

Team Members: Austin Blanchard, Richard Hutcheson, Noah Lambaria, Johnny Acosta

Team Leader: Richard Hutcheson

Project: Bear Market

Project Vision:

The vision for the project is to create a marketplace simulation that allows users to create and log into a marketplace account. Within their account, the user is able to browse and search a catalog of items other users are selling. The user is able to view item details and their price, and then purchase the item in whatever quantity desired and available. The user is also able to create market postings of their own to sell items and have those items be purchased by other users.

Issue Tracking Site: <https://github.com/Richard-Hutch/BearMarket/issues>

Website: <https://richard-hutch.github.io/BearMarket/>

Git Link: <https://github.com/Richard-Hutch/BearMarket>

Team Member	Use Case Responsibility
Austin Blanchard	search/sort Items, user purchase history/currently selling, edit user market posting
Richard Hutcheson	Create Account, Log-In, Edit Account
Noah Lambaria	featured items, product browsing, user review
Johnny Acosta	delete a market posting,create market post, purchase item

Time Tracker

Team Member	Hours Worked
Richard Hutcheson	27
Johnny Acosta	18.5
Austin Blanchard	18
Noah Lambaria	18.5

Functional Requirements:

- The system will return item results based on the user's input.
- If the user does not have an account to our website they should be able to create one to interact with our marketplace. Else if the user has an account they should be able to login and access all of their user information and sell history.
- Users should be able to sort items by price, rating, etc. to find help find their desired product to purchase.
- When a user selects the delete icon on one of their products, the system deletes the posting.
- System should be able to generate featured items
- The user is able to create a new posting and enter all of the product details. The system places that product on the public catalogue.
- The user is able to click and modify any of their product postings. The system then updates that product.
- The system gives a unique purchase ID for each order.
- The system records and gives user access to the selling history of each of their products.
- The user is able to view the rating of a product and see the reviews made of the item.
- The user is able to make a review on a product and rate it if that product has been on their purchase history.

Non-functional Requirements:

- The system shall return results in a reliable and fast manner
- The system shall store all users items
- The system should protect personal user information
- The system should be able to reliably retain login information to interact with the marketplace.
- The system should reliably save login information when a user is going through the account creation process

<p>ID: Create Account</p> <p>Scope: Marketplace system</p> <p>Level: user goal</p> <p>Stakeholders and Interests:</p> <p>User</p> <p>- person interested in creating a personal account</p> <p>Precondition:</p> <p>-user does not already have an account</p> <p>Postcondition:</p> <p>-user has a new account</p>	<p>Main success scenario:</p> <ol style="list-style-type: none">1. user wants to create an account2. user clicks button to create a new account3. user fills out first, last name4. user fills out payment information5. user fills out shipping address6. user provides username7. user provides password8. user clicks complete account <p>Extensions:</p> <p>a.* system stops responding</p> <ol style="list-style-type: none">1. user will restart application and can email bug to development team on website <p>1.a. user already has an account and wants to sign-in instead</p> <ol style="list-style-type: none">1. redirect user to sign in to pre-existing account <p>1.b user already has an account and wishes to edit account</p> <ol style="list-style-type: none">2. sign user in and allow user to edit existing account information <p>6.a user missed or improperly filled out account info</p> <ol style="list-style-type: none">1. inform user of what information needs provided or fixed and allow them to keep trying
---	---

<p>ID: Log-In</p> <p>Scope: Marketplace system</p> <p>Level: user goal</p> <p>Stakeholders and Interests:</p> <p>User</p> <ul style="list-style-type: none"> - person attempting to log-in to their account <p>Precondition:</p> <p>user has an account</p> <p>Postcondition:</p> <p>user has access to the account</p>	<p>Main success scenario:</p> <ol style="list-style-type: none"> 1. user wants to log-in 2. user inputs username 3. user inputs password 4. user clicks login button 5. user is logged in <p>Extensions:</p> <p>a.* system stops responding</p> <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website <p>4.a User's log-in information is invalid</p> <ol style="list-style-type: none"> 1. user is informed of error and asked to re input login information
--	---

<p>ID: Edit Account Information</p> <p>Scope: Marketplace system</p> <p>Level: user goal</p> <p>Stakeholders and Interests:</p> <p>User</p> <ul style="list-style-type: none"> - account holder who wishes to change account details <p>Precondition:</p> <p>user has an existing account</p> <p>Postcondition:</p> <p>user has updated/alterd their account details</p>	<p>Main success scenario:</p> <ol style="list-style-type: none"> 1. user wants to edit/update their account information 2. user clicks account panel 3. user clicks edit profile button 4. user edits desired fields of account details 5. user saves changes made to profile <p>Extensions:</p> <p>a.* system stops responding</p> <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website <p>5.a User makes invalid change to field</p> <ol style="list-style-type: none"> 1. user will be notified which field is incomplete or invalid and continue to prompt user until they address it
---	--

<p>ID: UC Search and Sort Items</p> <p>Scope: Sorting and finding products in the marketplace.</p> <p>Level: User Goal</p> <p>Stakeholders and Interests:</p> <p>Customer</p> <ul style="list-style-type: none"> - User with an account who is trying to search for desired item <p>Seller</p> <ul style="list-style-type: none"> - Person listing item on marketplace and potential buyers finding their item easier <p>Precondition: Two or more items are listed on the marketplace and the user is logged in to account.</p> <p>Postcondition: Items in marketplace are sorted and can be searched by user</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. Customers want to look for something to buy and search and sort for items based on attributes (price, rating, etc), or by name searching. 2. User enters item name they are looking for 3. User can sort by rating or money by clicking a sort by menu 4. User can scroll through items and clicked desired produced <p>Extensions:</p> <p>a.* system stops responding</p> <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website <p>2a. If searched by name item is not in the store</p> <ol style="list-style-type: none"> i. Expect an item not found error to pop up
--	--

<p>ID: UC user purchase history/currently selling</p> <p>Scope: User can track and log their purchase history and items currently listed</p> <p>Level: User Goal</p> <p>Stakeholder and Interest:</p> <p>Seller</p> <ul style="list-style-type: none"> - Seller can keep track of the items they are selling <p>Customer</p> <ul style="list-style-type: none"> - Customer who wants to check on previous items bought in the marketplace. <p>Precondition: User is logged in.</p> <p>Postcondition: Users can keep track of bought/selling products</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. User wants to explore their purchased/selling items 2. User clicks on purchased/selling tab 3. User can see items they bought and the status of items they are currently selling <p>Extensions:</p> <ol style="list-style-type: none"> a.* system stops responding <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website 2a. if user has no purchased/selling items <ol style="list-style-type: none"> i. Display a no items purchased/selling message
--	--

