אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

הפקולטה למדעי ההנדסה
המחלקה להנדסת תוכנה

Faculty of Engineering Sciences
Dept. of software Engineering

Ben-Gurion University of the Negev

“Trading System”

Version 0

Table of Contents

Correctness constraints: 3

Glossary of Terms: 4

Use Cases (including customer inspection): 5

Visitor 5

1. Use case: Entrance 5

2. Use case: Exit 5

3. Use case: Registration 6

4. Use case: Login 7

5. Use case: getting information 8

6. Use case: searching items 9

7. Use case: saving selected products in the shopping cart 10

8. Use case: cart content and modification 11

9. Use case: buying and paying 12

Registered User: 13

Use case : Exit Market. 13

Use case : Logout. 13

Use case : Opening a store. 14

Store Owner: 15

Use Cases - 4.1: Inventory management. 15

Use Cases - 4.1.1: Inventory management – adding product. 16

Use Cases - 4.1.2: Inventory management – remove product. 17

Use Cases - 4.1.3: Inventory management – Changing product details. 18

Use Cases - 4.2: Changing the types and rules (policy) of the store's purchase and discount. 19

Use Cases - 4.4: Appointment of store owner. 20

Use Cases - 4.6: Appointment of store manager. 21

Use Cases - 4.7: Changing store manager’s permissions. 22

Use Cases - 4.9: Closing store. 23

Use Cases - 4.11: Request for information about positions in the store. 24

Use Cases - 4.11.1: Request for information about store manager’s permissions. 25

Use Cases - 4.13: Receiving information about purchase history in the store. 26

System Administrator: 27

Use Cases - 6.4: Getting information about purchase history of the store or buyers. 27

Use Cases - 6.4.1: Getting information about purchase history of buyers. 28

Correctness constraints:

1. A Single name for each user – when an account is created the system asks for the visitor to give information such as email, password, and name. The name given by the visitor (now user) will be held by the object user (specified in the class diagram) and will serve as an indication for that user (with a unique id number given by the system).

2. According to the class diagram an admin (inherited by registered) must be registered to the system. Also, the system can only be initialized by an admin logged in (see use case 1 for system functional requirements).

3. According to the class diagram a store owner/ store manager (inherited by registered) must be registered to the system.

4. From the general build of the system, whenever a user is entering the system, he is classified as a guest and is able to do certain actions (according to his permission regarding each store or even all market).

5. According to the class diagram, a shop must have a registered store owner for it to be active.

6. From the class diagram, each shop will have its own modifiable purchase(a) and discount(b) policies according to the owner's needs.

7. In the system the shopping cart will hold different baskets, each one related to a different store and containing only items from that store.

8. In the system a user (visitor or registered) has his own shopping cart that no other user can access.

9. According to the class diagram, each shop has an inventory which holds all the products the store has. When ordering the system address the store about each product's quantity, if the quantity needed is over what the inventory holds then the system informs the user that his order cannot be executed.

10. From the system use cases (specifically the payment use case) if the transaction was successful then a receipt is given to the user, otherwise a "payment failed" message is presented and the purchase is canceled(a)(b)(c).

11. From both the class diagram and the system use cases it is a precondition that there be a connection for both a payment system(a) and a supply system(b).

Glossary of Terms:

Facade- this is where all the commands get assigns, meaning- his job is to interact with the client code and the subsystems and its used for comfortable interface

User controller- component that handles input and coordinates actions within a system and manages user objects

Guest – someone who uses the system but isn't registered meaning his actions will be limited

Registered – someone that uses the system and is registered meaning he have more options and actions (open store, saving shopping cart etc.)

Shop owner – registered user who owns a store.

Shop creator- registered user who created the store.

Shop manager- registered user who manages the store.

Seller- general term for shop manager, creator, opener,

Shopping cart- collection of product baskets that a user\guest is interesting to buy.

Basket- collection of products that a user\guest is interesting to buy.

Supplement system- external service that provides products for the store in the system.

Payment system- external service that handles the payment for the store in the system.

