Team Members: Austin Blanchard, Richard Hutcheson, Noah Lambaria, Joshua McKone
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/Marketplace-System/issues
Website: https://richard-hutch.github.io/BearMarket/
Git Link: https://github.com/Richard-Hutch/Marketplace-System

Team Member	Use Case Responsibility
Austin Blanchard	create, edit, delete market posting, generate review
Richard Hutcheson	product table, featured items, browse reviews
Noah Lambaria	create account, login, edit account,
Josh McKone	purchase history, Items Being Sold, purchase item

Time Tracker

Team Member	Hours Worked
Richard Hutcheson	50
Austin Blanchard	42
Noah Lambaria	48
Josh McKone	42

Bear Market Project Plan

Mar 31, 2021

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

Project manager	Richard Hutcheson
Project dates	Feb 22, 2021 - Mar 31, 2021
Completion	0%
Tasks	16
Resources	4

Tasks

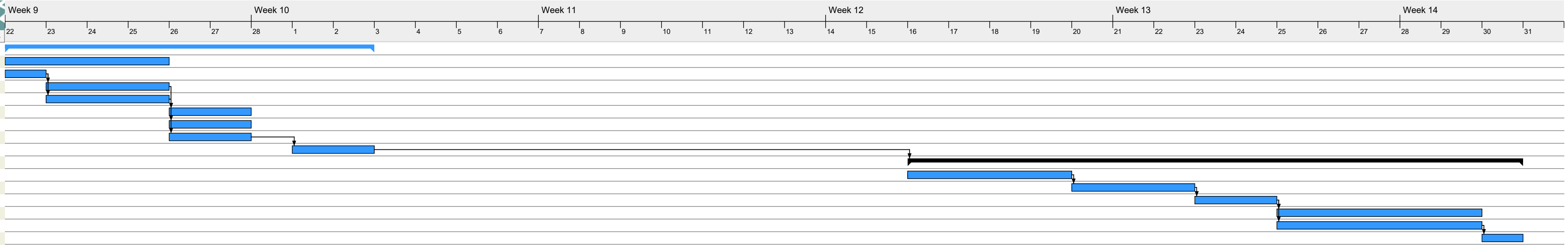
Name	Begin date	End date
Iteration1	2/22/21	3/2/21
Website Creation	2/22/21	2/25/21
Requirements	2/22/21	2/22/21
Use Cases	2/23/21	2/25/21
Wireframes	2/23/21	2/25/21
Use Case Diagrams	2/26/21	2/27/21
SSD	2/26/21	2/27/21
Domain Model	2/26/21	2/27/21
Presentation	3/1/21	3/2/21
Iteration2	3/16/21	3/30/21
ProductList	3/16/21	3/19/21
Sequence Diagrams	3/20/21	3/22/21
Package Diagram and Design Model	3/23/21	3/24/21
Programming prototype	3/25/21	3/29/21
GRASP patterns	3/25/21	3/29/21
Presentation	3/30/21	3/30/21

Resources

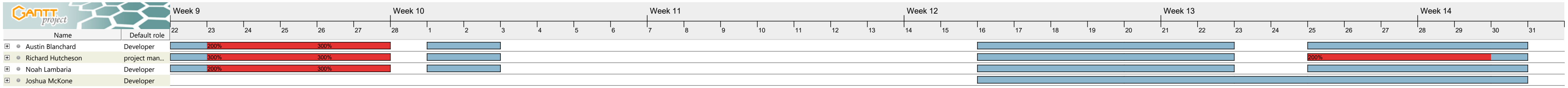
Name	Default role
Austin Blanchard	Developer
Richard Hutcheson	project manager
Noah Lambaria	Developer
Joshua McKone	Developer

Gantt Chart

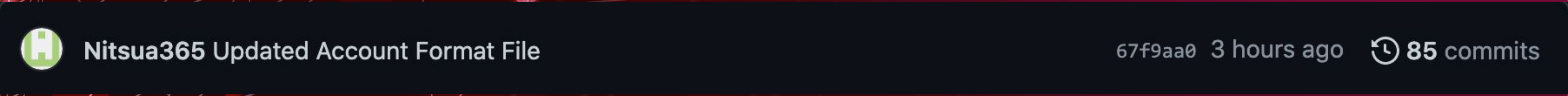
Name	Begin date	End date
⊟ Iteration1	2/22/21	3/2/21
Website Creation	2/22/21	2/25/21
Requirements	2/22/21	2/22/21
Use Cases	2/23/21	2/25/21
Wireframes	2/23/21	2/25/21
Use Case Diagrams	2/26/21	2/27/21
SSD	2/26/21	2/27/21
Domain Model	2/26/21	2/27/21
Presentation	3/1/21	3/2/21
⊟ Iteration2	3/16/21	3/30/21
ProductList	3/16/21	3/19/21
Sequence Diagrams	3/20/21	3/22/21
Package Diagram ...	3/23/21	3/24/21
Programming prot...	3/25/21	3/29/21
GRASP patterns	3/25/21	3/29/21
Presentation	3/30/21	3/30/21