<p>ID: UC edit user market posting</p> <p>Scope: User can edit listing</p> <p>Level: User Goal</p> <p>Stakeholder and Interest:</p> <p>Seller</p> <ul style="list-style-type: none"> - Seller can edit and refine his posting <p>Precondition: User is logged in and has items listed for sale</p> <p>Postcondition: User can edit those pre-existing listings</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. User has a listing they are unsatisfied with 2. User can click on items purchased/selling tab 3. User can go into edit menu on item listing 4. Users can change properties of the listing such as (name, price, etc.). 5. Until the user is satisfied he/she can confirm changes 6. Items will be updated successfully on the marketplace for other users to view. <p>Extensions:</p> <ol style="list-style-type: none"> a.* system stops responding <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website 2a. If the user does not have any items for sale <ol style="list-style-type: none"> i. Throw error that they do not have any items for sale
---	---

<p>ID: UC Featured Item</p> <p>Scope: Marketplace system</p> <p>Level: User goal</p> <p>Stakeholders and Interests:</p> <p>-User: Interested in items</p> <p>Precondition: item(s) exist for the featured display</p> <p>Postcondition: Items are displayed for feature to the user</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. System calculates which item should be featured 2. System returns the item(s) in the list to be featured 3. System outputs featured deals for the user 4. User can view featured items and click on them <p>Extensions:</p> <p>a.* If the system stops responding</p> <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website <p>2.a If there are no items to be featured</p> <ol style="list-style-type: none"> 1. System will output a message in the featured items section stating that there are no items currently on sale/featured
---	--

<p>ID: UC View items</p> <p>Scope: Marketplace system</p> <p>Level: User goal</p> <p>Stakeholders and Interests:</p> <p>-User: potential buyer who wants to view items</p> <p>-Seller: individual who wants to examine the item they listed</p> <p>Precondition: There is an item on the webpage</p> <p>Postcondition: An item is displayed to the user</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. User wants to examine an item 2. System displays list of items that are currently in the marketplace 3. User can click on an item to examine its information and reviews 4. System returns the specific item with detailed information 5. System gives the opportunity for the user to click either to purchase or reviews <p>Extensions:</p> <p>a.* If the system stops responding</p> <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website <p>2.a If there are no items in the marketplace</p> <ol style="list-style-type: none"> 1. System will output a message stating that the market is currently empty
---	---

<p>ID: UC User write review</p> <p>Scope: Marketplace system</p> <p>Level: User goal</p> <p>Stakeholders and Interests:</p> <p>-User: interested in writing a review</p> <p>Precondition: User is logged in</p> <p>Postcondition: a new review is created and displayed under the product</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. User examines an item that they want to provide feedback on 2. System returns past reviews 3. User inputs text comment that they want 4. System will update list of reviews with new review <p>Extensions:</p> <p>a.* If the system stops responding</p> <ol style="list-style-type: none"> 1. user will restart application and can email bug to development team on website <p>2.a If there are no past reviews</p> <ol style="list-style-type: none"> 1. System will inform user that review list is empty <p>3.a If the input is empty</p> <ol style="list-style-type: none"> 1. System will tell user that there must be text 2. System will give another opportunity for user to type their review
---	--

<p>ID: Delete a market posting</p> <p>Scope: Marketplace system</p> <p>Level: User goal</p> <p>Stakeholders and Interests:</p> <p>-User: Take down their own product for any reason</p> <p>Precondition: A product posting exists from the user</p> <p>Postcondition: Product is deleted from the marketplace</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. User wants to delete an item they have have for sell 2. User goes into their products for sell section 3. User clicks on the delete icon 4. System deletes selected product 5. User repeats steps 2-4 for each product posting they want to delete <p>Extensions:</p> <p>a.* system stops responding</p> <ol style="list-style-type: none"> 1. User will restart application and can email bug to development team on website
---	---

<p>ID: Create a market post</p> <p>Scope: Marketplace system</p> <p>Level: User goal</p> <p>Stakeholders and Interests:</p> <p>-User: Create a product for sale</p> <p>Precondition: Have an account and a product for sell</p> <p>Postcondition: Product is now on the marketplace</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. User has an item to put on sale 2. User selects on create a market post 3. System displays all necessary information to put an item for sale 4. User inputs all needed details on the product 5. User confirms information 6. System places the product on the marketplace <p>Extensions:</p> <p>a.* system stops responding</p> <ol style="list-style-type: none"> 1. User will restart application and can email bug to development team on website
---	--

<p>ID: Purchase Item</p> <p>Scope: Marketplace system</p> <p>Level: User goal</p> <p>Stakeholders and Interests:</p> <p>-Customer: Receives wanted item(s)</p> <p>-Seller: Earns profit from purchase</p> <p>Precondition: Customer is logged in and has proper payment.</p> <p>Postcondition: Seller gains profit and customer receives product</p>	<p>Main Success Scenario:</p> <ol style="list-style-type: none"> 1. User finds a desired item on the marketplace 2. User clicks buy on item 3. System displays the price for each item and the total cost before and after tax. 4. User confirms payment information 5. User clicks on complete purchase 6. System authorizes purchase 7. User obtains all items purchased <p>Extensions:</p> <p>a.* system stops responding</p> <ol style="list-style-type: none"> 1. User will restart application and can email bug to development team on website <p>4.a If customer has a coupon</p> <ol style="list-style-type: none"> 1. User inputs coupon code 2. System discounts total <p>5.a If customer wants to change payment details</p> <ol style="list-style-type: none"> 1. Customer selects change payment details 2. Customer edits details 3. System saves edited details <p>6.a If payment is declined</p> <ol style="list-style-type: none"> 1. System asks user to reenter billing information 2. User enters new information 3. Repeat steps 1-2 until the card is accepted
--	---