Purchase policy- rules to describe the purchasing options- min/max quantity, whom allowed to buy etc.

Discount policy- rules to define options for cheaper price for a product/purchase.

History- saves the former orderings.

Bid- buyer applying a bid for the product the bid can be accepted or rejected

Instant buy- the buyer pays the exact price of the product

Auction- as long as the selling is still open a buyer can offer a new bid higher than the current bid

Lottery- Buyers can purchase a chance to win the product, provided that the total of their bids does not exceed the price of the product.

Visible discount- built from discount percentage and duration can be applied to one product or group of them.

Conditional discount- built from discount percentage, duration, and conditions that's the buyers need to hold.

Hidden discount- built from discount percentage, duration and discount code only if the code is valid and the buyer will supply it he'll see the price after discount.

Permission- defines what is allowed for a client in a specific store.

Use Cases (including customer inspection):

Visitor

1. Use case: Entrance

-Actor: Visitor.

-Precondition: None.

-Parameter: None.

-Actions:

1. The visitor opens the app/web page of the system.

2. the system loads and shows the visitor popular items from several stores.

3. the system opens a shopping cart for the visitor and allows him to add items to it while he is shopping.

Acceptance testing:

1. Good scenario: when loading the system can successfully gather all the information needed and the app/web page opens.
2. Bad scenario: when loading the visitor’s internet crashes so the app/web page isn’t loaded.
3. Sad scenario: when loading the system can’t access all the information from the database and crashes.

2. Use case: Exit

-Actor: Visitor.

-Precondition: the visitor already has the app/web page of the system open.

-Parameter: None.

-Actions:

1. The visitor wants to close the app/web page of the system.

2. the system deletes the visitor’s shopping cart and all his records that were taken while he was online.

Acceptance testing:

1. Good scenario: when exiting the system all the needed actions occur and the system continues normally for other users.
2. Bad scenario: when trying to exit the visitor’s internet crashes which causes the system to malfunction.
3. Sad scenario: when exiting the system wrongfully saves the visitor’s information in the database (which causes an overload of unused data).

3. Use case: Registration

-Actor: Visitor.

-Precondition: None.

-Parameter: None.

-Actions:

1. The visitor is presented with a login page.
2. the visitor presses the button for registration.
3. a new page is loaded and asks the visitor to enter certain information such as his name and email.
4. after the visitor has finished, he selects continue
5. the system checks to see if there is any problem with the information given.

    5.1 If there is a problem the system informs the visitor to change the things that are needed.

    5.2 otherwise the system adds the new information to its database and transfers the visitor to the login page.

Acceptance testing:

1. Good scenario: when trying to register the visitor enters all the correct information and the system executes the registration process successfully.
2. Bad scenario: when trying to register the visitor enters an invalid email (either an already registered email or a non-existent one) so the system can't do the registration process and informs the visitor.
3. Sad scenario: at the registration the visitor enters a string to the birthday category and causes the system to malfunction (and possibly crash).

4. Use case: Login

-Actor: Visitor.

-Precondition: The visitor has already registered to the system.

-Parameter: None.

-Actions:

1. The visitor is presented with a login page.
2. the visitor enters his login information.
3. the system checks the username and password given to see if it is in the database.

    3.1 if there is a problem the system informs the visitor and asks him to fix whatever is needed.

    3.2 otherwise the system now sees the visitor as user and loads the information needed for him (his shopping cart and other information about him).

Acceptance testing:

1. Good scenario: when trying to login the visitor enters all the correct information and the system executes the login process successfully.
2. Bad scenario: when trying to register the login enters an invalid email (a non-registered email) so the system can’t do the login process and informs the visitor.
3. Sad scenario: at the login page the visitor enters an injection code to the password making the system crash.

5. Use case: getting information

-Actor: Visitor.

-Precondition: The visitor already has the app/web page of the system open.

-Parameter: None.

-Actions:

1. The visitor is presented with the main page of the system.

2. the visitor clicks on the “info” button.

3. the system transfers the visitor to the info page where all the stores are presented in alphabetical order with a picture and a short explanation for each store.

