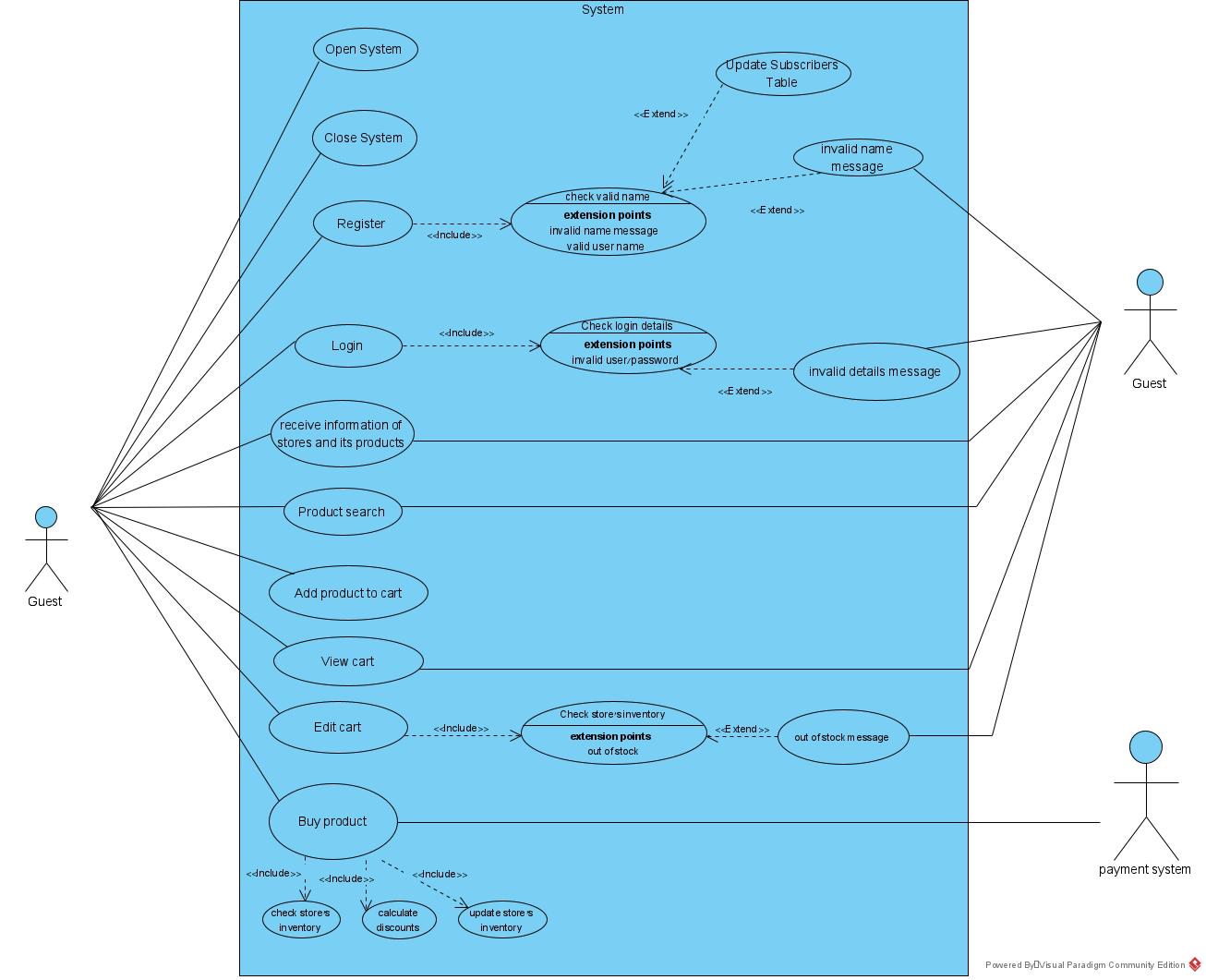
**Use Cases:**

**2- Guest Buyer**

**2.1: Enter the trading system  
Actor:** Any User  
**Precondition:** User is offline  
**Post condition:** User is onlineas a guest. **Parameters:** none **Actions:**

1. User enters the system
2. The System initiates (use case 1.1).
3. The user is defined as guest.

**Positive:**

User enters the system, the system initialization succeeded and the user is defined as guest.

