**Table of Contents**

System Boot Requirement: 1.1 page 3

Update External Systems Requirement: 1.2 page 3

Payment Requirement: 1.3 page 3

Supply Requirement: 1.4 page 4

Real time alerts Requirement: 1.5 page 4

Delayed alerts Requirement: 1.6 page 4

Enter System Requirement: 2.1.1 page 5

Exit System Requirement: 2.1.2 page 5

Register Requirement: 2.1.3 page 5

Login Requirement: 2.1.4 page 6

Receive Store Info Requirement: 2.2.1 page 6

Search Store/Item Requirement: 2.2.2 page 6

Save Item Requirement: 2.2.3 page 7

View ShoppingCart Items Requirement: 2.2.4 page 7

Purchase ShoppingCart Requirement: 2.2.5 page 7

Purchase ShoppingCart – Supply Requirement: 2.2.5 page 8

Purchase ShoppingCart – Payment Requirement: 2.2.5 page 8

Logout – Registered User Requirement: 2.3.1 page 8

Open Store Requirement: 2.3.2 page 9

Stock Management – Add New Item Requirement: 2.4.1 page 9

Stock Management – Remove Item Requirement: 2.4.1 page 9

Stock Management – Change Item Information Requirement: 2.4.1 page 10

Set Store Policy Requirement: 2.4.2 page 10

Appoint Store Owner Requirement: 2.4.4 page 10

: Appoint Store Manager Requirement: 2.4.6 page 11

Set Store Manager Permissions Requirement: 2.4.7 page 11

Close Store Requirement: 2.4.9 page 11

Get Store Staff List Requirement: 2.4.11 page 12

Get Store Sale History Requirement: 2.4.13 page 12

Store Manager Actions Requirement: 2.5 page 12

Get Store Sale History – System Admin Requirement: 2.6.4 page 12

**Usecase Scenarios:**

1. Name: System Boot

Requirement: 1.1

Actors: System Administrator

Preconditions: None

Postconditions: System is running.

Input Parameters: JSON file of external systems (at least one payment system and one supply system).

Actions Scenario:

1. System Administrator starts server and inputs the external systems configurations (at least one payment system and one supply system).
2. The system runs integrity checks:

* Checks for the existence of at least one administrator account.
* Creates connection with the external systems.

1. Name: Update External Systems

Requirement: 1.2

Actors: System Administrator.

Preconditions: None

Postconditions: New/updated external system is online.

Input Parameters: JSON configurations file.

Action Scenario:

1. Add:

* Create new config file for the new system.
* Call the “add external system” function that will read the file and create the connection to the external system.

1. Change Existing System:

* Change the JSON file of the existing system.
* Call the “update existing external system” function that will temporarily shut down the external system and recreate it with the new configuration.

1. Name: Payment

Requirement: 1.3

Actor: Client, external payment system .

Preconditions: Existing connection with an external purchase system.

Input Parameters: Transaction information.

Action Scenario:

1. User enters his/her credit card information.
2. The system forwards the payment request to the external purchase system through the PaymentController object.
3. If the payment is successful return confirmation number, otherwise return error message.
4. Name: Supply

Requirement: 1.4

Actor: External supply system

Preconditions: Existing connection with an external supply system.

Postconditions: Supply confirmation received.

Input Parameters: Item information and client information.

Action Scenario:

1. User enters his address into the system or confirms his saved address if he is registered.
2. The system passes the information to the external supply system through the SupplyController object.
3. The external system returns its response - success or failure.
4. Name: Real time alerts

Requirement: 1.5

Actor: Client A, Client B (Optional)

Preconditions: Notification-worthy event happens for Client A, Client A is online

Postconditions: Client A receives notification.

Input Parameters: Event

Action Scenario:

1. System/Client B causes an event that requires Client A to receive a notification.
2. The NotificationController receives the event and creates a notification object and adds it the user’s queue.
3. Client A checks the notification queue and the user receives the message.
4. Name: Delayed alerts

Requirement: 1.6

Actor: Registered User A, Registered User B (Optional)

Preconditions: Notification-worthy event happens for Registered User A, Registered User A is offline

Postconditions: Registered User A’s notification queue receives the notification.

