

M12-A02: Group Final Project

Forrest Moulin, Maxwell Naughton, and Brianna Price

Pennsylvania State University Abington

IST 311-001: Ood and Software

Shawn Lupoli

December 9th, 2020

Contents

1.0 Introduction.....	3
1.1 Objectives.....	3
1.2 Key Challenges and Constraints.....	3
2.0 Project Scope.....	5
2.1 Use Case Diagram.....	5
2.2 Specification Narratives.....	5
3.0 Structural Design.....	10
3.1 Implementation Class Diagram.....	10
3.2 Class Responsibilities.....	10
4.0 Flow-of-Control: Activity Diagrams.....	13
5.0 User Interface Classes: State Diagrams.....	17
6.0 Components and Distribution.....	21
7.0 User Manual.....	22

1.0 Introduction

The final development phase of the HIMA produced software capable of creating and storing (without a database) multiple user accounts, login functionality, storing inventory associated with user accounts, creating lists and items with a full list of attributes, as well as searching a user's inventory by list or item name. Development utilized the most current version of Java and NetBeans 8.2. The primary objectives and challenges are listed in the following sections.

1.1 Objectives

The objective of our team's conception for the Home Inventory Management Application was to develop an application which allowed users to log a collection of personal items to user created categorical lists (referred to as "Categories"). In addition to these primary objectives, auxiliary objectives such as the creation of private accounts, the ability to edit and modify the important details of the users private account, a search function to locate specific Items logged within the users' private account, and the ability to export a specified Category to either a PDF file or Text File were set. The collection of these objectives, both primary and auxiliary, were divided into a set of five Sprint implementation segments.

1.2 Key Challenges and Constraints

In our first Sprint implementation, our team devised a working project prototype which launched the application prototype and authenticated a test user. Once the test user passed the authentication check, our application displayed its main navigation window (referred to as the users' "Dashboard"). In addition to these features, our application's Dashboard presented visual references to the features our team intended to develop in the upcoming future Sprint implementation segments.

The second Sprint implementation handled the developed of the objective of allowing a user the ability to edit and modify the important details of the users private account. This feature allows the user to modify and save changes to their private accounts registered email address, first and last names, and password.

The third Sprint implementation concerned itself with the development of the of the Items/Inventory feature, allowing the user to log a collection of personal items and specify their category type, item name, item identification number, and item location. In addition to logging new items, the feature allows for both the removal and the duplication of items contained in the user's main inventory category (the default categorical list for the application at that time). A "Select Filter" drop down display was also added to the feature allowing for the user to display items according to which category they were assigned to.

The fourth Sprint implementation handled the creation of the categories feature, allowing a user to create their own categories to assign items in their inventory to. In this feature a user writes out a categories name and which identification number to assign to it, once this is done the user may create the category by clicking on the “Add Category” button next to the text entry fields for the category name and identification number. Additionally, the user is provided a function to remove existing categories by selecting an existing category and clicking on the “Delete Category” button, and a function to navigate to the Items/Inventory feature by selecting the “Inventory” button.

The fifth Sprint implementation concerned itself with the development of the search function to locate specific Items logged within the users' private account. In this feature a user may type a string of text into their Dashboards search bar and select the “Search” button, upon doing this the user will navigate to the Items/Inventory feature and display the item sought after if it is registered.

2.0 Project Scope

2.1 Use Case Diagram

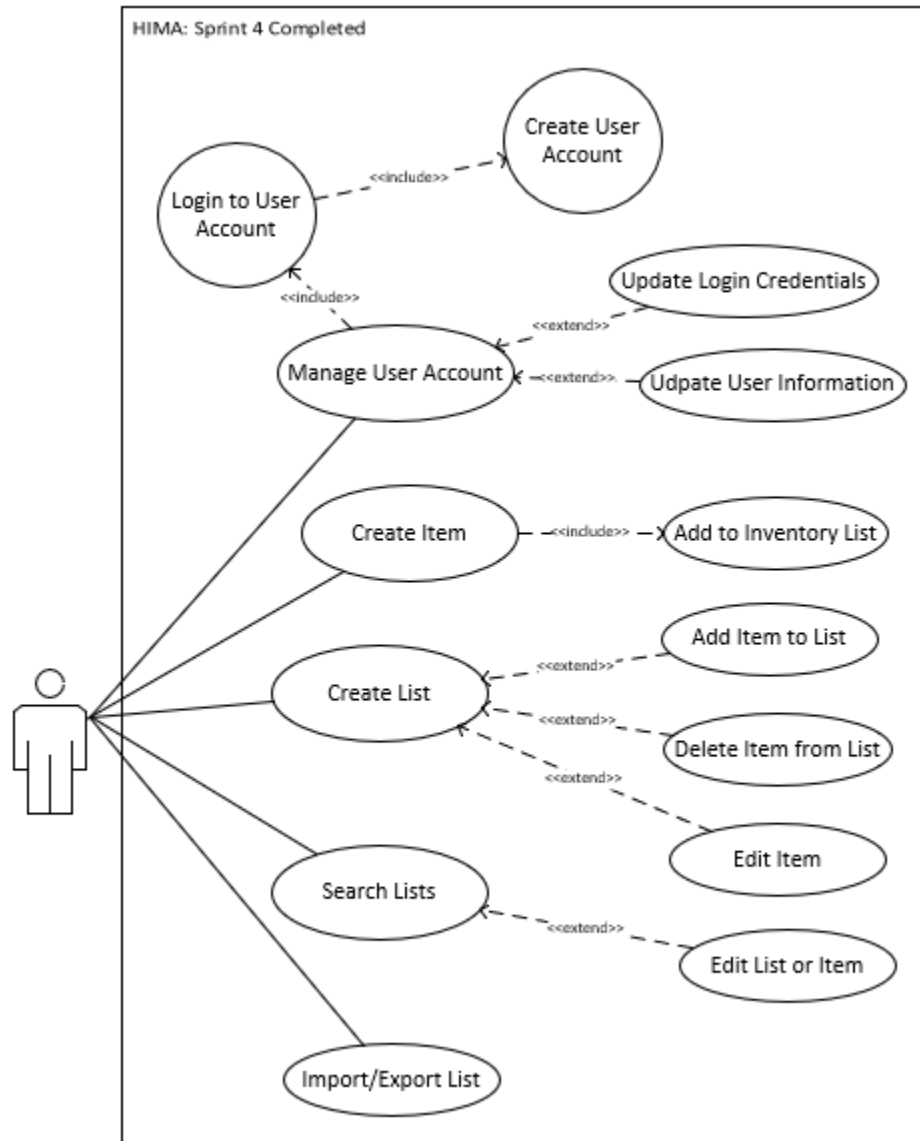


Figure 2.1 The HIMA Use Case Diagram: The diagram above represents the final product of The HIMA application, following Sprint 4. The two circles located at the top of the diagram are preconditions required to perform the functionalities displayed in the oval bubbles. Section 2.2 highlights the four primary use cases, excluding *Import/Export List*, which is an additional beta feature.

