UNIVERSITI MALAYA

FOP ASSIGNMENT: TECHNICAL REPORT

NAME	MATRIX NUMBER
SAYYID SYAMIL BIN SYED MOHAMED	S2194337
MUHAMMAD DANISH IMAN BIN ABD RAHMAN	22002450
AHMAD UZAIRIL AQIL BIN MOHD NAZZRI	22001832
ABDUL RAHMAN BIN SHAMSUDIN	22002623
MOHAMAD HAIQAL BIN BUJANG MOHAMAD ALI	22001995

COURSE CODE : WIX1002

COURSE NAME : FUNDAMENTALS OF PROGRAMMING

CLASS : OCCURRENCE 9

LECTURER : DR. LIM CHEE KAU

1. ASSIGNED PROJECT:

Thrifty Timmy wants to save money by finding the best prices for things he wants to buy. He noticed that prices could vary at different places and even change over time in the same place. Although he's good at finding deals, he needs help from a team of experts to create a program called **Price Comparison.** This program will make it easier for him to find the best prices.

2. REQUIREMENTS:

Price Comparison program required two things:

i. **File Description.** Three CSV files are needed for this project which is :

- **lookup_item.csv** (This file contains the relationship between item_code and item description)
  - item_code
  - item
  - unit
  - item_group
  - item_category

- **lookup_premise.csv** (This file contains the relationship between premise_code and premise description)
  - premise_code
  - premise
  - address
  - premise_type
  - state
  - district

- **pricecatcher_2023-08.csv** (This file contains relationship between date, premise_code, item_code and price in August 2023)
  - date
  - premise_code
  - item_code
  - price

ii. Basic Functionalities

- **Login / Registration**
  - Allow the user to register a new account and store his personal details.

- **Read the required three CSV files and store the data in classes.**

- **Item visualisation**
  - Display the top five cheapest sellers for particular item
  - Display a chart showing the price trend of particular item across the day in a month

- **User interface**
  - Allow the user to modify the account details (change password / username)
  - Create a selection screen based on item categories
  - Create a search function to allow the user search for specific item

- **User Experience**
  - Display selected item details and allow the user to modify the item details in the CLI
  - Create a shopping cart to allow the user to add multiple items. User can view the cheapest seller for all selected items in the shopping cart.
  - Suggest the cheapest shop that contains all the items selected.
    If you pick 10 items, the app suggests the fewest shops to get the best price. It prioritizes visiting the least number of stores. For example, if you can get all 10 items in Shop A for RM 100 or all in Shop B for RM 120, choose Shop A. If it's 9 items in Shop A and 1 in Shop B for RM 200 versus 8 in Shop A, 1 in Shop B, and 1 in Shop C for RM 150, still choose Shop A. You can also filter to only see shops in KL.

- **Data storage**
  - The program stores the user details, shopping cart details, and any changes you make to items. Next time you open it, the program automatically gets this information for you.

3. ADDITIONAL CHALLENGES

- How can we make my program more user-friendly by adding a Graphical User Interface (GUI)? Which tools like JavaFX or Swing can help create an attractive UI, and how do I improve chart visuals beyond basic command-line graphs?

- What's the benefit of using a database in my shopping app? How can we let users save their shopping cart and import data just once? Should we use MySQL, Oracle DB, or MongoDB, and is online hosting for the database a good idea?

- Implementing a fuzzy search function improves item search by accommodating partial names and typos. Consider using effective fuzzy matching algorithms for better results.

- Take inspiration from Shopee/ Lazada to make this a fully functional shopping application.

4. APPROACHES

Firstly, we simply coded the whole program to aid users create accounts and help them to find the item that has the best price. We coded the register and login for users to create their account and store the details locally, so that in the future, they can just login it. These codes ask the user to enter their username and password. Then, we debugged the program. After that, we had an online discussion between group members to think of the way to import the required three CSV files and how to make the interface of the program beautiful and user-friendly. So, we plan to use Java Databased Connectivity (JDBC) and Java Swing.

For the database, we use JDBC to connect with Java applications. Utilizing the MySQL JDBC driver and ‘java.sql.Connection’ interface, it enables the execution of SQL queries for data retrieval and updates. In handling CSV files, JDBC and some libraries allow us to read data and perform operations such as inserting or updating records in MySQL. Additionally, JDBC can export data from MySQL tables to CSV files, providing a comprehensive solution for data exchange and manipulation between Java applications, MySQL databases, and CSV formats.

Next, we coded the category and cart for the user to choose their items. All the items and shopping info will be stored locally and automatically retrieved the next time the user opens it. At the same time, the interface will show the top five cheapest, and the price trend based on the selected items. There is also a search button to help the user find the items easier as there are many items (also with variation). To make the interfaces more eye-catching and easier to use, we use Java Swing which is a GUI toolkit for Java. Java Swing provides a pluggable look and feel feature, which allows us to customize the appearance of the components. Moreover, it provides a rich set of components, including advanced components like JTree, JTable, and JSpinner.