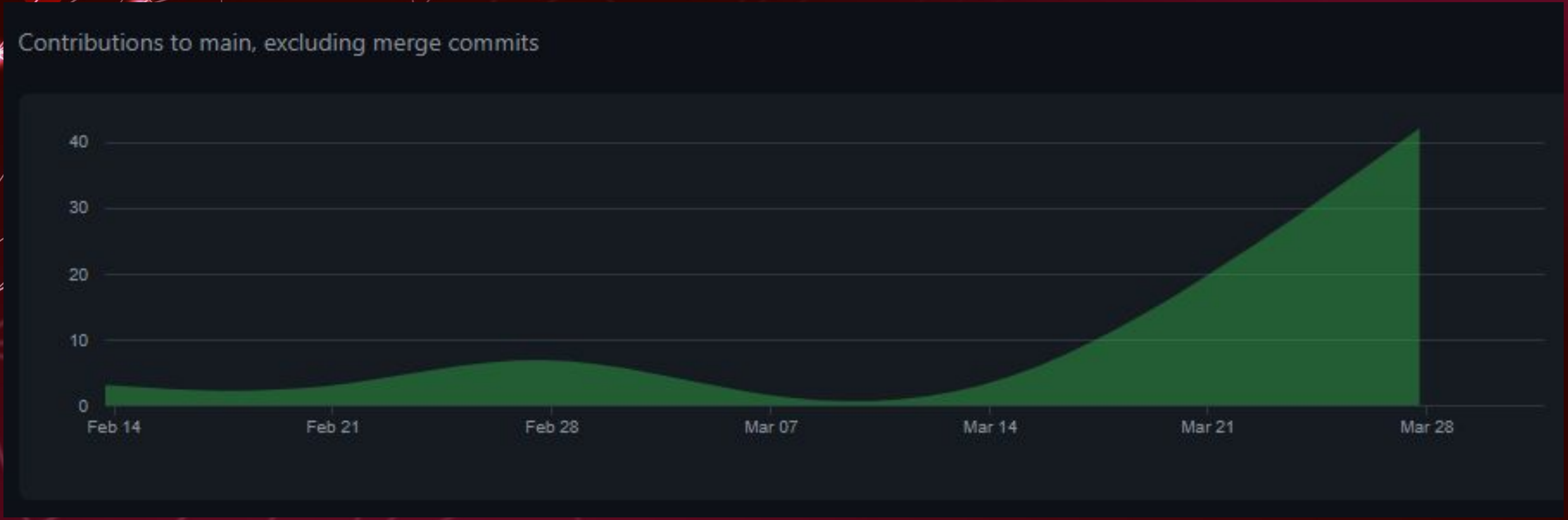
Resources Chart



COMMITS

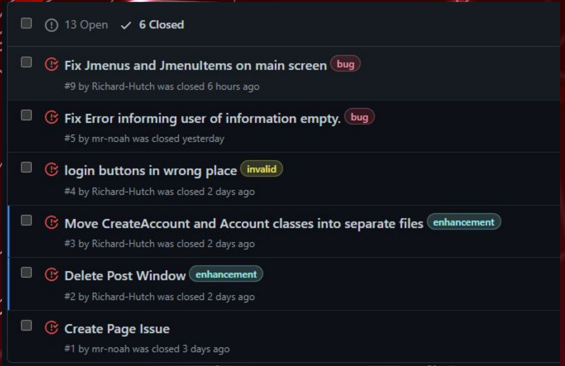


Nitsua365 Updated Account Format File 67f9aa0 3 hours ago 85 commits



Contributions to main, excluding merge commits

ISSUE TRACKING



□ 13 Open ✓ 6 Closed

□ Fix Jmenus and JmenuItems on main screen bug
#9 by Richard-Hutch was closed 6 hours ago

