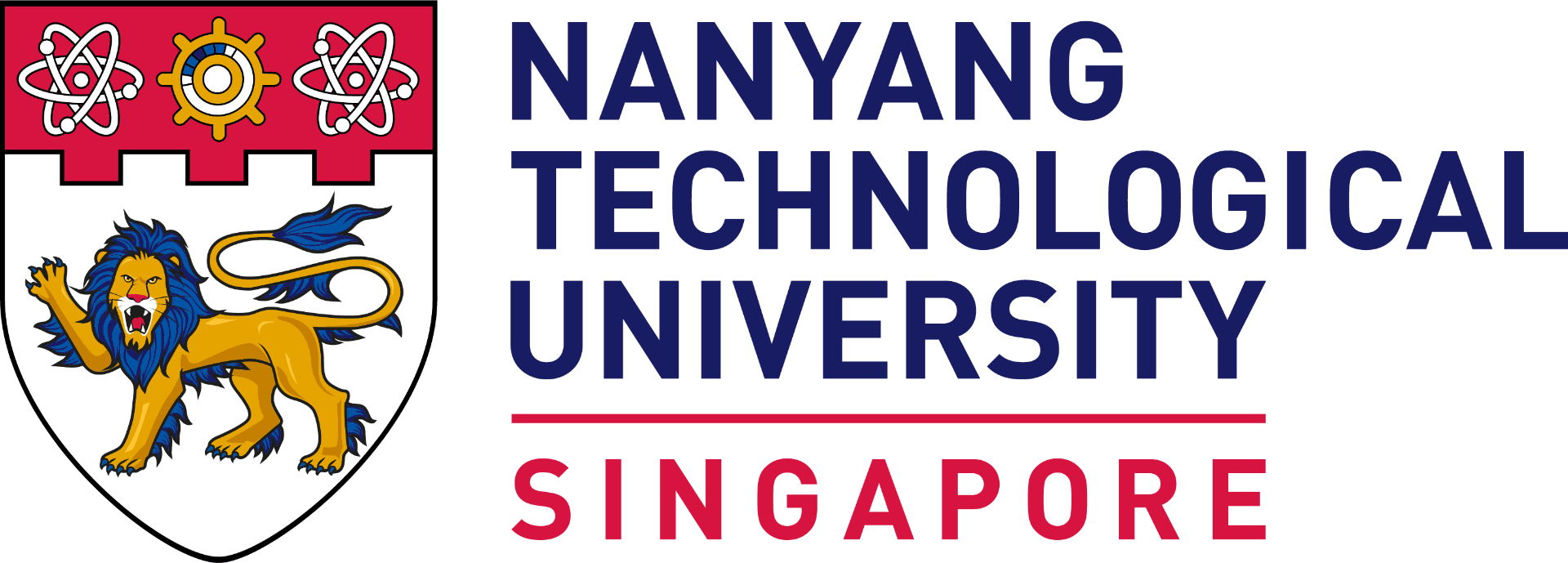
****

**CZ2006 Software Engineering:**

**Price Comparison App for Online Shopping**

****

**Lab 5**

**Group Name: JAWHO**

Joelle Thng (U1920277K)

Li Haocheng (U1921700J)

Wang Anyi (U1922992C)

Grace Ong (U1921756L)

Luo Wenyu (U1922009G)

**Contents**

[**Product Description**](#_p7x7og7qo38u) **4**

[Purpose](#_v1ofbxud7xis) 4

[Scope](#_v1ofbxud7xis) 4

[Users and stakeholders](#_v1ofbxud7xis) 4

[Assumptions](#_v1ofbxud7xis) 4

[Constraints](#_v1ofbxud7xis) 5

[Initial UI Mockups](#_v1ofbxud7xis) 5

[**Functional Requirements**](#_y4rtjelxm31l) **9**

[Registration](#_5c6ckof99z1b) 9

[Login](#_r6ntzrfmo0tx) 9

[User Profile](#_5zgzs1lgm89m) 9

[Search & Display](#_6h52g2md3l1m) 10

[Filter Result](#_apbqhp32p99t) 10

[Product Details](#_nf990s2hqg2n) 11

[Purchase & Payment](#_jz2q2zpfvdno) 11

[Post Purchase Requirements](#_x9bep17xmint) 12

[Use Case Diagram](#_30lx1y5l32r6) 13

[Use Case Descriptions](#_h43a8j7lsn3o) 14

[Registration](#_fsziir1ulalb) 14

[Login](#_nvzkeiv8nmuq) 16

[Verification](#_xopfk9j1sant) 18

[Log in with Facebook / Gmail](#_dulpzo8ahgbq) 20

[Request Password Change](#_iwcspweh010y) 22

[Edit User Profile](#_9w1qxtsamjig) 23

[Deregister account](#_kxrswysxnpg2) 25

[Search](#_juplawfg5a6k) 26

[Display Product](#_bjlsne821ubt) 29

[Filter](#_or9x4dbur71p) 31

[Pay using Credit Card or Grabpay](#_fmcf1fx08mc3) 33

[Review and Rating](#_9c9g4bjk9fow) 35

[User Contact Seller](#_p5684elim5rj) 36

[Class Diagram](#_gwawykizap17) 39

[Sequence Diagrams](#_8kvu9rfldbng) 40

[Registration](#_ec5u7fkpaben) 40

[Log In](#_wujwrnk0q32b) 41

[Filter](#_su4aazih2udp) 42

[Search](#_j0lnr8ahf8mp) 43

[Web Crawl](#_53qp7jynrrkc) 44

[Add to Cart & Pay](#_mgxb3hbcveqp) 45

[**Non Functional Requirements**](#_5vo38ptx3awc) **47**

[**Interface Requirements**](#_xrduyzxgx2ih) **48**

[User](#_kt78ymcgsbn) 48

[Hardware](#_ctvv8vyetgti) 48

[Software](#_s7mp15ljmme3) 48

[Communication](#_9dhwsld74v06) 48

[**System Architecture Design**](#_86bjosw1nt1w) **49**

[System Architecture Diagram](#_vrhn6dlnd4mi) 49

[Design Patterns](#_eogv33f5tt5z) 50

[**Data Dictionary**](#_pp1phd4jizm6) **52**

[**Testing**](#_m5xyoxqmdou) **55**

[Black Box Testing](#_i35j83hv6oxl) 55

[White Box Testing](#_xom496w0xz3r) 61

[Login](#_718r7a8ptv3m) 61

[Register](#_718r7a8ptv3m) 62

[Search](#_718r7a8ptv3m) 63

[Filter](#_718r7a8ptv3m) 63

[Checkout](#_718r7a8ptv3m) 64

[Forget password](#_s90ljk3o7thg) 65

[**Source Code**](#_tktidi511zzr) **66**

[**Appendix**](#_8e1xrxs274tp) **66**

# 

# Product Description

## Purpose

*Marketplace* aims to enable users to easily find the cheapest and best rated items from multiple websites in one single app.

Effectively, the app has the following purposes:

1. To allow users to compare prices of products that are originally listed on different E commerce websites.
2. To enable users to purchase the items directly from the apps using their preferred payment method.

## Scope

The app allows users to search for any item they wish to purchase, and subsequently see the details of the item such as the price and description of the item they are interested in. Users will be able to filter the results based on a few factors such as price and location from which the item is shipped from. Users can create their own Buyer account free of charge so that they can keep track of their purchases and store their preferred shipping address.

## Users and stakeholders

The stakeholders of the project consist of the external E commerce apps, Android mobile users, and *Marketplace*. Information of the items will be gathered from the various websites using Selenium webdriver, an open source API that can programmatically interact with browsers, the way a real user would. Android mobile users will use the application to compare prices of items across various E commerce websites.

## Assumptions

Users should have Internet access in order to use the application.

Users should have an Android device to run this application.

Users should have an email account in order to register for an account to use this application.

## Constraints

*Marketplace* only supports Android OS as of now.

The application is currently only available in English.

Users are unable to pay within the app now because of security and privacy issues caused by setting up a credit card payment gateway.

Sometimes the website may detect unusual activity due to webcrawling, resulting in a captcha page that blocks the API from website access.

## Initial UI Mockups

| Description | Image |
| --- | --- |
| Figure 1 shows the Home page interface where The user can search for the item name that they want to compare or choose product categories. Recent trending products will be recommended to the user at the bottom of this interface. | Figure 1. Homepage |
| Figure 2 shows the search interface. When a user searches for the name of the product, the search bar will auto fill information according to what the user has already entered. | Figure 2. Search bar autofill |
| Figure 3 shows the result interface which displays the search results of products crawled from different websites (indicate the origin on the listing). Clicking the filter icon on the top right corner brings the user to the Filter page. | Figure 3. Product display interface |
| Figure 4 shows the filter interface which allows the user to choose different filters and apply them to the search results. | Figure 4. Filter interface |
| Figure 5 shows the product information page for the individual product. This page displays the rating, price, origin and description of products. Buttons that allow the user to open the original websites or add to cart are located at the bottom of the page. | Figure 5. Product information page |

# Functional Requirements

## Registration

* 1. System should allow User to register for a new account.
     1. Information entered by User:
        1. Username
        2. Phone Number/Email Address
        3. Password & Confirm Password
        4. The system should validate all required fields are filled in.
        5. The system should check that the email/phone number is not taken

## Login

* 1. User should login with email address/phone number and password.
     1. User should be able to login using their Facebook/Google account
  2. The system should check all fields have been filled in.
  3. The system should validate the account login details.
  4. System will display error message if login information is incorrect.
  5. System will redirect user to the application’s main menu.
  6. User should be able to select ‘Forgot password?’ or request for change of password
     1. System should be able to send email verification code when User requests for change of password.