5. CLASSES AND METHODS

LOGIN CLASS (use Java swing application for handling user login with a JDBC-connection based)

- **Database Connection:**
  - JDBC is used for database connectivity.
  - Database credentials are stored in the **Relate** class.
- **Login Class:**
  - Extends **JFrame** for the main login window.
  - Initializes components, sets the icon, and updates database credentials.
- **Database Operations:**
  - **getUserId**: Retrieves user ID based on the username.
  - **checkCredentials**: Verifies entered username and password.
- **GUI Components:**
  - JLabels, JTextFields, JPasswordFields, and JButtons for login/register.
  - Images used for branding.
- **ActionPerformed Methods:**
  - **jButton1ActionPerformed**: Checks credentials, retrieves user ID, and opens the main window.
  - **jButton2ActionPerformed**: Opens the registration window.
- **Main Method:**
  - Initializes the login window, sets the look and feel, and displays it on the Event Dispatch Thread.
- **Static Variables:**
  - **userId**, **getUsername**, and **getPassword** are static variables for user information.
- **Notes:**
  - Includes error handling for SQL exceptions.
  - Passwords are stored and compared in plain text (for simplicity).

REGISTER CLASS (use Swing-based GUI and JDBC connected database in Java)

- **Database Connection:**
  - Uses JDBC for database connectivity.
  - Database credentials stored in **Relate** class.
- **Register Class:**
  - Extends **JFrame** for registration.
  - Initializes components, sets the icon, and updates database credentials.

- **Database Operations:**
  - **registerNewAccount**: Inserts a new user into the database.
  - **isUsernameTaken**: Checks if a username is already in use.
  - **createUsersTable**: Creates the "users" table if it doesn't exist.
- **GUI Components:**
  - JLabels, JPasswordField, JTextField, and JButton for registration.
  - Images used for branding.
- **ActionPerformed Method:**
  - **jButton2ActionPerformed**: Registers a new account if passwords match, username is available, and registration is successful.
- **Main Method:**
  - Initializes the registration window, sets the look and feel, and displays it.
- **Notes:**
  - Error handling for SQL exceptions is included.
  - Password confirmation ensures accurate registration.
  - Checks if the username is available before attempting registration.

CART CLASS (use Swing to manage shopping cart, interacting with a database for cart data, and using CSV data for additional information. Users can find the best shop, remove items, and get the best prices for selected items.)

- **Variables:**
  - **public static String publicPath**: Public path variable.
  - **public List<String[]> itemDetails**: List for storing item details from a CSV file.
- **Constructor (Cart()):**
  - Initializes components, sets the window icon, and populates the table with cart data.
  - Calls methods to load districts and display cart data.
- **Methods:**
  - **loadDistricts()**: Populates a district dropdown from the database.
  - **displayCartData()**: Fetches and displays user-specific cart data.
- **GUI Components:**
  - JTable (**jTable1**), JButtons (**jButton1**, **jButton2**, **jButton3**), JComboBox (**jComboBox1**), JTextArea (**jTextArea1**), and JLabels with Icons.

- **ActionPerformed Methods:**
  - **jButton2ActionPerformed**: Removes selected items from the cart.
  - **jButton1ActionPerformed**: Finds the best shop for selected items and displays details.
  - **jButton3ActionPerformed**: Gets the best prices and sellers for selected items.
- **Helper Methods:**
  - **getBestShop(List<String> items)**: Finds the best shop based on selected items and district.
  - **calculateBestShop(List<String[]> filteredData)**: Calculates the best shop from filtered data based on total price.
  - **getBestPriceAndSeller(String selectedItem, String selectedDistrict)**: Gets the best price and seller information for a selected item and district.
- **CSV Data Reading:**
  - **readCSVData()**: Reads data from a CSV file.
- **Main Method (main()):**
  - Sets the look and feel, creates an instance of the **Cart** class, and makes it visible.

CATEGORY CLASS (key functionalities and sections of the code.)

- **Variables:**
  - **private JTree jTree**: Swing JTree for displaying item groups, categories, and items.
  - **public static String publicPath**: Path to a CSV file.
- **Constructor (`Category(String publicPath)'):**
  - Initializes the class with a CSV file path, sets window icon, and calls **populateTree()**.
- **Method (`populateTree()'):**
  - Creates a tree structure and populates it with data from a database.
  - Configures a **TreeSelectionListener** to handle item selection events.
- **Database Query Methods:**
  - **readItemGroupsFromDatabase()**, **readItemCategoriesFromDatabase(String itemGroup)**, **readItemsFromDatabase(String itemGroup, String itemCategory)**: Retrieve data from a database.
- **Generated GUI Components:**
  - **JTree** (**jTree1**), **JButtons** (**jButton1**-**jButton4**), **JPanel** (**jPanel1**), **JSeparator** (**jSeparator1**), **JTextArea** (**jTextArea1**).
- **ActionPerformed Methods:**
  - **jButton1ActionPerformed**, **jButton2ActionPerformed**, **jButton3ActionPerformed**, **jButton4ActionPerformed**: Handle button actions.

