An Agile Course Project

CSCI 3060U – Winter 2022

Project Teams

You are to form a team of three or four people to design, implement, document and deliver a software product. All phases to follow Agile development philosophy and practices (e.g., XP, Scrum, or Kanban). Within Agile development it is expected that you will:

- continuously maintained test suites as requirements and quality control
- pair program all code
- use the simplest possible solution to every problem
- continuously redesign and re-architecting
- automate testing and integration
- frequently or continuously release the software

Every two weeks (or so, see the schedule below) you will deliver concrete evidence of your team's progress as required by project phases.

Project Phases

The project will be done in five phases, each of which will be an assignment. Phases will cover steps in the process of creating a quality software product in the context of Agile development.

Assignments will be on the quality control aspects of requirements, rapid prototyping, design, coding, integration, and analysis of the product you are building. Throughout the project, you should keep records of all evidence of your product quality control steps and evolution, to make the marketing case that you have a quality result at the end of the course.

Your final product will be tested and evaluated.

Project Schedule

The course project consists of six assignments, with separate handouts for each one. Assignments are scheduled to be due as follows:

- Phase #1: Front End Requirements
  – Tuesday, Feb. 21, 2022
- Phase #2: Front End Rapid Prototype
  – Sunday, Mar. 6, 2022
- Phase #3: Front End Requirements Testing
  – Sunday, Mar. 20, 2022

- Phase #4: Back End Unit Testing
  – Sunday, Apr. 3, 2022
- Phase #5: Integration and Delivery
  – Sunday Apr. 17, 2022

Assignment/Phase Hand-Ins – GitHub & Blackboard

While working on a given phase of the project you are expected to maintain a GitHub repository and regularly commit updates to source code, tests and all project documents. Your GitHub repository must be non-public and should include your teaching assistant as a team member.

Unless otherwise specified, all assignments/phases *must* be handed in through Canvas by 11:59pm on the due date. For all submissions clearly indicate your team's name and all team member names.

Late assignments/phases will be accepted through Canvas during the first 24 hours after the submission deadline with a 20% penalty. After 24 hours, late assignments/phases will not be accepted. Exceptions to this policy may be made for valid reasons (e.g., medical).

As part of the assessment of each assignment/phase, your teaching assistant will review your submitted Canvas materials and may review your GitHub project activity. Each team member will also be expected to complete a peer evaluation of their team's work on the current phase.

Project Requirements

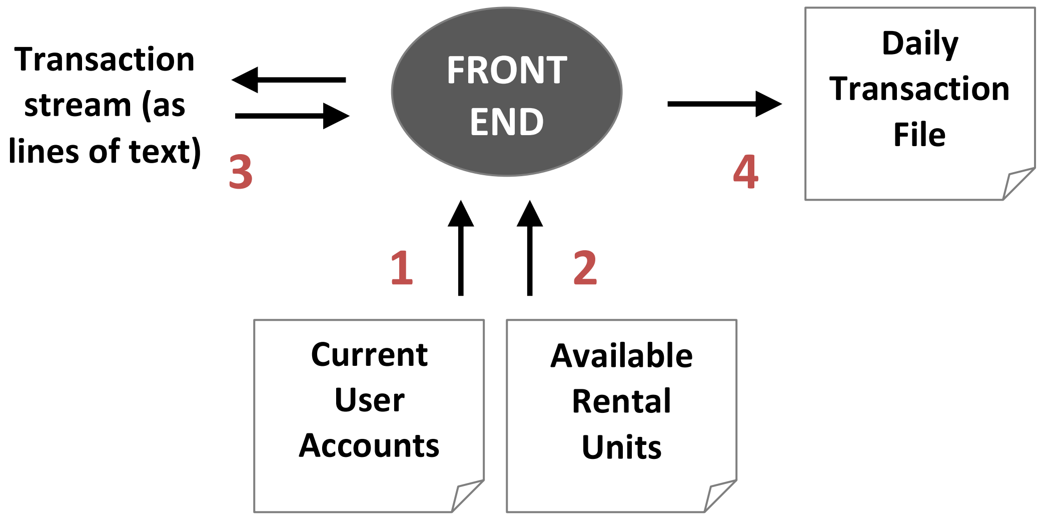
The product you are to design and build is a short-term rental system called OT-Bnb. The system consists of two parts:

- the Front End, a point of purchase terminal for renting short-term accommodations (written in Java)
- the Back End, an overnight batch processor to maintain and update a master rental files (also written in Java)

Both parts will be run as console applications, that is, they are to be invoked from a command line and use text and text file input/output only (this is an important requirement for assignments later in the project, so don't ask for exceptions).

THE FRONT END

The Front End reads in a file of rental units available (1) and a file containing information regarding current user accounts in the system (2), it processes a stream of rental requests one at a time (3), and it writes out a transaction file of rental activities at the end of the session (4).



Informal Customer Requirements for the Front End

The Front End handles a sequence of transactions, each of which begins with a single transaction code (word of text) on a separate line. The Front End must handle the following transaction codes:

login	- start a Front End session
logout	- end a Front End session
create	- add a user with the ability to rent and or list rental units (privileged transaction)
delete	- delete a user's account
post	- posts a new rental unit in the system
search	- search for available rental units by city
rent	- rent a unit in the system

Transaction Code Details

login - start a Front End session

- before processing a login transaction, the Front End reads in the current user accounts file.
- should ask for the username
- after the username is accepted, reads in the available rental units file (see below) and begins accepting transactions
- <u>Constraints</u>:
    - no transaction other than login should be accepted before a login
    - no subsequent login should be accepted after a login, until after a logout
    - after a non-admin login, only unprivileged transactions are accepted
    - after an admin login, all transactions are accepted

<u>logout</u> - end a Front End session

- should write out the daily transaction file (see below) and stop accepting any transactions except login
- <u>Constraints</u>:
    - should only be accepted when logged in
    - no transaction other than login should be accepted after a logout

<u>create</u> – creates a new user with renter and/or rentee privileges.

- should ask for the new <u>username</u> (as a text line)
- then should ask for the <u>type of user</u> (admin or full-standard, rent-standard, post-standard)
- should save this information to the daily transaction file
- <u>Constraints</u>:
    - privileged transaction - only accepted when logged in as admin user
    - new user name is limited to at most 15 characters
    - new user names must be different from all other current users

<u>delete</u> - cancel any outstanding rentals and remove the user account.

- should ask for the <u>username</u> (as a text line)
- should save this information for the daily transaction file
- <u>Constraints</u>:
    - privileged transaction - only accepted when logged in as admin user

    - username must be the name of an existing user but not the name of the current user
    - no further transactions should be accepted on a deleted user's available inventory of rental units.

<u>post</u> – post a new rental unit in the system

- should ask for the <u>city</u> (as a text line)
- should ask for a <u>rental price</u> in dollars (e.g. 15.00)
- should ask for the <u>number</u> of bedrooms
- should set the <u>rented flag</u> for the unit to false
- should save this information to the daily transaction file
- <u>Constraints</u>:
    - Semi-privileged transaction - only accepted when logged in any type of account except rent-standard.
    - the maximum rental amount per night is 999.99
    - the maximum length of a city name is 25 characters
    - the maximum number of bedrooms is 9
    - no further transactions should be accepted on a new rental unit until the next session

<u>search</u> – search for all available rental units in a given city with a maximum rental price and a minimum number of bedrooms

- should ask for the <u>city</u> (as a text line)
- should ask for a <u>rental price</u> in dollars (e.g. 15.00)
- should ask for the <u>number</u> of bedrooms
- should save this information to the daily transaction file
- <u>Constraints</u>:
    - should only list units with a <u>rented flag</u> set to false
    - accepts * as a wildcard value