Bear Market Project Plan

Feb 28, 2021

<https://richard-hutch.github.io/Marketplace-System/>

Project manager

Richard Hutcheson

Project dates

Feb 22, 2021 - Mar 3, 2021

Completion

0%

Tasks

9

Resources

4

Tasks

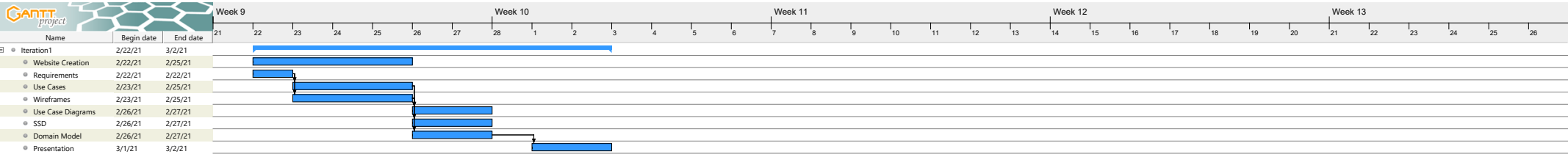
2

Name	Begin date	End date
Iteration1	2/22/21	3/2/21
Website Creation	2/22/21	2/25/21
Requirements	2/22/21	2/22/21
Use Cases	2/23/21	2/25/21
Wireframes	2/23/21	2/25/21
Use Case Diagrams	2/26/21	2/27/21
SSD	2/26/21	2/27/21
Domain Model	2/26/21	2/27/21
Presentation	3/1/21	3/2/21

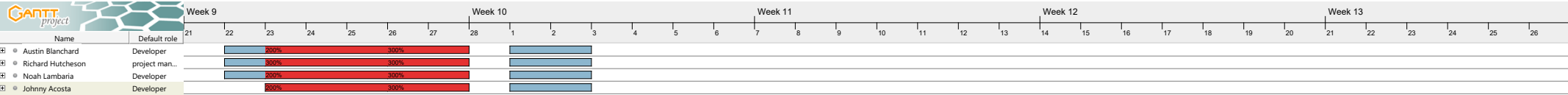
Resources

Name	Default role
Austin Blanchard	Developer
Richard Hutcheson	project manager
Noah Lambaria	Developer
Johnny Acosta	Developer

Gantt Chart



Resources Chart



	System will return item results based on the user's input	Customer stores account information	customer finding item	User control over items sold on our platform	General accounting of items	User providing feedback on products
User searches/sorts for items	X		X			
User purchase history	X				X	X
Create Account		X				
Log-In	X	X	X	X	X	X
Edit Account		X				
Edit user selling product				X		
Featured Items	X		X	X		
View Items	X		X			X
Write review				X		X
Delete market posting				X		
Create market posting				X		
Purchase item	X				X	

Operation Contracts

Operation: viewPurchaseHistory()

Cross References: Use Cases: Purchase History

Pre-Conditions: List of items purchased from account exists.

Post-Condition(s): -Displays frame with table and buttons. -Table is populated with items from list -Buttons are activated to lead to other menus

Operation: removeItem()

Cross References: Use Cases: Selling Items

Pre-Conditions: List of items currently on sale from account exists. Item being removed is valid.

Post-Condition(s): -Item is removed from the list. -Account is updated -Table is updated.

Operation: purchase()

Cross References: Use Cases: Purchase Item

Pre-Conditions: Item is valid. Item is in stock.

Post-Condition(s): -Item is decremented from stock. -Item is added to purchase history.

Operation: createTable()

Cross References: Use Cases: Use Store

Pre-Conditions: a mainScreen object exists

Post-Conditions: -a table object is created, a table model is created and assigned to table, data populates table

Operation: viewReviews()

Cross References: Use Cases: Browse Reviews

Pre-Conditions: Product object exists and user clicks on review button

Post-Conditions: a dialog panel displays with the product name and a list of all of the reviews tied to that account, the ratings, and the username of individuals who left reviews

Operation: findFeaturedItems()

Cross References: Use Cases: Featured Items

Pre-Conditions: products have already been read from file and stored, user is on main screen

Post-Conditions: three Products assigned to featured items panel and their information displayed

Operation: createAccount(File accountList)

Cross References: Use Case(s): Create Account

Pre-Conditions: -all text field information is filled in (not empty) and meets the specific criteria
-the username hasn't already been taken

Post-Conditions: -an account is created

-the accounts username and password are written to the master login file/database

-another file is created for that account, storing all the information from the text fields

Operation: editAccount(Account, File)

Cross References: Use Case(s): Edit Account

Pre-Conditions: -an account exists along with the associated files with that account

-the information written in the text field matches the criteria needed for an account to be edited

Post-Conditions: the account and account file is modified with the changes the user makes

Operation: login()

Cross References: Use Case(s): Log-In

Pre-Conditions: the username and password are valid and an account has been created

Post-Conditions: -the main screen frame is now displayed

-the account is logged in

Operation: createNewProduct(File)

Cross References: Use Case(s): Create Market Posting

Pre-Conditions:

- The Account exists within the Account file.

Post-Condition:

- The function gets the data from the create account window.
- The function writes the new listing to the file successfully.
- The function displays the new product in the currently selling tab.
- The function displays the new product in the main market place window.

Operation: DeleteItemFromAccountFile(File)

Cross References: Use Case(s): Delete Market Posting

Pre-Conditions: The file exists and the user is found within the file with all of their current listings.

Post-Conditions:

- The item the user wants to delete will be found and removed from their listings.
- The item the user wants to delete will be removed from the marketplace