4. the visitor can click on each store picture and get more detailed information about the store and about the items it is selling.

Acceptance testing:

1. Good scenario: when trying to get information he finds the store he wants, and the system can load the information successfully and present it to the visitor.
2. Bad scenario: when trying to get information the visitor searches for a store that does not exist, and the system informs the visitor of his mistake.
3. Sad scenario: when trying to get information the system does not present all the stores that are in the market and their information becomes inaccessible.

6. Use case: searching items

-Actor: Visitor.

-Precondition: The visitor already has the app/web page of the system open.

-Parameter: None.

-Actions:

1. The visitor is presented with the main page of the system.

2. the visitor clicks on the “search” button.

3. the system opens a search bar and modification options for the search (such as price range, selected categories and more).

3. the visitor writes can write the product name, or he can leave the search bar empty and run a search using only the modification options.

    3.1 if there is something written in the search bar then the system goes over all the products in the database and checks if their name matches the search or if the search is included in their description. The system then presents all matching products sorted by their rating.

    3.2 if nothing is written then the system searches only by the modification options for the best matches and presents them to the visitor, sorted by rating.

Acceptance testing:

1. Good scenario: when trying to get information he finds the item he wants and can successfully add it to his cart.
2. Bad scenario: when trying to find an item the visitor searches for an item that does not exist, and the system informs the visitor of his mistake.
3. Sad scenario: when trying to get find an item the system does not present all the items related to the search and their information becomes inaccessible.

7. Use case: saving selected products in the shopping cart

-Actor: Visitor.

-Precondition: The visitor already has the app/web page of the system open.

-Parameter: None.

-Actions:

1. The visitor is presented with the main page of the system.

2. the visitor goes through the different stores and selects an item he is thinking about buying.

3. the visitor clicks on the “add to cart” button presented near the item.

4. the system checks if the visitor already has the item in his cart.

    4.1 if he does then the system sends a message to the visitor telling him that the item was already in his cart and asks the visitor to specify the quantity he wants to buy.

    4.2 otherwise the item is added to the cart and the only message that pops asks for the needed quantity.

Acceptance testing:

1. Good scenario: when trying to add an item to the cart the process is successful, and the visitor can continue to shop.
2. Bad scenario: when trying to add an item to the cart the visitor chooses an item that is out of stock and the system informs the visitor.
3. Sad scenario: when trying to add an item to the cart the visitor chooses an item that is out of stock, but the system does not prevent it.

8. Use case: cart content and modification

<u>-Actor:</u> Visitor.

<u>-Precondition:</u> The visitor already has the app/web page of the system open.

<u>-Parameter:</u> None.

<u>-Actions:</u>

1. The visitor is presented with the main page of the system.

2. the visitor clicks on the "cart" button.

3. the system opens a new page showing the cart content sorted by stores (in alphabetical order). Under each store there is a list of all the products selected from that store.

4. near each item there are 3 buttons, a "remove all" button and a "-" button and "+" button.

    4.1 if the "remove" button is pressed then the item is removed from the cart.

    4.2 if the "+" button is pressed then the quantity of the item is increased by 1 and if the "-" button is pressed the quantity id decreased by 1.

<u>Acceptance testing:</u>

1. <u>Good scenario:</u> when a visitor is trying to view his cart's content the system can successfully load all the information and present it to the visitor.
2. <u>Bad scenario:</u> when a visitor is trying to view his cart's content the system can't successfully load all the information and some of it becomes inaccessible to the visitor.
3. <u>Sad scenario:</u> when a visitor is trying to view his cart's content the system tries to load his cart but accidentally loads another user's cart and shows it to the visitor.

9. Use case: buying and paying

-Actor: Visitor.

-Precondition: The visitor already has the app/web page of the system open and the system is already connected to a payment system.

-Parameter: None.

-Actions:

1. The visitor is presented with the main page of the system.

2. the visitor clicks on the “cart” button.

3. the visitor goes to the end of his shopping list to click on the “order and pay” button.

4. the system loads a new page, showing the total price of the order and asks the visitor to enter payment information.

