Stag Bar
Bar Inventory System
Iteration 1
Version .1
Due: 3/2/2016

Abstract

The purpose of this project is too create a user friendly bar inventory system. The system will allow a inventory manager to keep track of their inventory and monitor internal theft. The system will allow a manager to create custom drinks, in order to avoid calculating how much spirits, and what spirits, was used for cocktails each time inventory is done.

Team

Thomas Andrikus

Gregory Dudar

Samuel McAdams

Anthony Whitaker

Table of Contents

1.0 High Level Goals

1) The user should be greeted with a UI to create an account with a custom user name and password.
2) The user should be allowed to login into the system (provided they have created a login) using their custom login and id.
3) The user should be able to create custom groups for the various types of beer and spirits sold at their establishment.

1.1 Deployment System

Currently the system will be compiled and executed using a Java IDE and build file. The build file will compile and run the software. For final release, the software will be opened or installed through the use of an executable or JAR file (TBA which will be used). Upon execution the software will communicate with a cloud based database rather then forcing the user to download a database locally. Using a cloud-based database will be less overhead with for the consumer.

2.0 Work Items

1) Create the database. An account with Amazon Web Services will need to be made in order to create the cloud-based database. Once the account has been created a relational database, probably MYSQL, will need to then be established.
2) Create methods to allow the system to communicate with the database. This will include establishing a connection to the remote database, creating a new user in the database, and creating a new group (table) in the database. When establishing the connection with the database the software should allow the

user to access using a custom id and password. When a new user is created, using a custom id and password, the system should first check the database to make sure the user doesn't already exist. When creating a new group the user should be able to use custom names (I.E. bottled beer). The groups will be one of three types: bottled beer, spirits, and draft beer. This is necessary since the different types will have different properties. For example bottled beer is represented by and integer (20 bottles bud light) and liquor needs to be expressed as a float (23.5 fluid ounces of Barton's vodka).

3) A UI will need to be created to allow the user to create a new user. The user should be allowed to use a custom username and password. When a user creates a new user the password should be entered twice to insure that the user has entered the password they want correctly. The username and password should then be stored in the database. When creating a new user the user should be allowed one of two permissions. Either they are a master user (can enter inventory, add new groups, etc.), or they are a guest user with read only permissions.
4) A main button driven menu will need to be created for user navigation. The interface should be clean with button labels clearly indicating what they do (EX: "Enter Sales"). The button should then create a pop up window or refresh the frame to allow the user to do their selected action.
5) A UI will need to be created to allow the user to create a new group, which they want to maintain inventory for. For example, a user may create a group called "bottled domestic beer". The group will then be established as a table

within the database to hold the inventory data. The name of the group should be stored locally so that a later UI can recall which groups the user has created, so the user can then select the group and manage the inventory.

6) A build file will need to be created to compile the program. The build file should take the .java files and compile them into .class files. The build file should also make sure that any external jar files are included in the build path (such as the SQL jar file needed for database connections). The build file will be a .xml file.

2.2 Assignments

Thomas Andrikus: Work on the build file and help where needed with the UI. Create Design Document for iteration one.

Greg Dudar: Create the main menu and the enter group UI. Create System Sequence Diagrams for iteration one.

Sam McAdams: Create database and the necessary methods to interact with database. Create the iteration plan for iteration 1 deliverables.

Anthony Whitaker: Create the UI for creating a new user, and login. Create Domain Model for Iteration 1.

2.3 Testing Criteria

Thomas Andrikus: Once the build file has compiled the code it should run the program every time. The build file should not have any compilation errors or dependency errors.

Greg Dudar: For main menu there will only be one working button. The button should be clicked multiple times to make sure that it opens every time in the same

manner. The window should also close every time when either a "back" button is pushed or the close window icon is pushed ("X" in upper corner). Dummy data should also be put in of multiple types to make sure they are properly entered into the system from the UI.

Sam McAdams: The method to create connection should return a connection to the database each time it is called. When creating a user the database should be queried to make sure that the user has been created in the database. If an attempt to create a new user in the database that is already present is executed then the method should return false. A new group should be put into the database and the database queried to make sure that group has been added to the database.

Anthony Whitaker: When creating a new user the database should be queried to make sure that the user has been successfully created in the through the UI. For the password field there should be both a positive and negative test case. One in which the passwords match and one where they do not. When a user has successfully logged in a non-null connection object should be returned. If the user is not then the system should return a null connection object. Both cases should be tested.

3.0 Evaluation Criteria

Given the limited number nature of the iteration one use cases and the limited, amount of variables, all tests should be at least >= 99%. At this current stage creating a user and connecting to the database are the most critical. If a user cannot be successfully created or a connection to the database not made then any future use cases will not be able to be implemented. At the current stage our project will be very bare bones, with most of the features not implemented. We should however

be able to show the vision and potential for our software, and garner a favorable response from the viewers of our demo.

2/25/2016 Edit: It appears that our database connection is not going to work in the demo environment. This issue has been discussed with the professor and a video demo in place of a live demo may be used. This is unfortunate, but not a critical failure. Hopefully, a video demo rather then a live demo will not have a negative impact on the response from the class.