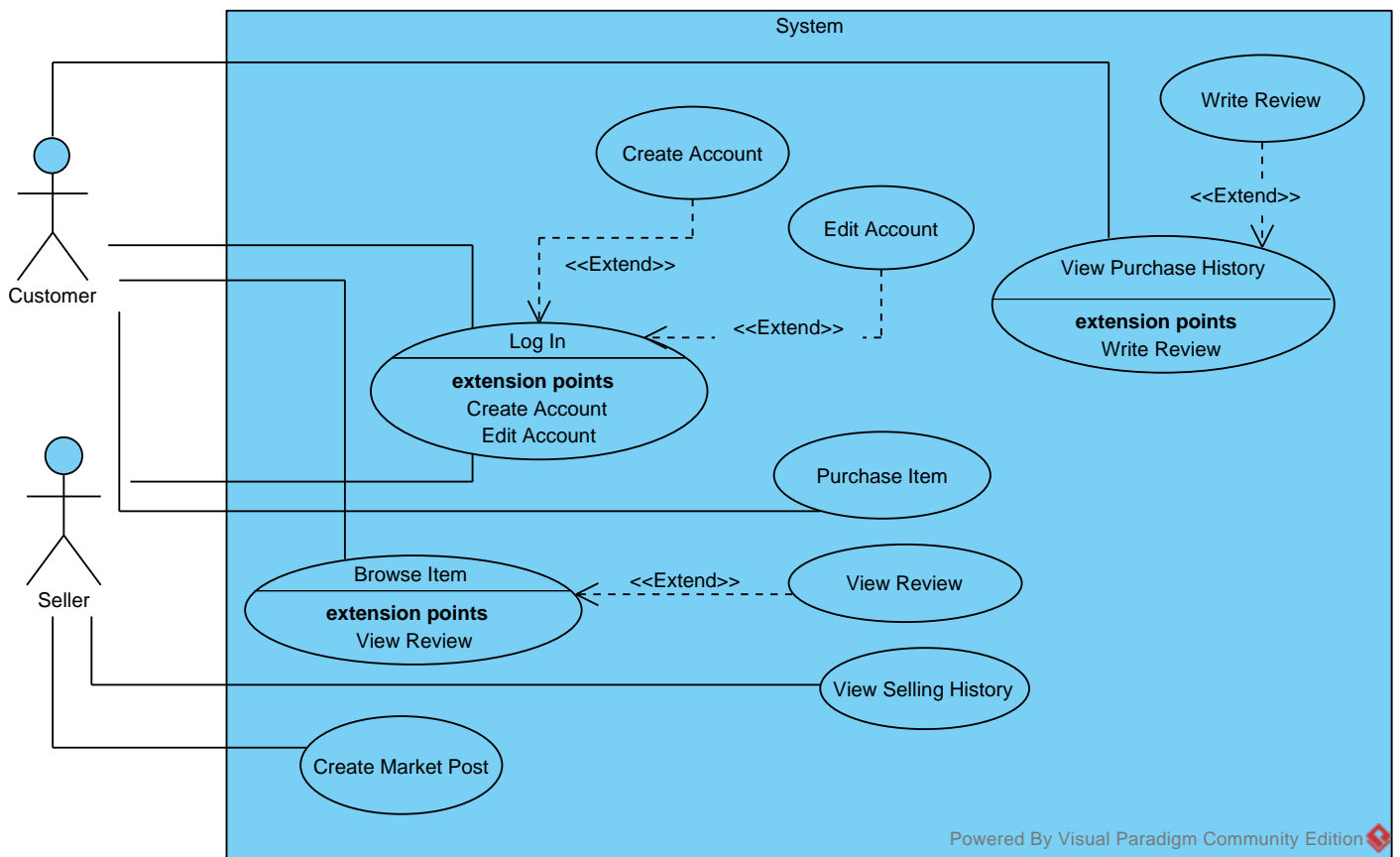
Operation: readCurrentProduct()

Cross References: Use Case(s): Edit Market Posting

Pre-Conditions: The item is a valid listing connected with the user's account.

Post-Condition:

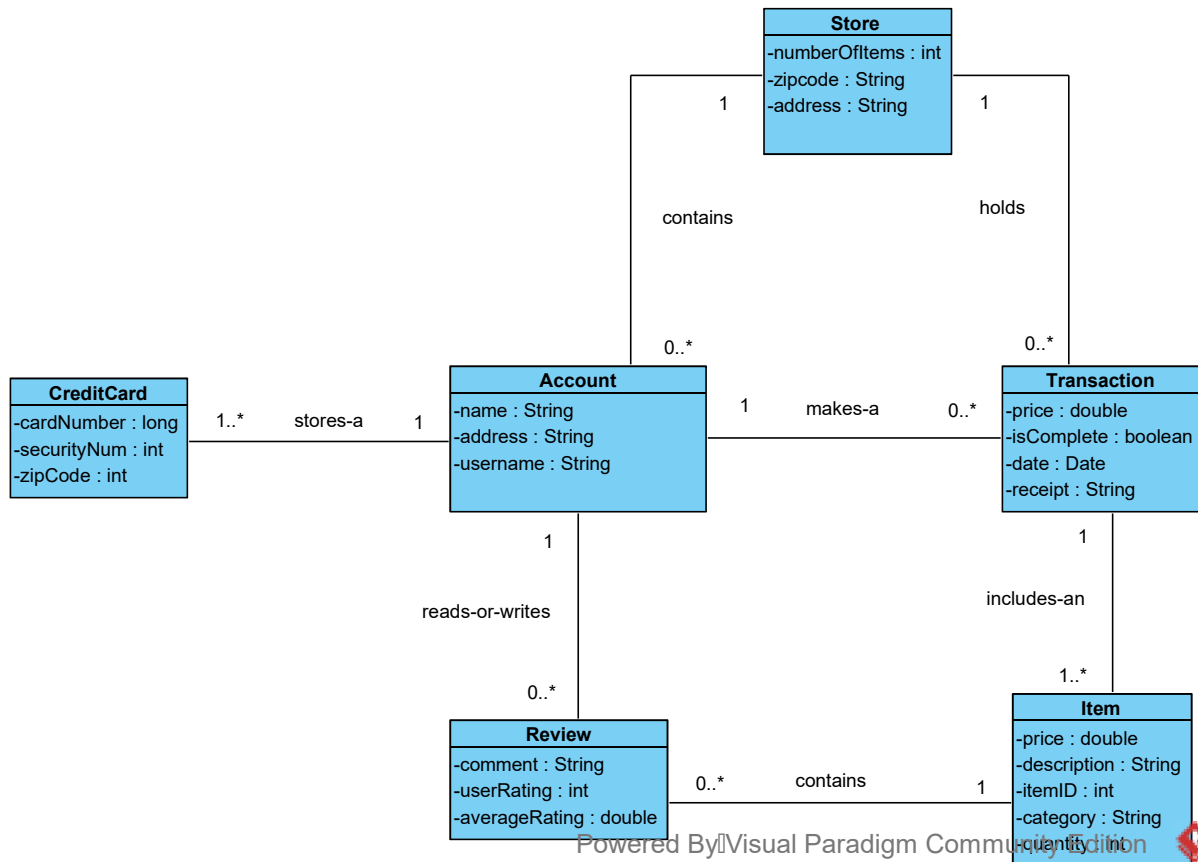
- The new information edited by the user about the product is displayed in the main market and in the currently selling tab.