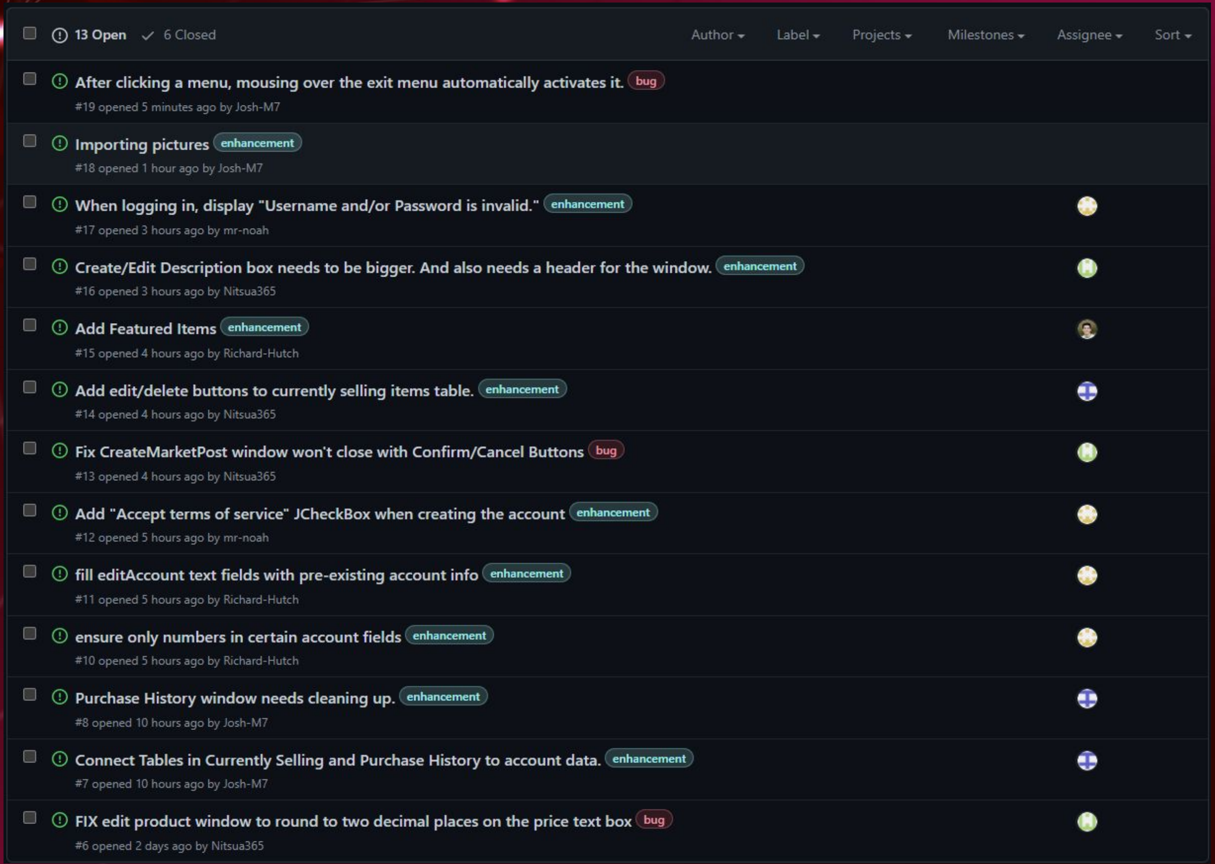
□ Fix Error informing user of information empty. bug
#5 by mr-noah was closed yesterday

□ login buttons in wrong place invalid
#4 by Richard-Hutch was closed 2 days ago

□ Move CreateAccount and Account classes into separate files enhancement
#3 by Richard-Hutch was closed 2 days ago

□ Delete Post Window enhancement
#2 by Richard-Hutch was closed 2 days ago

□ Create Page Issue
#1 by mr-noah was closed 3 days ago



□ 13 Open ✓ 6 Closed

□ After clicking a menu, mousing over the exit menu automatically activates it. bug
#19 opened 5 minutes ago by Josh-M7

□ Importing pictures enhancement
#18 opened 1 hour ago by Josh-M7

□ When logging in, display "Username and/or Password is invalid." enhancement
#17 opened 3 hours ago by mr-noah

□ Create/Edit Description box needs to be bigger. And also needs a header for the window. enhancement
#16 opened 3 hours ago by Nitsua365

□ Add Featured Items enhancement
#15 opened 4 hours ago by Richard-Hutch

□ Add edit/delete buttons to currently selling items table. enhancement
#14 opened 4 hours ago by Nitsua365

□ Fix CreateMarketPost window won't close with Confirm/Cancel Buttons bug
#13 opened 4 hours ago by Nitsua365

□ Add "Accept terms of service" JCheckBox when creating the account enhancement
#12 opened 5 hours ago by mr-noah

□ fill editAccount text fields with pre-existing account info enhancement
#11 opened 5 hours ago by Richard-Hutch

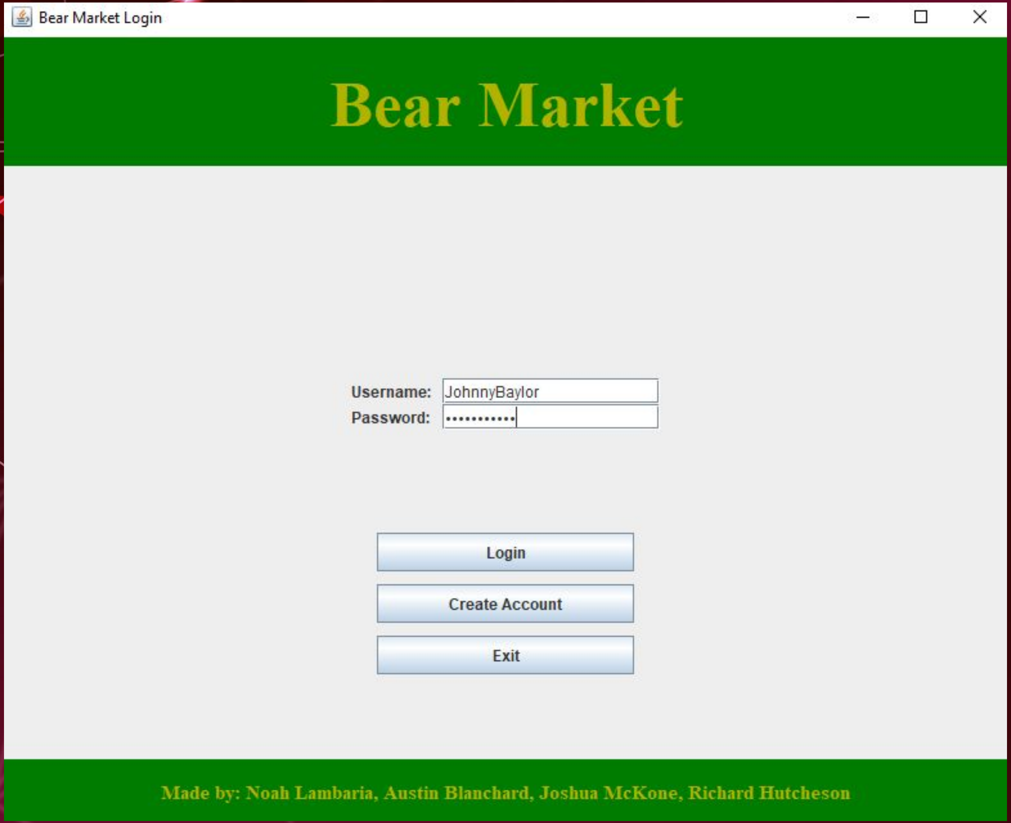
□ ensure only numbers in certain account fields enhancement
#10 opened 5 hours ago by Richard-Hutch