2.2 Specification Narratives

Update Login Credentials and User Account Information Use Case

Pre-condition: Create User Account and Login

1. Launch the HIMA software and select “Create Account”
2. Enter user account information including email, first name, last name, password, then confirm password
3. Select “Add” to create the account
4. Select “Confirm” to save submission
5. Sign into user account with email and password

Task Analysis Summary: The HIMA application will allow the user to create and manage their user account. An account is made up of information that includes:

- First name
- Last name
- Email address
- Username
- Password

All of these fields are String variables entered by the user and validated for uniqueness and against formatting constraints. These inputs are required to create an account and will persist until the program is ended or the IDE is exited. After creation, users can manage their account information by editing the fields independently. If the user logs out of their account or ends the program, accurate login is required to access the user’s account information.

Task Analysis:

1. Upon login and within the user dashboard, select “Account”
2. To edit account information, select “Edit”
3. After editing account information, select “Save All”
4. Return to the dashboard by selecting “Dashboard”

Functional Requirements:

- String input validation for account creation, login credentials, and email address
- Account login validation
- Option to independently manage (“update”) first/last name, email, username, password
- Logout of user account

Non-functional Requirements:

- Access to IDE using personal computer
- Keyboard & Mouse
- Monitor

Primary Design Challenge: Persistent data is required for the user to be able to manage their account information. This includes maintaining user accounts for future logins once the application is started and ended, displaying user information like first/last names when they attempt to manage their accounts. Persistent data was challenging to design due to the learning curve associated with JavaFX, FXML, and CSS.

Post- requisites: The HIMA application should indicate if the user's input was validated. This includes String variable input for account creation and updates, as well as login. If the user accidentally causes an error, the system should display a message relative to the error that occurred.

Create Category Use Case

Pre-requisites: Create User Account and Login (see page 6)

Task Analysis: The Home Inventory Management application user will need to create specific categorical lists, or "Categories", to divide their inventory under. The user can select the "Categories" button in their Dashboard to enter the category feature, whereafter the user can write out a category name and its ID number and click the "Add Category" button to register a category for item classification.

Task Analysis Summary:

1. The user logs in with proper credentials
2. The application displays the Dashboard GUI
3. The user selects "Categories"
4. The user enters category name, and category ID
5. The user selects "Add Category"

Functional Requirements: The User can create and log into a private account so that they may access the "Categories" feature provided in each private accounts' dashboard. An additional functional requirement would be that the user created categories become available in the Item/Inventory features "Filter Categories" operation, where the user can filter through which items have been associated to which user created category.

Non-functional Requirements: The user can create a new category and instantly see that it has been added to their list of users created categories in a table above the category creation text fields and buttons. The user should also be able to see instantly that a category has been removed once they have chosen a category to delete.

Primary Design Challenges: One of the primary design challenges was to have newly created categories register in the "select filter" drop down box in the Inventory feature; additionally, any category that was deleted would also need to be removed from the "select filter" drop down box in the Inventory feature. This required that we use additional array lists to filter the data based on the user's selection. Once this data was filtered, it could be loaded into an ObservableList, which was loaded into the TableView.

Post-requisites: The user must be able to add new categories and delete existing categories and see that those categories are added or removed from both their categories list and their Inventory features' "select filter" operation.

Create Item Use Case

Pre-requisites: Create User Account and Login (see page 6)

Create Item Task Analysis: The Home Inventory Management Application (HIMA) user will need to be able to create and add items to their Inventory and affix categories to categorize those items. The user can use the Inventory feature to select an items relevant category, provide a name for the new item, provide an ID number for the new Item, and provide a location to specify where the item is in their homes. The Create Item use case includes the following steps:

1. The user logs in with proper credentials.
2. The application displays the Dashboard GUI.
3. The user selects the "Inventory" button.
4. The user selects a category to assign to a new item, provides the items name, the items ID number, and the Items location in their homes.
5. The user selects "Add Item"

Functional Requirements: The user can create and log into a private account so that they may access the "Inventory" feature provided in each private account dashboard. An additional functional requirement would be that the user created Items are immediately visible within the users Inventory table upon creation, and the existing Items are immediately removed from the users Inventory table upon deleting an item.

Non-functional Requirements: The Items the user creates are still held within the Inventory table of the Inventory feature when the user logs out and back into their account. An additional non-functional requirement would be that once a user deletes a certain Item from their Inventory and logs out and then back into their account, that that Item deleted doesn't persist in their Inventory table.

Primary Design Challenges: The primary design challenge in the Create Item use case was that each users Items are privately stored into their own private account, and that any changes to their list of Inventory Items persisted between application use sessions. To accomplish this our team created a series of Array Lists for a user's Inventory that were associated with each private user accounts email address and had each operation (Adding and Deleting Items) either store a new Item into the private user accounts specified individual Array List or remove a specified item from the private user accounts specified individual Array List.

Post-requisites: The user must be able to see the new Item that they have added to their Inventory and be able to manage the Inventory Item (Delete or Copy).

Dashboard Search Use Case

Pre-requisites: Create User Account and Login (see page 6)

Dashboard Search Task Analysis: The Home Inventory Management Application (HIMA) user may need to search for specific keywords, items, or item IDs in their inventory. Instead of manually searching through the inventory table. The user can use the search text field on the Dashboard GUI. The Dashboard Search use case includes the following steps:

1. The user logs in with proper credentials.
2. The application displays the Dashboard GUI.
3. The user types a search term and selects search or types enter on keyboard while text field is focused.
4. Any items that contain the string entered in the search term will appear in the inventory table.
5. The user can review these items or select Dashboard to return to another search.

Functional Requirements: HIMA allow users to create, view, manage, and search inventory and category data to provide a smoother experience for householders. This use case specifically addresses the process of searching a user's inventory data from the Dashboard GUI. The search functional requirement allows the user to quickly find specific items by searching the item name, location, ID, or category instead of having to search through the inventory manually.

Non-functional Requirements: A functioning Windows or Mac computer with access to download NetBeans IDE in order to run the application. The application also requires a mouse or touchpad, keyboard, and a monitor to interact with the user interface.