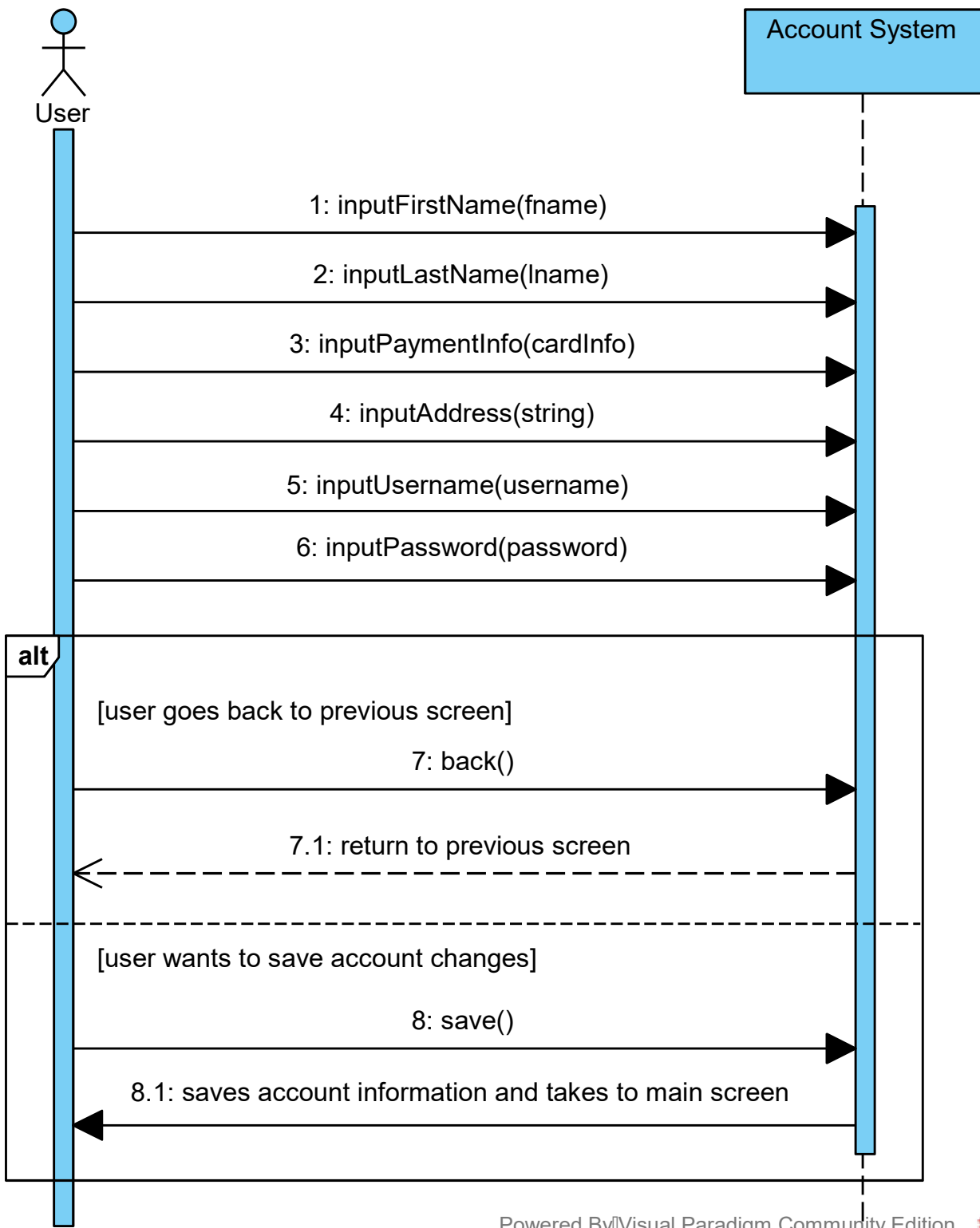
System

+viewAccount()
+viewPSHistory()
+createPosting()
+searchForItem(item)
+sortItems(category)
+purchaseItem(itemID)
+viewItemReview(review)
+inputName(first, last)
+inputAddress(address)
+inputUsername(username)
+inputPassword(password)
+createAccount()
+writeReview(review)
+login()
+inputPrice(amount)
+inputCategory(category)
+purchaseItem(item)
+changePayment(payment)
+getSorted(comparator)