- **Shopping Cart Methods:**
  - **addToShoppingCart(String itemCode)**, **updateQuantityInCart(String itemCode)**, **insertNewItemToCart(String itemCode)**: Manage shopping cart.
- **CSV Data Reading:**
  - **getItemDetailsFromCSV(String itemCode)**, **displayCheapestSellers(List<String[]> itemDetails)**, **readCSV(String filePath)**: Read data from a CSV file.
- **Price Trend Display:**
  - **displayPriceTrend(List<String[]> priceTrendData)**: Display the price trend chart.
- **Main Method (main()):**
  - Set look and feel, create an instance of **Category**, and make it visible.

RELATE CLASS (handles Database Interactions)

- **Database Connection:**
  - Manages JDBC operations for MySQL using stored credentials, allowing connections to the database.
- **Methods:**
  - **getItemNameFromCode**, **getItemCodeFromName**, **getItemGroupFromCode**, **getItemCategoryFromCode**: Retrieve item-related details based on item codes or names.
  - **getPremiseNameFromCode**, **getPremiseCodeFromName**, **getPremiseAddressFromCode**, **getPremiseTypeFromCode**, **getPremiseStateFromCode**, **getPremiseDistrictFromCode,** **getPremiseCodeFromDistrict**: Fetch premise-related information from the database using codes or names.
  - **getStringFromCode**: Executes SQL queries with prepared statements to obtain specific data from the database.

SEARCH CLASS (GUI for Item Search and Modification)

- **Variables:**
  - **jTextField1**: Text field for entering search keywords.
  - **jButton6**: Triggers the search process.
  - **jTable1**: Display search results.
  - **jButton1**, **jButton2**: Buttons to remove and modify selected rows.
  - **tableModel**: Manages table data.

- **Constructor (`Search()'):**
  - Initializes the class with a CSV file path, sets window icon, and calls **populateTree()**.
- **Methods:**
  - **initComponents()**: Sets up the graphical user interface components.
  - **jButton6ActionPerformed()**: Executes a search operation based on the entered text in jTextField1. Displays the fetched results in jTable1.
  - **jButton1ActionPerformed()**: Removes a selected row's item from the database and updates the table.
  - **jButton2ActionPerformed()**: Modifies selected row information in the database and table.
- **Database Query Methods:**
  - **removeItemFromDatabase()**: Deletes an item entry from the database based on the provided item code.
  - **modifySelectedRow()**: Updates an item's information in the database, considering changes made to the table's selected row.
- **Helper Methods:**
  - **getWordList()**: Retrieves items from the database matching the search key using prepared statements.
  - **displaySearchResults()**: Populates the table with fetched search results.
- **Main Method (main()):**
  - Initializes the search GUI and displays it on the Event Dispatch Thread.

SETTINGS CLASS (Handles User Settings)

- **Database Connection:**
  - Uses JDBC for database connectivity.
  - Database credentials stored in **Relate** class.
- **Variables:**
  - **jLabel, jPasswordField, jTextField, jButton, jPanel**: Swing components for user interface elements.
  - **jLabel3**: Displays the current username.
- **Constructor (`Settings()'):**
  - Initializes components, sets the window icon, and updates the username label.
  - Sets the default text for **jLabel3** to the username fetched from **Login.getUsername**.
- **Methods:**
  - **initComponents()**: Initializes GUI components and sets their layout.