□ Purchase History window needs cleaning up. enhancement
#8 opened 10 hours ago by Josh-M7

□ Connect Tables in Currently Selling and Purchase History to account data. enhancement
#7 opened 10 hours ago by Josh-M7

□ FIX edit product window to round to two decimal places on the price text box bug
#6 opened 2 days ago by Nitsua365

LOGIN SCREEN



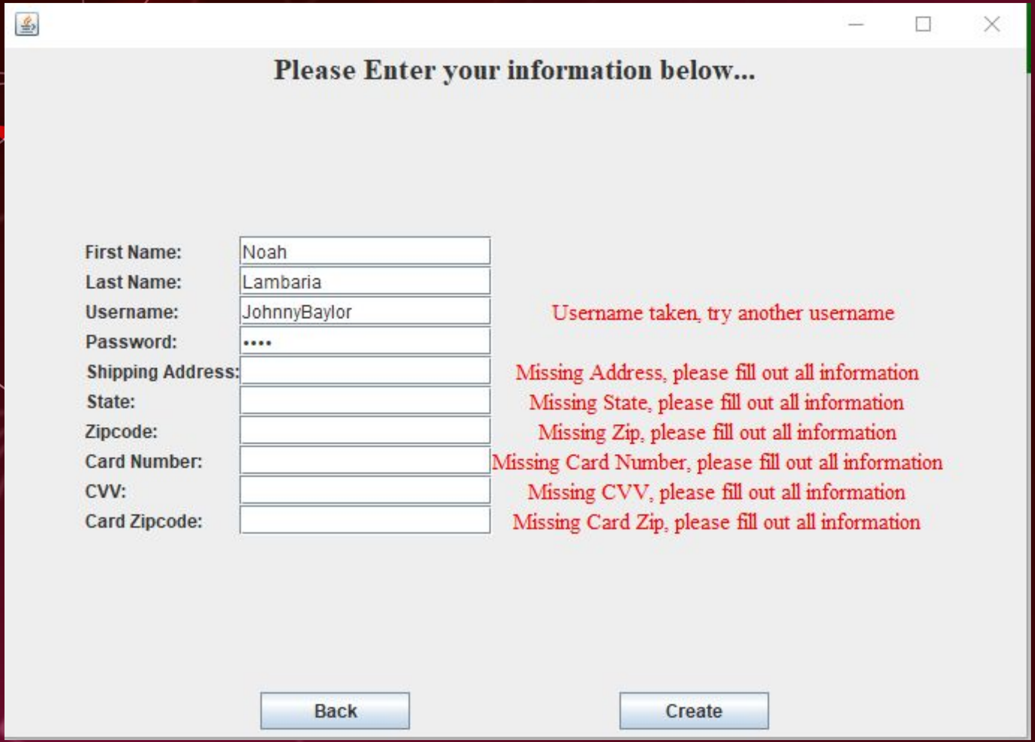
Bear Market

Username: JohnnyBaylor

Password: •••••••••••

Made by: Noah Lambaria, Austin Blanchard, Joshua McKone, Richard Hutcheson

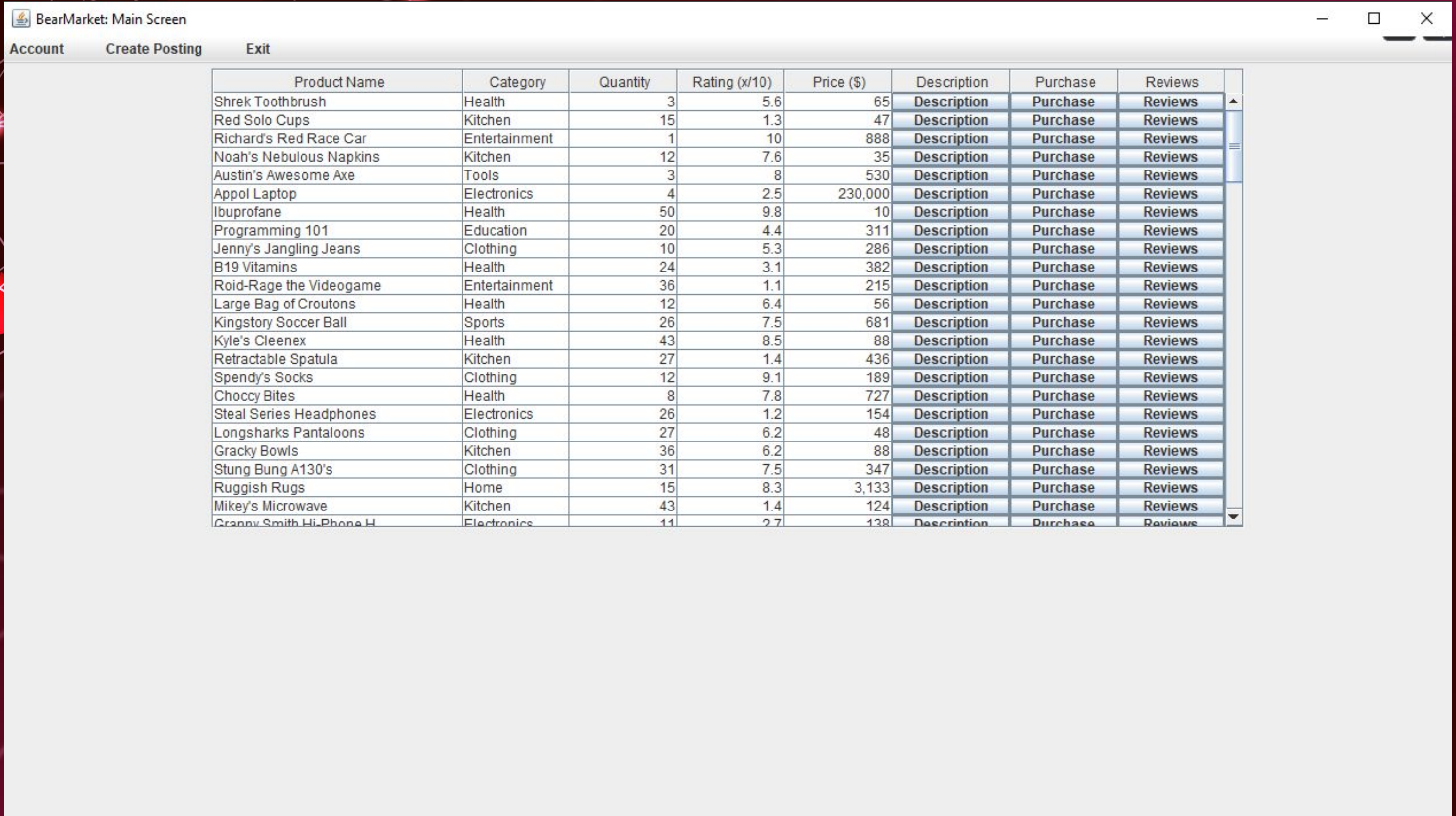