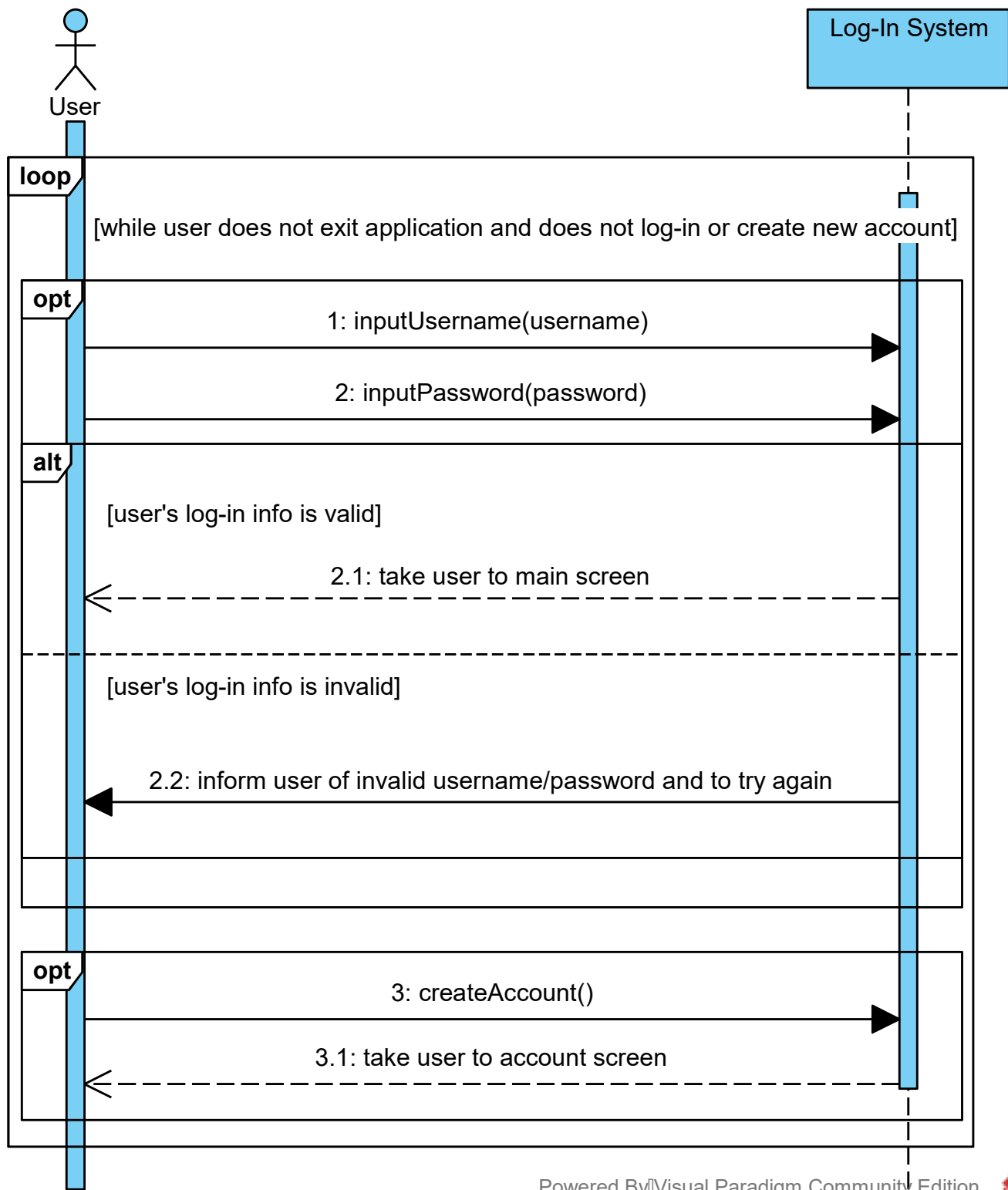
Domain model - Software Engineering Group Project