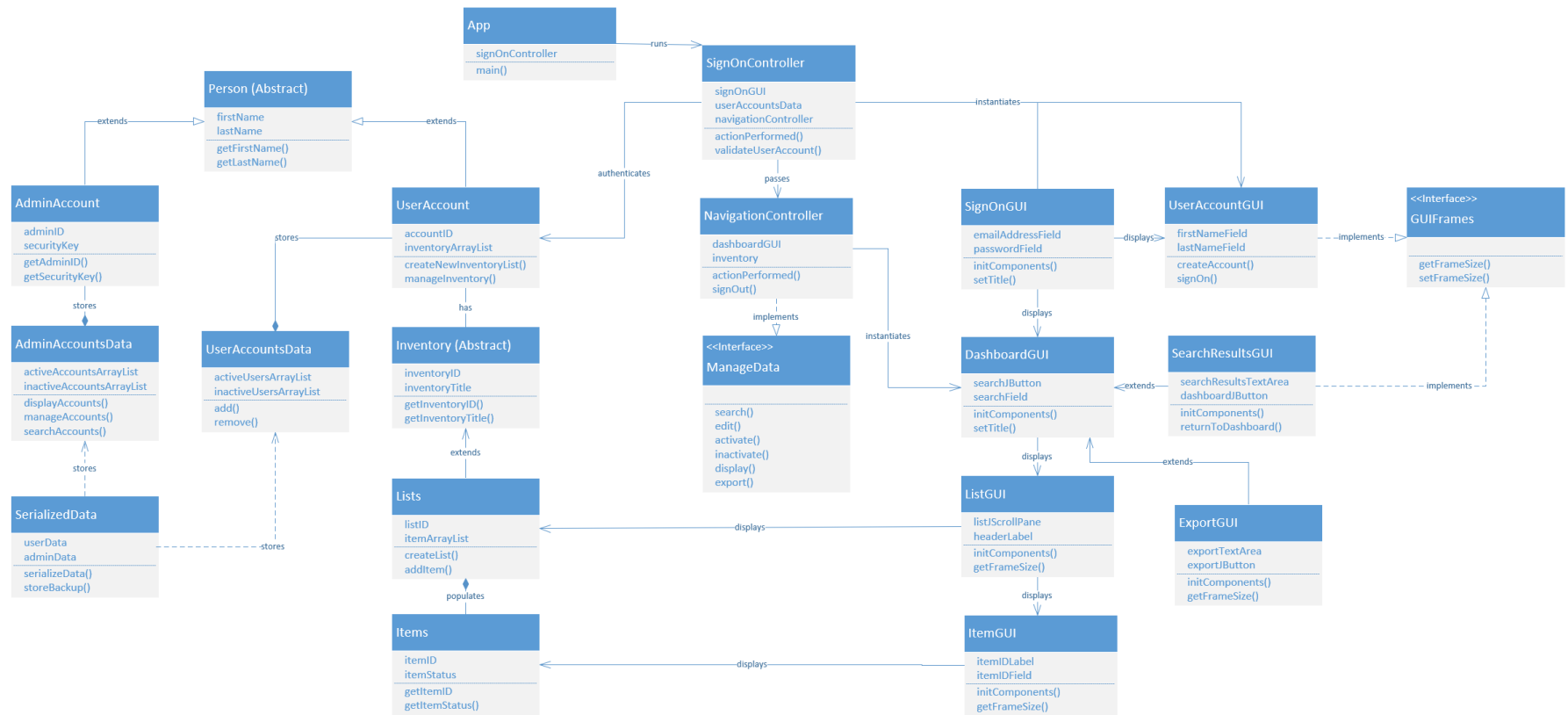
Primary Design Challenges: One of the primary challenges with this use case was creating a search function that only retrieves inventory items that belong to the user that is currently logged in. Additionally, our goal was to provide visibility of system status to indicate whether search results were found. This required the use of Cascading Style Sheets (CSS) to edit class styles based on an error message or search results match. Furthermore, any styles that were added needed to be removed using a `refreshFormatting()` method each time a new button was clicked.

Pre-requisites: the user must have login credentials to sign in. Therefore, the user has already created a username and password before logging in.

Post-requisites: the user must be able to either return to the dashboard or use the table to manage their inventory.

3.0 Structural Design

3.1 Implementation Class Diagram



3.2 Class Responsibilities

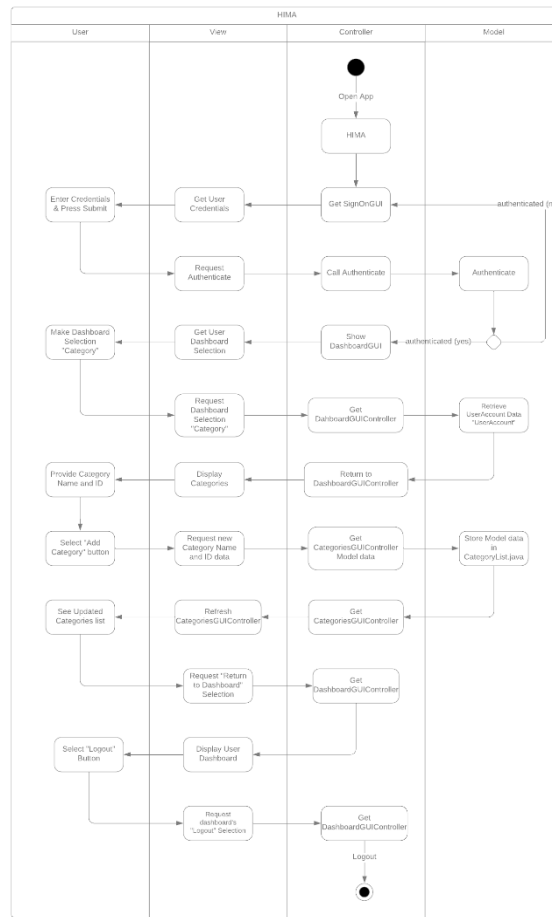
Class	Responsibility
HIMA_FX	User will run this class to start the application. Contains a start method, which loads the SignOnGUI.fxml file.
Category	Model class that constructs category objects. Contains methods to get and set category attributes.
CategoryList	Model class that stores an array list of categories. Contains methods to add items to the category list.
CategoryGUI	Graphical user interface that displays list model data and allows the user to view specific list or item data. Accessed once the user selects the categoriesButton on the DashboardGUI.
CategoryGUIController	Controls the fxml view and uses serialized data collection to write updates to categories as well as providing logic for changes to view states of the CategoryGUI.
CreateAccountGUI	Graphical user interface that allows user to enter text fields to create account and return to sign on.
CreateAccountGUIController	Controls the fxml view and uses serialized data collection to write updates to user account as well as providing logic for changes to view states.
DashboardGUI	Primary GUI containing the search bar and Buttons to navigate GUIs.
DashboardGUIController	Primary controller class that is called once user has been authenticated by the SignOnController. This controller will hold logic to navigate GUIs and update model data based on user input.
ExportGUI	Graphical user interface app that displays a text area field containing the data in lines of text. This class will also contain controller code that exports the data to a .txt file.
ExportGUIController	Controls the fxml view and uses serialized data collection to get inventory items to export as text based on the user category selections.
ForgotPasswordGUI	Graphical user interface that displays forgot password text field and button for user to reset password to a designated default.
ForgotPasswordGUIController	Controls the fxml view and uses serialized data collection to write updates to user account as well as providing logic for changes to view states.
himaCSS	Customized CSS file designed for the HIMA application.
Item	Model class building item attributes as itemID and itemName. Items collaborate with the ItemList class, which stores item objects.
ItemList	Provides methods and objects to store items in an array list.
InventoryGUI	Graphical user interface that displays item model data and allows the user to cycle through, edit, and add items to a list. Accessed once the user selects the inventoryButton on the DashboardGUI.

Class (Continued)	Responsibility (Continued)
InventoryGUIController	Controls the fxml view and uses serialized data collection to write updates to inventory items as well as providing logic for changes to view states.
ManageData	Interface allowing classes to implement abstract methods. Multiple GUI classes implement this interface.
Person	Abstract class holding person attributes such as firstName and lastName that are inherited by UserAccount and AdminAccount classes.
SerializedDataCollection	Utilizes Serializable interface to store and backup data between sessions. Collaborates with UserAccountList, ItemList and CategoryList to create serializable files.
SignInGUI	Creates graphical user interface for user sign on. This is the first Stage that will be displayed when the application is run.
SignInGUIController	Communicates with SignInGUI to authenticate user data and contains logic to update fxml view state.
UserAccount	Model class that constructs user account. This class extends person to help create new accounts.
UserAccountList	Provides methods and objects to store UserAccounts in array lists and manage the list of active and inactive user accounts.

Figure 3.2 The HIMA Class Responsibilities: The table above details the 25 classes developed for the final product of the HIMA software.