Input Parameters: Event

Action Scenario:

1. System/ Registered User B causes an event that requires Registered User A to receive a notification.
2. The NotificationController receives the event and creates a notification object and adds it the user’s queue.
3. When Registered User A becomes online the system will check the queue and the user will receive the notification.
4. Name: Enter System

Requirement: 2.1.1

Actor: Client

Preconditions: None

Postconditions: None

Input Parameters: None

Action Scenario:

1. Client enters the system via the specific interface.
2. UserController object logs the entry to the system and creates a new client object, along with a ShoppingCart object.
3. The client is now able to interact with the system.
4. Name: Exit System

Requirement: 2.1.2

Actor: Client

Preconditions: None

Postconditions: None

Input Parameters: None

Action Scenario:

1. Client chooses to exit the system by closing his/her interface.
2. If the client had an active ShoppingCart then it is deleted.
3. UserController deletes the client object and updates the DB appropriately.
4. Name: Register

Requirement: 2.1.3

Actor: Client

Preconditions: Client is not logged in.

Postconditions: New Registered User is added to the system.

Input Parameters: Identifying information

Action Scenario:

1. Client chooses the registration option.
2. Client enters identifying information.
3. UserController checks that the unique identifiers aren’t present in the system:

* Success: The identifiers are found to be unique and the registration is successful. UserController passes the password to the SecurityController to be encrypted and saved, and passes the client a success message.
* Failure: The identifiers aren’t unique – failure message sent to client.

1. Name: Login

Requirement: 2.1.4

Actor: Client

Preconditions: Client is not logged in.

Postconditions: Client is logged in.

Input Parameters: Username, password

Action Scenario:

1. Client chooses the login option.
2. Client enters username and password and confirms.
3. UserController dispatches call to SecurityController with the username and password for validation.
4. Response:

* Success: SecurityController validates the login request successfully and returns a success response. UserController marks the user as logged in and returns a success message to the client.
* Failure: SecurityController finds that there is a mismatch between the username and the given password or does not find the specified username and returns a failure response. UserController returns a failure message to the client.

1. Name: Receive Store Info

Requirement: 2.2.1

Actor: Client

Preconditions: None

Postconditions: None

Input Parameters: Store/Item information.

Action Scenario:

1. Client chooses a store/item to view.
2. The StoreController object receives the request and returns the item list (if store) or specific item information.
3. Name: Search Store/Item

Requirement: 2.2.2

Actor: Client

Preconditions: None

Postconditions: None

Input Parameters: Store/Item information to search.

Action Scenario:

1. Client enters search parameters to the system.
2. The SearchController object receives the request and searches the database/cache for items or stores that match the parameters.
3. SearchController returns and indexed list of relevant results.
4. Name: Save Item

Requirement: 2.2.3

Actor: Client

Preconditions: None

Postconditions: Item is added to a new/existing ShoppingBasket.

Input Parameters: Item

Action Scenario:

1. Client finds an item.
2. Store object checks the quantity of the item and shows to client.
3. Client chooses amount of the item to add to cart and confirms.
4. If the ShoppingCart object does not have a ShoppingBasket object of the item’s store, it creates one.
5. Either way, the item is added to the relevant ShoppingBasket.
6. Name: View ShoppingCart Items

Requirement: 2.2.4

Actors: Client

Preconditions: None

Postconditions: None

Input Parameters: None

Action Scenario:

1. Client requests to view the shopping cart.
2. If the shopping cart contains items, it will show them sorted by store.
3. Otherwise, the client will receive a message that the shopping cart is empty.
4. Name: Purchase ShoppingCart

Requirement: 2.2.5

Actors: Client

Preconditions: ShoppingCart is not empty

Postconditions: Items purchased do not appear in ShoppingCart, Client Purchase History contains the items purchased and Store Sale History contains the items sold.

Input Parameters: Items to purchase, client shipping address and payment information.

