# 

**Lab Assignment #4**

**Submitted to: Professor SHAHID BHATTI**

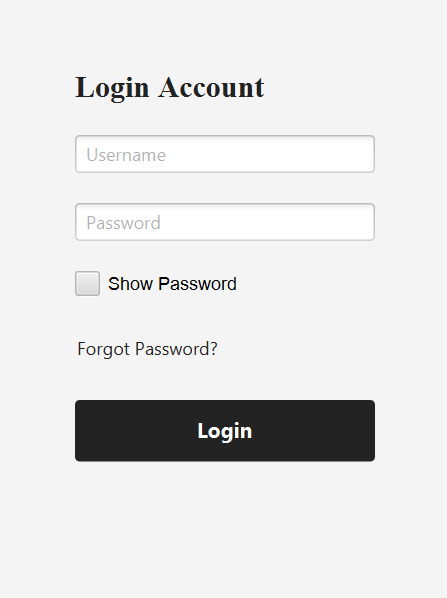
**Submitted by:**

**1:Muhammad Saad Badar (sp24-bse-082)  
  
2:Hafiz Abdullah Aslam (sp24-bse-028)  
  
3:Sohaib Athar (sp24-bse-115)**

**Food Ordering and Management Application**

**Overview:**

The Cafe Shop Management System is a JavaFX application designed to handle **login**, **registration**, and **password recovery forms**. It uses the **AccountManager** class for user authentication and account management, and a **Dashboard** class for displaying features. It has external CSS for styling the app.

**Login Form**

* Allows existing users to log in.
* Validates user credentials.
* Toggles password visibility.
* Check if the fields are empty or not
* Hovering effect for **Login** button through **CSS**

**createLoginForm()**

* **Components:**
  + **TextField** for username input.
  + **PasswordField** for password input, with dynamic visibility toggle.
  + **CheckBox** to toggle password visibility.
  + **Hyperlink** for password recovery.
  + **Button** to initiate login.

### **Registration Form**

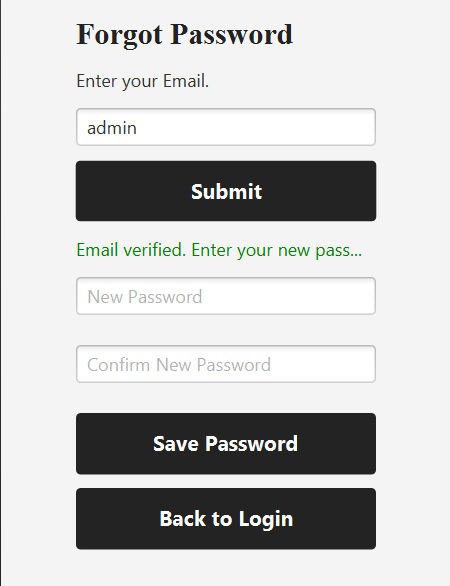
### **createRegisterForm** ()

* **Components:**
  + **TextField** for username and email input.
  + **PasswordField** for password and confirmation input, with visibility toggle.
  + **CheckBox** to toggle password visibility.
  + **Button** to register a new account.
  + **Label** to display validation messages.
* **Functionality:**
  + Validates username, email, and password fields.
  + Ensures password strength and matching.
  + Uses **AccountManager**.**register()** to create a new account.