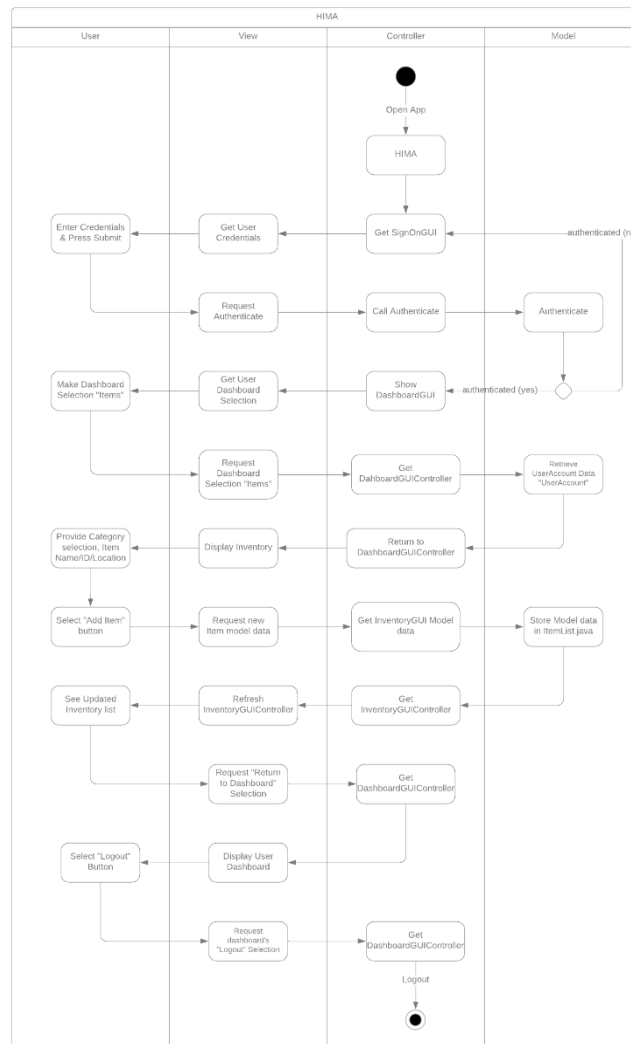
4.0 Flow-of-Control: Activity Diagrams

Create Categories Flow of Control Diagram



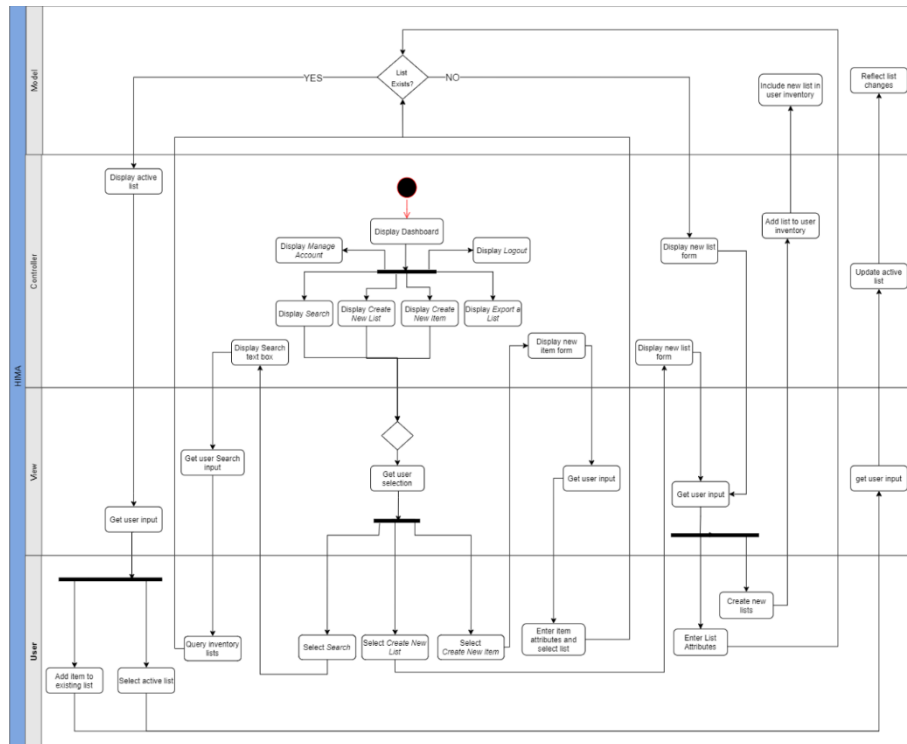
The Add Category UML Activity Diagram shows the sequence of activities which occur when a user is authenticated after signing on and selects the “Categories” selection of their Dashboards options to utilize the Add Category feature in their Categories feature. After sign on, the user can select the Category button to be presented with their private accounts Categories list. Once the user accesses their Categories, they are presented with two text entryfields to enter a Categories name and ID number. Upon providing a new Category Name and ID number the user may click on the Add Category button to create a Category to add to their Categories table. The CategoriesGUIController then passes the new Categories data to the CategoriesList.java class where it uses the writecategoriesListFile() method to store the new Category data into a serializable file, then refreshes the CategoryGUIController to display the newly added Category in the users Category table. Once the user has finished adding new Categories to their Categories table, they may exit the program by clicking on the Dashboard button to return to their Dashboard and the Logout button to fully log out of the Application.

Create Items Flow of Control Diagram



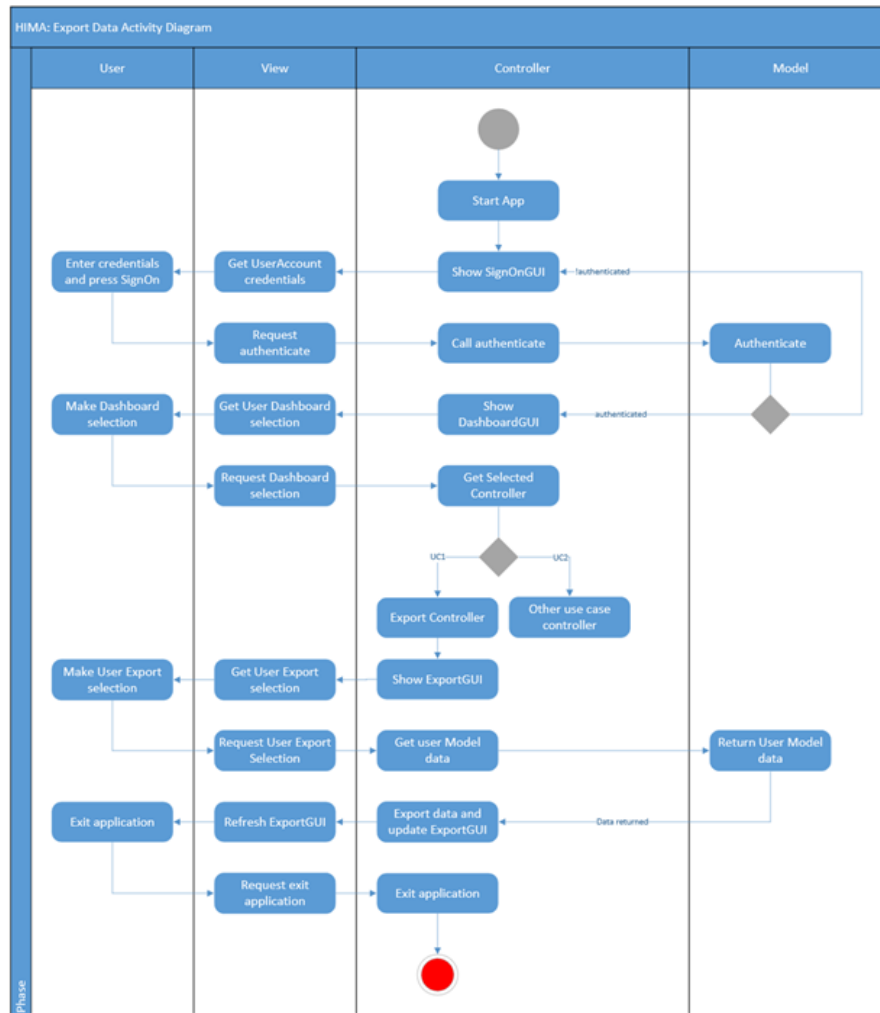
The Add Item UML Activity Diagram shows the sequence of activities which occur when a user is authenticated after signing on and selects the “Inventory” selection of their Dashboards options to utilize the Add Item feature in their Inventory. After sign on, the user can select the Inventory button to be presented with their private accounts Inventory list. Once the user accesses their Inventory, they are presented with a categories drop-down box selector, and text entry fields for the name, ID number, and location text values for a newly created Item to add to their Inventory table. Upon selecting the new Items categorical definition and providing the name, Item ID number, and Item location, the user may click on the “Add Item” button to register their new Items to their Inventory table. The InventoryGUIController then passes the new Item data to the ItemList.java class where it uses the writeItemListFile() method to store the new Item data into a serializable file, then refreshes the InventoryGUIController to display the newly added Item in the users Inventory table. Once the user has finished adding new Items to their Inventory, they may exit the program by clicking on the Dashboard button to return to their Dashboard and the Logout button to fully log out of the Application.