Action Scenario:

1. Client selects the Purchase Cart option.
2. Client enters shipping address and confirms.
3. SupplyController passes the information to the external supply system for verification (see Usecase #16). On success:
4. Client enters payment information and confirms.
5. PaymentController passes the information to the external payment system for verification (see Usecase #17). On success:
6. Client gives final confirmation for all information and the confirmation finalizes the purchase with regards to the Payment and Supply systems.
7. StoreController object creates new Purchase & Sale objects, saves them to the database and deletes the purchased items from the user’s cart.
8. Name: Purchase ShoppingCart – Supply

Requirement: 2.2.5

Actors: External Supply System

Preconditions: Supply system connected to market system

Postconditions: Supply request confirmed by external supply system.

Input Parameters: Items to purchase, client shipping address.

Action Scenario:

1. StoreController passes supply information to SupplyController.
2. SupplyController verifies the passed information and creates a verification request from the external system.
3. If the request is successful and the information is verified, the SupplyController saves the information internally and returns the success response to the StoreController.
4. StoreController passes final confirmation to the SupplyController which sends an order request to the external supply system.
5. SupplyController returns the appropriate response to the StoreController.
6. Name: Purchase ShoppingCart – Payment

Requirement: 2.2.5

Actors: External Payment System

Preconditions: Payment system connected to market system

Postconditions: Payment request confirmed by external payment system.

Input Parameters: Items to purchase, client payment information.

Action Scenario:

1. StoreController passes payment information to PaymentController.
2. SupplyController verifies the passed information and creates a verification request from the external system.
3. If the request is successful and the information is verified, the PaymentController saves the information internally and returns the success response to the StoreController.
4. StoreController passes final confirmation on the payment to the PaymentController which then sends a payment request to the external payment system.
5. PaymentController returns the appropriate response to the StoreController.
6. Name: Logout – Registered User

Requirement: 2.3.1

Actor: Registered User

Preconditions: Registered User is logged in

Postconditions: Registered User is logged out and is now a client only.

Input Parameters: None

Action Scenario:

1. Logged in user selects the logout option.
2. UserController removes the user from the LoggedInUsers dictionary.
3. The user will receive a message stating that the logout action was successful.
4. Name: Open Store

Requirement: 2.3.2

Actor: Registered User

Preconditions: User is logged in

Postconditions: New store opened

Input Parameters: Store name, store description

Action Scenario:

1. Registered User chooses the option to create a new store.
2. Registered User enters store name and description.
3. StoreController creates a new store instance with the inputted parameters.
4. StoreController saves the store to the database and returns to the user object a new StoreFounder Role object.
5. The user receives a message notifying him that the store opened successfully.
6. Name: Stock Management – Add New Item

Requirement: 2.4.1

Actor: Store Owner

Preconditions: User has an open store.

Postconditions: New item added to store.

Input Parameters: Item information

Action Scenario:

1. Store owner selects store.
2. Store Owner selects item.
3. Store Owner selects the Add New Item option.
4. Store Owner enters relevant item information.
5. StoreController checks item information entered for faults:

* Success: StoreController adds item to database.
* Failure: Store Owner receives error message.

1. Name: Stock Management – Remove Item

Requirement: 2.4.1

Actor: Store Owner

Preconditions: User has an open store and an item.

Postconditions: Item removed from store.

Input Parameters: Item ID

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects item.
3. Store Owner selects the Remove Item option.
4. StoreController checks item information entered for faults:

* Success: StoreController removes item from the database.
* Failure: Store Owner receives error message.

1. Name: Stock Management – Change Item Information

Requirement: 2.4.1

Actor: Store Owner

Preconditions: User has an open store and an existing item.

Postconditions: Item information changed.

Input Parameters: Item information to change.

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects item.
3. Store Owner selects the Change Item Information option.
4. Store Owner edits item information.
5. StoreController checks item information entered for faults:

* Success: StoreController modifies item in database.
* Failure: Store Owner receives error message.

