System

System init

- **Actor:** System
- **Preconditions:**
- **Parameters:**
- **Actions:**
  1) Create Admin User
  2) Init different external components – TBD based on which external service is chosen
     a. Payment System
     b. Notification System
     c. Delivery System

Contact with external payment systems Use Case

- **Actor:** seller
- **Precondition:**
- **Parameter:** Payment info, Store id
- **Actions:**
  1. The seller inserts the payment system info for the relevant store.
  2. The system adds/removes the payment system accordingly.
- **PostCondition:** External system added\removed from the system

Payment Use Case

- Actor: user
- Precondition: user has a cart, the cart has at least 1 basket with at least 1 item.
- Parameter: transaction details (price, id of user), type of system to use (PayPal...)
- Actions:
  1. The system searches all stores for the relevant items.
  2. If at least one item is missing in stock:
     1. The system cancels the purchase and warn the user the following item is missing.

  Else:

  3. The system chooses the right payment system the user asked for.

4. The system sends the transaction information to the external system.
5. If the payment succeeded:
    1. The system sends to the user that the transaction completed.
    2. The system sends to the user his receipt and his receipt.

Else:

1. The system returns to the user that something went wrong.

- **PostCondition:** Cart is emptied, items have been sold and money was transferred from the user to the store.

Supplying Use Case

- Actor: user
- Precondition: user has a cart, the cart has at least 1 basket with at least 1 item
- Parameter: contact info, ordering information
- Actions:
    1. The user inserts his information.
    2. The system sends to the supplier's system the user information and the purchase details (items info).
    3. If the order succeeded:

    The system notifies the user that the order completed.
    Else:
    The system notifies the user that something went wrong.

- PostCondition: The order is saved in the system.

Guest Visitor Actions

Registration

- **Actor:** Guest User.
- **Preconditions:** No user is logged in in the current session.
- **Parameters:** username, password (and more info?).
- **Actions:**
    1) Check username is unique, and password meets requirements (TBD)
    2) Create new User in UserController list and add appropriate permissions in authentication handler
    3) If username/password are ok, user can now log in
    else: get corresponding message

Log in

- **Actor:** Registered User.
- **Preconditions:** Not currently in a session.
- **Postconditions :** The user now can see his saved data,(cart, info, messages).
- **Parameters:** username, password.
- **Actions:**
  1) Check credentials
  2) If credentials are correct, a session is created with a unique ID that maps to the user
     else: get corresponding message
  3) User now has all his system options including stores he owns/manages and cart
  4) If user has notifications waiting, pop-up with info to check the messages

System Entry (Visit the website)

- **Actor:** Guest.
- **Preconditions:**
- **Parameters:**
- **Actions:**
  1) Load all stores
  2) Create an empty cart

System Exit

- **Actor:** Any.
- **Preconditions:**
- **Parameters:**
- **Actions:**
  1) If user is logged in, save cart

2) If no user was logged in, delete cart

Guest purchase Actions

Get store Info

- **Actor**: Guest user
- **Preconditions**: store exists, store is open
- **Parameters**: store id
- **Actions**:
  1. User requests to see store info by store id

2. The system searches for the given store by the store id
3. The system returns the store info.

Get Item Info

- **Actor**: Guest user
- **Preconditions**: item exists, the store of the item is open
- **Parameters**: item id
- **Actions**:
  1. User requests to see item info by item id
  2. The system searches for the given item id
  3. The system returns the item info

Search Items (by name)

- **Actor**: Guest user
- **Preconditions**: None
- **Parameters**: Item name
- **Actions**:
  1. User requests to search for all items with the given string as its name.
  2. The system searches for all items with the given name.
  3. The system returns the list of items with the given name.

Search Items (by category)

- **Actor**: Guest user
- **Preconditions**: Category exists
- **Parameters**: Category name or id
- **Actions**:
  1. User requests to search for all items in the given category.
  2. The system searches for all items in the given category.
  3. The system returns the list of items in the given category.

Filter Searched Items (price range, item rating, category, store rating…)

- **Actor:** Guest user
- **Preconditions:** User has a list of searched items
- **Parameters:** filters and list of items
- **Actions:**
  1. User requests to filter his searched items list.
  2. The system removes from the list all the items that don't meet the filters.
  3. The system returns the filtered list of items.