Search Inventory Flow of Control Diagram



The Search Inventory button is located in the DashboardGUI, and it allows the user to search for item names, IDs, locations, and categories from the dashboard. The control may follow in different directions based on the actions taken by the user. For example, if the user searches for an item that is found in their inventory, the data will be passed to the InventoryGUIController to display those items. However, if the users search does not return any results, the control remains with the DashboardGUIController.

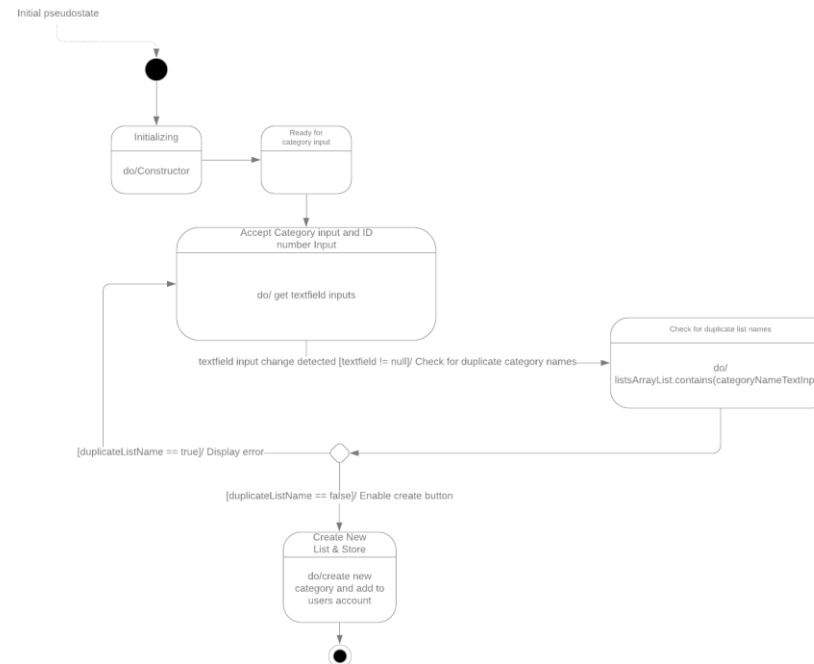
Export Data Flow of Control Diagram



The Export Data UML Activity Diagram shows the sequences of activities that happen when a user is authenticated after signing on. After sign on, the user can view multiple options in the Dashboard window. Once the user selects the Export button, they are able to view the ExportGUI to confirm the export of the selected user inventory data. The controller retrieves the data from the model classes and exports the data to a .txt file. Once the file is exported, the ExportGUI refreshed with a message indicating that the export was successful, and the user closes the application by selecting the exit button.

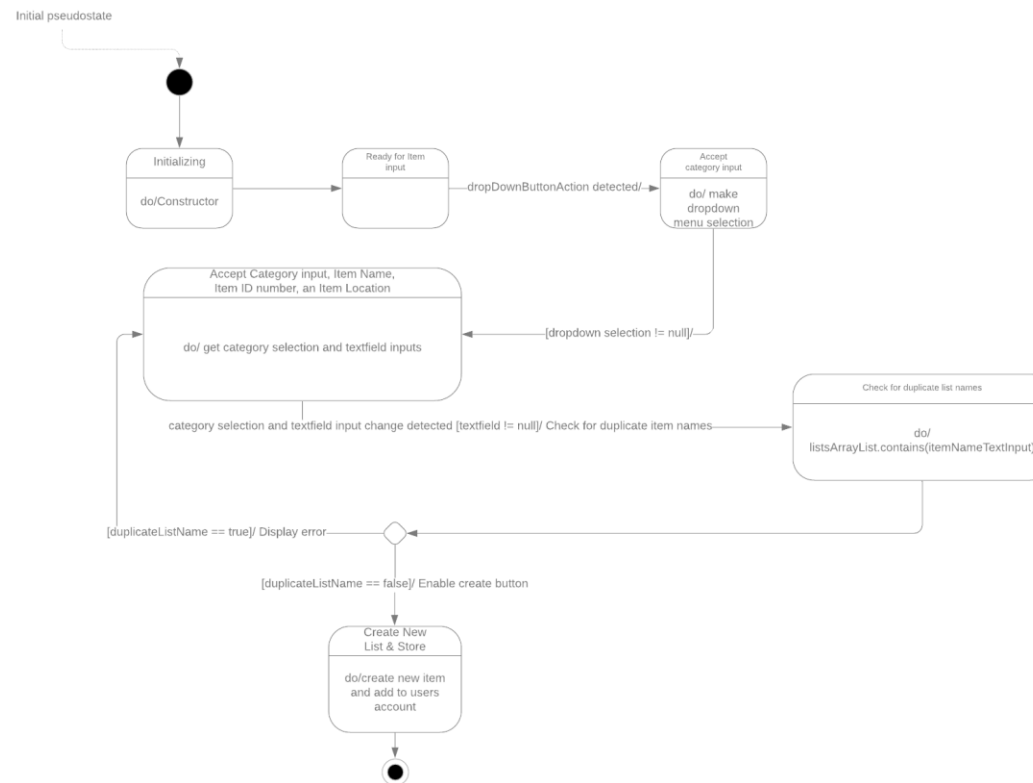
5.0 User Interface Classes: State Diagrams