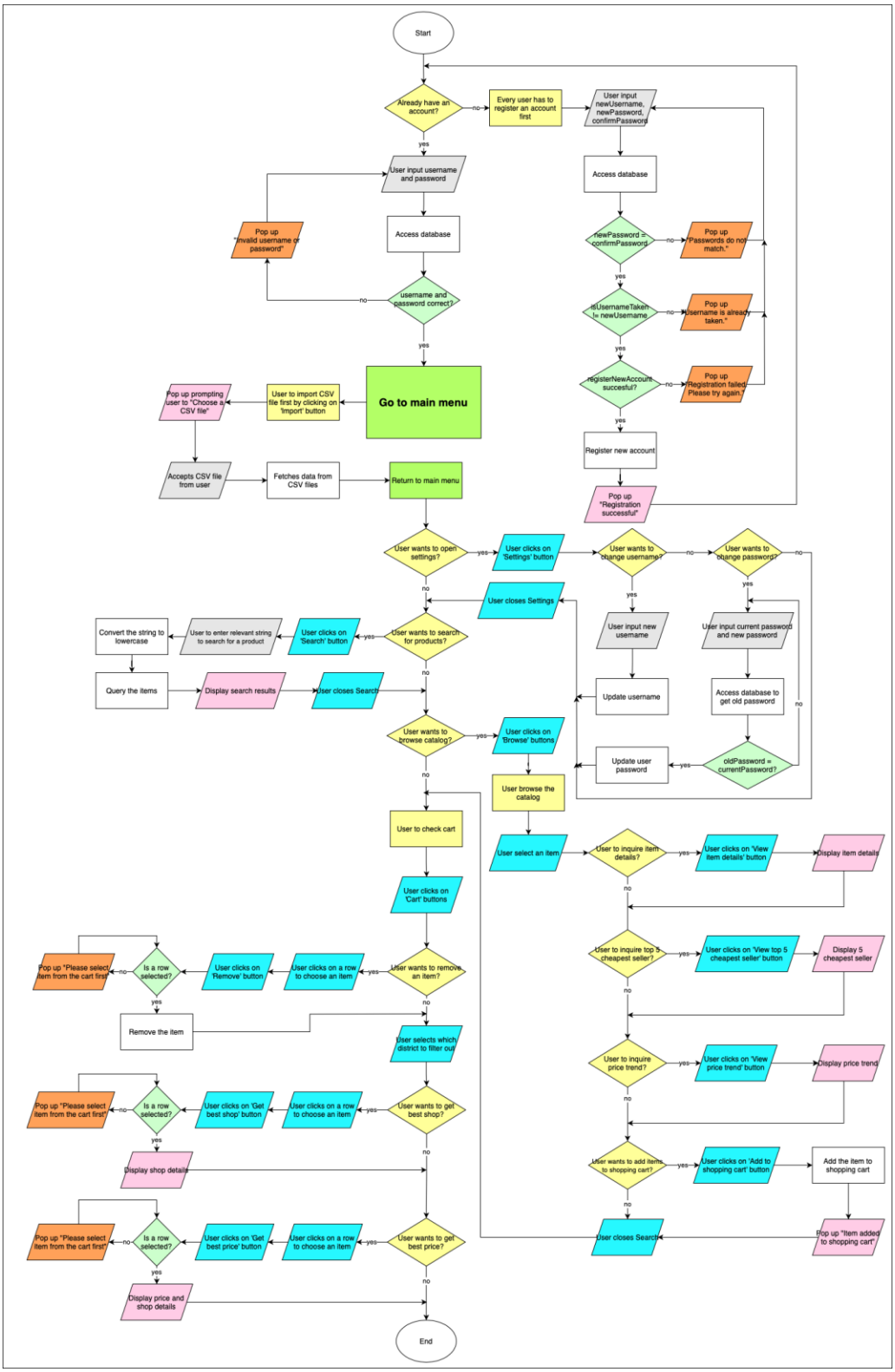
- **GUI Components:**
  - **JTable (jTable1)**, **JButtons (jButton1, jButton2, jButton3)**, **JComboBox (jComboBox1)**, **JTextArea (jTextArea1)**, and **JLabels.**
- **ActionPerformed Methods:**
  - **jButton5ActionPerformed**, **jButton6ActionPerformed**: Placeholder methods for handling button actions related to changing username and password.
- **Helper Methods:**
  - **validateOldPassword(String oldPassword)**: Validates the old password before allowing a password change.
  - **changeUsername(String newUsername)**: Changes the username after validating its availability.
  - **isUsernameAvailable(String username)**: Checks if a new username is available in the database.
  - **changePassword(String newPassword)**: Changes the user's password in the database.
- **Main Method (main()):**
  - Set look and feel, create an instance of **Settings**, and make it visible.

MAIN CLASS (Price Tracker Application Entry Point)

- **Variables:**
  - **userId**: Stores the current user ID.
  - **publicPath**: Path to the default CSV file.
  - **JDBC_URL, DB_USER, DB_PASSWORD**: Database connection details.
  - **GUI components**: Buttons, Labels, Image icons, etc.
- **Constructor (` Main(int userId1)'):**
  - Initializes the main frame of the application. Sets the user ID, initializes components, establishes a database connection, and inserts data from CSV files into the database.
- **Methods:**
  - **initComponents()**: Initializes GUI components and their layouts.
  - **Button Action Methods (jButton1ActionPerformed, jButton2ActionPerformed, etc.)**: Define actions for button clicks to perform various tasks like browsing files, managing the cart, searching, etc.
- **GUI Components:**
  - **Buttons (jButton1 to jButton5)**: Actions include importing CSV, managing the cart, browsing files, searching, and accessing settings.