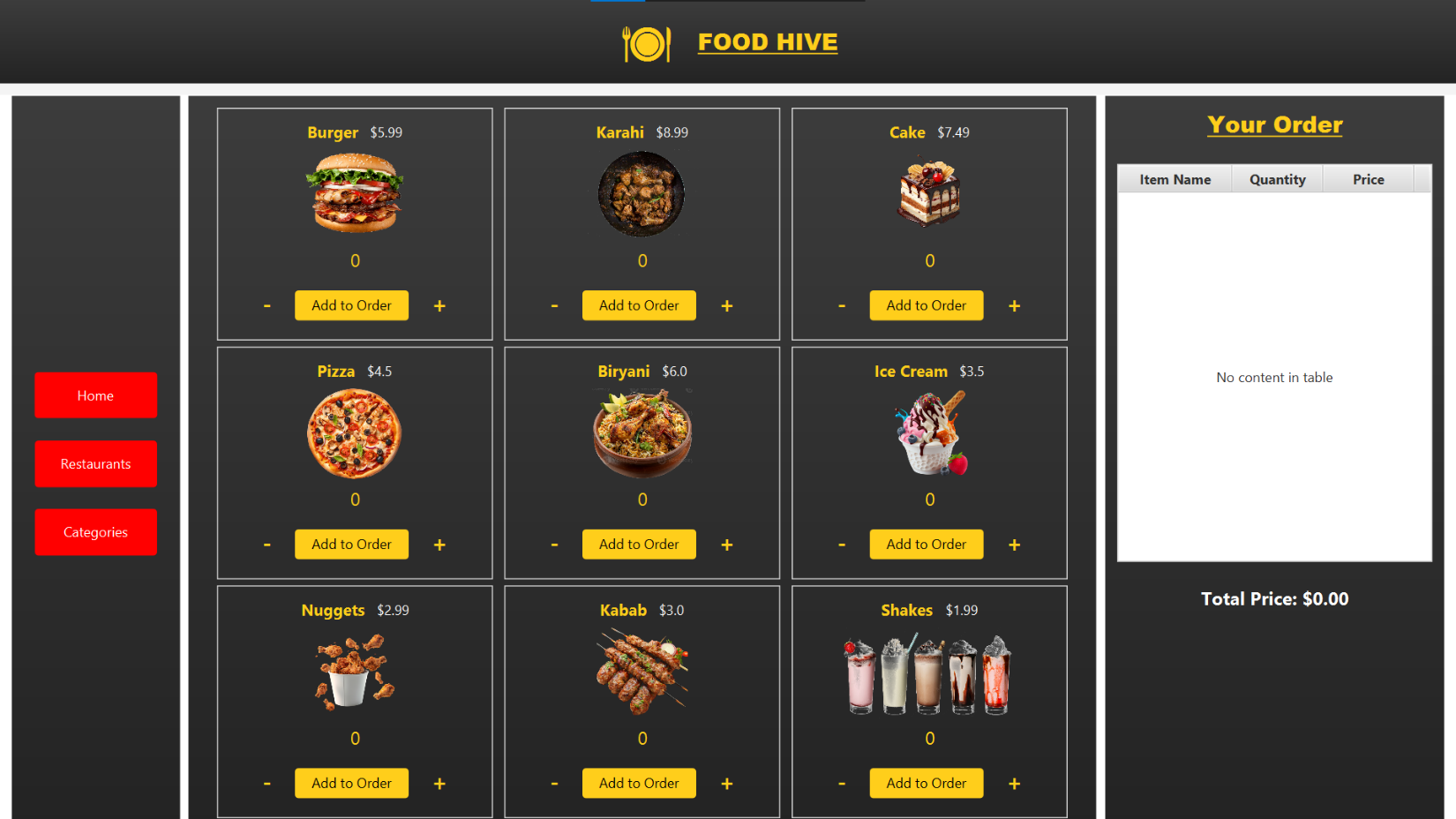
**Password Recovery Form**

**createForgotPasswordForm()**

* **Components:**
  + **TextField** for email input.
  + **TextField** for new password and confirmation input, with visibility toggle.
  + **Button** to submit email for verification and save the new password.
  + **Label** to display validation and success messages.
* **Functionality:**
  + Verifies email with AccountManager.isEmailRegistered().
  + Allows password update using AccountManager.updatePassword().
  + Validates new password strength and matching.



**Dashboard**

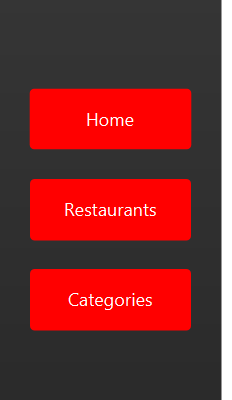
****

* Manages the layout and transitions between the home, restaurant, and order views.
* Implements multiple VBox containers for logical organization of UI elements.
* Utilizes TableView to manage and display the order summary.



**showDashboard()**

* **Purpose:** Initializes and displays the main dashboard.
* **Components:**
  + HBox for the top bar containing the application title and logo.
  + VBox containers (vBox1, vBox2, vBox3, showRestaurant) for managing different sections.
  + TilePane for item layout.