CREATE AN ACCOUNT



Please Enter your information below...

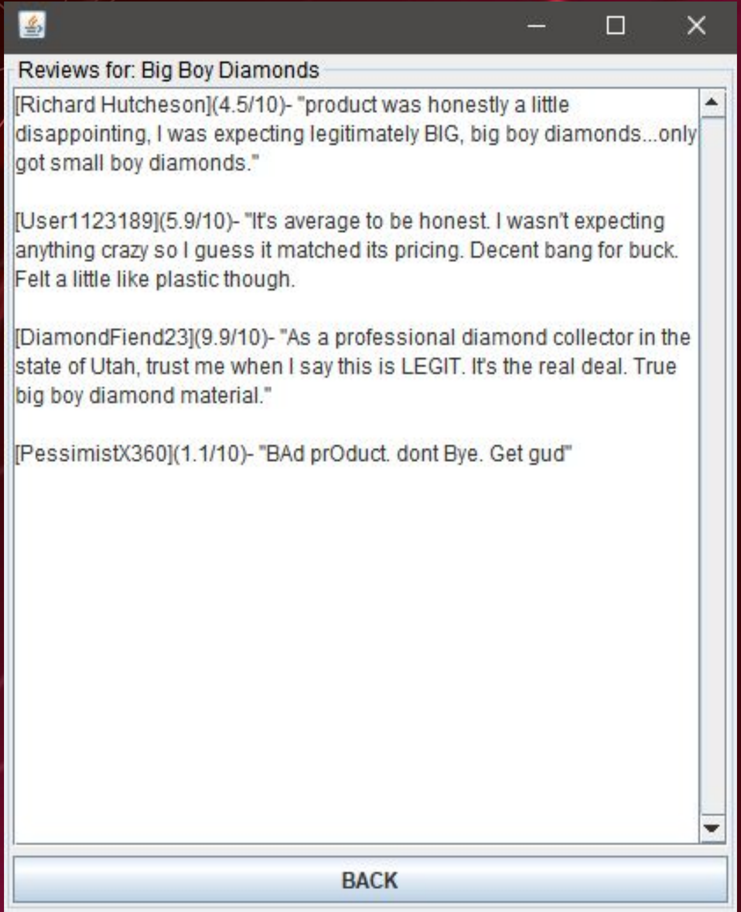
First Name:	Noah	
Last Name:	Lambaria	
Username:	JohnnyBaylor	Username taken, try another username
Password:	••••	
Shipping Address:		Missing Address, please fill out all information
State:		Missing State, please fill out all information
Zipcode:		Missing Zip, please fill out all information
Card Number:		Missing Card Number, please fill out all information
CVV:		Missing CVV, please fill out all information
Card Zipcode:		Missing Card Zip, please fill out all information