**Negative:**User initiates the system, the connections to the external systems initiated successfully but no system manager was detected, the operation fails and an error message was sent to the user.

**2.2: Exit the trading system  
Actor:** Any User  
**Precondition:** User is online **Post Condition:** User is offline **Parameters:** none **Actions:**

1. Guest user exits the system
2. The system closes the connection to the external systems
3. The system saves all unsaved data in its tables

**Positive:**

User exits the system closes the connection to the external systems and saves all relevant data.

**Negative:**

User forces exit on the system; some temporary data wasn't saved in the systems tables.

**2.3 : Register to the trading system**  
**Actor**: Guest User  
**Precondition**: username is not already used  
**Post Condition**: username and password were added to the users table  
**Parameters**: username , password  
**Actions**:

1. Guest user chooses to register and enters his username and password
2. The system checks the username is not already used
3. The system checks for the username and password validity
4. If all details are valid a new user is created and saved in the system tables
5. Otherwise the process fails

**Positive:**

Guest user registers to the system with a username and password, the system verifies the username uniqueness, and a new subscriber is saved in the systems tables.

**Negative:**

Guest user tries to register with a new username and password, the system checks for the username uniqueness but discovers the username is already taken by another subscribed user. The process fails.

**2.4: Login to the trading system  
Actor:** Guest user  
**Precondition:** no logged on user in current session, user is already registered **Post Condition:** User is logged onas Subscriber **Parameters:** username and password **Actions:**

1. Guest user enters his username and password
2. The system checks for a match between the username and password
3. If the username and password matches the guest user changes his state to logged on
4. Otherwise the process fails

**Positive:**

The user logs into the system with his details, the system checks successfully for a match between the username and the password. The user is now logged on as a subscriber.

**Negative:**

The user tries to log into the system with a username and password, the system checks for match between the username and password but finds out there is no such user in the system. The process fails.

**2.5: View a store and its products.**  
**Actor:** Any User  
**Precondition:** The store exists in the system  
**Post Condition:** none  
**Parameters:** The store  
**Actions:**

1. The user chooses a store
2. The system returns all details about the store and the products in the store

**Positive:**

The user chooses a store, the system returns all relevant details about the store and its products.

**Negative:**

The user chooses a store to view, but the store doesn’t exist in the system.

**2.6: Search for products which can be filtered by some parameters.  
Actor:** Any User  
**Precondition:** none **Post Condition:** only data filtered by the parameters is presented **Parameters:** one or more of: product name, category and keywords**.**optional : any number of product's attributes such as: price range, ranking , etc. **Actions:**

1. Guest user asks to see details about products according to one of the following: product name, category and keywords
2. The trading system searches all stores and return only products that correspond to the parameters given.
3. If optional parameters are given the system filters the results from the previous stage according to these parameters.
4. The filtered data is returned

**Positive:**

The user searches for products by product name – "basketball", and filters the data by price range, the system searches for basketball in all stores and returns only those in the relevant price range.

**Negative:**

The user tries to search products without any parameters, the search fails.

**2.7: Add products from a store to store-cart.  
Actor:** Any User  
**Precondition:** The store's inventory contains all requested products. **Post Condition:** The products has been added to the users old shopping cart. **Parameters:** products and quantities to add. **Actions:**

1. The user asks to add products to his store-cart
2. If the user has no store-cart for this store, a new store-cart is added to his shopping-cart and the products are added to it.
3. Otherwise the products are added to his old shopping cart
4. If the user is a subscriber the store-cart is saved in the system's tables.

**Positive:**

The user adds products to his store-cart, the system checks the store's inventory, and adds the products to his old store-cart.