Add Categories State Diagram



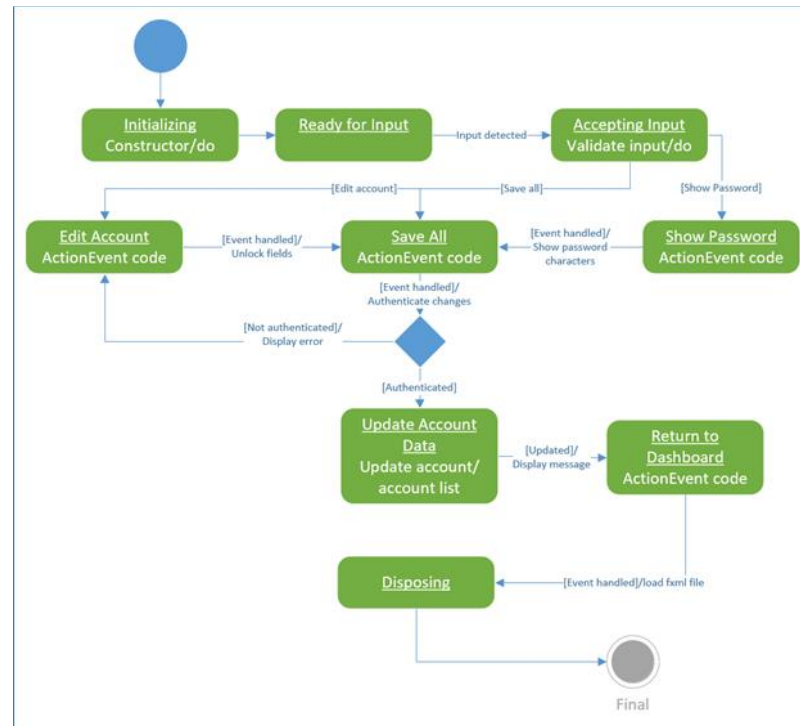
The Add Categories State Diagram considers the different actions and states that may take place in the creation of a new Category in the Categories feature of the application. In this diagram we see that the user must simply supply both a name and Id number for a category. If the category name textfield detects that the user has already created a category with a similar name, it will display an error message describing that a similar category already exists. If the name for the categories is unique, the category will be created and added to the users private account Categories Table.

Add Item State Diagram



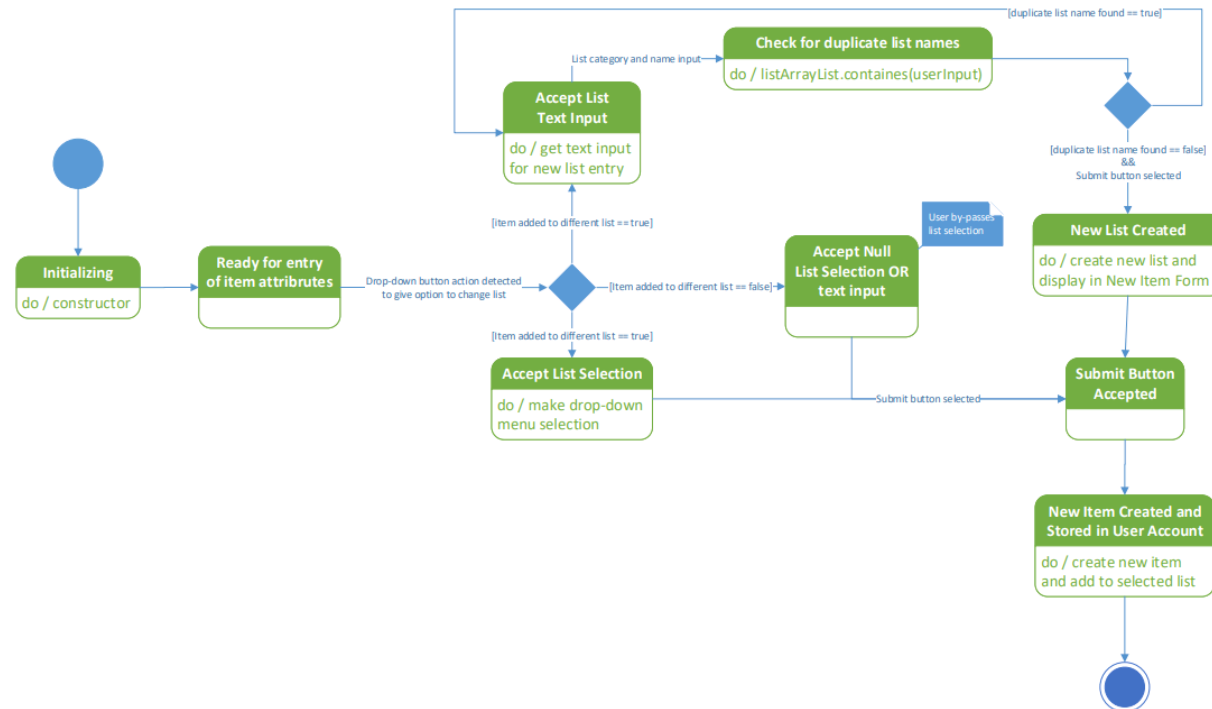
The Add Item State Diagram considers the different actions and states that may take place in the creation of a new Item in the Inventory feature of the application. In this diagram we see that the user must first select a category to categorize the new Item under by choosing a drop-down box option in the new Item creation field below the Inventory table. Once the category for the Item has been selected, the user will be allowed to enter the new items name, id number and location. If the Item name is not a duplicate, the creation button will register the new Item in the Inventory table; should the Item name be a duplication of an existing name, the user will receive an error message and be asked to rename the Item.

Edit Account State Diagram



The Edit Account GUI object state diagram considers the different actions and states that may take place within the application's user interface. To begin, the view class is initialized and becomes ready for user input. The input occurs in the form of button clicks, check box clicks, and text field entries. The text fields to edit account information are initialized with `setDisable()` in order to lock the fields. Once the edit button is selected, the text fields are unlocked, allowing the user to enter text to update their account. The user may change their personal information, including the email address if another user is not registered with the same address. Once the user has finished making changes, they can select the "Save All" button. This button will authenticate the change and return an authentication Boolean. After changes are saved, a message is displayed to acknowledge the update, and the text fields are locked again. While making these changes, the user can also choose to select the "Show Password" check box in order to display the text in the password fields. Once finished with the GUI, the user can select the "Dashboard" button to dispose of the `EditAccountGUI` and load "`DashboardGUI.fxml`".

Add Item from List State Diagram



Create New Item from Existing Category (List) State Diagram: The diagram above demonstrates a user creating a new item from an existing list of items. The user has the option to change the list (as seen in the first diamond) by adding it to an existing category or creating a new category from the drop-down selection. This is one of two routes a user can take for creating a new item in the HIM application.