   4.1 if the information is false an error message will pop informing the visitor.

   4.1 otherwise, the transaction will go through, and the system will send a receipt to the visitor’s email.

Acceptance testing:

1. Good scenario: when a visitor is trying to purchase his cart’s content the system can successfully make the transaction and send a receipt to the visitor.
2. Bad scenario: when a visitor is trying to purchase his cart’s content the payment system returns an error, so the system cancels the transaction and informs the visitor.
3. Sad scenario: when a visitor is trying to purchase his cart’s content the system takes the money from the visitor but without completing the transaction and giving the visitor the items he paid for.

Registered User:

Use case : Exit Market.

- **Actor:** User
- **Pre-condition:** User is logged in.
- **Parameter:** None.
- **Action:**
  1. User selects the "Exit Market" button.
  2. The system receives the input and triggers the logout sequence. (See logout use-case below (2, 2.a 2.b))
     0. The system presents the user with "Exit successful" and exits the market.

  Acceptance Tests:

  - Good: The user selects "Exit Market" button and the system triggers the logout sequence, the user is then presented with an "Exit successful" message.
  - Bad: The user selects "Exit Market" but the system crashes instead of exiting correctly.
  - Sad: The user is not logged in but is still presented with an "Exit Market" button.

Use case : Logout.

- **Actor:** User
- **Pre-condition:** User is logged in.
- **Parameter:** None.
- **Action:**
  1. User selects the logout button.
  2. The system receives the input and handles the logout:

. The system removes all logged in permissions from the user.

. The system saves all data regarding shopping cart and baskets to the database.

  2. The system presents the user with "Logout successful" and returns the user to the home page.

  Acceptance Tests:

  - Good: The user selects the logout button, and the system saved all data correctly.
  - Bad: The user selects the logout button, but instead of handling the request something went wrong and the system crashes.
  - Sad: The user selects the logout button, but the system is treating him as a guest and not saving all relevant information.

Use case : Opening a store.

- **Actor:** User
- **Pre-condition:** User is logged in and chose the open store option.
- **Parameter:** None.
- **Action:**
  1. The system requests for some information about the store.
  2. The user inputs the information (such as the stores' name and description).
  3. The system validates the information.

. If **false**:

a.1. The system presents the user with an error message describing what's wrong.

b. If **true**:

b.1. The system identifies the store with a unique ID number and sets it
in the store.

b.2. The system presents the user with "Shop has been created" message.

b.3. The system saves all store data in the systems' data base.

<u>Acceptance Tests:</u>

- Good: The user tried to open a store, the system identifies the store with a unique ID and presents the user with "Shop has been created", all store data is saved in the database.
- Bad: Good: The user tried to open a store, the system identifies the store with a unique ID and presents the user with "Shop has been created". while trying to save all store data in the database, the system crashes.
- Sad: Good: The user tried to open a store, the system identifies the store with a unique ID and presents the user with "Shop has been created", but store data isn't saved in the database.

Store Owner \ Store manager with the precondition that he has permission for that action:

Use Cases - 4.1: Inventory management.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. System presents an option to manage the inventory in the store **s**.
  2. User selects the inventory management option.
  3. System presents options of adding and removing products, and changing their details.
  4. If the user selects a add product. $\rightarrow Usecase\ 4.1.1$
  5. If the user selects a removing product$\rightarrow Usecase\ 4.1.2$
  6. If the user selects a changing product details$\rightarrow Usecase\ 4.1.3$
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and selects one of the options.
       Excepted Output: The system will present the next screen by the option which the user selects.
  - **Negative:**
    1. **Scenario:** The user isn't logged in to the trading system. He is the owner of store s (the store s is open) and selects one of the options.
       Excepted Output: The system will present a login screen.
    2. **Scenario:** The user is logged in to the trading system. He isn't the owner of store s (the store s is open) and selects one of the options.
       Excepted Output: The system will return alert that the user isn't owner of store s.

