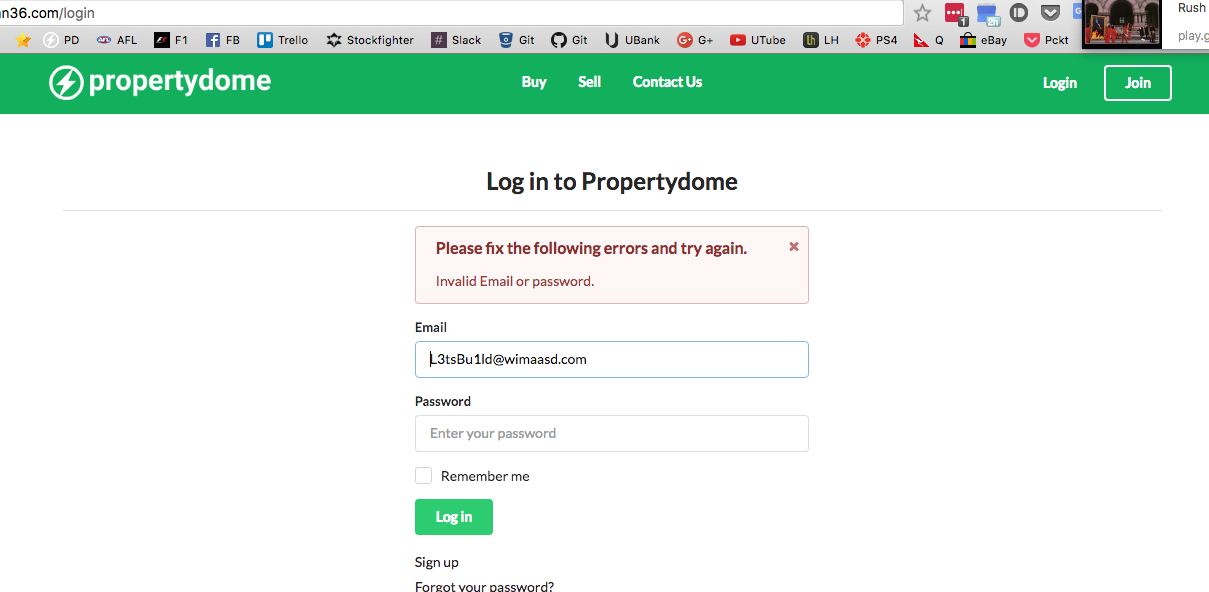


Again, redirected to root\_url/login with the error message. Entering the correct details redirected back to /buy as expected.

**Screenshots for Test 4:**



Incorrect email and password test.



Again, we are redirected to the root\_url/login page with the error message shown. Entering the correct details redirected back to /buy as expected.

The site successfully passed all four tests of the Login action as built using Devise and both a modal and login page. The benefit of testing this way also verifies that the login page works (if a user was to login without using the modal), as all 3 error tests test what would happen when the user logs in from the root\_url/login page to confirm they get redirected back to the /buy page.

## *Test 2: Suburb/Postcode invalidation on Sell Page*

A useful functionality on our website when a user is adding or editing a property listing is the ability to select the suburb and postcode for the property from a large database of choices. This helps ensure the integrity of the listing data as well as provides consistency for our search functions as we don’t need to cover for Perth, perth, Perth (WA) and perth (WA).

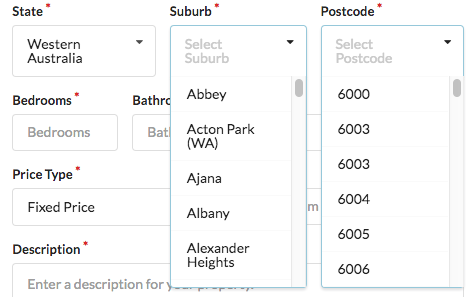
I built a JavaScript function that fires when the user selects a state in the Add/Edit page. When a state is chosen from the state dropdown the suburb and postcode choice lists are invalidated and an Ajax call is made to the server to get the list of suburbs and postcodes for that state. When this call succeeds the suburb and postcode choices are filled with these new values, ensuring only choices within the chosen state can be made.

This function is a critical feature as it provides a front end check for the data integrity of the listing before it is sent to the server. As such I performed the following test scenarios against it and reviewed the expected results against the performant results.