## User Profile

* 1. System should allow user to set up/edit their personal profile
     1. User should be able to edit their username
     2. User should be able to edit their password
     3. User should be able to edit their shipping address
     4. User should be able to edit their payment method and payment information
  2. User should have the option to deregister their account.

## Search & Display

* 1. User should be able to key in search term (product keyword)
     1. System should display list of recommended results while user is keying
  2. Users should be able to search with URL of the website/app they found the item
  3. Users should be able to search for similar products using image search
  4. System should be able to compare products based on different factors and user should be able view comparison results
     1. The default comparison results shown should be based on price and relevancy

## Filter Result

* 1. User should be able to use filters to view different comparison results
     1. User should be able to use filter to view the comparison of rating of the products by other users
     2. User should be able to use filter to view the comparison of reliability of shops
     3. User should be able to use filter to view if the product is shipped from domestic or overseas
     4. User should be able to filter products that have free shipping.
     5. User should be able to find products by the following categories:
        1. Home & Living
        2. Women’s Clothing
        3. Men’s Clothing
        4. Computers & Gadgets
        5. Toys & Babies
        6. Food & Beverages
        7. Health & Wellness
        8. Sports & Outdoors
  2. User should be able to use filters to sort products when they key in search terms and after results are shown.
     1. User should be able to sort by ascending or descending price.
     2. User should be able to sort by number of sales.
     3. User should be able to sort by ratings of products.
  3. System should display recommended products for each user based on past activities.
  4. System should be able to compare products from the following e commerce websites
     1. Lazada
     2. Qoo10
     3. Aliexpress
     4. Ezbuy

## Product Details

* 1. Users should be able to see the descriptions and prices pulled from the respective ecommerce websites.
     1. User can select items to add to cart
     2. User can click to redirect to the website if they want
  2. Users should be able to browse ratings and reviews posted by previous buyers.
  3. System should display 4-5 other products and respective prices related to current product

## Purchase & Payment

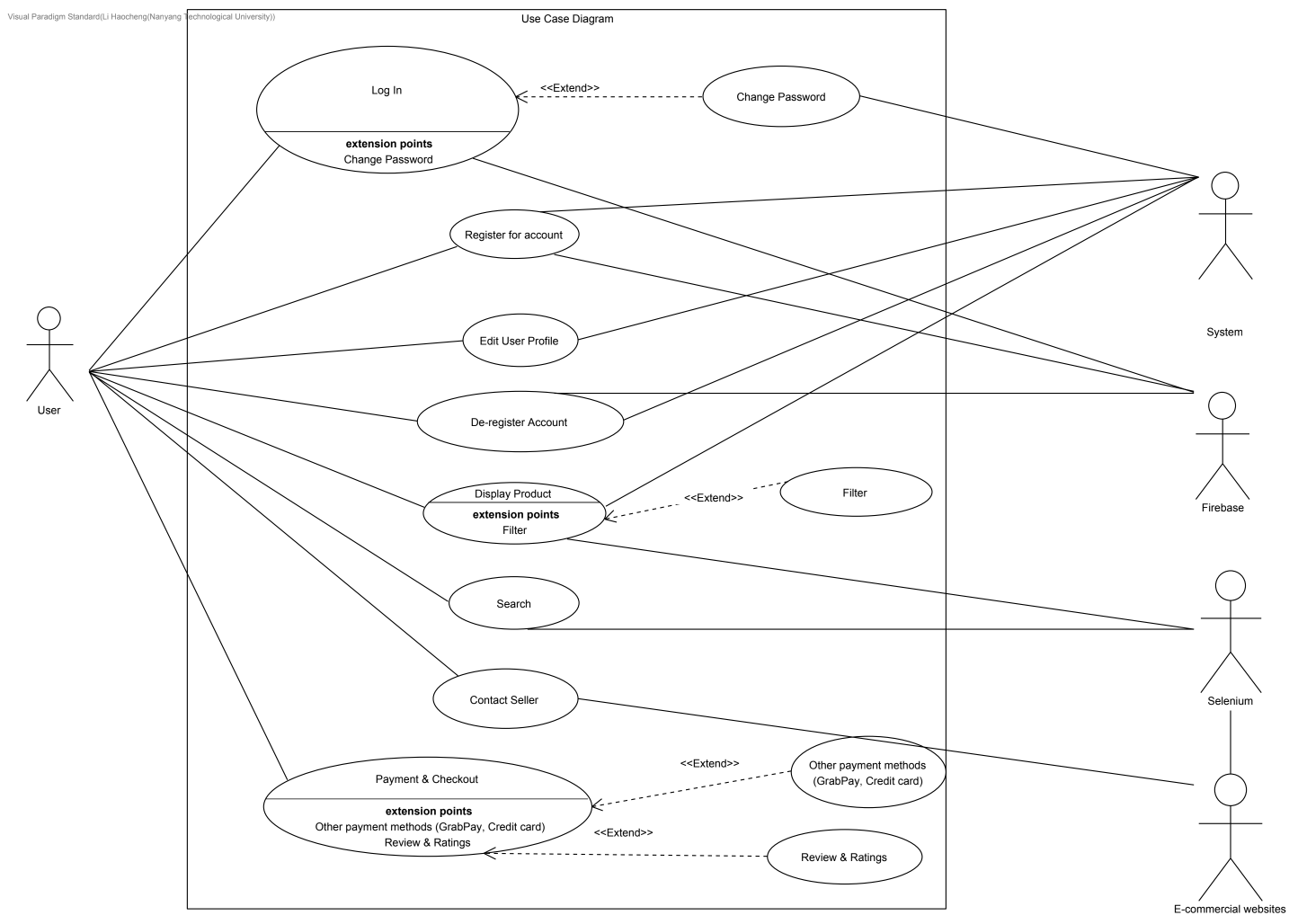
* 1. User should be able to see their shopping cart with all the items they selected.
     1. User should be able to unselect item(s) from cart
     2. User should be able to checkout from cart.
  2. User should be able to choose between checking out on this app or the app from which the product originates from.
     1. If User decides to checkout on this app, they should be able to pay via credit card, Grabpay.
     2. If User selects to check out from the original app, System should direct user to the original app.
  3. User should be able to see a list of promotion codes and vouchers available for their purchase.
     1. User should be able to apply code or voucher upon checkout.
  4. User should be able to contact the original seller for more information.
     1. The communication process should be done on the original app.

## Post Purchase Requirements

* 1. User should be able to leave reviews and ratings for the products they bought.
     1. System should update reviews and ratings on the original E-Commerce app.
  2. User should be able to request refund if item is damaged or delivered wrong product
  3. User should be able to contact seller for any queries on the item.
  4. User will be redirected to the respective E-Commerce website to contact the seller.

## 

## Use Case Diagram



## 

## Use Case Descriptions

### Registration

| Use Case ID: | 1.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Registration | | |
| Created By: | Wang Anyi | Last Updated By: | Wang Anyi |
| Date Created: | 5 Feb 2021 | Date Last Updated: | 15 April 2021 |

| Actor: | User, System, Firebase |
| --- | --- |
| Description: | To allow User to register for an account. |
| Preconditions: | 1) User must not already have an existing account that was registered using the same email/phone number.  2) User has the application downloaded and is at the welcome interface before logging in. |
| Postconditions: | 1) Firebase contains information for newly registered User. User is now able to login.  2) User is returned to the welcome interface page. |
| Priority: | High. User has to have an account in order to browse and purchase items |
| Frequency of Use: | Once per User |
| Flow of Events: | 1) User taps on “Register for an account”.  2) System displays the registration page requesting for information: Email address, password, confirm password.  3) User enters the required information and clicks “Register” button.  4) System validates if all fields are filled in.  5) System validates that the password and confirm password fields are identical.  6) System calls <<Verification>> and validates that email or phone number entered are not already taken, and starts verification process.  7) System displays “Account successfully registered” and brings the user back to login page. |
| Alternative Flows: | AF-S4: System detects field(s) left unfilled.  1) System displays an error message “ Please ensure all fields are filled” and highlights unfilled field(s) in red.  2) Return to Step 3.  AF-S5: System detects mismatch in password and confirm password.  1) System displays error message “ Password and Confirm Password do not match”.  2) Return to Step 3.  AF-S6: System detects existing account with the same email address/phone number.  1) System displays error message “Account with this contact already exists”. |
| Exceptions: | - |
| Includes: | 3. Verification |
| Special Requirements: | 5.2 User’s information should not be shared with connected social media platforms. |
| Assumptions: | - |
| Notes and Issues: | - |