6.0 Components and Distribution

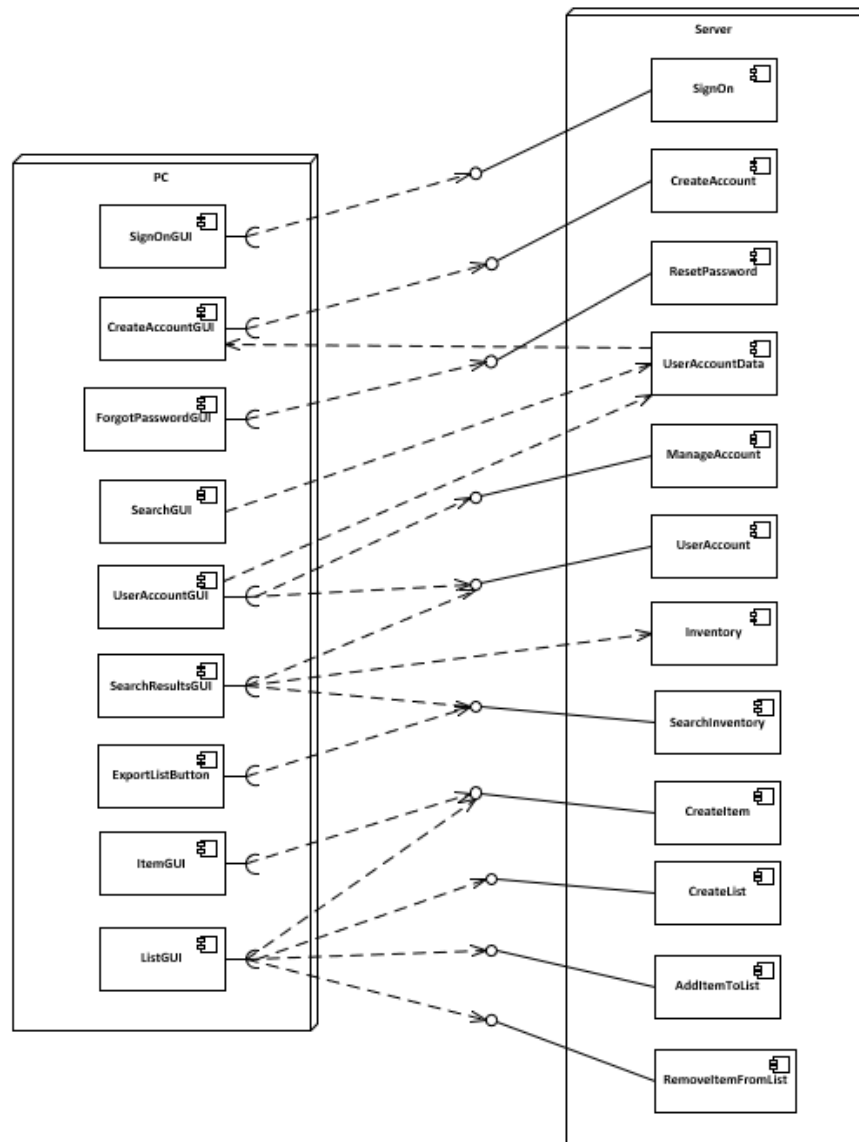


Figure 6.0 The HIMA Component-Deployment Diagram: The diagram above displays the deployment strategy for the HIMA software. The anticipated number of users is expected to reach 10,000 within the initial deployment phase.

7.0 User Manual



The screenshot shows a web application window titled "HIMA: Sign On". At the top center is a blue house icon. Below it, the text "HIMA: Sign On" is displayed in bold. There are two input fields: "Email" with the text "testUser" and "Password" with the text "testPass". To the right of the password field is a checkbox labeled "Show password" which is checked. Below the input fields are three buttons: "Sign On", "Forgot Password", and "Create Account". At the bottom center, the text "Welcome home!" is displayed in blue.

This screen shot is of the initial sign on screen for the Hima Application. In it the user will enter their Email and Password to log in to their private user account. If the user does not have already have an account to log into, they may use the "Create Account" feature to create a new account (details on the creation of a new account are below). The user may enjoy two auxiliary features of showing their written password by clicking on the check box next to the password field, and ask that an email be sent in the event that they forget their password.

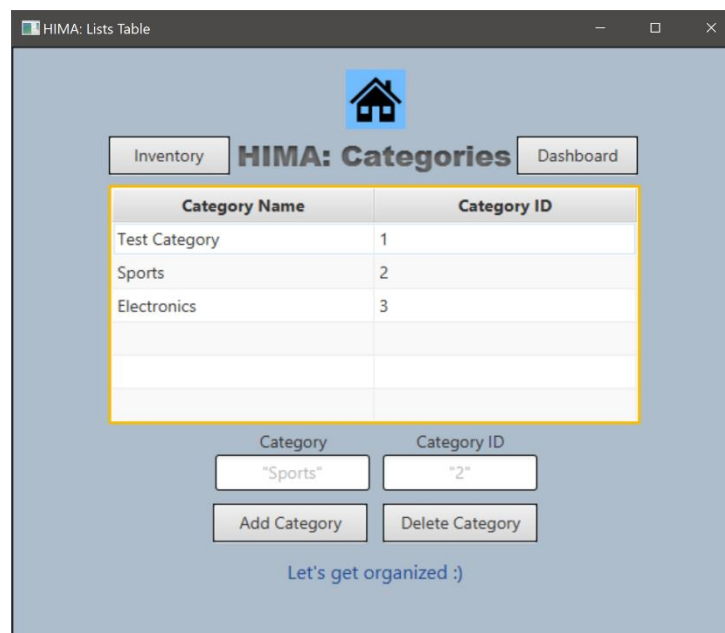


The screenshot shows a web application window titled "HIMA: Create Account". At the top center is a blue house icon. Below it, the text "HIMA: Create Account" is displayed in bold. To the right of this text is a "Sign On" button. There are five input fields: "Email:" with the text "Email@address.net", "First Name:" with the text "Create", "Last Name:" with the text "Account", "Password:" with four dots, and "Confirm Password:" with four dots. To the right of the password fields is a checkbox labeled "Show Password" which is unchecked. Below the input fields are two buttons: "Add" (highlighted in yellow) and "Confirm". At the bottom center, the text "Select Add to create an account and Confirm to save your submission." is displayed in blue.

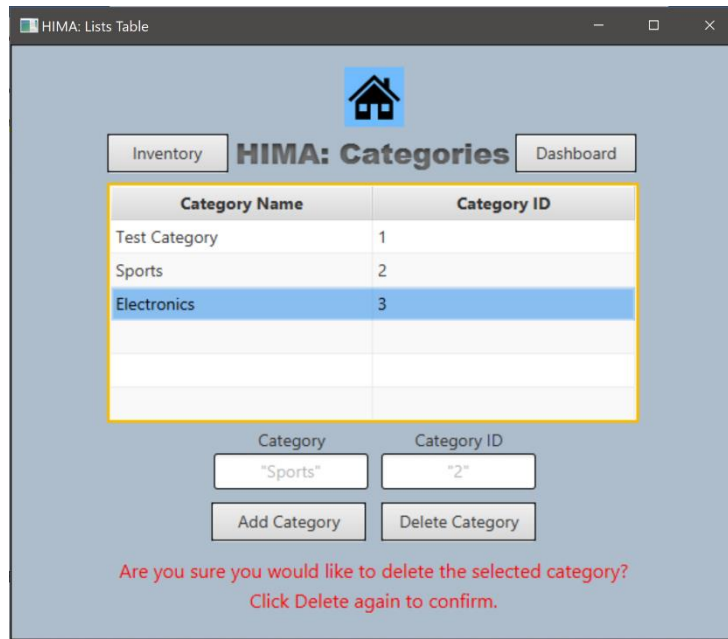
This screen shot is of the Create Account feature of the HIMA application, this is where a new user will enter their email, first and last name, and password to create their account. As a security feature the user must enter their password again in the confirm password text field to confirm their password submission.



This screen shot is of the private user accounts Dashboard. The dashboard is the main navigational hub for the HIMA. The user may search for existing Items by entering the name of an Item in the search bar and clicking on the “Search” button to navigate to their Inventory. The “Categories” button will allow the user to navigate to their Categories repository where they will be able to manage their created categories for items. The “Inventory” button will allow the user to navigate to their Inventory repository where they will be able to manage their created Items for their Inventory. The “Export” button is a beta feature for premium users.