MAIN WINDOW



Product Name	Category	Quantity	Rating (x/10)	Price ($)	Description	Purchase	Reviews
Shrek Toothbrush	Health	3	5.6	65	Description	Purchase	Reviews
Red Solo Cups	Kitchen	15	1.3	47	Description	Purchase	Reviews
Richard's Red Race Car	Entertainment	1	10	888	Description	Purchase	Reviews
Noah's Nebulous Napkins	Kitchen	12	7.6	35	Description	Purchase	Reviews
Austin's Awesome Axe	Tools	3	8	530	Description	Purchase	Reviews
Appol Laptop	Electronics	4	2.5	230,000	Description	Purchase	Reviews
Ibuprofane	Health	50	9.8	10	Description	Purchase	Reviews
Programming 101	Education	20	4.4	311	Description	Purchase	Reviews
Jenny's Jangling Jeans	Clothing	10	5.3	286	Description	Purchase	Reviews
B19 Vitamins	Health	24	3.1	382	Description	Purchase	Reviews
Roid-Rage the Videogame	Entertainment	36	1.1	215	Description	Purchase	Reviews
Large Bag of Croutons	Health	12	6.4	56	Description	Purchase	Reviews
Kingstory Soccer Ball	Sports	26	7.5	681	Description	Purchase	Reviews
Kyle's Cleenex	Health	43	8.5	88	Description	Purchase	Reviews
Retractable Spatula	Kitchen	27	1.4	436	Description	Purchase	Reviews
Spendy's Socks	Clothing	12	9.1	189	Description	Purchase	Reviews
Choccy Bites	Health	8	7.8	727	Description	Purchase	Reviews
Steal Series Headphones	Electronics	26	1.2	154	Description	Purchase	Reviews
Longsharks Pantaloons	Clothing	27	6.2	48	Description	Purchase	Reviews
Gracky Bowls	Kitchen	36	6.2	88	Description	Purchase	Reviews
Stung Bung A130's	Clothing	31	7.5	347	Description	Purchase	Reviews
Ruggish Rugs	Home	15	8.3	3,133	Description	Purchase	Reviews
Mikey's Microwave	Kitchen	43	1.4	124	Description	Purchase	Reviews
Granny Smith Hi-Phone H	Electronics	11	2.7	138	Description	Purchase	Reviews

PRODUCT REVIEW



Reviews for: Big Boy Diamonds

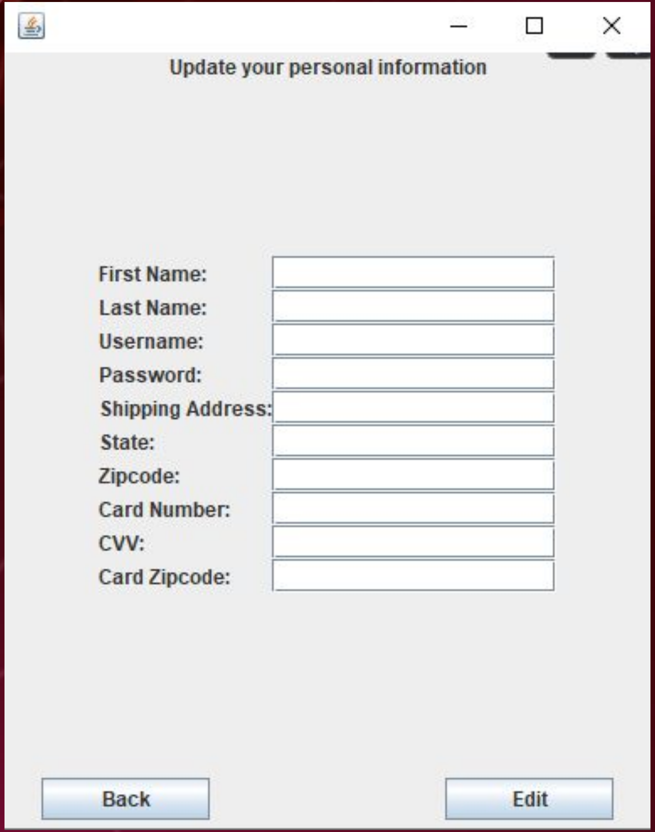
[Richard Hutcheson](4.5/10)- "product was honestly a little disappointing, I was expecting legitimately BIG, big boy diamonds...only got small boy diamonds."

[User1123189](5.9/10)- "It's average to be honest. I wasn't expecting anything crazy so I guess it matched its pricing. Decent bang for buck. Felt a little like plastic though.

[DiamondFiend23](9.9/10)- "As a professional diamond collector in the state of Utah, trust me when I say this is LEGIT. It's the real deal. True big boy diamond material."