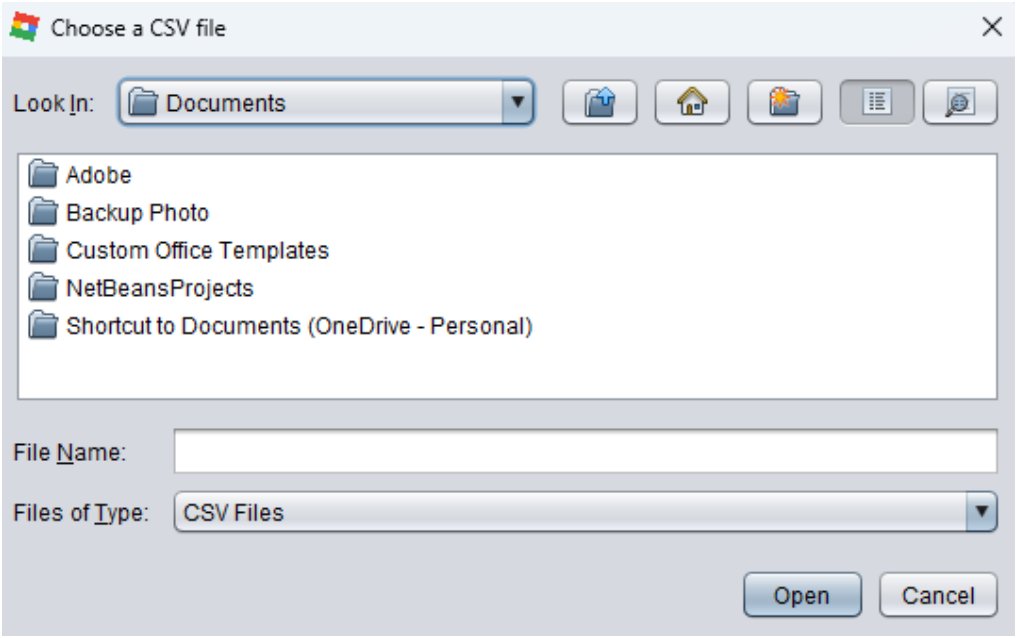
- **Database Operations:**
  - **connectToDatabase()**: Connects to the database and creates necessary tables if they don't exist.
  - **insertDataFromCSV(String csvUrl, String tableName)**: Reads data from a CSV file via URL and inserts it into the specified table in the database.
  - **tableHasData(Statement statement, String tableName)**: Checks if a table in the database has existing data.
- **Main Method main(String args[]):**
  - Entry point of the application. Sets the look and feel for the GUI. Launches the main frame and makes it visible.