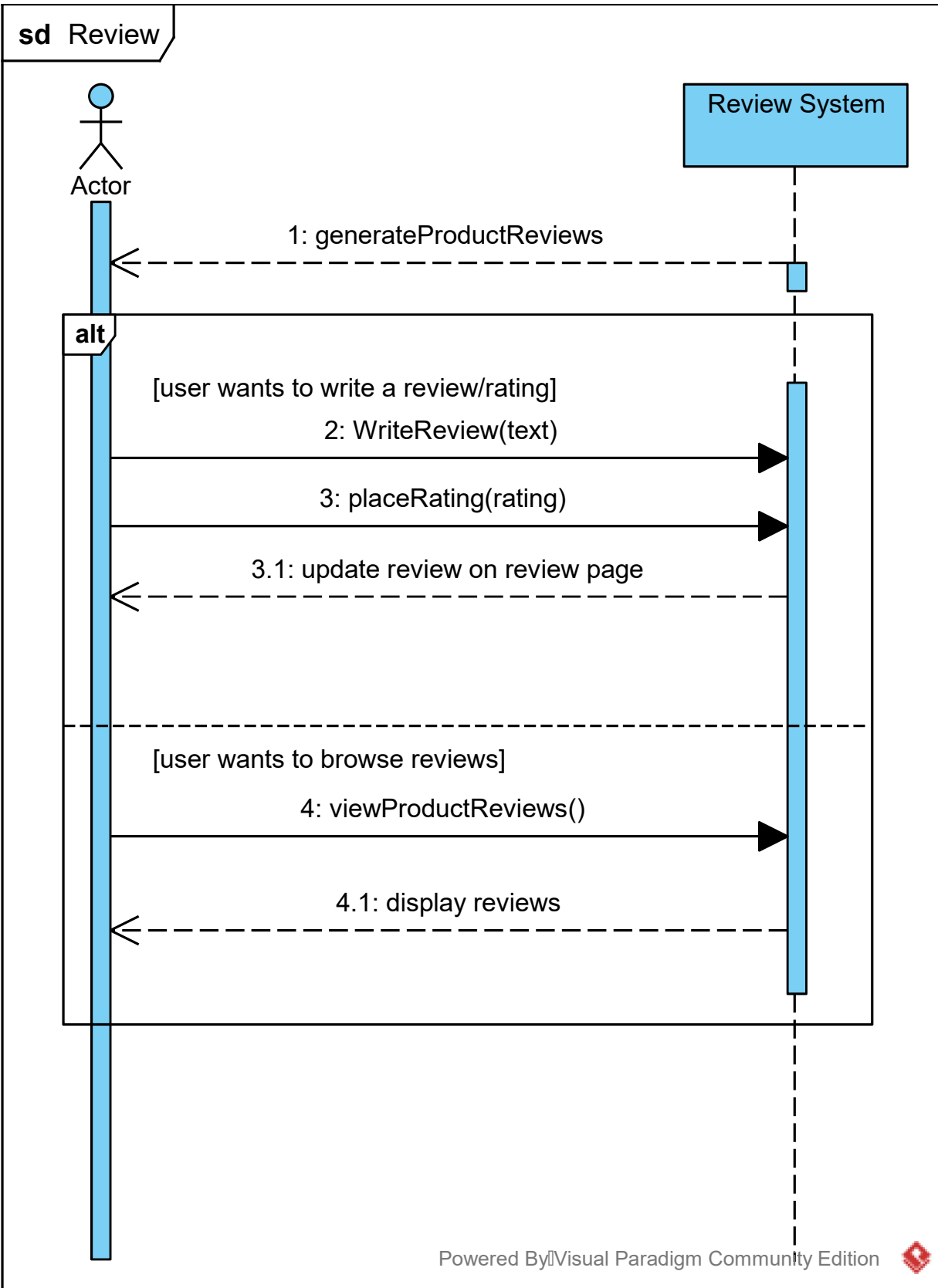


sd create/EditAccount

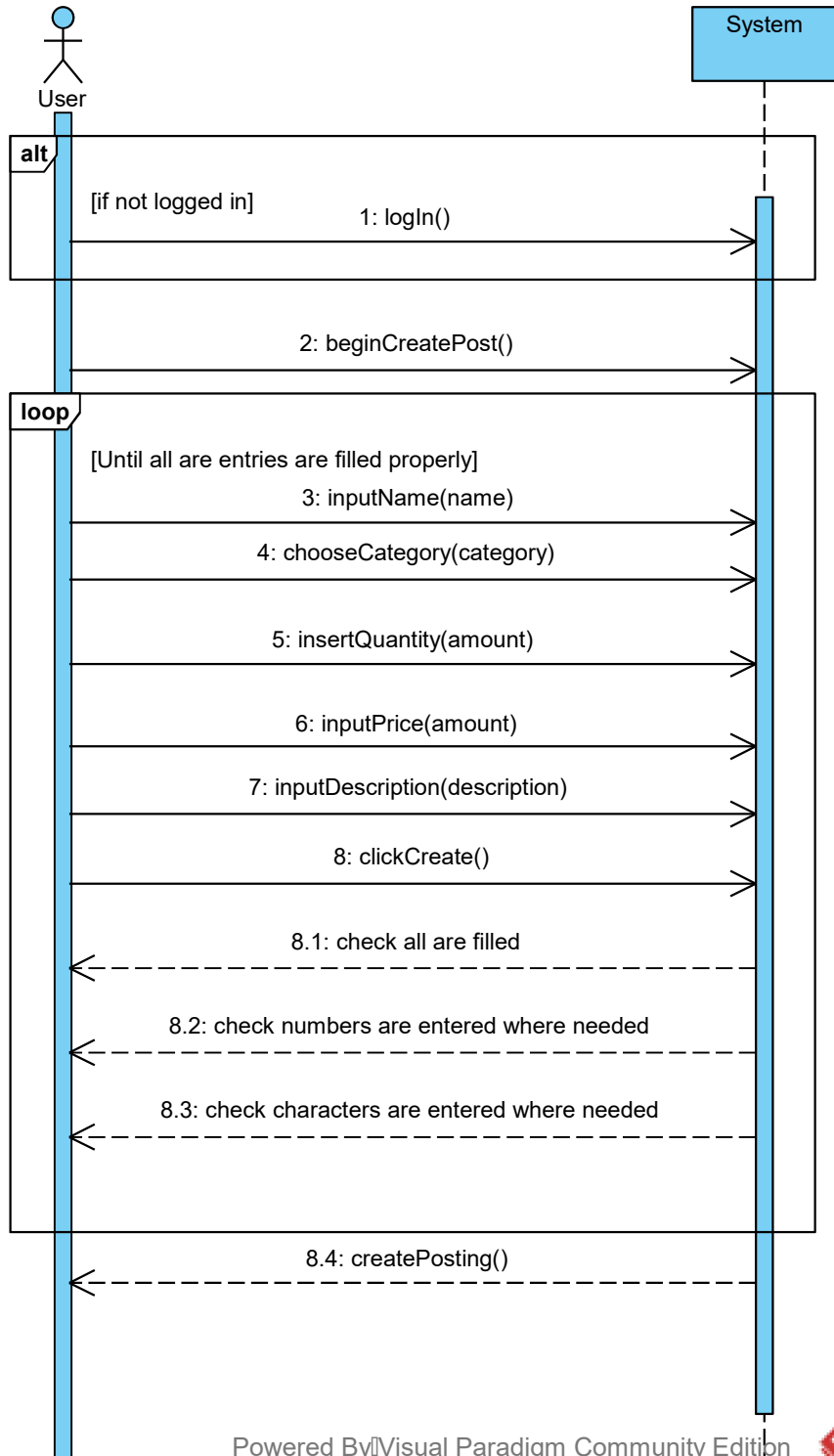


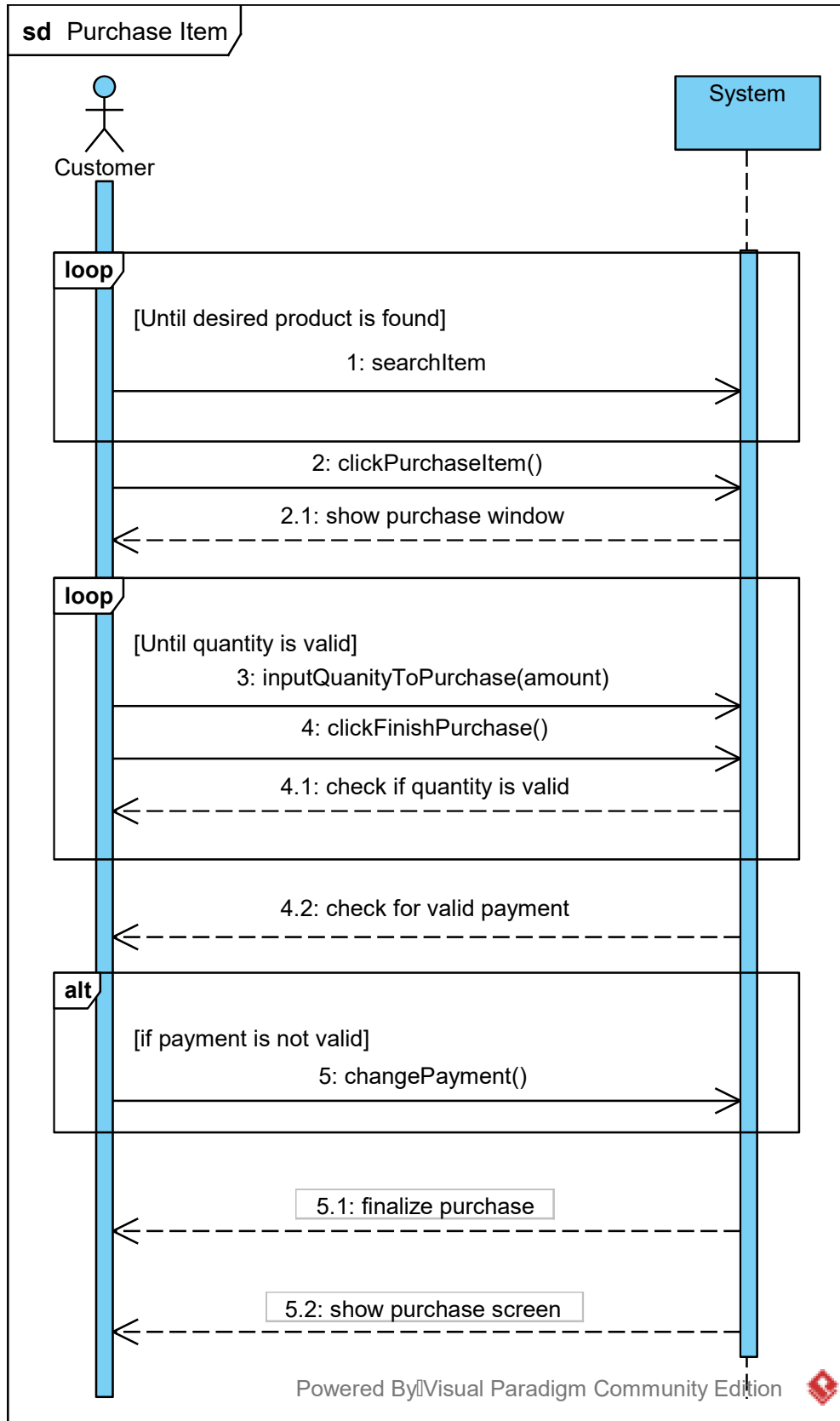
sd Log-In Screen





sd Create Market Post





sd SearchItems

User

Search

1: takeUserInput()

2: searchItem()

alt

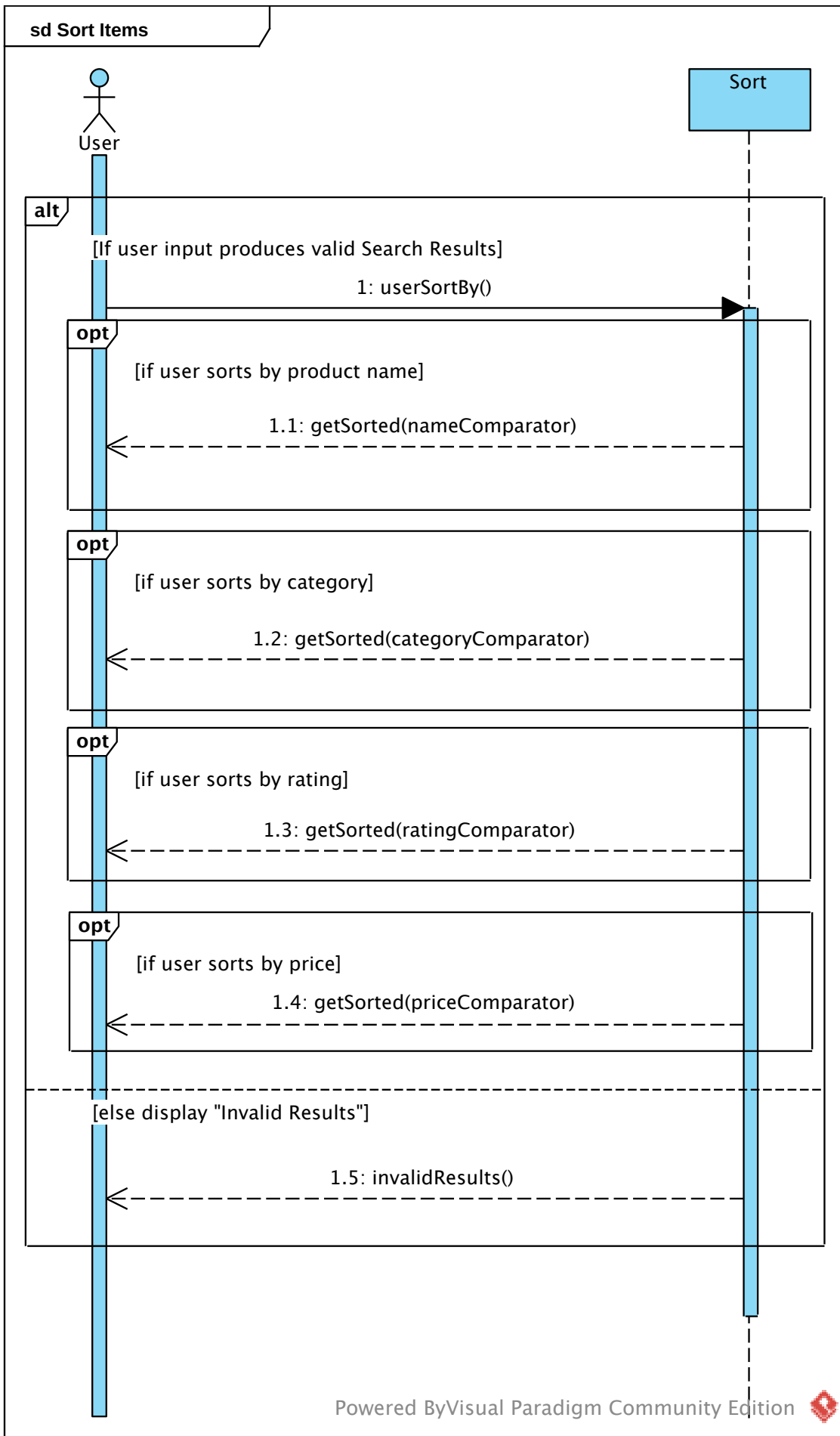
[if search queries are found]

2.1: getSearchItemsList()

[search item not found]

2.2: displayNotItemsFound()





Create Market Post

Name*

Create

Choose Category

Quantity

Description

Your item has been purchased!

Quantity: _____ Total Price:\$ _____

Close

First Name

Last Name

Shipping Address

State

ZIP

Payment Information

Card Number

CSV

ZIP

Username

Password

Back

Save

Account	Purchase History and Selling	Create Posting		Exit
---------	------------------------------	----------------	--	------

Today's Featured Items:

<Item> <Item> <Item> <Item> <Item> <Item>

Type to search...

Sort By:	Name	Category	Description	Quantity	Rating	Price	
----------	------	----------	-------------	----------	--------	-------	--

<Item>

ReviewsPurchase

<Item>

ReviewsPurchase

<Item>

ReviewsPurchase

<Item>

ReviewsPurchase

Poggers Market

Username:

Password:

Login

create account

Exit

Product Details and Information here

NAME

REVIEW TEXT

NAME

REVIEW TEXT

BACK

Purchase / Selling History

currently selling

Purchased

item 1

edit

item 2



edit

item 3

edit

item 4

edit

	Write review below	
		Submit