Use Cases - 4.1.1: Inventory management – adding product.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. System presents the possibility of adding products with all the necessary fields to be filled in regarding the product.
  2. User enters the product data and adds a product.
  3. System processes the request to add the product.
  4. If the addition of the product is possible.
     a. System adds the product to the store inventory.
     b. System updates the user that the product has been added to the store's inventory.
  5. If adding the product is not possible.
     a. System updates that the product cannot be added and returns the reason.
- **Acceptance Test:**
  - **Positive:**
    2. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he enter all the necessary fields of the product and add the product.
       Excepted Output: The system will return that the product is added.
  - **Negative:**
    3. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he enter all the necessary fields without one necessary field of the product and add the product.
       Excepted Output: The system will return that the product isn't added and the reason.
    4. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he enter all the necessary fields but the product already exists in the store of the product and add the product.
       Excepted Output: The system will return that the product isn't added and the reason.

Use Cases - 4.1.2: Inventory management – remove product.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. System presents the list of products in store’s inventory.
  2. User selects from the list a product that he wishes to remove from the inventory of the products in the store **s**.
  3. If the remove of the product is possible.
     a. System removes the product from the store **s** inventory.
     b. System updates the user that the product has been removed to the store's inventory.
  4. If remove of the product is not possible.
     a. System updates that the product cannot be removed and returns the reason.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he selects one product from the list to remove and remove the product.
       Excepted Output: The system will return that the product is removed.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he try to remove product that doesn’t exist in the store s.
       Excepted Output: The system will return that the product isn’t removed and the reason.

Use Cases - 4.1.3: Inventory management – Changing product details.

- **Actor:** User
- **Pre-condition:**
  5. User is logged in.
  6. The user is a store owner of store **s**.
  7. The Trading system is initialized.
  8. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. System presents the list of products in store's inventory.
  2. User selects from the list a product for which the user wants to change the details.
  3. System presents an option to update the details of the product with all the data fields regarding the product.
  4. User enters the product details and updates the product.
  5. System processes the request to update the product.
  6. If the update of the product is possible.
     b. System updates the product to the store inventory.
     c. System updates the user that the product has been updated.
  7. If the update of the product is not possible.
     d. System updates that the product cannot be added and returns the reason.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he update all the necessary fields of the product and update the product.
       Excepted Output: The system will return that the product is updated.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he enter invalid input field of the product and update the product.
       Excepted Output: The system will return that the product isn't updated and the reason.

Use Cases - 4.2: Changing the types and rules (policy) of the store's purchase and discount.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. System shows the possibility of changing the purchase and discount policy.
  2. User chooses which policy he wants to change.
  3. If user select changing the purchase policy.
     a. System presents to user with a selection option for the various purchase policies with a description for each one.
     b. User selects the policy for the store.
     c. System defines for the store the policy that the user defined.
     d. System updates the employees in the store about the update of the policy.
  4. If user select changing the discount policy.
     a. System presents to user with a selection option for the various discount policies with a description for each one.
     b. User selects the policy for the store.
     c. System defines for the store the policy that the user defined.
     d. System updates the employees in the store about the update of the policy.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he changed policy of the stores purchase.
       Excepted Output: The system will return that the policy is changed.
    2. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he changed policy of the stores discount.
       Excepted Output: The system will return that the policy is changed.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He isn’t the owner of store s (the store s is open) and he changed policy purchase or discount of the stores.
       Excepted Output: The system will return an alert that the user isn’t the owner of store s.

Use Cases - 4.4: Appointment of store owner.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. System presents an option to search for users registered in the system.
  2. User enters in the search bar the name of the registered user he wants to appoint to the store owner.
  3. System presents a pool of registered users that the user has searched for.
  4. User selects from the pool the user **u** he wants to appoint as the owner of the store.
  5. System will check if the user is registered and not the owner of the store already.
  6. If the appointment is possible,
     a. System that will define the user **u** as the owner of the store **s**.
     b. System will grant permissions to the user **u** as the store owner of the store **s**.
     c. System will send the user **u** a message about the appointment.
     d. System will return a message to the user that the appointment was successfully made.
  7. If the appointment is not possible,
     a. System will return a message to the user that the appointment is not possible with the reason.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he select user from the list to be new owner.
       Excepted Output: The system will return that the appointment done and send alert about the appointment.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he selects to appointing user that already owner of store s to be store owner.
       Excepted Output: The system will return that the appointment failed with the reason.
    2. **Scenario:** The user is logged in to the trading system. He isn't the owner of store s (the store s is open) and selects to appoint new owner to store s.
       Excepted Output: The system will return alert that the user isn't owner of store s.