### Login

| Use Case ID: | 2.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Login | | |
| Created By: | Wang Anyi | Last Updated By: | Wang Anyi |
| Date Created: | 5 Feb 2021 | Date Last Updated: | 15 April 2021 |

| Actor: | User, System, Firebase |
| --- | --- |
| Description: | To allow User to log in to an existing account by entering the relevant information. |
| Preconditions: | 1) There must be an existing account in the database that was registered using the same email/phone number.  2) User has the application downloaded and is at the welcome interface before logging in. |
| Postconditions: | 1) System displays the search and product browsing page.  2) User can now access their personalised cart and profile. |
| Priority: | High, user must log in in order to proceed to using this application |
| Frequency of Use: | High  Every time User logs out and re-login or log in on new device |
| Flow of Events: | 1) System requests for email/phone number and password.  2) User enters the required information and clicks “Login” button.  3) System calls <<Verification>> to check if the account exists, and validates the fields against information in the database.  4) User proceeds as System displays the search and product browsing page. |
| Alternative Flows: | AF-S2: User taps on “Forgot password”.   1. System sends 6 digit code to registered email address/phone number. 2. User receives code and enters the code. 3. System validates code. 4. System requests for new password and confirm new password. 5. System verifies that “new password” and “confirm new password” fields are identical. 6. System displays “Password changed successfully”. 7. Return to Step 1.   AF-S3: System does not find account in database with the email/phone number entered.   1. System displays error message “This account has not been registered”. 2. Return to Step 1.   AF-S4: System finds account in database with the email/phone number entered, but password does not match.   1. System displays error message “Incorrect password, please try again. Account will be locked after 5 incorrect attempts”. 2. Return to Step 1. |
| Exceptions: | EX1: User enters incorrect password for 5 times in a row.  1) System displays “Account is locked. Please contact customer service”. |
| Includes: | 3. Verification |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Verification

| Use Case ID: | 1.1.1.5 | | |
| --- | --- | --- | --- |
| Use Case Name: | Verification | | |
| Created By: | Wang Anyi | Last Updated By: | Wang Anyi |
| Date Created: | 5 Feb 2021 | Date Last Updated: | 8 Feb 2021 |

| Actor: | User, Firebase |
| --- | --- |
| Description: | To verify if User exists in Firebase or if password and email/phone number are valid. |
| Preconditions: | 1) User entered an email address/phone number and password in the login fields.  OR  2) User has entered relevant details and clicks on “Register” |
| Postconditions: | 1) System has successfully verified the new User account  OR  2) System is unable to verify account because User already has an existing account  OR  3) System verifies that email/phone number and password entered by User are valid  OR  4) System verifies that email/phone number and password entered by User are invalid |
| Priority: | High, both registration and login process involves verification. |
| Frequency of Use: | Low  When the User first creates an account, or when the User has been logged out of their account and they try to log in |
| Flow of Events: | 1) User tries to create a new account with their email and password  2) System checks the Firebase if the account (and password) is valid.  3) If there is no existing account with that email, Firebase will proceed with registration of a new User. |
| Alternative Flows: | AF-S2: User forgot they registered before, there is an existing account under the email address that the User keyed in   1. System will display “There is already an existing account under that email. Please try again.” for 3 seconds 2. System returns to Step 1   AF-S1: Login with Valid Account Details   1. User enters log in details (email/phone number and password) 2. System checks database if the account details are valid. 3. If valid, login is successful, System returns to Step 7   AF-S1: Login with Invalid Account Details   1. User enters log in details again 2. System checks database if the account details are valid 3. If account details are not valid, login is unsuccessful. 4. If login is successful, System returns to Step 7 |
| Exceptions: | EX1: User account has been locked due to multiple incorrect attempts   1. Return “Account is locked. Please contact customer service”. |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Log in with Facebook / Gmail

| Use Case ID: | 2.1.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Log in with Facebook/Gmail | | |
| Created By: | Joelle Thng | Last Updated By: | Joelle Thng |
| Date Created: | 10 Feb 2021 | Date Last Updated: | 15 April 2021 |

| Actor: | User, Firebase system, Facebook, Gmail |
| --- | --- |
| Description: | To allow the User to use their existing Facebook/Gmail account to continue using the App instead of having to create an entirely new account |
| Preconditions: | 1) User must have an existing Facebook/Gmail account  2) User’s mobile/computer must be connected to Wifi/mobile data |
| Postconditions: | User has successfully logged in with their Facebook/Gmail account |
| Priority: | Medium |
| Frequency of Use: | Low if the User decides to remain signed in on the app  High if the User logs out after each use and has to log in again whenever they use the app |
| Flow of Events: | 1) On the log in page, User clicks “Log in with Facebook” or “Log in with Gmail”  2) System prompts the User to key in their Facebook/Gmail email and password  3) System validates account details with Facebook/Gmail  4) User will be prompted to allow System to access their Facebook/Gmail account details  5) User grants permission  6) System extracts User data from Facebook/Gmail account and save in database  7) User has successfully logged in with their Facebook/Gmail account |
| Alternative Flows: | AF-S2: User keys in invalid email and/or password  1) System displays “Invalid email/password. Please try again” for 5 seconds  2) System returns to Step 2 |
| Exceptions: | EX1: App is unable to log in due to network error  1) System displays “A network error occurred. Please try again”  2) System returns to Step 2 |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Request Password Change

| Use Case ID: | 2.6 & 3.1.2 | | |
| --- | --- | --- | --- |
| Use Case Name: | Request password change | | |
| Created By: | Joelle Thng | Last Updated By: | Joelle Thng |
| Date Created: | 10 Feb 2021 | Date Last Updated: | 16 April 2021 |

| Actor: | User, System, Firebase |
| --- | --- |
| Description: | To allow the User to change their password |
| Preconditions: | 1) User logs in and wants to change password  OR  2) User forgot password and cannot log in  AND  3) User’s mobile/computer must be connected to Wifi/mobile data |
| Postconditions: | 1) User has successfully changed their password |
| Priority: | High |
| Frequency of Use: | Low. User only uses this function when they want to change their password or when they’ve forgotten their old password |
| Flow of Events: | 1) User keys in their registered email  2) System verifies that the email is inside the Firebase  3) User enters new password and confirm password  4) System checks if the 2 passwords match  5) System saves User’s new password in Firebase |
| Alternative Flows: | AF-S2: There is no account associated with the email that User has keyed in  1) System displays “There is no account associated with this email. Please try again” for 5 seconds  2) System remains at Step 2    AF-S7: The 2 passwords entered by the User do not match  1) System displays “The 2 passwords do not match. Please try again”  2) App returns to Step 6 |
| Exceptions: | EX1: System is unable to change password due to network error  1) System displays “A network error occurred. Please try again”  2) System returns to Step 2 |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Edit User Profile

| Use Case ID: | 3.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Edit user profile | | |
| Created By: | Luo Wenyu | Last Updated By: | Luo Wenyu |
| Date Created: | 13 Feb 2021 | Date Last Updated: | 15 April 2021 |

| Actor: | User, System |
| --- | --- |
| Description: | To allow User to edit their personal information such as payment information, shipping address, phone number, email address etc. |
| Preconditions: | 1) User must have an existing account |
| Postconditions: | 1) System displays “User profile updated successfully”. |
| Priority: | Low |
| Frequency of Use: | Low |
| Flow of Events: | 1) User taps on “Edit user profile” to update their information.  2) System displays existing user profile.  3) User taps on the section he/she wants to change.  4) System displays the keyboard to allow user to type and update user information.  5) User keys in new information.  5) User taps on “Save” to save changes on user profile.  6) System displays “User profile is updated successfully”. |
| Alternative Flows: | AF-S5: User did not key in new information after tapping the section and deleting the existing information.   1. System keeps and displays the old information. 2. Return to step 3. |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Deregister account

| Use Case ID: | 3.2 | | |
| --- | --- | --- | --- |
| Use Case Name: | Deregister account | | |
| Created By: | Li Haocheng | Last Updated By: | Li Haocheng |
| Date Created: | 15 Feb 2021 | Date Last Updated: | 15 April 2021 |

| Actor: | User, System, Firebase |
| --- | --- |
| Description: | To allow the User to deregister their account |
| Preconditions: | 1) User must have an existing account within the Database  2) User’s mobile/computer must be connected to Wifi/mobile data |
| Postconditions: | 1) User has successfully deregistered their account  2) User’s private information has been removed from Firebase |
| Priority: | Low |
| Frequency of Use: | Low |
| Flow of Events: | 1) On the profile page, User clicks “Deregister account”  2) System displays an alert window  3) User confirm by clicking on the yes button  4) System verify that there is no on-going purchase  5) System displays deregistration successful window  6) User logout from the application  7) System removes User’s information from Firebase |
| Alternative Flows: | AF-S4: There is on-going purchase for the current account  1) System displays “There is an on-going purchase for the current account. Please deregister your account after the purchase is complete.”  2) System terminates deregistration |
| Exceptions: | EX1: System is unable to deregister account due to network error  1) System displays “A network error occurred. Please try again”  2) System returns to Step 2 |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