[PessimistX360](1.1/10)- "BAd prOduct. dont Bye. Get gud"

EDIT ACCOUNT

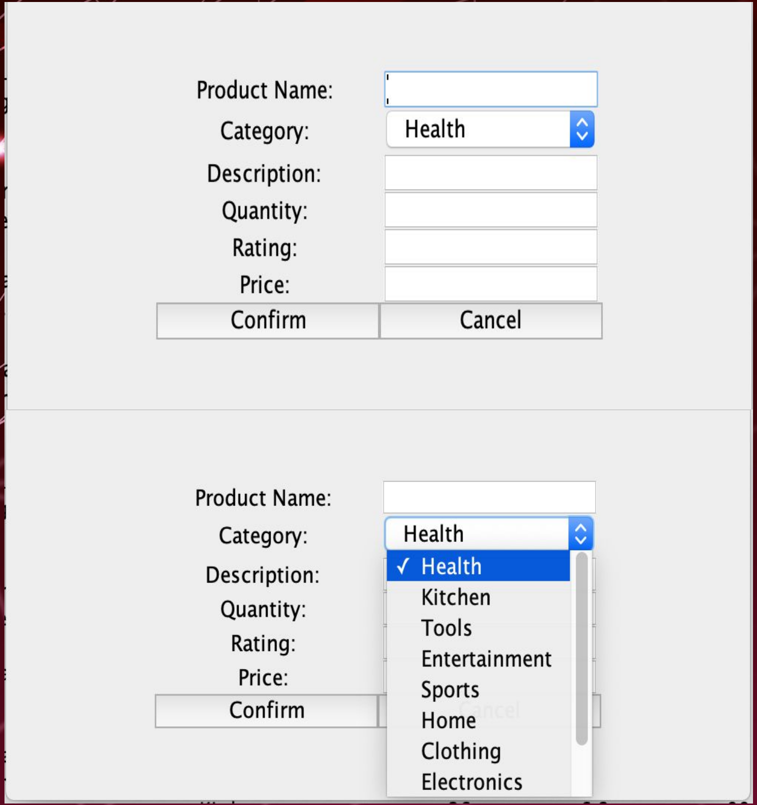


Update your personal information

First Name:	
Last Name:	
Username:	
Password:	
Shipping Address:	
State:	
Zipcode:	
Card Number:	
CVV:	
Card Zipcode:	

CREATE/EDIT/DELETE

Create Window



Product Name:

Category: Health

Description:

Quantity:

Rating:

Price:

Product Name:

Category: Health

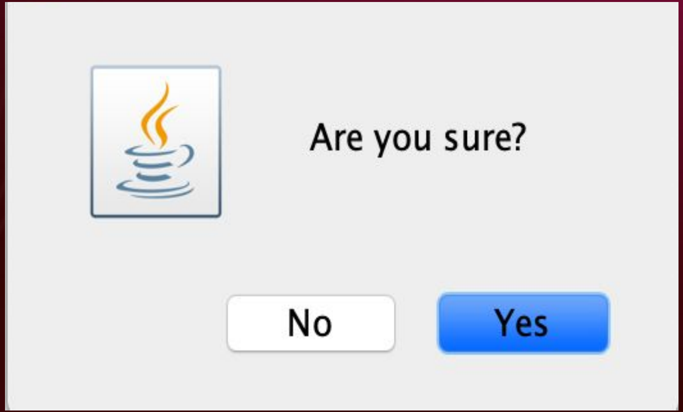
Description:

Quantity:

Rating:

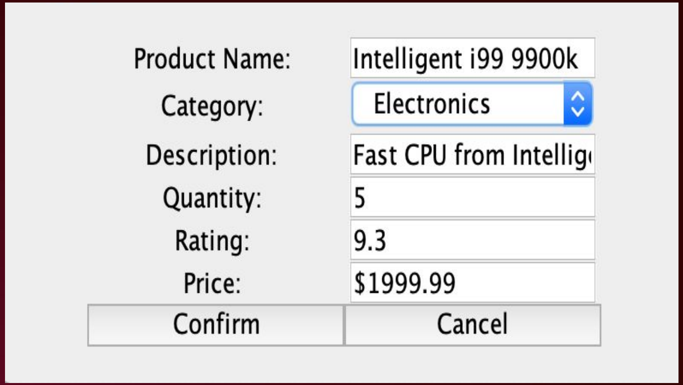
Price:

Delete



Are you sure?