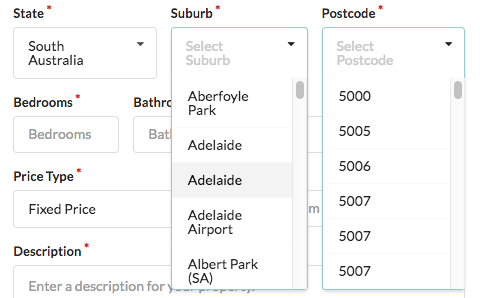
|  |  |  |  |
| --- | --- | --- | --- |
| # | Task | Expected Result | Successful |
| 1 | On the add view, select a state | * This should clear any selections from the Suburb and postcode field; * Change the suburbs to suburbs from the chosen state; and * Change the postcodes to postcodes from the chosen state. | * Y * Y * Y |
| 2 | On the edit view, change a state with data already present in the suburbs/postcode fields | * This test has the same expected outcomes as in test 1. | * Y |
| 3 | On the add view, change the suburb selection after a state has been chosen. | * This should clear any selections from the postcode field; and * Change the postcodes field choices to postcodes that match the chosen state and suburb. | * Y * Y |
| 4 | On the edit view, change the suburb selection after a state has been chosen | * This test has the same expected outcomes as in test 3. |  |

Test methods and passing criteria for the Suburb/Postcode invalidation functionality

**Screenshots for Test 1:**

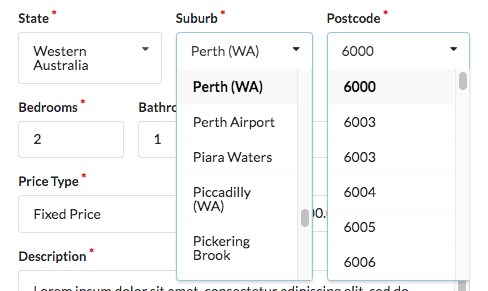


This image shows the field choices available to the user before the state selection.

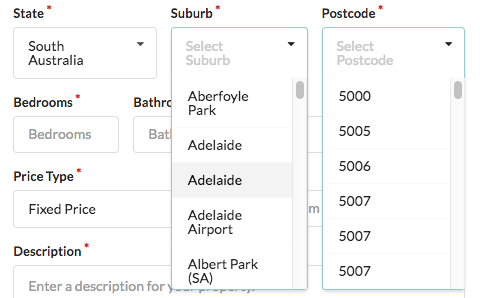


This image shows that the suburbs and postcode choices has changed to match the chosen state.

**Screenshots for Test 2:**

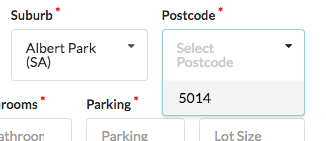


This image shows the field choices available to the user before the change of state.

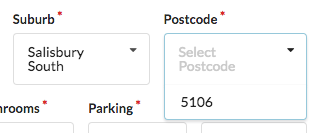


This image shows that the suburbs and postcode choices have changed to match the chosen state.

**Screenshots for Test 3:**

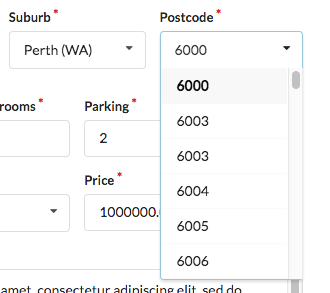


This image shows the field choices available to the user before the change of suburb.

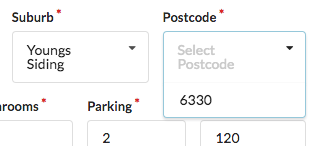


This image shows that the postcode choices have changed to match those that are valid for the chosen state and suburb.

**Screenshots for Test 4:**



This image shows the field choices available to the user before the change of suburb.



This image shows that the postcode choices have changed to match those that are valid for the chosen state and suburb.

The site successfully passed all four tests of the Suburb/Postcode invalidation action as built for both the add and edit views on the sell listing page. This is an important front end check to help ensure that we’re getting data for a listing that is consistent as this reduces the complexity of the search methods for the website.

# **Conclusion:**

Throughout this project all team members performed continuous testing of their code and modules. Team members also collaborated extensively through the chosen project tools. This helped to smooth development tasks and fostered an excellent team environment which improved the overall productivity.

I feel that we were able to complete the project tasks to the level of polish that we have is due to team members following the project practices and procedures as they were set at the beginning of the project. However, we were also able gain the benefits of an agile project management methodology by updating and tweaking our procedures and standards throughout the lifecycle of the project.