6. FLOWCHART

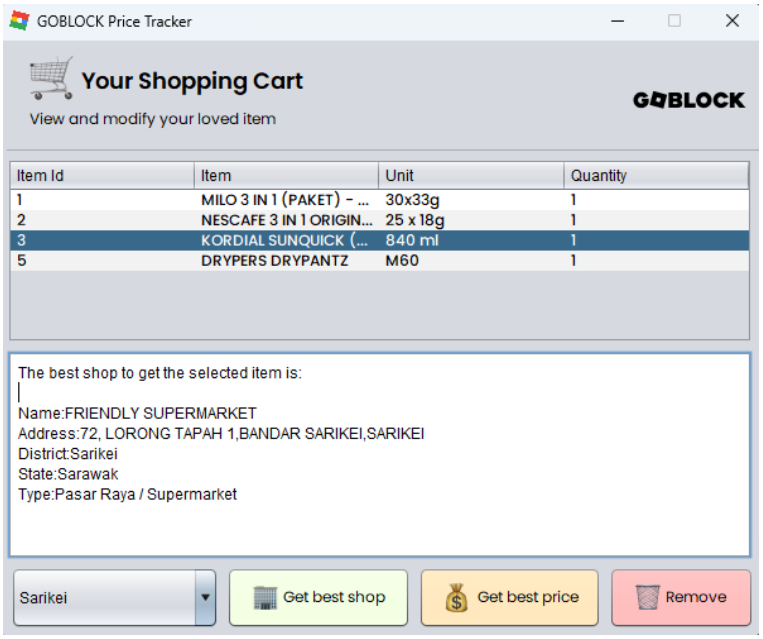

Start
Already have an account?
no
Every user has to register an account first
User input newUsername, newPassword, confirmPassword
yes
User input username and password
Access database
Access database
newPassword = confirmPassword
no
Pop up "Passwords do not match."
Pop up "Invalid username or password"
yes
username and password correct?
no
isUsernameTaken != newUsername
no
Pop up "Username is already taken."
yes
yes
registerNewAccount succesful?
no
Pop up "Registration failed. Please try again."
Go to main menu
yes
Register new account
Pop up prompting user to "Choose a CSV file"
User to import CSV file first by clicking on 'Import' button
Pop up "Registration successful"
Accepts CSV file from user
Fetches data from CSV files
Return to main menu
User wants to open settings?
yes
User clicks on 'Settings' button
User wants to change username?
no
User wants to change password?
no
yes
yes
no
User closes Settings
User input new username
User input current password and new password
Convert the string to lowercase
User to enter relevant string to search for a product
User clicks on 'Search' button
yes
User wants to search for products?
Query the items
Display search results
User closes Search
no
Update username
Access database to get old password
no
User wants to browse catalog?
yes
User clicks on 'Browse' buttons
Update user password
oldPassword = currentPassword?
yes
no
User browse the catalog
User to check cart
User select an item
User to inquire item details?
yes
User clicks on 'View item details' button
Display item details
User clicks on 'Cart' buttons
no
Pop up "Please select item from the cart first"
Is a row selected?
no
User clicks on 'Remove' button
User clicks on a row to choose an item
User wants to remove an item?
yes
User to inquire top 5 cheapest seller?
yes
User clicks on 'View top 5 cheapest seller' button
Display 5 cheapest seller
yes
no
no
Remove the item
User selects which district to filter out
User to inquire price trend?
yes
User clicks on 'View price trend' button
Display price trend
Pop up "Please select item from the cart first"
Is a row selected?
no
User clicks on 'Get best shop' button
User clicks on a row to choose an item
User wants to get best shop?
yes
no
User wants to add items to shopping cart?
yes
User clicks on 'Add to shopping cart' button
Add the item to shopping cart
yes
Display shop details
no
Pop up "Please select item from the cart first"
Is a row selected?
no
User clicks on 'Get best price' button
User clicks on a row to choose an item
User wants to get best price?
yes
User closes Search
Pop up "Item added to shopping cart"
yes
Display price and shop details
no
End

