Anti-Theft Project Final Documentation

CS 3354.504

Prepared by:

Web Development Team
Labeeba Rana - LFR160130
Gentry Jenkins - GLJ180000
Georpi Ikamba - GXI170230
Varika Pinnam - VXP171430
Sindura Boppudi - SLB170330
Christopher Guerra - CJG160230
Noman Syed

Table of Contents

Project Description

The basis of this project is tracking chips that allow people to trace personal objects to prevent theft and reclaim stolen property. We created a database that allows users to create an account to register and store their products, so they can later view or track their items.

Architecture and Design

Our team utilized a GitHub repository to collaborate and commit code from different branches. There exists a DatabaseQuerying branch, a GUI branch, a User branch, a password branch, and finally the master branch. These branches have been merged and integrated to develop a functioning system. The initial Graphical User Interface (GUI) consists of a “Sign Up” and Sign In” button. Action listeners have been implemented for each of these buttons to direct users to a separate GUI. The “Sign Up” button leads to fields to input various personal information and a Sign Up button that will create a user. The “Sign In” button leads to fields to input Username, which would be an email, and Password, followed by a Login or Back button.

When a User is created, the information will be stored using queries to a Database. Oracle Database 19c is used for data storage on a remote server which makes use of Oracle's JDBC driver for connecting the server application and the database. The client application sends query requests to the database which the server then processes and returns necessary values. The client receives the result data and parses it into a form that may be displayed or stored temporarily on the client-side.

Within the password branch, the primary classes we worked with were User and Product. We used the password branch to work on the back-end and test sign up, product creation, and login separate from the database. We later synced the methods in the classes of the password branch, such as creating a new account, logging in, adding a product, and displaying products, to the GUI to test it and follow the path of the data before syncing up the GUI with the actual database.

In the User class, there are attributes that pertain to an account such as first and last name, middle initial, email, password, userID, and address. There is also a static counter we used to assign the userID number and increment with the creation of each new user. There are also two HashMaps, which acted as a temporary data storage before we linked to the actual server. The HashMaps contain the user email and ID or user email and password. The User class contains methods for creating a new user as well as returning attributes like name, address, email, and ID.

Both the Product class and Login class within the password branch are inherited from User due to needed access to the HashMaps. In the Product class, there is an additional HashMap, in which the key is the UserID and the same for every product that is added by a

user, and the value is the description of the product. In this way, we created a list of products attached to the same UserID.

Running Tests Appendix

In order to test the overall system, simply Sign Up or Sign In to create a product. If you enter empty fields into the database, it is not going to return anything or be entered into the database. Oracle databases do not allow null values for unique variables such as username and password. However, if you enter empty fields for the items, it will allow you to do so. Once you enter personal information on yourself and your products, our team can cross-check with the database to test its accuracy.

JUnitConTest.java Class is a JUnit test case that verifies a connection with the server. It utilizes an “assertNotNull” function that checks it against a null string. If it is not null, the test case passes. Simply run to the program and check for the bar to turn green to indicate passing.

JUnitQueryMethodsTest.java is a JUnit test case that verifies whether a method is returning correctly. It also utilizes an “assertNotNull”, checking for a non-empty string.

The productjunit.java class is a JUnit test case that asserts equal when the number of products owned by a user is verified. Figure 1 below demonstrates a passing test case:

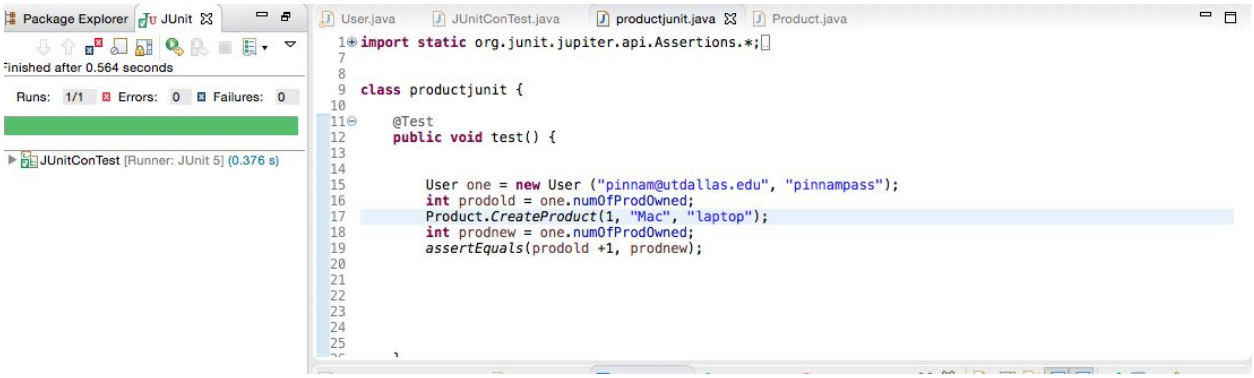

Figure 1.

RowsUpdated.java is a JUnit test case that returns the correct number of rows, or users if a new user is created. It utilizes the “assertTrue” function that checks it against a concrete parameter.