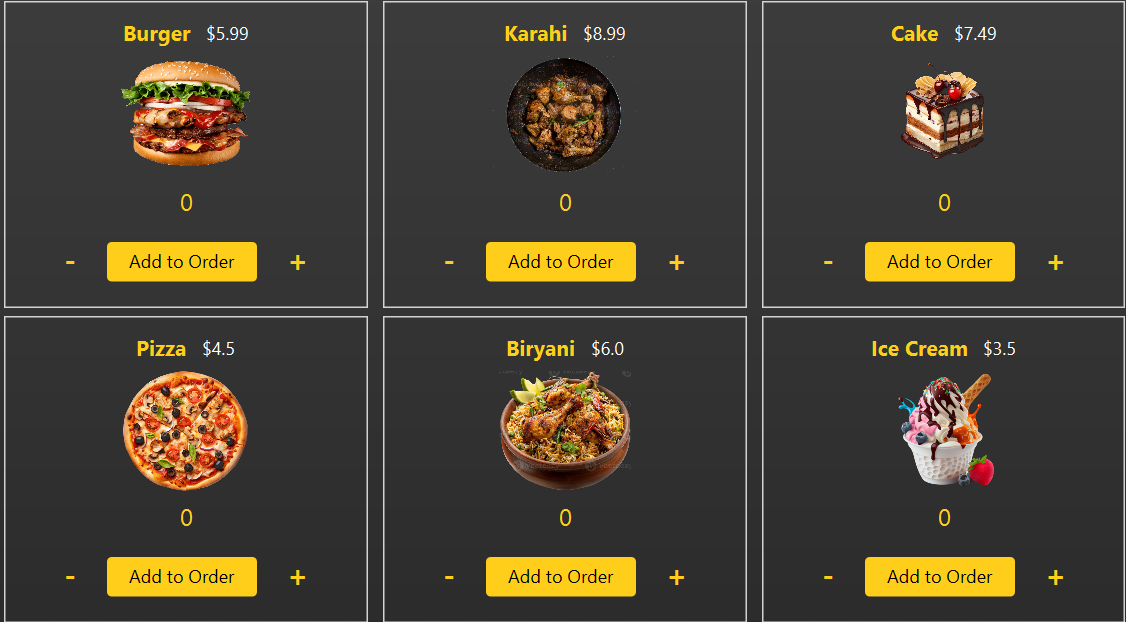


**setupVBox1()**

* **Purpose:** Configures the navigation menu with buttons for Home, Restaurants, and Categories.
* **Components:**
  + Buttons styled with CSS classes and assigned action handlers to toggle views.

**Navigation Bar (VBox1)**

* Contains buttons for navigation:
  + **Home Button:** Switches to the main item grid view.
  + **Restaurants Button:** Displays the restaurant listings.
  + **Categories Button:** (Future use).



**setupVBox2()**

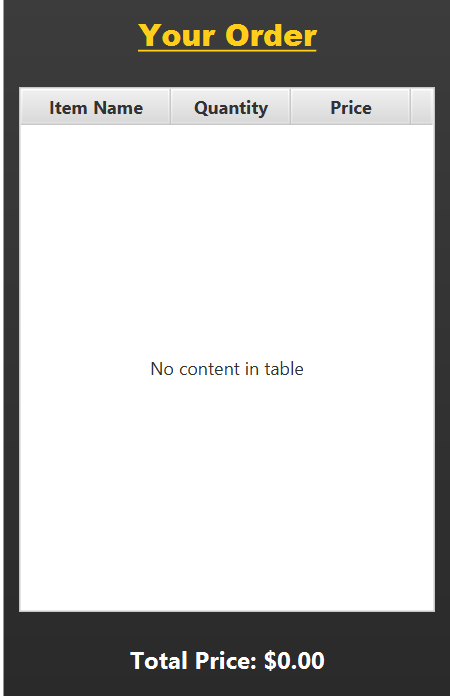
* **Purpose:** Sets up the item grid using TilePane.
* **Components:**
  + Displays items with images, names, prices, and quantity controls.
  + Adds functionality for adding items to the order.

**addItemToTable(String itemName, int quantity, double itemPrice)**

* **Purpose:** Adds an item to the order table or updates an existing item’s quantity and price.
* **Logic:**
  + Checks for existing items in the table before adding or updating.
  + Dynamically updates the total price.

**Item Grid (VBox2)**

* **Layout:** Uses TilePane for a flexible grid display.
* **Components:**
  + **ImageView:** Displays the item image.
  + **Name and Price Labels:** Show item details.
  + **Quantity Controls:** Includes "+" and "-" buttons for customization.
  + **Add to Order Button:** Adds selected items to the order table.