<u>rent</u> – rent an available unit

- should ask for the <u>rental id #</u> and the <u># of nights</u> requested. System assumes the bookings start on the current night.
- should display the rent per night and the total cost to the user and ask for confirmation in the form of <u>yes</u> or <u>no.</u>
- should set the <u>rented flag</u> for the unit to true
- should save this information to the daily transaction file
- <u>Constraints</u>:
    - Semi-privileged transaction - only accepted when logged in any type of account except post-standard.
    - Rental id # must be for a currently available unit

    - A rental can be for at most 14 nights

General Requirements for the Front End

The Front End should never crash or stop except as directed by transactions.

The Front End cannot depend on valid input - it must gracefully and politely handle bad input of all kinds (note: but you can assume that input is at least lines of text).

Daily Transaction File

At the end of each session, when the logout transaction is processed, a daily transaction file for the day is written, listing every transaction made in the session.

Contains fixed-length text lines of the form:

```
XX_UUUUUUUUUU_TT_IIIIIIII_CCCCCCCCCCCCCCC_B_PPPPPP_NN
```

Where:

`XX`	is a two-digit transaction code: 01-create, 02-delete, 03-post, 04-search, 05-rent, 00-end of session
`UUUUUUUUUU`	is the username (renter if two users in transaction)
`TT`	is the user type (AA=admin, FS=full-standard, BS=buy-standard, SS=sell-standard)
`IIIIIIII`	is the rental unit id (alpha-numeric)
`CCCCCCCCCCCCCCC`	is the city
`B`	number of bedrooms
`PPPPPP`	rental price per night
`NN`	number of nights
`_`	is a space

Constraints:

- every line is a fixed length
- numeric fields are right justified, filled with zeroes
  (e.g., 09 for number of nights)
- alphabetic and alphanumeric fields are left justified, filled with spaces
  (e.g. User001________ for username "User001")
- unused numeric fields are filled with zeros
  (e.g., 0000)

- In a numeric field that is used to represent a monetary value, “.00” is appended to the end of the value
  (e.g. 00110.00 for 110)
- unused alphabetic and alphanumeric fields are filled with spaces (blanks)
  (e.g., _________________________)
- the sequence of transactions ends with an end of session (00) transaction code

Current User Accounts File

Consists of fixed length (13 characters) text lines in the form:

```
UUUUUUUUUUU_TT
```

where:

`UUUUUUUUUUU`
: is the username

`TT`
: is the user type (AA=admin, FS=full-standard, RS=rent-standard, PS=post-standard)

`_`
: is a space

Constraints:

- every line is exactly 13 characters (plus newline)
- alphabetic fields are left justified, filled with spaces (e.g., User001________ for username “User001”)
- unused alphabetic fields are filled with spaces (blanks) (e.g., ________________________)
- file ends with a special user named END with an empty user type and the credit field empty.

Available Tickets File

Consists of fixed length (45 characters) text lines in the form:

```
IIIIIIII_UUUUUUUUUU_CCCCCCCCCCCCCCC_B_PPPPPP_F_NN
```

where:

`IIIIIIII`
is the rental unit id (alpha-numeric)

`UUUUUUUUUU`
is the username

`CCCCCCCCCCCCCCC`
is the city

`B`
number of bedrooms

`PPPPPP`
rental price per night

`F`
rental flag

`NN`
number of nights remaining if currently rented

`_`
is a space

Constraints:

- every line is a fixed length
- alphabetic and alphanumeric fields are left justified, filled with spaces (e.g., Toronto_________________ for the city "Toronto")
- unused numeric fields are filled with zeros (e.g., 0000)
- in a numeric field that is used to represent a monetary value, if the value is only in dollars, then ".00" is appended to the end of the value (e.g. 110.00 for 110)
- unused alphabetic and alphanumeric fields are filled with spaces (blanks) (e.g., _________________________)
- file ends with a special rental unit named END with all other fields empty.