### Search

| Use Case ID: | 4.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Search | | |
| Created By: | Joelle Thng | Last Updated By: | Joelle Thng |
| Date Created: | 5 Feb 2021 | Date Last Updated: | 16 April 2021 |

| Actor: | User, System, Selenium, Firebase |
| --- | --- |
| Description: | To allow User to search for an item |
| Preconditions: | 1) User has account that is stored in Firebase |
| Postconditions: | 1) System has successfully produced search results for user’s search conditions  OR  2) System is unable to find any products matching the user’s search conditions |
| Priority: | High, this function is always required for users to check out the products they want to compare. |
| Frequency of Use: | High. The purpose of this app is to allow users to search and compare items from different E Commerce sites, therefore the User will always have to use the search function to search for items to compare. |
| Flow of Events: | 1) User clicks on the search icon on the Home Page.  2) User searches for an item using any of the following options   * Type keywords in search bar * Photo search (by uploading a picture of a product for app to conduct image recognition to find similar products) * URL search   3) System uses the Selenium Web Driver to access the ecommerce websites and search for the required keyword entered by User.  4) Driver extracts web elements including product titles, product images, product price, product URLs from the list of products on the ecommerce websites by inspecting the webpage and looking for the respective classnames.  5) For each product URL, the driver accesses the individual pages and extracts web elements including shipping location, shipping price, rating, product description, and reviews.  6) System sorts the products by price and rating by default.  7) System displays the list of products on the page. |
| Alternative Flows: | AF-S3: System is unable to find what the User is searching for  1) System displays “No results found. Please enter another search”  2) System returns to step 2  AF-S6: User clicks on filter button and applies filter to results   1. System calls <<Filter>> to filter list of products according to the selected filter options. 2. System returns to step 7. |
| Exceptions: | EX1: System is unable to produce search results due to network error   1. System displays “A network error occurred. Please try again” 2. System returns to step 2 |
| Includes: | - |
| Special Requirements: | 1.1 System must provide search results within 2 seconds |
| Assumptions: | - |
| Notes and Issues: | - |

### Display Product

| Use Case ID: | 4.4 | | |
| --- | --- | --- | --- |
| Use Case Name: | Display Product | | |
| Created By: | Grace Ong | Last Updated By: | Grace Ong |
| Date Created: | 5 Feb 2021 | Date Last Updated: | 16 April 2021 |

| Actor: | User, Firebase, Selenium |
| --- | --- |
| Description: | After searching the product in the search bar, the system displays the results |
| Preconditions: | 1) User has searched product |
| Postconditions: | 1) User has successfully found the item they were looking for within the app  OR  2) User was unable to find the item they wanted to buy |
| Priority: | High, Users can only see the results when product information is displayed. |
| Frequency of Use: | High, every time the user searches something, the results will be displayed according to what the user has searched for. |
| Flow of Events: | 1. User searches for an item using the search bar 2. System finds all available information for the user using Selenium 3. System displays the relevant products:    1. Results are by default sorted according to price and relevancy    2. User can select <<Filter>> to sort or refine search results according to other options 4. User clicks on product of interest 5. System displays product price and information pulled from the respective E-commerce app, with options to go to the app, or add items to cart. |
| Alternative Flows: | AF-S3: System is unable to find what the User is searching for  1) System displays “No results found. Please enter another search”  2) System returns to step 1 |
| Exceptions: | EX1: System is unable to find search results due to network error  1) System displays “A network error occurred. Please try again”  2) System returns to Step 1 |
| Includes: | Filter |
| Special Requirements: | 3.1 System must be able to show the E-Commerce website that the products originated from  3.2 System should reflect accurate information  3.3 System must not crash |
| Assumptions: | - |
| Notes and Issues: | - |

### Filter

| Use Case ID: | 5.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Filter | | |
| Created By: | Grace Ong | Last Updated By: | Grace Ong |
| Date Created: | 5 Feb 2021 | Date Last Updated: | 15 April 2021 |

| Actor: | User, Firebase, Selenium |
| --- | --- |
| Description: | After system displays the search results, filter option can be applied to manipulate results |
| Preconditions: | 1. User has searched product 2. System has displayed search results |
| Postconditions: | User is able to sort or narrow down search results successfully |
| Priority: | Medium |
| Frequency of Use: | Medium |
| Flow of Events: | 1. User selects the filter icon. 2. System displays the filter options. 3. User selects a category to refine search options: 4. User selects/unselects multiple variables to sort the search results by:    1. Price (default)    2. Relevance (default)    3. Sales    4. Ratings    5. Shipping location and fees 5. User selects the sorting order (Ascending/ Descending). 6. User clicks to “Apply” the filter. 7. System passes control back to search page to display the filtered results. |
| Alternative Flows: | AF-S6: Reset filter options   1. User clicks to “Reset” filter options. 2. System displays filter options reset to default options (sort by Price and Relevancy, in ascending order) 3. System returns to Step 2 |
| Exceptions: | EX1: System is unable to filter search results due to network error  1) System displays “A network error occurred. Please try again”  2) System returns to Step 1 |
| Includes: | - |
| Special Requirements: | 3.2 System should reflect accurate information  3.3 System must not crash |
| Assumptions: | - |
| Notes and Issues: | - |

### 

### Pay using Credit Card or Grabpay

| Use Case ID: | 7.2.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Pay using credit card or grab pay | | |
| Created By: | Wang Anyi | Last Updated By: | Wang Anyi |
| Date Created: | 15 Feb 2021 | Date Last Updated: | 16 April 2021 |

| Actor: | User, System, Firebase |
| --- | --- |
| Description: | To allow User to pay for purchases using a choice of credit card or grab pay. |
| Preconditions: | User has selected items to checkout. |
| Postconditions: | User successfully makes payment and System displays “Payment made successfully”. |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1) User taps on “Pay using credit card” or “Pay using grabpay”.  2) System displays the payment page containing payment information (of credit card or grabpay) retrieved automatically from user’s profile, which is stored in Firebase.  3) User taps “Confirm payment”.  4) System contacts issuing bank to verify information and process payment.  5) System displays “Payment made successfully”. |
| Alternative Flows: | AF-S2: User chooses not to pay using default payment method retrieved from User Profile.   1. User taps on “Use another credit card”. 2. System prompts User to enter information including card number, expiry date, and CVC. 3. System contacts Bank to verify card and send verification code via SMS. 4. User enters verification code. 5. System verifies the code. 6. Return to Step 3.   AF-S2: User choose to make payment on the original website   1. User click on the button to make payment on original website 2. System open a new webpage that direct User to the original website |
| Exceptions: | EX1: System unable to verify User’s payment information.   1. System displays “Payment failed. Please verify your payment information”. 2. System directs User to the payment information page.   EX2: Payment declined by issuing bank.   1. System displays “Payment failed. Please check with your card issuing bank”. |
| Includes: | - |
| Special Requirements: | 5.2 User’s personal information should not be shared with connected social media platforms. |
| Assumptions: | - |
| Notes and Issues: | - |

### Review and Rating

| Use Case ID: | 8.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | Review and rating | | |
| Created By: | Wang Anyi | Last Updated By: | Wang Anyi |
| Date Created: | 10 Feb 2021 | Date Last Updated: | 15 April 2021 |

| Actor: | User, System |
| --- | --- |
| Description: | To allow User to leave reviews and ratings to evaluate the quality of products and service. |
| Preconditions: | User has completed the purchase of product(s) |
| Postconditions: | System displays “Review and rating has been submitted successfully”. |
| Priority: | Low |
| Frequency of Use: | Low |
| Flow of Events: | 1) System displays five white stars and “Leave a review” button at the bottom of the page after user completes payment.  2) User taps on stars to rate the product and service. Stars turn yellow upon tapping.  3) User taps on “Leave a review” button.  4) System displays textbox and photos icon beside.  5) User types in review into textbox and clicks on “Submit review” button.  6) System updates review and/or rating onto external e-commerce platform.  7) System displays “Review and rating has been submitted successfully”. |
| Alternative Flows: | AF-S5: User attaches a photo to the review   1. User clicks on photo icon next to textbox. 2. System opens User’s phone gallery. 3. User taps on photo(s) to be uploaded and taps on “upload”. 4. Return to step 5. |
| Exceptions: | - |
| Includes: | - |
| Special Requirements: | 5.2 User’s personal information should not be shared with connected social media platforms. |
| Assumptions: | - |
| Notes and Issues: | - |

### User Contact Seller

| Use Case ID: | 8.3 | | |
| --- | --- | --- | --- |
| Use Case Name: | User Contact Seller | | |
| Created By: | Luo Wenyu | Last Updated By: | Luo Wenyu |
| Date Created: | 16 Feb 2021 | Date Last Updated: | 15 April 2021 |