**Negative:**

The user tries to add products to his store-cart, the system checks the store's inventory and finds out the store doesn’t have enough of the requested product

**2.8.1: View shopping cart**.  
**Actor**: Any user  
**Precondition**: none  
**Post Condition**: none  
**Parameters**: none  
**Actions**:

1. The user asks to see his shopping-cart
2. The system returns a list of all of his store-carts

**Positive:**

User chooses to view his shopping cart, the system collects all of his store-carts and returns it.

**2.8.2 : Edit shopping-cart  
Actor:** Any user  
**Precondition:** none **Post Condition:** the shopping-cart has changed according to the user's request. **Parameters:** list of <product, quantity> to be changed **Actions:**

1. The user views his shopping-cart (use case 2.8.1) and chooses a store-cart to edit
2. The user enters list of products that exists in his store-cart with their new quantities
3. If the user chooses to increase the amount of products in his store-cart, the system checks for the product availability in the store
4. The store-cart is saved in the system's tables

**Positive:**

The User views his shopping-cart and chooses to edit specific store-cart, he decided to increase the amount of a product he previously added, the store's inventory is checked successfully, and the store cart is saved.

**Negative:**

The User views his shopping-cart and chooses to edit specific store-cart, he decided to increase the amount of a product he previously added, the store's inventory doesn’t contain enough of this product and the process fails.

**2.9: Guest User asks to purchase products  
Actor:** Any user  
**Precondition:** The user have at least one none empty store-cart **Post Condition:** The user's store-cart and the store's inventory is updated according to the amount of purchased products **Parameters:** products, amounts and purchase type **Actions:**

1. User asks to buy products from a store
2. The system verifies that all products are in the store's inventory
3. The system calculates discounts in the store according to its buying and discount policies.
4. The system notifies the user about the final price and asks for his approval to commit payment
5. The system transfers the payment request to the external payment system and waits for its response (use case 7).
6. If the payment has been approved the user's cart and the store's inventory are updated
7. Otherwise the error message sent to the user

**Positive:**

The user requests to pay for his shopping cart, the system verifies all stores has the requested products and then calculates the final price, the user accepts to pay and a payment request is sent to the payment system. The payment request is approved and the store's inventory is updated.

**Negative:**

The user requests to pay for his shopping cart, the system verifies all stores has the requested products and then calculates the final price, the user accepts to pay and a payment request is being sent. The payment request is rejected and the items stay in the user's shopping cart.

**3- Subscribed Buyer**

**3.1: logout  
Actor:** Subscriber  
**Precondition:** User is subscribed **Post Condition:** User is logged off as Guest **Parameters:** none **Actions**:

1. Subscribed user logs into the system
2. After the user finished he decides to logout from the system
3. The system backs up all unsaved data.
4. The system marks the user as logged out
5. The user returns to the guests Users menu

**Positive:**

Logged in buyer decides to logout, the system saves all data and the user now acts as a guest.

**Negative:**

Logged in buyer decides to logout, the system fails and the user still acts as a subscriber.

**3.2: open store  
Actor:** Subscriber  
**Precondition:** User is subscribedand logged on, parameters are valid **Post Condition:** A new store was created in the system, the subscribed user is its only store owner **Parameters:** Name, Description, policies, products and amounts. **Actions**:

1. A Subscribed user decides to open a store
2. He enters the stores name, description, policies and products
3. The system verifies the parameters validity and specifically checks for at least one buying policy and one discount policy
4. If the parameters aren’t valid the system cancels the request and the user receives error message
5. Otherwise a new store is created, in addition the user is marked as the store's owner and gets all available permissions over the store

**Positive:**

Subscribed user decides to open a store; he provides valid parameters. A new store is created and the user is marked as its only owner.

**Negative:**

Subscribed user decides to open a store; he provides all relevant parameters but it turns out the buying policy he provided is illegal. The system informs the user that buying policy is necessary in order to open a store. The process fails

**3.7: get purchase history  
Actor:** Subscriber  
**Precondition:** User is subscribedand logged on **Post Condition:** none

**Parameters:** none **Actions**:

1. A Subscribed user logs into the system
2. The user requests to view his purchase history
3. The system returns the purchase history to the user