# SW Engineering CSC648/848

# Rently

Tool and Equipment Rentals

**Section 01 - Team 01**

Chase Alexander - Team Lead, full-stack

Benjamin McCullough - Github Master, frontend

Lauren Barer - Scrum master, database, cloud, security

Yu Hang Lee - Frontend lead

Chu Cheng Situ - Backend lead

**Milestone Four**

1 December 2021

**Revision History**

| Revision ID | Revision Date | Revised By |
|---|---|---|
| 01 | 12/15/2021 | Chu Cheng Situ |
| 02 | 12/15/2021 | Lauren Barer |
| | | |

## ***1. Product Summary***

Our product is Rently and it is marketed as a platform that can be used to rent out specialty equipment. For this product, the user can rent equipment that is available in the listing page and choose the item that they want and check the price for the equipment. The user must be aware that if they damage the equipment, they need to pay a certain amount of money to the owner of the equipment. The rental listing will show the price to rent the equipment per day and security deposit to make sure that the equipment will be returned safely to the owner. The user can browse through the category pages to find what equipment they want to rent. Users can contact the rental poster if they want to rent the item. If the user has extra equipment that they are storing in their garage and want to clear out the space, then the user can list the equipment using this product by setting up the listings with rental price per day, security deposit, and price for damaging the equipment.

Product Link: http://54.183.152.191/

## ***2. QA testing***

**I.**

Unit Test 1: For the first major user story, we will test our program's function which sorts each listing by category. The functions that we will need to test are getCategoryInfo and getPostCards. In our test cases, we will pass a category name, and the testing method will expect that the listings returned all have the exact category ID corresponding to the category name. If the listings do not have the same category ID, the test should fail. These test statements have full function coverage, as they test the functions which receive the data by category and the function that displays the given data. The test executes most of the statements in each function, but not all.

Unit Test 2: For the second major user story, We will test our program's registration feature. Specifically, we will be testing the functions within our Registration component to make sure they only send valid accounts to the database.  For the first of our  test cases, we will pass an account with valid and unique account information. When tested, the testing method will expect the account to be created successfully. For another test case, we will pass an account with a password that is different from its confirmed password. The testing method will expect an error thrown. This test has partial function coverage, as it will cover each function in the Registration component. This test has partial statement coverage as well, as only some of the statements in the program will be executed.

Unit Test 3: For the third major user story, We will test our program's posting feature. Similar to the testing done for our registration, we will be testing the functions in the PostPage component. For our test cases, we will pass a post with complete information. The testing method will expect the post to create successfully, and should pass if it does so. For another test case, we will pass a post with some of the fields empty. The testing method will expect an error thrown. The test only has partial function coverage, as it does not  test the functions in the back end, but only the front end post component. The test  also only has partial statement coverage, as only specific lines from the component's functions are used for testing.

**II.**

Test 1:

#1- #8 The user can create an account by entering the information. Afterward, he can log in to the account. After logging in, the user can search the categories to see the information. In the search the categories section, the user can click on the item they want to see to display the price and the information with pickup location to let the user to rent. Also, at the bottom of the item, it has the message button to contact the renter. For the user that wants to rent their item, the user can also fill out the information on the item to post the item they want to rent.

Test Case Id 1:

Test Data: Once you are on the website, users can click on the “create an account” button to redirect to create an account. Then after finishing creating the account, the user can log in and list the item. If you want to rent out the item, you can click on the “list an item” to post your item with information that you filled out.

Step1: Input the information for creating an account and click on the “create an account” button on the bottom. The user can log in to their account on the login page.

Step2: The user can log in to their account by entering their username and password

Step3: After logging in, the user can click on the “Search By Category” to see the items that are listed.

Step4: After the user finds the item you want to see, there are price information with the pickup location listed on the post.

Step5: If the user wants to rent and contact the renter, there is an “message” button to contact the renter at the bottom of the post. The user can click on it and redirect to send the email.

Step6: If the user wants to rent out the item, just click on “list an item” and fill out the information and click post.

| Test Case Id | Test Case Description | Date Tested | Test Scenario | Prerequisites | Test Result |
|---|---|---|---|---|---|
| 1 | Creating an account, Login, and Searching for items for the user “Maya” | 11/30/2021 | -Enter information<br><br>-Click on “create an account”<br><br>-Enter username and password<br><br>-Click “login”<br><br>-Click “Search By Category”<br><br>-Click on item want to rent<br><br>-Click on “message” button if want to want or contact the renter<br><br>-Click on “List an Item”<br><br>-Fill out the information with pickup location<br><br>-Click on “post” | -Website is able to be run locally.<br><br>-Keyboard to type in information | Chrome: Pass<br><br>Safari: Pass |

***3. Code Review***

a. Coding Style:
   i. The coding style we chose is as follows:
      1. Space between parameters and arguments
      2. Opening curly brace on the same line after a space
      3. 2 space indentations
      4. Spaces around operators and nested calls
      5. Else on the same line as if statement closing curly brace
      6. Mandatory semi-colons
      7. 1 statement per line
      8. camelCase variables and function names
      9. UpperCamelCase for class names
      10. Must have default case for switch statements
   ii. To enforce this coding style, we use auto-formatting tools built into our individual IDEs
b. Code Review Practices:
   i. Check for logical errors
   ii. Check for files that should be ignored
   iii. Check for any obvious errors in formatting
   iv. Leave at least one piece of constructive feedback
   v. Links:
      1. Initial request: https://github.com/CSC-648-SFSU/csc648-fa21-01-team01/pull/25
      2. Fixed request: https://github.com/CSC-648-SFSU/csc648-fa21-01-team01/pull/26

***4. Non Functional Requirements Specs***

1. Ideally, our application shall run in any web browser and any operating system but also be used on any mobile device through any browser.
   - This is **DONE** our application runs in any browser
2. The users' passwords and usernames will be encrypted with the Advanced Encryption Standard method, possibly using an AWS Lambda.
   - This is **ON TRACK** and will be completed closer to M5
3. All data will be completed in the MySQL Workbench and stored in the Amazon AWS RDS server
   - This is **DONE** all of our data is stored on RDS through the SQL workbench
4. The website/app would be very easy to access, meaning we would have clear icons and not too many options so people can navigate through the site easier.
   - This is **ON TRACK,** we have icons at the moment but they need more color

5. The expectancy of our website is a 2 second launch time
    - This is **DONE** the website has a 2 second launch time.
6. When switching from one window to another, the switch time should be roughly 2 seconds as well.
    - This is **DONE** when you switch to a new window on our app. It's roughly 2 seconds.
7. The goal is to make the website an easy experience, meaning getting people to their transaction as quickly as possible, so people get what they want, have a good experience, and leave a good review for the app.
    - This is **DONE** the website is clear with understandable navigation.
8. Navigation will be clear, payment processing and information will be secure, and communication between people will be trouble-free
    - This is **ON TRACK** we are finalizing the payment process.
9. Our code will be processed in the Git Repository and managed not only by the git master but also by every team member. Each push will be discussed in the team. Git Repo will be cleaned and organized before each submission.
    - This is **DONE** all of our code gos through github and is managed by the git master.

## ***5. List of P1 Features***

1. User can log in / log out
2. User can create an account
3. Users can filter by category
4. Users can see rental price of equipment
5. Users can rent out with info for pick up location
6. Users can select the amount for security deposit
7. Users can "rent" a post (see contact info/communicate with renter)
8. Users will receive a penalty if item is missing or damaged / can also set penalties