7. SAMPLE OUTPUT

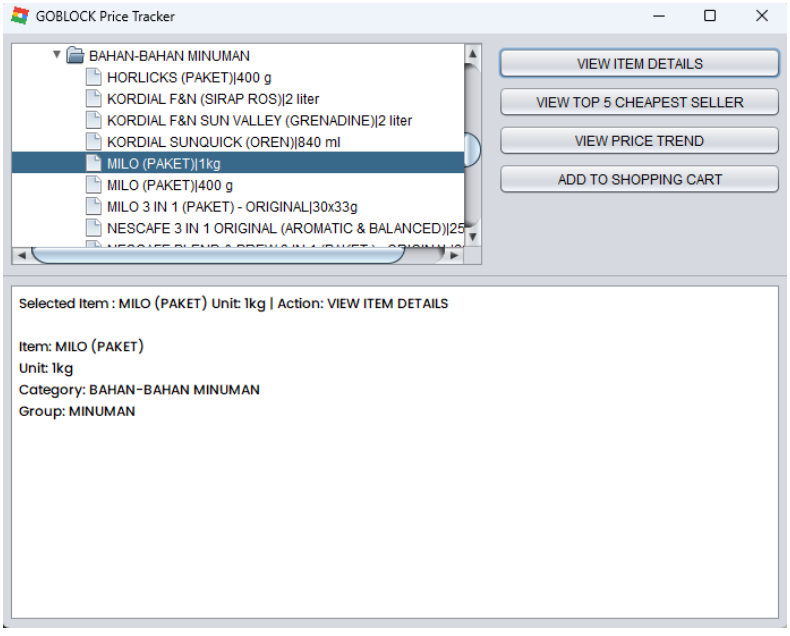
Login and register



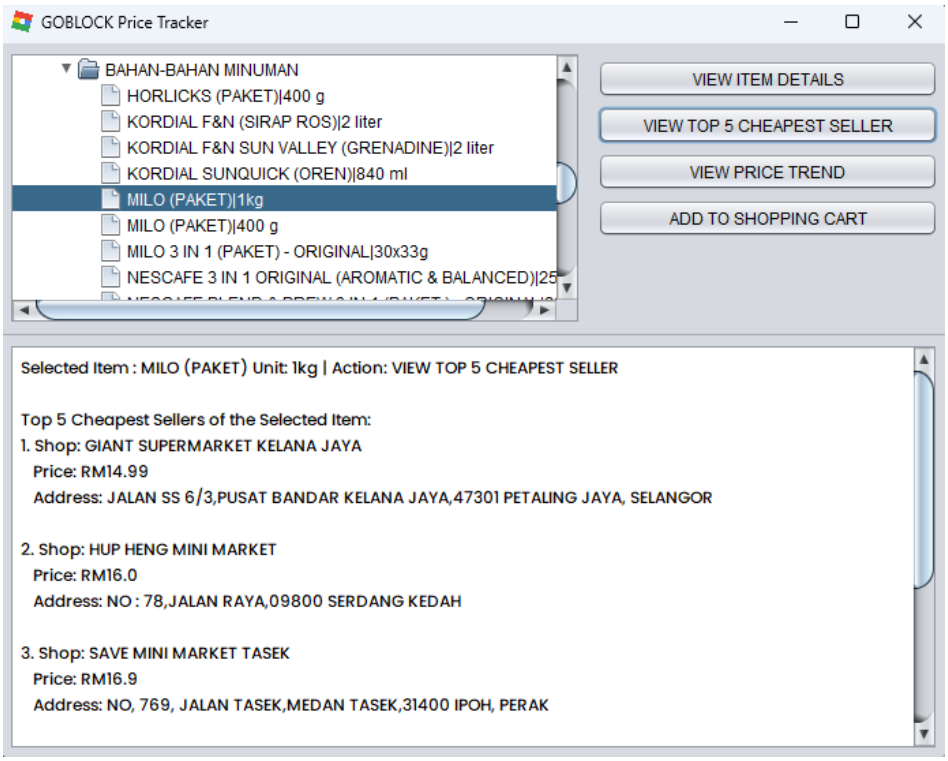
Import database



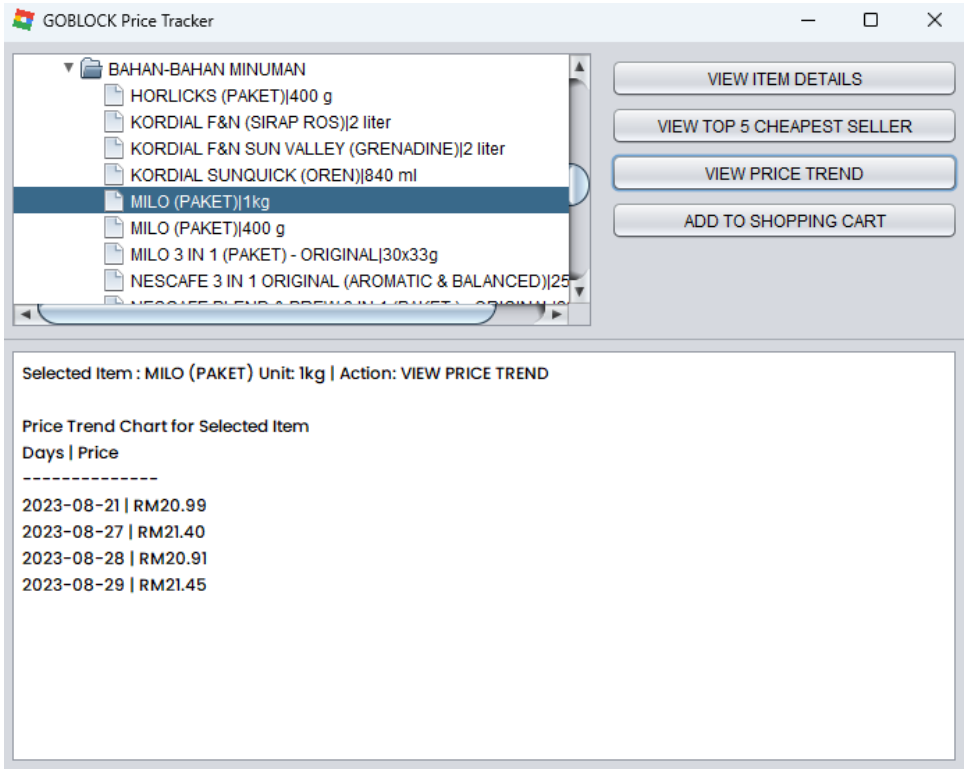
View cart



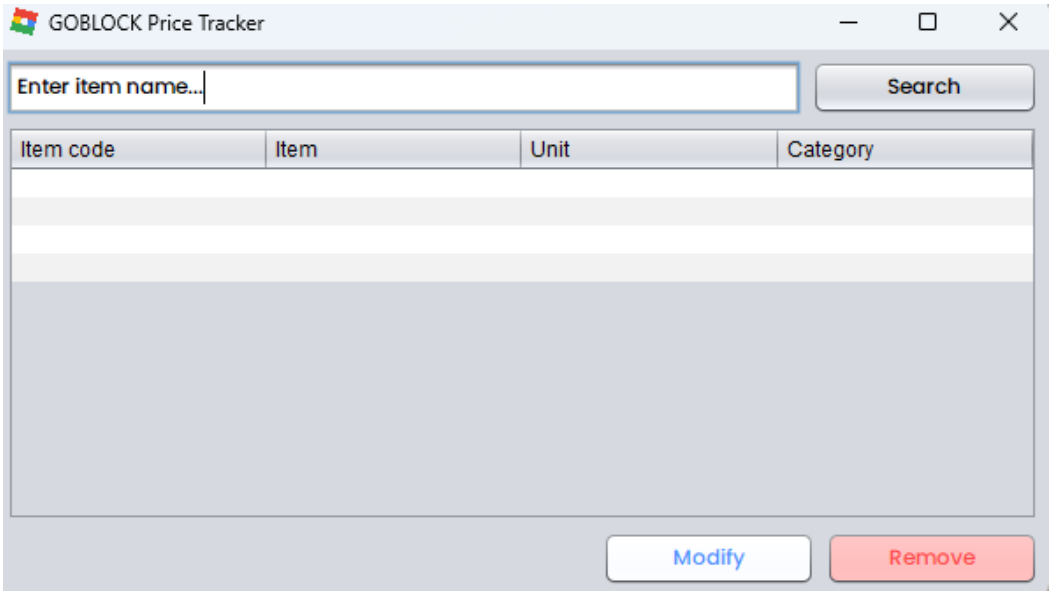
View item details

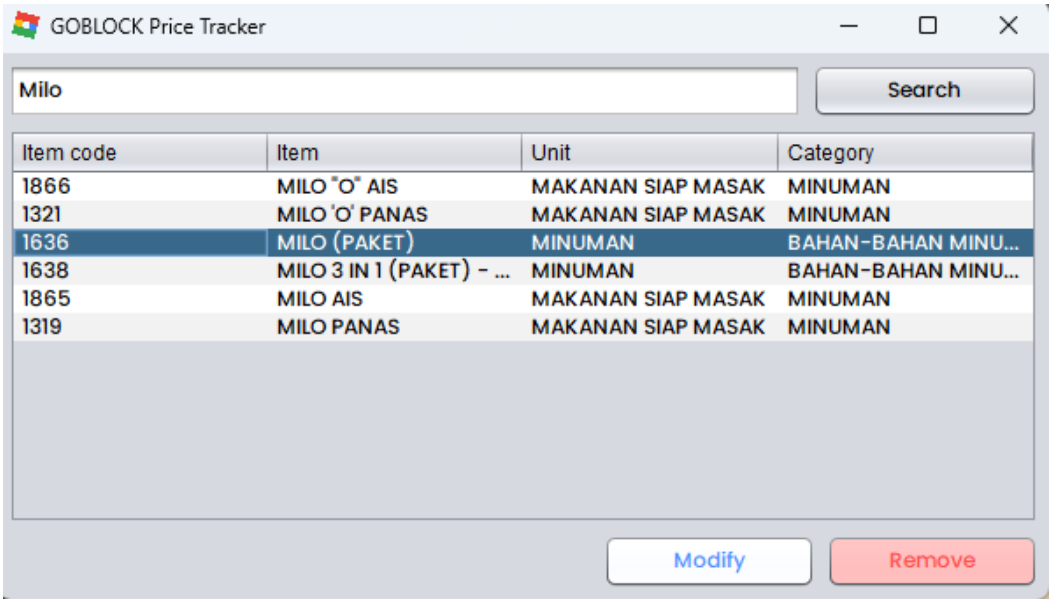