Add To Basket

- **Actor:** Guest user
- **Preconditions:** User has cart, store and item exists, store is open
- **Parameters:** User id, store id, item id, amount
- **Actions:**
  1. The system searches for the item in the store.
  2. If the user's cart doesn't have a basket for the given store, then the system creates a new basket for the store in the user's cart with the item.

3. Else, the system adds the item to the basket.
- **Postconditions**: The user's basket contains the item and its amount increased by <Parameter: amount>.

Show Cart

- **Actor:** Guest user
- **Preconditions:** User has a cart
- **Parameters:** User id
- **Actions:**
  1. User requests to see his current cart.
  2. The system searches for the user's cart
  3. The system returns the user's cart.

Modify Cart

- **Actor:** Guest user
- **Preconditions:** User has a cart
- **Parameters:** User id, parameters to change (remove item, add/remove amount of an item, remove basket…)
- **Actions:**
  1. User requests to change his cart.
  2. The system searches for the user's cart
  3. The system changes the user's cart with the requested parameters.
- **Postconditions:** The user's cart has changed by the parameters (remove item, add/remove amount of an item, remove basket…)

Buy Cart

- **Actor:** Guest user
- **Preconditions**: User has a cart, cart has at least 1 basket with at least 1 item, stores of the baskets are open
- **Parameters**: User id, user's payment info and his preferred payment method
- **Actions:**
  1. The system searches for the user's cart
  2. If all items of the cart exist in stock:
     1. The system interfacing with the chosen external payment system and makes the purchase.
     2. The system removes the items from the stocks.
     3. Deletes the cart with its baskets.

     Else:
     1. The payment is being canceled.
     2. The user gets an error message.
- **Postconditions:** The user's cart is empty, the amounts of items in the stores is lower by a corresponding amount.

Registered User Actions

Log out

- **Actor:** Logged in User.

- **Preconditions:** User is logged in in current session.
- **Parameters:**
- **Actions:**
  1) Remove User from session.
  2) User now only has contact with system as a guest

Create a Store

- Actor: Registered User.
- Preconditions:
- Parameters: Store Info
- Actions:
  1) Check that info is ok
  2) Create a store with matching info and status “Open”
  3) User becomes Store Founder and is registered to receive appropriate messages
  4) Store management options appear

Store Owner Actions

Change discount and buying policies as Store Owner Use Case

- **Actor:** Store Owner
- **Precondition:**
- **Parameter:** new Policies
- **Actions:**
  1. The Store Owner inserts the new policies information.
  2. The new policies are saved in the relevant store.
- **PostCondition:** the new policies are saved in the system.

Managing Inventory as Store Owner Use Case

- **Actor:** Store Owner
- **Precondition:**
- **Parameter:**
- **Actions:**
  1. If Store Owner chooses "Adding items" -> (1)
  2. If Store Owner chooses "Removing items" -> (2)
  3. If Store Owner chooses "Changing details" -> (3)
- **PostCondition:** Item was added\removed\updated in the inventory.

(1) Adding item Use Case

- **Actor:** Store Owner
- **Precondition:**
- **Parameter:** item details

- **Actions:**
    1. The Store Owner inserts item details to the system.
    2. The system saves the item in the store.

(2) **Removing item Use Case**

- **Actor:** Store Owner
- **Precondition:**
- **Parameter:** item id
- **Actions:**
    1. The Store Owner inserts the item's id to the system.
    2. The system removes this item from the store.

(3) **Changing item details Use Case**

- **Actor:** Store Owner
- **Precondition:**
- **Parameter:** item id, new item details
- **Actions:**
    1. The Store Owner inserts item's id and new details to the system.
    2. The system updates the item details.

Show information about store staff.

- **Actor:** store owner.
- **Preconditions:** User is logged-in, store is not permanently closed.
- **Parameters:** store id, user id.
- **Actions:**
    1) System verifies the ownership of the user on current store.
    2) System iterates through staff tree (pre-order) and returns a list of all the information in the nodes.
    3) System shows the list on screen.