1. Name: Set Store Policy

Requirement: 2.4.2

Actor: Store Owner

Preconditions: Store exists and open.

Postconditions: New policy applied.

Input Parameters: Policy parameters

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects the Set Store Policy option for a specific store.
3. Store Owner inputs the policy parameters.
4. Store object checks its previous policies and Founder policy for inconsistencies:

* Success: Store policy is saved to database by StoreController object.
* Failure: Inconsistency found and error message shown – go to step b.

1. Name: Appoint Store Owner

Requirement: 2.4.4

Actors: Store Owner, Registered User

Preconditions: Store exists and is open, Registered User isn’t one of the store’s owners.

Postconditions: Registered User is now an owner of the store under the appointing owner.

Input Parameters: None

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects the Appoint Store Owner.
3. Store Owner searches for the Registered User to appoint.
4. Store Owner selects the user and confirms the selection.
5. StoreController object checks that the user is not already an owner of the store.
6. Registered User receives the appointment, which he may confirm.
7. StoreController object returns the user object a StoreOwner Role assigned to the store and saves the role in the database.
8. Name: Appoint Store Manager

Requirement: 2.4.6

Actors: Store Owner, Registered User

Preconditions: Store exists and is open.

Postconditions: Registered User is a manager of the store.

Input Parameters: None

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects the Appoint Store Manager option.
3. Store Owner searches for the user to appoint.
4. Store Owner selects the user and confirms the selection.
5. StoreController object checks that the user isn’t already a manager of the store.
6. Registered User receives the appointment, which he may confirm.
7. StoreController object attaches a StoreManager Role (with information reading permissions only) to the user object that is assigned to the store and saves the role and permissions in the database.
8. Name: Set Store Manager Permissions

Requirement: 2.4.7

Actors: Store Manager, Store Owner

Preconditions: None

Postconditions: Store Manager Permissions changed.

Input Parameters: Permission parameters.

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects the Set Store Manager Permissions option.
3. Store Owner selects the desired manager out of a list.
4. Store Owner inputs the new permissions for the manager.
5. Store Owner confirms the selection.
6. StoreController object adds the new permissions to the manager’s Role object’s Permissions collection.
7. Store Manager receives a notification of the changed Permissions.
8. Name: Close Store

Requirement: 2.4.9

Actors: Store Founder

Preconditions: Store is open.

Postconditions: Store is closed.

Input Parameters: None

Action Scenario:

1. Store Founder selects store.
2. Store Founder selects the Close Store option and confirms it.
3. StoreController object updates the database and cache: Sets the store to closed and all its items as hidden.
4. All managers and owners receive a notification that the store has been closed.
5. Name: Get Store Staff List

Requirement: 2.4.11

Actors: Store Owner

Preconditions: None

Postconditions: None

Input Parameters: None

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects the Get Store Staff List option.
3. StoreController object searches the database for all users with a role assigned to the store ID and returns the list along with the users’ permissions for viewing.
4. Name: Get Store Sale History

Requirement: 2.4.13

Actors: Store Owner

Preconditions: None

Postconditions: None

Input Parameters: None

Action Scenario:

1. Store Owner selects store.
2. Store Owner selects Get Store Sale History option.
3. StoreController object queries the database for all sales made by the store and compiles a collection of sales and returns them to the Store Owner for viewing.
4. Name: Store Manager Actions

Requirement: 2.5

Actors: Store Manager

Preconditions: None

Postconditions: None

Input Parameters: None

Action Scenario: All Store Manager actions are to be consistent with his permissions. All actions permitted to him work as if the Store Owner performs them, and so for each particular action see Use Cases for Store Owner Requirements (2.4.x).

1. Name: Get Store Sale History – System Admin

Requirement: 2.6.4

Actors: System Admin

Preconditions: None

Postconditions: None

Input Parameters: Store ID

Action Scenario:

1. System Admin selects store.
2. System Admin selects Get Store Sale History option.
3. StoreController object collects the store that has the passed Store ID as its ID’s sales from the database and returns them to the admin for viewing.