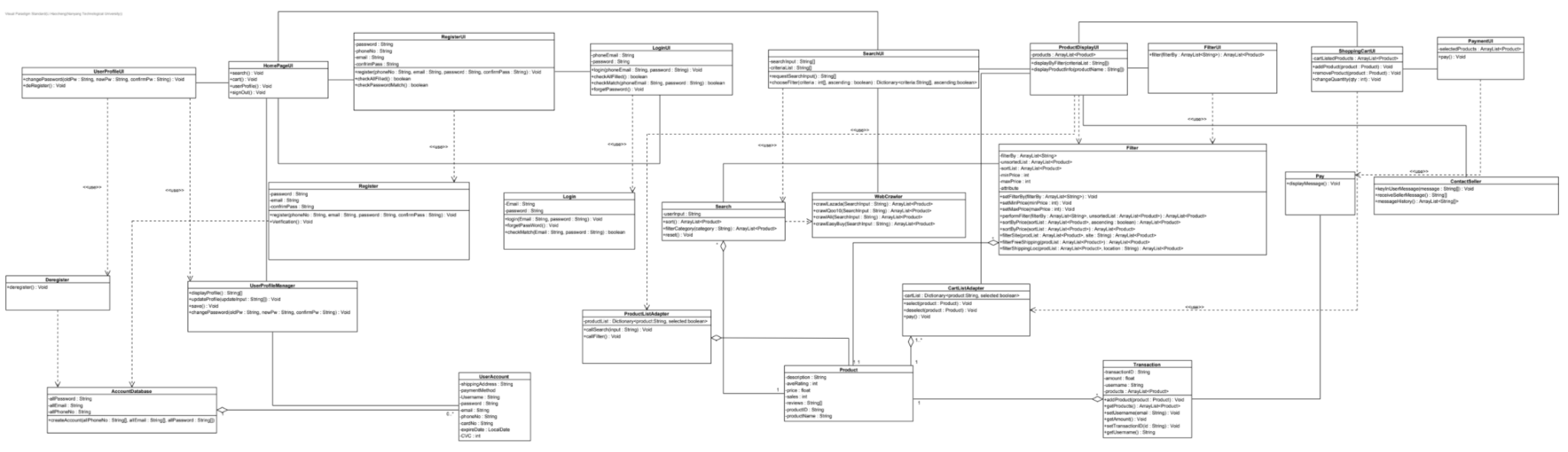
| Actor: | User, System, External E-Commerce Apps |
| --- | --- |
| Description: | To allow User to contact the seller for any questions involving the product. |
| Preconditions: | 1) User has selected a product to contact respective seller |
| Postconditions: | 1) User is redirected to respective E-commerce app and has successfully contacted seller |
| Priority: | Low |
| Frequency of Use: | Low |
| Flow of Events: | 1. User selects product they are interested in. 2. System displays product information 3. User selects ‘Contact Seller’ button 4. System redirects user to the respective chat functions in the E-Commerce App selling the product 5. User uses chat function to communicate with seller of product |
| Alternative Flows: | - |
| Exceptions: | EX1: Unable to connect to seller   1. System displays “Unable to contact seller” |
| Includes: | - |
| Special Requirements: | - |
| Assumptions: | - |
| Notes and Issues: | - |

## 

## Class Diagram

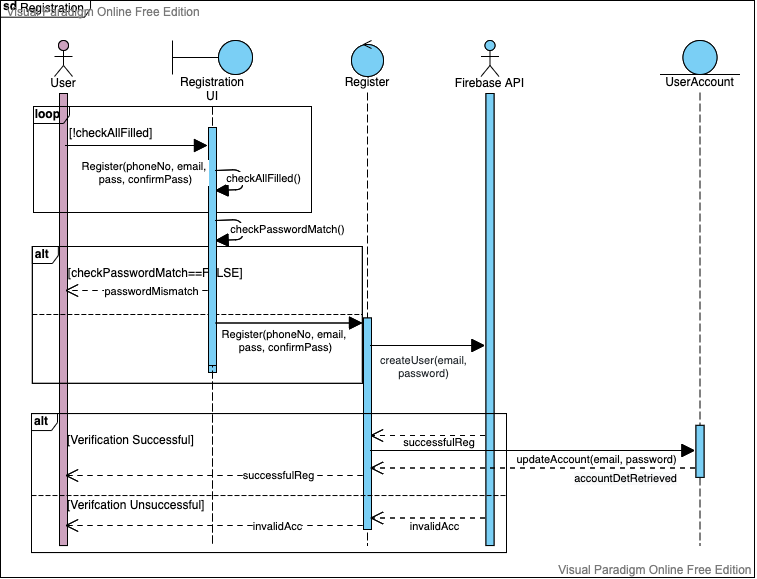
[a separate file for the class diagram is also submitted in the folder for clearer view]

[or a [link](https://drive.google.com/file/d/17YCkmBbx4wra4za6ADD-cKv0j5v49WR7/view?usp=sharing) to the class diagram]

