System and Unit-Test Report

Team/Product Name: Rent’s Planet

Date: 11/23/2016

Final System Tests (manual semi-unit-testing):

Testing SQL Database:

Pre-condition: No rows in database exist with address: ‘1 address’

* On Server In SQL console
  + List ‘Houses’ table rows
  + Verify row containing address: ‘1 address’ does not exist
* Insert new entry
  + address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’, review: ‘review’
* List table rows
* Verify new entry in ‘Houses’ table containing the above information
  + address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’, review: ‘review’
* Delete this row
* Verify the row is deleted

Testing Backend REST Service with Database: view-table, view-one, create, delete

Pre-condition: No rows in database exist with address: ‘1 address’

* Using xmlhttp requesting program access REST service (<http://rentsplanet.com/RentsPlanet-1.0-SNAPSHOT/rest/>)
  + GET all reviews in database by /display REST extension
  + Verify table does not contain row with ‘1 address’
  + POST a review to /create REST extension
    - address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’, review: ‘review’
  + GET all reviews in database by /display REST extension
  + Verify table contains the added row
    - address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’, review: ‘review’
  + GET ‘1 address’ review by /entry REST extension
    - ?address=1%20address
  + Verify the added row is returned
  + DELETE this review by /delete REST extension
    - address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’, review: ‘review’
  + GET all reviews in database by /display REST extension
  + Verify table does not contain row with ‘1 address’

Testing Front-End View: prior-to-search, after-search, create review, edit, delete

* Verify only map and search bar are visible, and map is expanded to fit the window
* Search an address
  + Verify map shrinks and the space is allocated to scraped data
  + Verify creation form (button) is visible and list of reviews is visible
  + After a moment
    - Verify scraped data and address’ reviews are retrieved and visible
* Create a review and verify that it is added to the list of reviews
* In the reviews list, choose a review to edit
  + Verify that editing a field works, but doesn’t change review when cancelled
  + Verify that deleting the review removes it from the list of reviews

Verification of each internal method of the vue objects was done by inserting logging calls into each method and manually comparing input and expected output.

Testing Front-End API Calls: create, delete, view-one

Pre-condition: row with ‘1 address’ does not exist in database

* get\_housing\_data(‘1 address’)
  + Log output to console with a 1 second delay
  + Verify empty data is retrieved
* submit\_review(“{address:’1 address’, landlord:’landlord’, rent:’12345’, review:’review’}”)
* get\_housing\_data(‘1 address’)
  + Log output to console with 1 second delay
  + Verify submitted review is retrieved
* get\_rid\_of(“{address:’1 address’, landlord:’landlord’, rent:’12345’, review:’review’}”)
* get\_housing\_data(‘1 address’)
  + Log output to console with 1 second delay
  + Verify empty data is retrieved
* get\_info(‘325 Ocean Street, Santa Cruz, CA, 95060’)
  + Log output to console with 1 second delay
  + Verify “3 bedrooms\n 2 full bathrooms\n 1,995 sqft” is retrieved

Testing Sprint 1:

A. Story 1: As a renter, I want to be able to post review information to this website

B. Story 2: As a future tenant, I want to be able to see other reviews for an address

Scenario: (Pre-condition: No rows in database exist with address: ‘1 address’)

* User submits ‘1 address’ into the search bar
* User verifies that there is no existing review for this address as website doesn’t update
* User submits creation form by below specs
  + address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’
* Website updates locally with the form information
  + List of reviews has new review inserted at top
    - address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’
* User refreshes browser to clear the list of reviews
* User submits ‘1 address’ into the search bar
* Website updates with available review(s) from the server
  + List of reviews has review retrieved from server inserted into list
    - address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’

Testing Sprint 2:

A. Story 1 (Sprint 1): As a renter, I want to be able to post review information to this website

B. Story 2 (Sprint 1): As a future tenant, I want to be able to see other reviews for an address

C. Story 3: As a developer, I want to store and pull reviews in Database to save reviews.

Scenario:

Pre-condition: No rows in database exist with address: ‘1 address’

* User submits ‘1 address’ into the search bar
* User verifies that there is no existing review for this address as website doesn’t update
* User submits creation form
  + address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’, review: ‘review’
* Website updates locally with the form information
  + List of reviews has new review inserted at top containing the above information
* User refreshes browser to clear the list of reviews
* User submits ‘1 address’ into the search bar
* Website updates with available review(s) from the server
  + List of reviews has retrieved review inserted at top
    - address: ‘1 address’, landlord: ‘landlord’, rent: ‘12345’, review: ‘review’

D. Story 4: As a product owner, I want a presentable website because I want it easier to use

* Better looking buttons
* Formatted reviews and forms

Testing Sprint 3:

A. Story 1,2 (Sprint 1): As a user, I want to be able to send/receive data through the website, because it will share my experiences or I will learn more about the house’s history.

B. Story 3 (Sprint 2): As a backend developer, I want to be able to store reviews onto the DB, because the website stores and loads reviews from the DB

C. Story 5: As a developer, I want to check if the house is a valid address, in order to prevent inaccurate reviews.

Scenario:

Pre-condition: No row containing address ‘111 Beach Street, Santa Cruz, CA, 95060’ in db

* Input/Select ‘111 Beach Street, Santa Cruz, CA, 95060’ in autocomplete search bar
* Verify that no reviews exist for this address, as none are added
* Verify that the search was completed as the map updates to the searched location
* Submit creation form:
  + landlord: ‘landlord’, rent: ‘12345’, review: ‘review’
* Website updates locally with the form information
  + List of reviews has new review inserted at top containing the above information
* User refreshes browser to clear the list of reviews
* Input/Select ‘111 Beach Street, Santa Cruz, CA, 95060’ in autocomplete search bar
* Website updates with available review(s) from the server
  + List of reviews has retrieved review inserted at top

D. Story 4 (Sprint 2): As a product owner, I want a presentable and professional website, because it will be more marketable.

* Format map and search bar
* Improve formatting/styling of reviews

E. Story 6: As a web developer, scrape apartment websites for housing info, decrease load on the server

Scenario:

* Input/Select ‘325 Ocean Street, Santa Cruz, CA, 95060’ in autocomplete search bar
* Verify that to the right of the map the following is shown
  + 3 bedrooms, 2 full bathrooms, 1,995 sqft