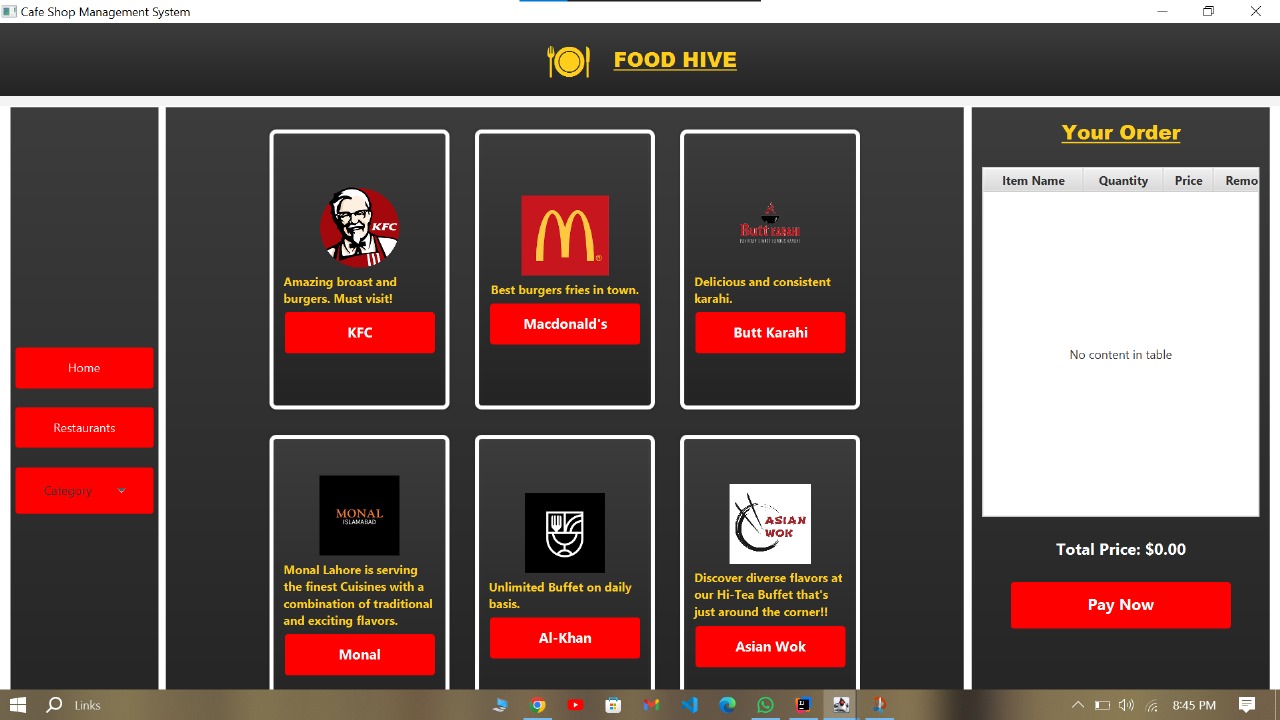


**setupVBox3()**

* **Purpose:** Configures the order summary table and calculates the total price dynamically.
* **Components:**
  + TableView with columns for item name, quantity, and price.
  + A total price label updated on table changes.