Use Cases - 4.6: Appointment of store manager.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. System presents an option to search for users registered in the system.
  2. User enters in the search bar the name of the registered user he wants to appoint to the store manager.
  3. System presents a pool of registered users that the user has searched for.
  4. User selects from the pool the user **u** he wants to appoint as the manager of the store.
  5. System will check if the user **u** is registered and not the owner or manager of the store already.
  6. If the appointment is possible,
     a. System that will define the user **u** as the manager of the store **s**.
     b. System will grant permissions (Getting of information) to the user **u** as the store manager of the store **s**.
     c. System will send the user **u** a message about the appointment.
     d. System will return a message to the user that the appointment was successfully made.
  7. If the appointment is not possible,
     a. System will return a message to the user that the appointment is not possible with the reason.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he select user from the list to be new manager in store s.
       Excepted Output: The system will return that the appointment done and send alert about the appointment.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he selects to appointing user that already owner or manager of store s to be store manager.
       Excepted Output: The system will return that the appointment failed with the reason.
    2. **Scenario:** The user is logged in to the trading system. He isn’t the owner of store s (the store s is open) and selects to appoint new owner to store s.
       Excepted Output: The system will return an alert that the user isn’t owner of store s.

Use Cases - 4.7: Changing store manager’s permissions.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
  5. **m** is store manager of store **s**.
- **Parameter:** Store **s** and store manager **m**.
- **Actions:**
  1. User requests to manage the **m** manager's permissions.
  2. System presents a list of employees in the store **s**.
  3. System presents the permissions that the manager has and the permissions that are not set for him.
  4. User decides which permissions to add and which to cancel and saves changes.
  5. System makes the changes to the permissions.
  6. System sends an alert to the store manager **m** about the changes in his permissions.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he select user who manger in the store s and change his permissions.
       Excepted Output: The system will return that the permission change and update the user about the changes.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he selects user who isn’t manger in the store s and change his permissions.
       Excepted Output: The system will return that the change of the permissions of this user cancel.
    2. **Scenario:** The user is logged in to the trading system. He isn’t the owner of store s (the store s is open) and change permissions of store manager in store s.
       Excepted Output: The system will return an alert that the user isn’t owner of store s.

Use Cases - 4.9: Closing store.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a founder-store of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. User (founder-store) submits a request to close the store.
  2. System checks the request.
  3. If the request is possible,
     a. System will close the store.
     b. System will define the store as inactive and does not allow access to receive information for users who are not managers or owners of store **s**.
     c. System will send a notification to store owners and managers about the store's closing.
  4. If the request is not possible,
     a. System will return a message to the user that the request to close the store is not possible with the reason.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the founder of store s (the store s is open) and close the store.
       Excepted Output: The system will return that the store close and send about the closed message to the employees in the store.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He is the founder of store s (the store s is already closed) and close the store.
       Excepted Output: The system will return that the store already closed.
    2. **Scenario:** The user is logged in to the trading system. He isn’t the founder of store s (the store s is open) and he try close the store.
    2.
       Excepted Output: The system will return an alert that the user isn’t founder of store s.

Use Cases - 4.11: Request for information about positions in the store.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. User asks to receive information about the positions in the store **s**.
  2. System presents to the user a list of the store's employees with their position in store **s**.
  3. System presents an option to get information about permissions for the managers of the store **s**.
  4. If the user chooses to receive information about the store manager's permissions in store **s**. → $Usecase\ 4.11.1$
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he request for information about positions in the store.
       Excepted Output: The system will return the information which the user requested.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He isn't the owner of store s (the store s is open) and he request for information about positions in the store.
       Excepted Output: The system will return an alert that the user isn't owner of store s.

