CSCI 3060U Course Project

Winter 2022

Phase #5 - Integration and Delivery

April 24th, 2022

Report

Group members:

Ben Tsoumagas 100751395

Thinh Le 100741899

Muaz Rehman 100553376

Cody Malcolm 100753739

Project Summary:

In this project, we have created a rental booking system according to Agile principles. This project was broken down into 5 phases:

1. Analysis and creation of a Black Box test suite based on project requirements
2. UML design and implementation of the Front End
3. Executing tests created in #1 against application created in #2 via script, and correction of deficiencies identified by these tests
4. White Box statement, loop, and decision coverage testing of selected methods
5. Implementation of a draft Back End to process output from the Front End created in #2, and a script to demonstrate the processing of a week's worth of activity

The final product being delivered is a robust Front End with a comprehensive Black Box test suite and representative White Box tests, a script to execute the Black Box test suite, a draft Back End that can correctly process valid inputs and produce the correct output for the next day's Front End session, and a script that executes five day's worth of usage of the system simulating a week of use.

Phase 5 Implementation:

In phase 5, one of the major deliverables was supposed to be a script that runs the application several times to simulate one day's worth of activity. However, our application had already incorporated several "user sessions" in one "front-end session" - for more details on this, please refer to the consolidated list of requirements problems in the Phase 1 Test Plan Document. As a result, implementation of a script to perform this functionality would be redundant and unnecessary. Therefore, we extended the requested "Back End Stub" to produce correct input for the next day, rather than just acceptable input for the next day. In order to do this, we implemented a "BackEnd" class that was integrated with the existing Front End project, it made use of some methods and classes already implemented for the main project. However, it should be noted that since we did this as an extra, the Back End is a "draft" and does not contain the same degree of robust guarding against bad inputs that the Front End application does. However, as long as it is used "properly" (ie. the inputs to the Back End are the inputs that are generated by the thoroughly tested Front End application and/or the included starter files in the repository), it does not have any identified crashes or other unexpected behaviours.

We also implemented a weekly script, which requires a directory ./daily_tests/ that contains the initial input files (current_user_accounts.txt and available_rental_units.txt)

and a directory called “inputs” that contains files of the format “day1input.txt”, “day2input.txt”, “day3input.txt”, etc. The weekly script, when placed in the ./out/production/Vamonos-Pest-SQA-Project/ directory and executed with the “bash ./weekly_script.sh” command, will execute the Application and BackEnd for each day there is an input file for, creating a directory in the ./daily_tests/ directory for each day’s Application output files. The BackEnd output (the input for the following day’s Application) will be put into the next day’s directory. If there are 5 day’s worth of input, for example, the final produced current_user_accounts.txt and available_rental_units.txt files will be found in a directory at “./daily_tests/day6/” and would be considered the input for the following week.

Please note that sample input files have been provided for 5 days. Unlike the minimalist Black Box testing inputs, these involve multiple transactions/user sessions for each day, and are intended to demonstrate the full breadth of capabilities of the implementation, including such scenarios as logging in as users created in a previous day, renting units that “used to be” rented but are now available again, and correct handling of deleted users and their rentals.

Phase 5 files:

Since the BackEnd is integrated with the main Front End application, it is not a trivial matter to submit “just” the BackEnd code. Rather, we will submit the full repository and specify here where code was added or changed to implement the BackEnd:

- The only new class is the BackEnd class, of course everything contained in this class is new for the BackEnd
- As the Front End did not have any reason to care about the days remaining on a rented rental, there was no field in a Unit object to store this information. However, the BackEnd does use this information, so a daysRemaining field and a setter for this field were added to the Unit class. Additionally, a method was added to decrement the number of days remaining by 1 representing the passage of a day.
- The BackEnd needed a way to convert Users and/or Units to a String of the correct format for output to a current_user_accounts.txt file and and available_rental_units.txt file. As a result, the Users and Units classes had “getUserAsString()” and “getUnitAsString()” methods added, respectively.