This screen shot is of the private user accounts’ Categories repository. In this feature the user can create new categories or delete existing categories. To create new categories the user must supply a category name and ID and click on the “Add Category”.



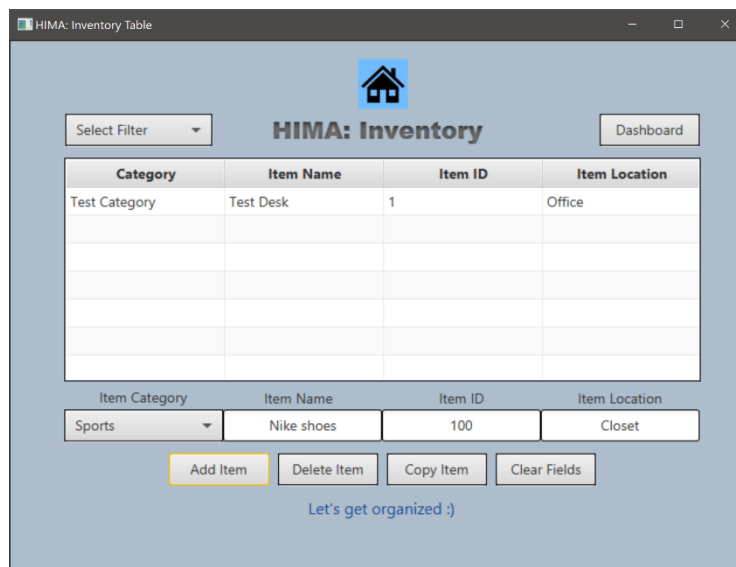
Category Name	Category ID
Test Category	1
Sports	2
Electronics	3

Category: "Sports" Category ID: "2"

Add Category Delete Category

Are you sure you would like to delete the selected category?
Click Delete again to confirm.

This screen shot describes the actions needed to delete an existing category. To delete an existing category, the user must select a category and then click on the “Delete Category” button. Once the user has clicked on the “Delete Category” button, they will be prompted to confirm their deletion of the category in question by re-clicking the “Delete Category” button again.



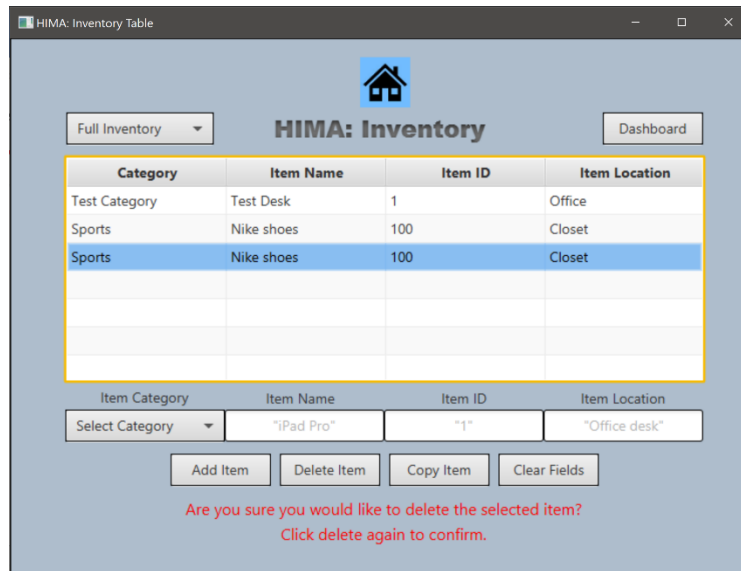
Category	Item Name	Item ID	Item Location
Test Category	Test Desk	1	Office

Item Category: Sports Item Name: Nike shoes Item ID: 100 Item Location: Closet

Add Item Delete Item Copy Item Clear Fields

Let's get organized :)

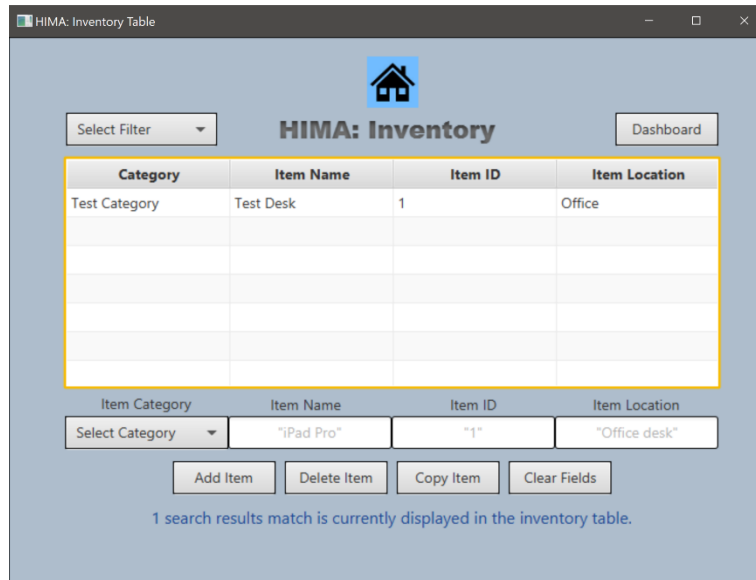
This screen shot describes the process of adding a new item to the users Inventory. The user must select a category definition from the drop-down box, add text for the new item’s name, item ID number, and item location, and then click on the “Add Item” button to register a new item in their Inventory table.



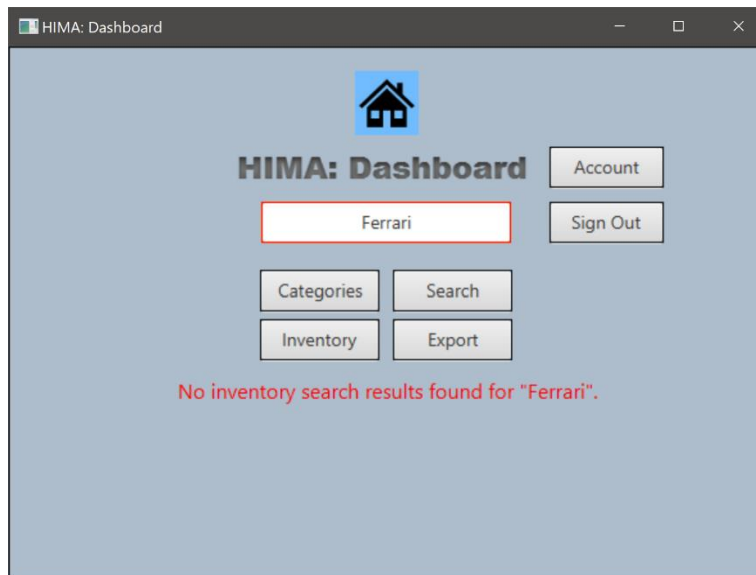
This screen shot describes the process of deleting an existing inventory item from the users Inventory. The user must select an Item to delete, click on the “Delete Item” button and follow the confirmation directions.



This screen shot describes the process of searching for an existing item in the users Inventory. The user must type out the name of an existing Item and click on the “Search” button to search for the item in their inventory.



This screen shot describes the positive result action of finding an item after using the search feature on the Dashboard.



This screen shot describes the negative result action of failing to find an item after using the search feature on the Dashboard.