Get selling history (store owner)

- **Actor:** store owner.
- **Preconditions:** User is logged-in, store is not permanently closed.
- **Parameters:** store id, user id.
- **Actions:**
    1) System verifies the user is indeed the store's owner.
    2) System search for the store's receipts.
    3) System iterates through the store's receipts and copy their content.
    4) System returns the list.

5) The receipts list is shown on screen.

Define Co-Owner

- **Actor:** Store Owner.
- **Preconditions:** Owner is logged in, User to be added is not currently one of the store's owners.
- **Parameters:** Store, User to be added
- **Actions:**
  1) Add user to list of store's owners
  2) Add store to list of user's owned stores
  3) Add store ownership permissions for specified user
  4) Notify user of new role

Remove Co-Owner

- **Actor:** Store Owner.
- **Preconditions:** Actor is Logged in, Owner to remove was defined by actor
- **Parameters:** Store, Owner to remove
- **Actions:**
  1) Remove all owners and managers the removed defined
  2) Notify those removed that they were removed

Define Manager

- **Actor:** Store Owner.
- **Preconditions:** Owner is Logged in, user to be added is not currently one of the store's managers
- **Parameters:** store, user to be added
- **Actions:**
  1) Add user to list of store's managers
  2) Add store to list of user's managed stores
  3) Add store managing permissions for specified user
  4) Notify user of new role

Edit Manager Permissions

- **Actor:** Store Owner.
- **Preconditions:** Owner is Logged in, user to be added is not currently one of the store's managers
- **Parameters:** store, manager, permissions, to add or remove (Boolean)
- **Actions:**
  1) If adding permissions:
     a. Check if manager has these permissions
     b. If not, add the permissions
     c. Notify manager of change

2) If Removing permissions:
    a. Check if manager has these permissions
    b. If so, remove the permissions
    c. Notify manager of change

Closing a Store

- **Actor:** Store Founder.
- **Preconditions:** Founder is Logged in.
- **Parameters:** Store
- **Actions:**
  1) Remove Store from system list of stores
  2) Notify other store owners of the store closing

Remove Manager

- **Actor:** Store Owner.
- **Preconditions:** Actor is Logged in, Manager to remove was defined by actor
- **Parameters:** Store, Manager to remove
- **Actions:**
  1) Remove manager's permissions to access store info
  2) Notify those removed that they were removed

Store Manager actions

Get store's information and answer requests.

- **Actor:** store manager.
- **Preconditions:** User is logged-in, store is not permanently closed.
- **Parameters:** store is, user id
- **Actions:**
  1) System verifies the user is the manager of the current store.
  2) If "View store's information" is chosen:
     1) System shows basic store information, in addition to any information the manager has access to.
  3) If "Answer questions and requests" is chosen:
     1) System gets all the messages from the store's mailbox.
     2) System shows all the messages on the screen.
     3) System gives the ability to choose a message to answer.
     4) System sends the manager's answer to the mailbox of the sender.

Get selling history (store manager)

- **Actor:** store manager.
- **Preconditions:** User is logged-in, store is not permanently closed.
- **Parameters:** store id, user id.

- **Actions:**
  1) System verifies the user is indeed the store's manager and that the manager has the proper permissions.
  2) System search for the store's receipts.
  3) System iterates through the store's receipts and copy their content.
  4) System returns the list.
  5) The receipts list is shown on screen.

System manager actions

Get selling history (system manager)

- **Actor:** System manager.
- **Preconditions:** user is logged in.
- **Parameters:** user id.
- **Actions:**
  1) System verifies the user is a system manager.
  2) System asks for a store's id / name to search.
  3) If found, system iterates through all the store's receipts and returns the list.
  4) The receipts list is shown on the screen.

All users actions

Send Notification

- **Actor:** Any.
- **Preconditions:**
- **Parameters:** Message type, SenderID, Message, store ID (sometimes)
- **Actions:**
  1) Action happens that requires a message – i.e. purchase from owners store, or direct message
  2) Message is added to receiver’s inbox
     a. If receiver is logged in receives notification of incoming message
     b. If receiver is not logged in, will receive immediately after logging in