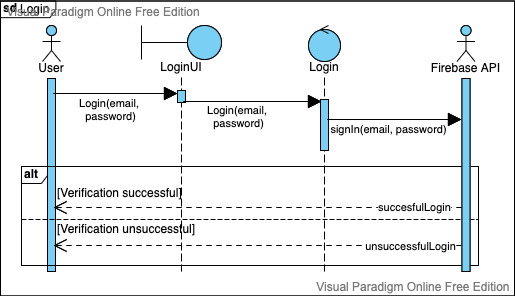


## Sequence Diagrams

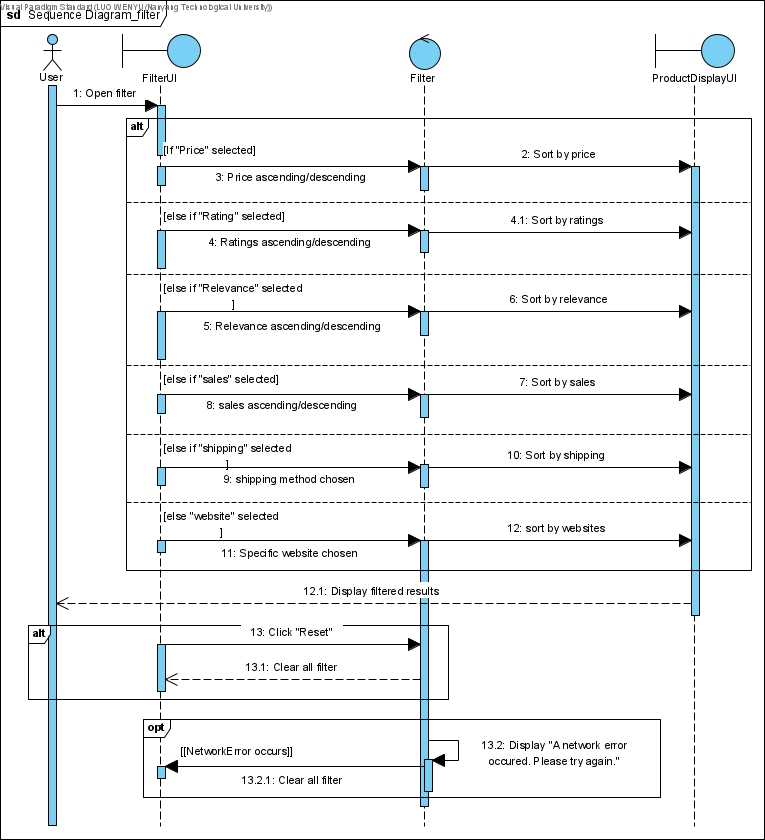
### Registration

****

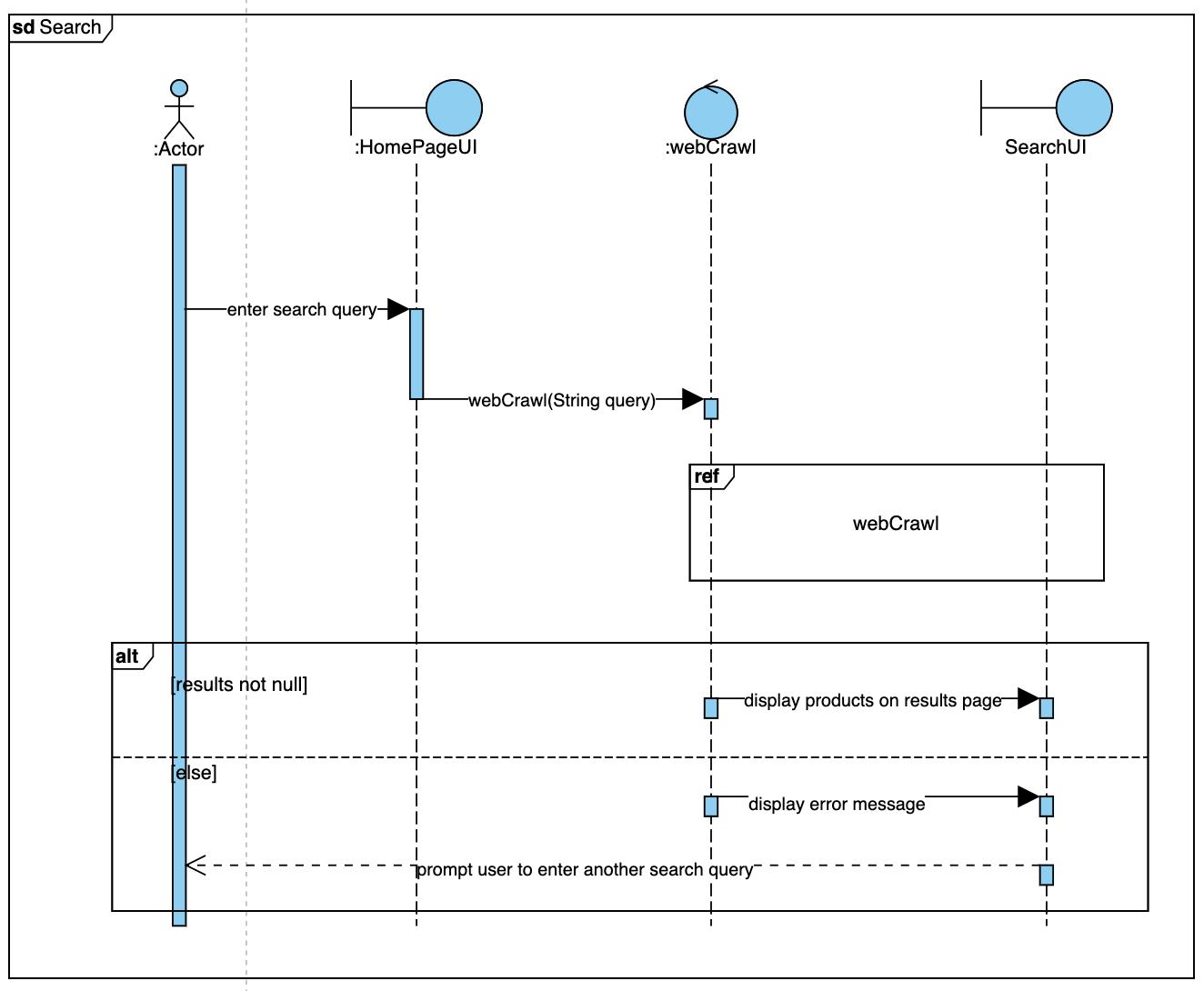
### Log In

****

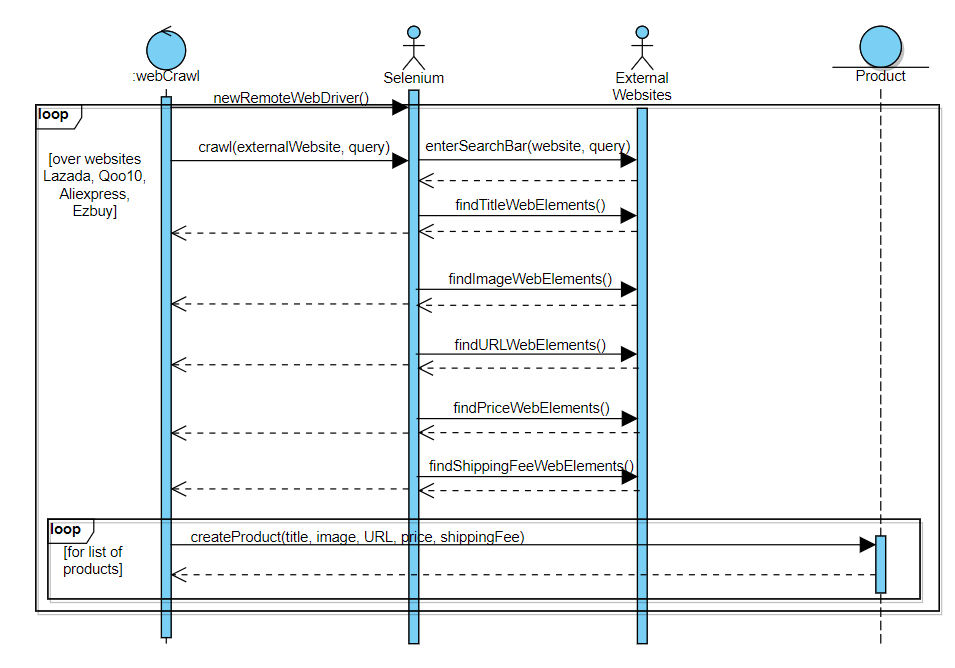
### Filter



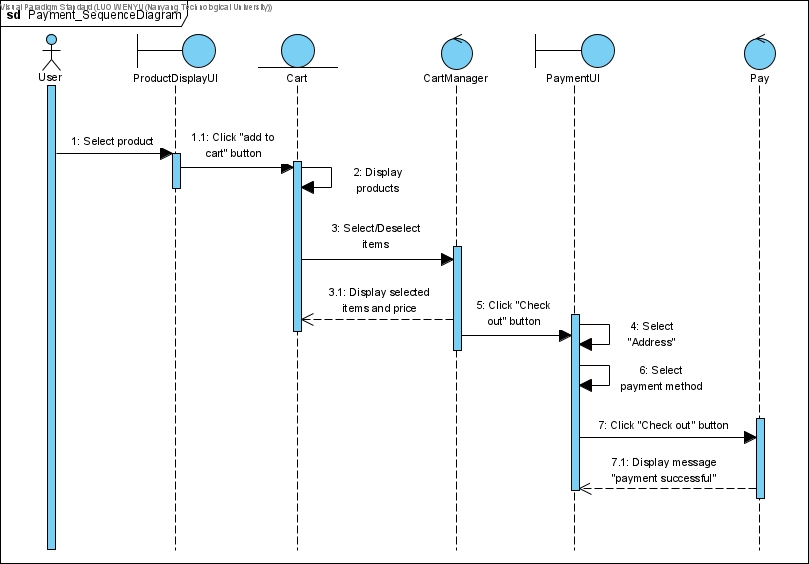
### Search



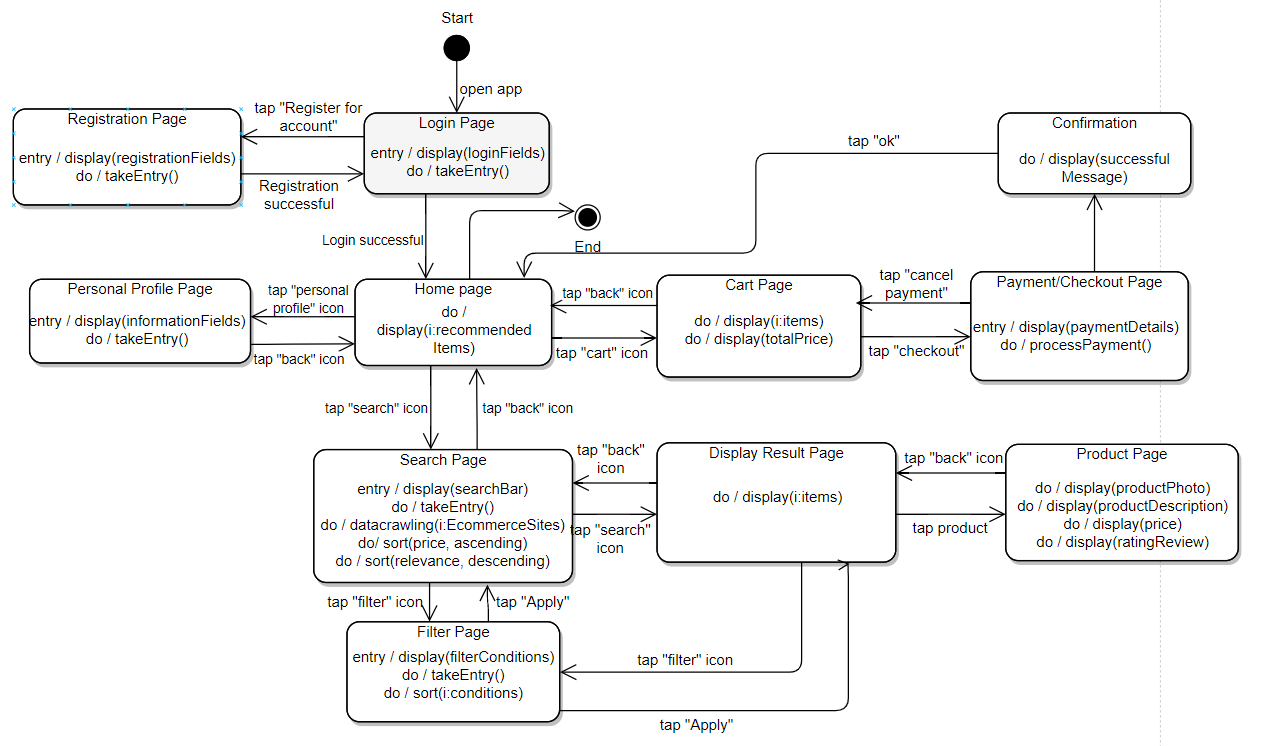
### Web Crawl



### Add to Cart & Pay



* 1. **Dialog Map**



# 

# Non Functional Requirements

1. Performance Requirements
   1. System must provide search results within 2 seconds
2. Usability Requirements
   1. Users have the option of changing the language
   2. User should be able to clear the cache
3. Reliability Requirements
   1. System must be able to show the E-Commerce website that the products originated from
   2. System should reflect accurate information
   3. System must not crash
4. Supportability Requirements
   1. Application must be able to support iOS and Android
5. Security Requirements
   1. User’s payment information and addresses must not be shared with other parties, including seller and delivery companies.
   2. User’s information should not be shared with connected social media platforms.

# 

# Interface Requirements

## User

*Marketplace* works for all users who want to shop online and find the most economical price for the products across different online shopping websites.

## Hardware

*Marketplace* requires a hardware device with internet connection as well as Chrome web browser in order to search and crawl for product results.