**Order Summary (VBox3)**

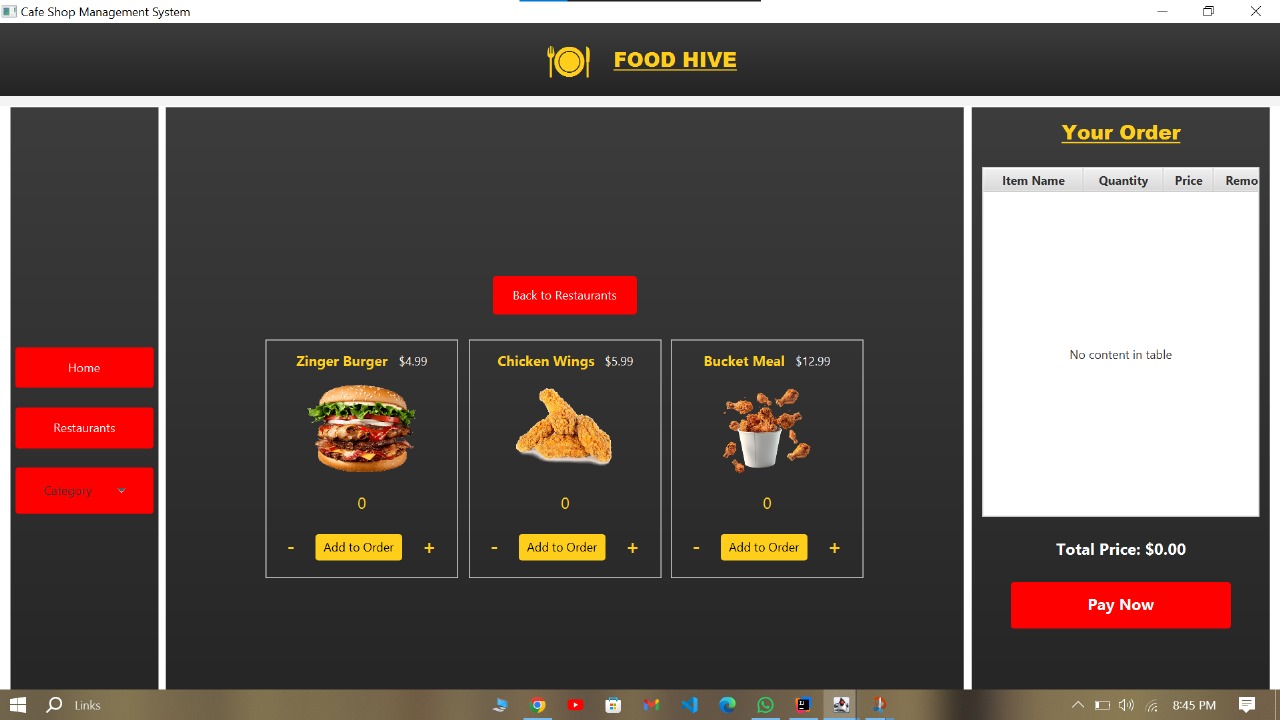
* **Layout:** Uses TableView to display order details.
* **Columns:**
  + Item Name
  + Quantity
  + Price
* **Dynamic Updates:** Automatically recalculates total price as items are added or modified.
* **Enhancements:**
  + Designed for easy integration with payment systems.
* **showRestaurants()**



* **Purpose:** Displays a list of restaurants with their names, descriptions, and images.
* **Components:**
  + VBox for restaurant details.
  + Action handlers for navigating to restaurant-specific details (future enhancement).
  + Includes placeholder images and error handling for missing resources

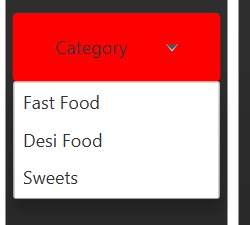
**Restaurants View (showRestaurant)**

* Displays a list of restaurants with:
  + Name (Button).
  + Description (Label).
  + Future functionality for restaurant-specific navigation.
  + Placeholder for ratings or user reviews.
* **RestaurantDishes**
* In the show restaurants method   
  when any restaurant is pressed the selected restaurant dishes are presented as shown



* The above given image shows the dishes from the specific restaurant “KFC”

**showCategories()**

****

* **CategoriesButton**

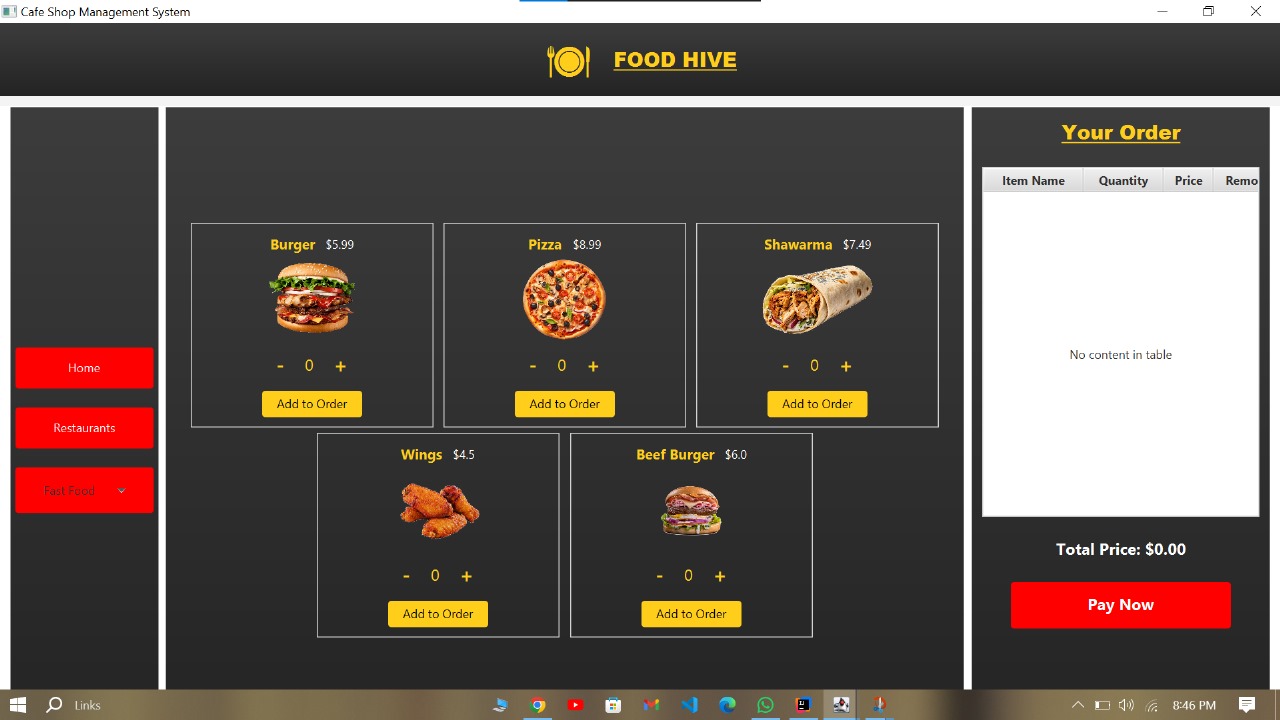
A **ComboBox** that displays three:

* FastFod
* Desi Food
* Sweets

***Category Selection***

If any category is selected , the selected buttons method is called which have a certain vbox that displays the require data

* As in the image below
* Each category has its own vbox having items of that category.
* The below give image shows the category of Fast Food



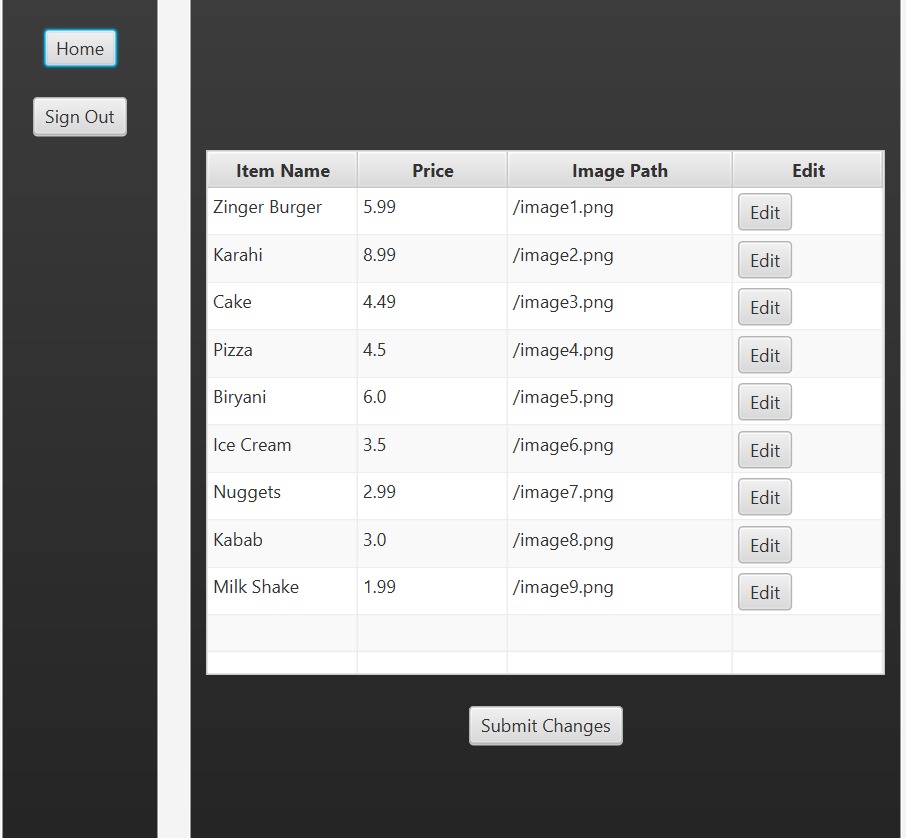
* **ADMINDASHBOARD**
* Dashboard for admin to change the item

NAME

PRICE  
IMAGE

* **Components:**

It uses a tableview and ask implementation as per request from Admin



* When edit button is pressed it ask from Admin the specific changes, admin wants.
* When the admin, presses the submit button, it saves it in file .
* This changes will also appear in the customer dashboard and work as it is .