View top 5 cheapest seller

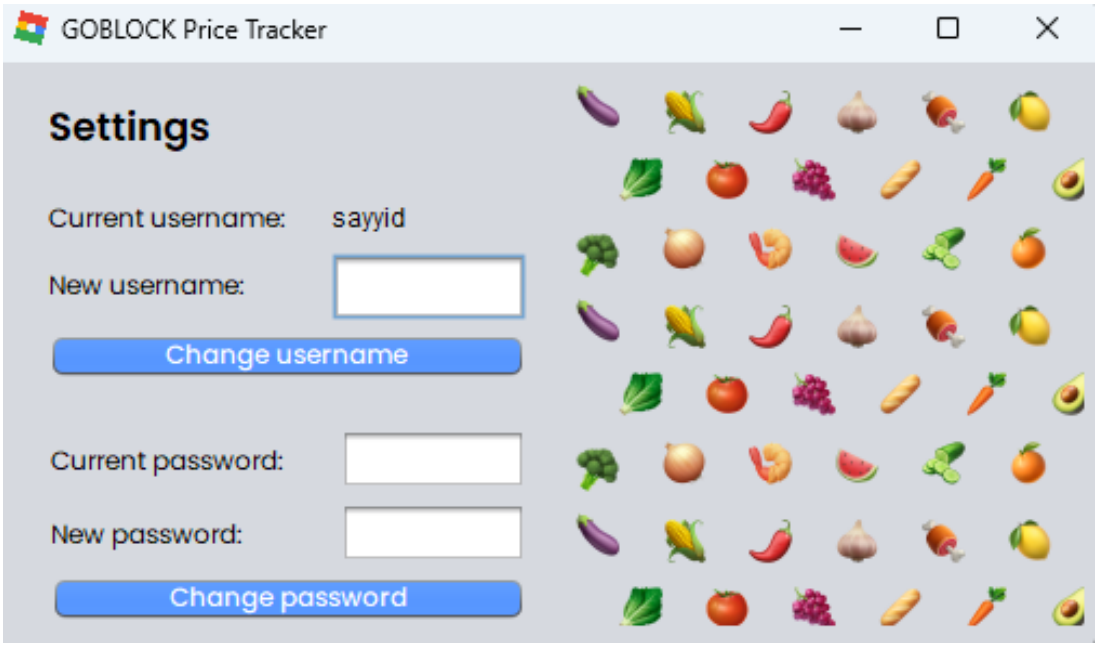


View price trend





Search UI and example


GOBLOCK Price Tracker
Settings
Current username:
sayyid
New username:
Change username
Current password:
New password:
Change password

Settings