## Software

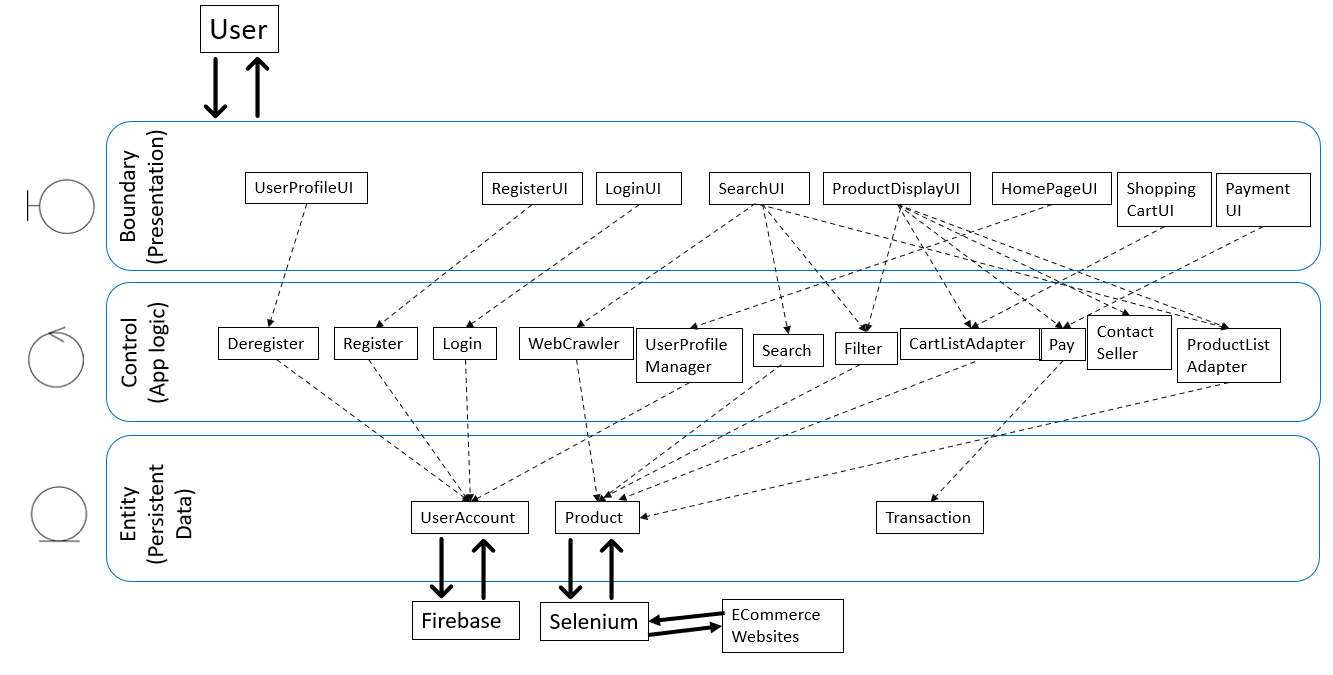
*Marketplace* is designed to work on Android Devices.

## Communication

All features of *Marketplace* will be accessible through the application. To perform search, *Marketplace* will be accessed over the Internet.

# System Architecture Design

## System Architecture Diagram



## 

## Design Patterns

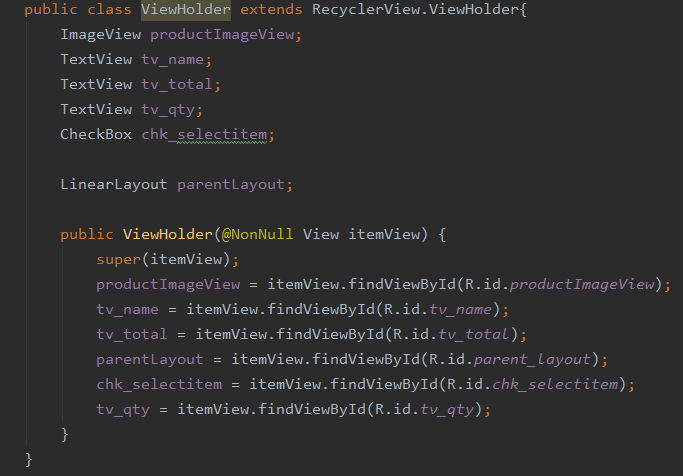
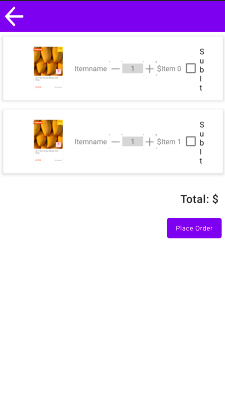
* + 1. **Adapter Pattern**

**Problem**

There are many different product objects with similar structure but slightly different attributes (name, image, price). The number and order of products to display are also unknown, especially after applying filters. Hence, there is a need for something to dynamically read the attributes of the product instead of hardcoding into the shopping cart and search results display UI.

**Solution**

We implemented the View Holder pattern to display lists of products in the Shopping Cart UI and SearchResultsUI. The View Holder holds generic elements such as the product image, title and price. The values reflected in these elements will vary for each product. This is performed by the Adapter class, which will link the View Holders to the RecyclerView in the UI pages by mapping each ViewHolder to the products elements. This makes our app more flexible and dynamic, at the same time vastly improving its performance and responsiveness.



* + 1. **Singleton Pattern**

**Problem**

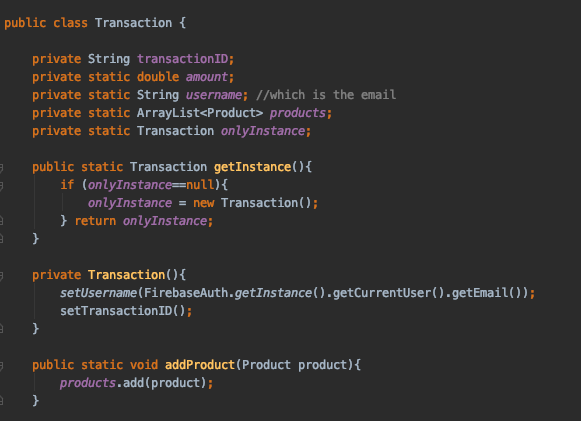
There should be only one instance of Transaction when recording the products bought. Additionally, global access is necessary for all classes.

**Solution**

Inside the Transaction class, a static function getInstance() is created. In the function, a single instance of Transaction is created and the private constructor instantiated only once. The private constructor prevents other classes from instantiating it.

Later on, Transaction.getInstance() will be called when any changes are made to the attributes.

This prevents other objects from instantiating other copies of transaction , ensuring that all other classes such as Payment access this single Transaction instance and that it is coordinated across all classes.



# Data Dictionary

| **Terms** | **Definition** |
| --- | --- |
| User | A person who uses the application to browse and purchase products. |
| System | The android mobile application. System performs functions in response to user inputs. |
| Personal profile | A collection of User’s related information, consisting of username, contact, shipping address, payment method and information. This can only be viewed and edited by User after logging in. |
| Search | Finding desired products by typing key words in the search bar. |
| Filter | A tool which allows the user to input certain requirements and then proceed accordingly. |
| Login | An action of logging in to the application system using an existing user account. |
| Register | To set up an account in the system by putting relevant information, including username, phone number/email address and password. |
| URL | The address of a web page. Brings the user to the web page upon clicking. |
| E-commerce | Commercial transactions conducted electronically on the internet. |
| Relevancy | The degree to which attributes of a product is related to the key words specified by User. |
| Rating | User’s evaluation of the products in terms of quality, on a scale of 1 to 5. Users are entitled to provide rating only after the delivery of product. The rating will subsequently be shown on the product description page of the original app. |
| Comment | An open ended text created by User to evaluate the products in terms of quality and/or the general service or delivery. Users can choose to attach photos of the products received to the comment. The comment will subsequently be shown on the product description page of the original app. |
| Cart | A place to keep records of the items chosen from an online store by the user. Users can check out from cart to purchase all items that have been saved into the cart. |
| Shipping | The physical movement of product from the seller to the buyer. Shipping may be in the form of air or sea shipping and involves a shipping company to manage this process. Duration and costs of the shippings are dependent on the mode of shipping, the shipping company, location of 2 parties (domestic or overseas) and properties of the product itself (weight and volume). |
| Checkout | Process in which User confirms items of purchase, selects shipping method, acknowledges price and proceeds to make payment for the total amount. |
| Promotional codes/ vouchers | Alphanumeric strings that can be applied by User to product payments upon check out. This offers a reduction to the original price of the products. |
| Cache | Short term information relating to the application that is stored in the device’s memory for faster retrieval. It can be cleared regularly to free up space. |
| Deregister | To remove an existing User account and all related information from System. |
| Verification code | 6 digit numeric string sent to User’s contact (SMS or email) to verify User’s identity for security reasons when User requests for a change in password. |
| Refund | The return of payment from seller to buyer in the case of a damaged or wrong product. |
| Categories | After clicking any of the categories, the User will be able to see recommended items for that category and search for items from that category only. |
| Selenium WebDriver | It is an open-source API that can programmatically interact with a browser on an operating system the way a real user would. |
| Appium | It is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms. |
| Firebase | A platform developed by Google to let your users authenticate with using their Google Accounts by integrating Google Sign-In into this app. |