Use Cases - 4.11.1: Request for information about store manager's permissions.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a store owner of store **s**.
  3. The Trading system is initialized.
  4. The user **u** manager in store **s**.
  5. Store **s** is existing and open.
- **Parameter:** Store **s** and user **u**.
- **Actions:**
  1. User asks to receive information about store manager **u** permissions in store **s**.
  2. System presents the permissions of the store manager **u** in the store **s**.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he request for information about store manager permissions.
       Excepted Output: The system will return the information which the user requested.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He isn't the owner of store s (the store s is open) and he request for information about store manager permissions.
       Excepted Output: The system will return an alert that the user isn't owner of store s.
    2. **Scenario:** The user is logged in to the trading system. He is the owner of store s (the store s is open) and he request for information about store manager permissions but this employee isn't manger in store s.
       Excepted Output: The system will return an alert that the user isn't manager in store s.

Use Cases - 4.13: Receiving information about purchase history in the store.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a system administrator or store owner or manager of store **s**.
  3. The Trading system is initialized.
  4. Store **s** is existing and open.
- **Parameter:** Store **s**.
- **Actions:**
  1. User requests purchase history in the store **s**.
  2. System checks the request.
  3. If the request is possible,
     a. System returns a list of purchase history of store **s**.
     b. System presents a list of the purchase history to the user.
  4. If the request is not possible,
     a. System will return a message to the user that the request is not possible with the reason.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is system administrator or store owner or manager of store **s** (the store s is open) and he request for information about purchase history in the store.
       Excepted Output: The system will return the information which the user requested.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He isn't system administrator or store owner or manager of store **s** (the store s is open) and he request for information about purchase history in the store.
       Excepted Output: The system will return an alert that the user isn't system administrator or store owner or manager of store **s**.

System Administrator:

Use Cases - 6.4: Getting information about purchase history of the store or buyers.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. User is a system administrator.
  3. The Trading system is initialized.
- **Parameter:** None.
- **Actions:**
  1. User asks to see purchase history.
  2. System presents an option to search for the buyer or the store for which the user would like to see the purchase history.
  3. User enters the name of the buyer or store for search.
  4. System presents a list of buyers and stores related to what the user entered in the search.
  5. User selects from the list the store or buyer whose purchase history he wants to see.
  6. If the user selects a buyer $\rightarrow Usecase\ 6.4.1$
  7. If the user selects a store $\rightarrow Usecase\ 4.13$
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is a system administrator and selects one of the options.
       Excepted Output: The system will present the next screen by the option which the user selects.
  - **Negative:**
    1. **Scenario:** The user isn’t logged in to the trading system. He is a system administrator and selects one of the options.
       Excepted Output: The system will present a login screen.
    2. **Scenario:** The user is logged in to the trading system. He isn’t the system administrator and selects one of the options.
       Excepted Output: The system will return alert that the user isn’t system administrator.

Use Cases - 6.4.1: Getting information about purchase history of buyers.

- **Actor:** User
- **Pre-condition:**
  1. User is logged in.
  2. The user is a system administrator.
  3. The Trading system is initialized.
  4. Buyer **b** is existing.
- **Parameter:** Buyer **b**.
- **Actions:**
  1. User requests purchase history of buyer **b**.
  2. System checks the request.
  3. If the request is possible,
     a. System returns a list of purchase history of buyer **b**.
     b. System presents a list of the purchase history to the user.
  4. If the request is not possible,
     a. System will return a message to the user that the request is not possible with the reason.
- **Acceptance Test:**
  - **Positive:**
    1. **Scenario:** The user is logged in to the trading system. He is system administrator and he requests information about purchase history of buyer.
       Excepted Output: The system will return the information which the user requested.
  - **Negative:**
    1. **Scenario:** The user is logged in to the trading system. He is system administrator and he requests information about purchase history in the buyer who don't register to the system.
       Excepted Output: The system will return an alert that the buyer isn't exists in the system.