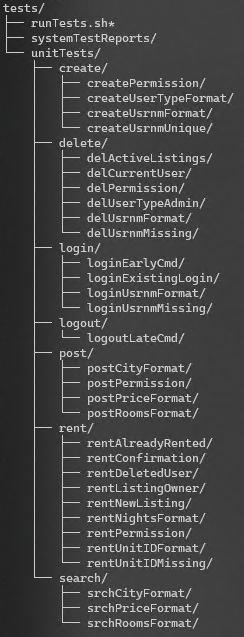
# KMD LLC – Test Organization and Testing Plan

All information and functionality related to the testing of our software resides in a directory named “test/”, which must be located at the top level of the software’s working directory. This directory’s structure can be seen in Figure 1, and its contents are follows: 1) a subdirectory named “unitTests/” that contains the many individual tests which comprise the overall system test, 2) a directory named “systemTestReports/”, which contains system reports describing the aggregated results of the individual tests, and 3) a shell script named “runTests.sh”, which will load and run the individual tests, as well as record their individual results into an overall system test report. Further details on these items are as follows:

1) The “unitTests/” subdirectory: This subdirectory contains further subdirectories with names indicating the specific area of system functional requirements that their contained tests target. These requirement categories are summarized as: create, delete, login, logout, post, rent, and search. Each of these functional requirement subdirectories further contains a subdirectory for each individual unit test related to that requirement. These tests are prefaced with a shorthand version of their related functional requirement for easy identification when selecting which tests to run in a targeted test (see below). The unit test subdirectory’s structure can be seen in Figure 2, and its contents are as follows:  
  
- A subdirectory named “inputs/”, which contains the users.txt and listings.txt files to be used as backend-input for that test

- A subdirectory named “expectedOutputs/”, which contains the expected outputs of that test, which the actual test outputs will be compared against to determine the test results

- A text file named “commands.txt”, which contains the list of program commands to be executed by that test

- A subdirectory named “testResults/”, which contains the results (including all past results) of that test, named in the following format in order to indicate the time and date of the result generation: “unit\_<year>\_<month>\_<day>\_<hour>\_<minute>\_<second>”

2) The “systemTestReports/” subdirectory: This subdirectory contains reports indicating the results of conducted system tests (including all past reports), named in the following format in order to indicate the time and date of report generation: “report\_<year>\_<month>\_<day>\_<hour>\_<minute>\_<second>”

Figure - The “test/” directory structure

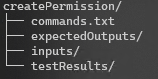
3) The “runTests.sh” shell script: This script verifies and summarizes the contents of the “unitTests/” subdirectory in a prompt to the user, then lets the user select which unit tests they wish to conduct. The script then backs-up the software’s production data files and runs each test with the test’s provided unit files and list of program commands. The output of the test’s program session is compared to the unit test’s expected outputs, and the result of the comparison is recorded in the unit file’s test results subdirectory. Once all tests have been conducted, the production data files are restored, and all test results are compiled into a single report for that system test, outputted to the system test reports subdirectory.

Figure - The structure of a unit test subdirectory (example "createPermission" seen here)

Each component of the system will be targeted for testing as it is implemented, and an overall system test will be conducted prior to every iteration’s release to ensure that none of the software’s existing functionality has been impacted by the implementation of newly developed features. Additionally, the testing will be run at a weekly frequency post-software launch to ensure that the quality and consistency of the software has not been impacted by any changes to the software hosts’ environment, such as system updates and library dependency updates.