# 

# Testing

## Black Box Testing

1. Login
   1. Generic case

| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| 1 | Login with valid account username and password | The system displays the homepage for user to continue the operation | The system displays the homepage for user to continue the operation |
| 2 | Login without filling up the required fields | The system prompts the user to fill up all the required fields in order to log in | The system prompts the user to fill up all the required fields in order to log in |
| 3 | Login without valid credentials | The system prompts the user to enter the credentials again | The system prompts the user to enter the credentials again |
| 4 | Forgot Password with valid user email account | The system sends email to user to reset password | The system sends email to user to reset password |
| 5 | Forgot Password with invalid user email account | The system prompts the user to enter the credentials again | The system prompts the user to enter the credentials again |

* 1. Specific cases (Combination)

| **Username** | **Password** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| testuser | testpass | Successful login | Successful login |
| wronguser | testpass | Invalid email/password | Invalid email/password |
| Empty(“”) | testpass | Please fill in all the required fields | Please fill in all the required fields |
| testuser | Empty(“”) | Please fill in all the required fields | Please fill in all the required fields |
| testuser | wrongpass | Invalid email/password | Invalid email/password |

1. Registration
   1. Generic cases

| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| 1 | Register with valid account username and password | The system displays the homepage for user to continue the operation | The system displays the homepage for user to continue the operation |
| 2 | Register without filling up all the required fields | The system prompts the user to fill up all the required fields in order to register | The system prompts the user to fill up all the required fields in order to register |
| 3 | Register with password mismatch | The system prompts the user to check and re-enter the password | The system prompts the user to check and re-enter the password |

* 1. Specific cases (Email)

| **Email** | **Expected Result** | **Actual Result** |
| --- | --- | --- |
| testEmail | The system displays “Error. Badly formatted email address.” | System displays “Error. Badly formatted email address.” |
| testEmail@aldkjfasdk | The system displays “Error. Badly formatted email address.” | System displays “Error. Badly formatted email address.” |
| testEmail@gmail.com | The system allows user to create a new account | System allows user to create a new account |

* 1. Specific cases (Combination)

| **Username** | **PhoneNo** | **Password** | **Confirm Password** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- | --- | --- |
| testuser | testno | testpass | testpass | Successful registration | Successful registration |
| Empty(“”) | testno | testpass | testpass | Please fill in all the required fields | Please fill in all the required fields |
| testuser | Empty(“”) | testpass | testpass | Please fill in all the required fields | Please fill in all the required fields |
| testuser | testno | Empty(“”) | testpass | Please fill in all the required fields | Please fill in all the required fields |
| testuser | testno | testpass | Empty(“”) | Please fill in all the required fields | Please fill in all the required fields |
| testuser | testno | testpass1 | testpass2 | Please check and re-enter the password | Please check and re-enter the password |
| testuser | testno | testpass2 | testpass1 | Please check and re-enter the password | Please check and re-enter the password |

1. Change Password
2. Generic cases

| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| 1 | Change with a new password | Firebase saves new password | Firebase saves new password |
| 2 | Change with password mismatch | The system prompts the user to check and re-enter the password | The system prompts the user to check and re-enter the password |

1. Specific cases

| **Old Password** | **New Password** | **Confirm Password** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- | --- |
| testpass | test | test | System displays “Password must be more than 6 letters” | System displays “Password must be more than 6 letters” |
| testpass | testpassNew | testpassNew | Password is changed successfully | Password is changed successfully |
| testpass | test | testpassNew | Please check and re-enter the password | Please check and re-enter the password |

1. Search
2. Generic cases

| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| 1 | User enters a valid search term | System displays all results | System displays all results |
| 2 | Search bar is empty but user presses the search button | System displays error message “Please enter a search query” | System displays error message “Please enter a search term” |
| 3 | User entered invalid search query that does not return any results | System displays error message “No results found. Please enter another search query” | System displays error message “No results found. Please enter another search query” |

1. Specific cases

| **Search Query** | **Expected Result** | **Actual Result** |
| --- | --- | --- |
| Milo | System displays all matching products | System displays all matching products |
| Empty(“”) | System displays “Please enter a search query” | System displays “Please enter a search query” |
| jcikshd | System displays error message “No results found. Please enter another search query” | System displays error message “No results found. Please enter another search query” |

1. Filter

| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| 1 | User applies a set of valid filters | System displays products that satisfy the filter | System displays products that satisfy the filter |
| 2 | User applies a set of filters that do not return any results | System displays “No results found. Please try again.” | System displays “No results found. Please try again.” |

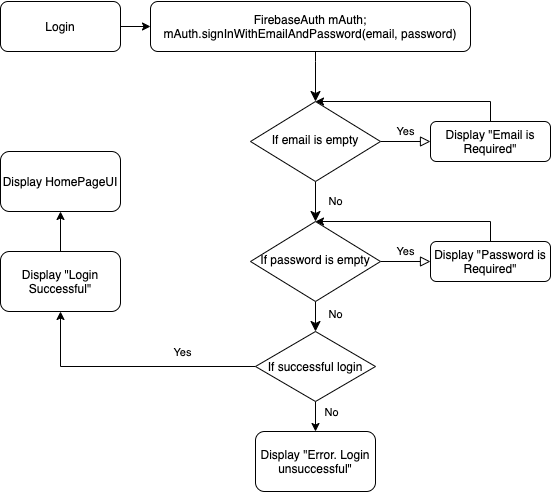
1. Checkout & Payment

| **Test Id** | **Scenario** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| 1 | User selects item and checks out | System prompts user to key in payment details | System prompts user to key in payment details |
| 2 | User enters valid payment details | Successful payment message is displayed | Successful payment message is displayed |
| 3 | User tries to checkout with no items in cart | System displays “Please add item(s) to cart before checking out.” | System displays “Please add item(s) to cart before checking out.” |
| 4 | User entered invalid payment details | System displays “Invalid payment method. Please try again.” | System displays “Invalid payment method. Please try again.” |

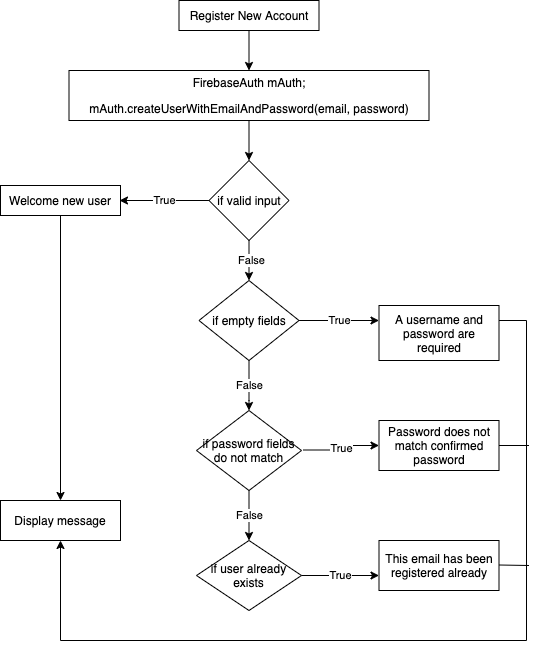
## White Box Testing

JUnit was used for unit testing in the application. Unit testing was applied on specific units and portions of code.

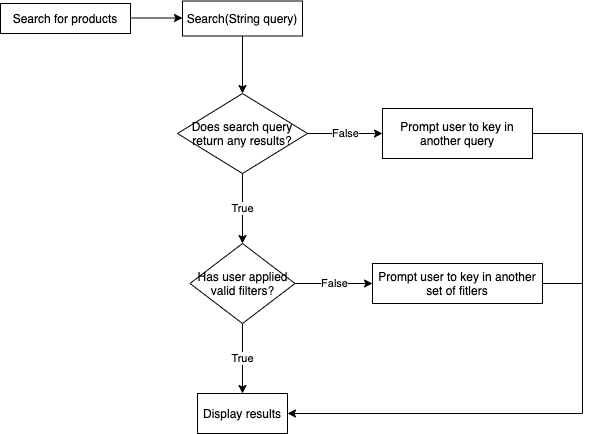
### Login



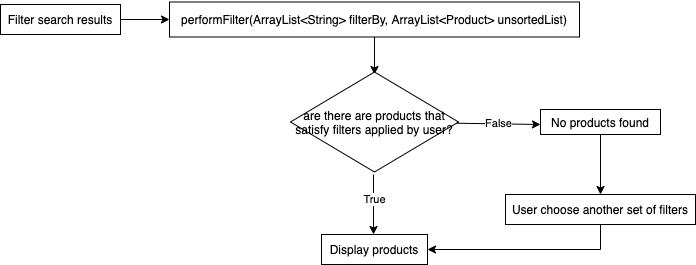
### Register



### Search



### Filter



### Checkout



### Forget password



# Source Code

**Refer to the attached file.**

Link to Github repository: <https://github.com/wanganyi1102/ecommTest>

# Appendix

Link to Use Case Diagram: <https://drive.google.com/file/d/17BFOJxsXqhy5wj0hy1ixBgcQo6rmWMG9/view?usp=sharing>

Link to Class Diagram:

<https://drive.google.com/file/d/17YCkmBbx4wra4za6ADD-cKv0j5v49WR7/view?usp=sharing>

Link to Video Demo:

<https://drive.google.com/file/d/1salTaMxJiVicIlur-AkMJxD05dMurwae/view?usp=sharing>