Edit Window



Product Name: Intelligent i99 9900k

Category: Electronics

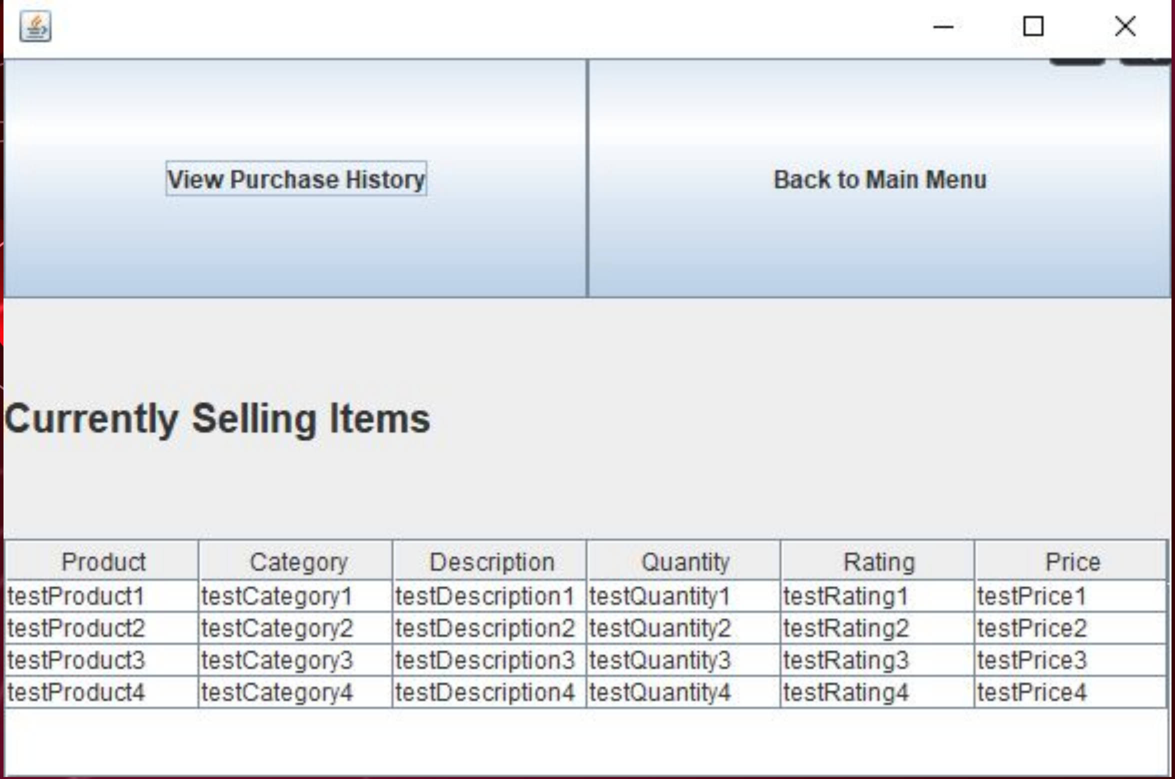
Description: Fast CPU from Intellig

Quantity: 5

Rating: 9.3

Price: $1999.99

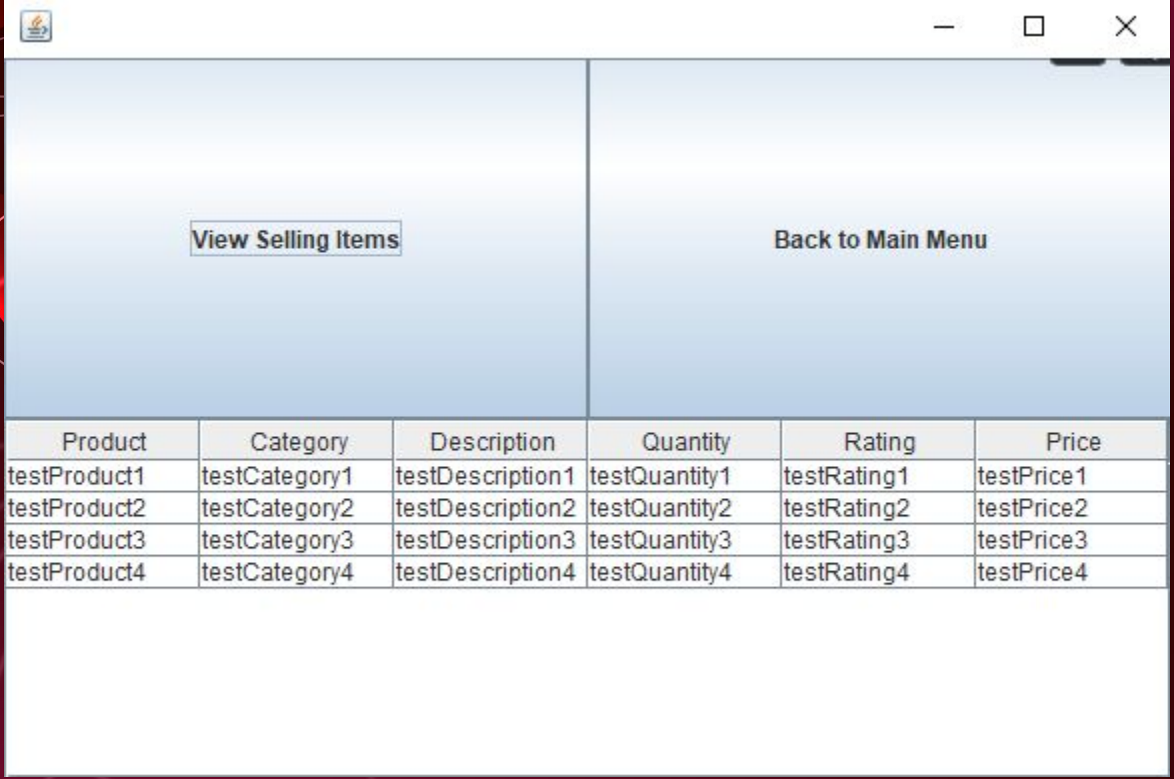
CURRENTLY SELLING ITEMS



Currently Selling Items

Product	Category	Description	Quantity	Rating	Price
testProduct1	testCategory1	testDescription1	testQuantity1	testRating1	testPrice1
testProduct2	testCategory2	testDescription2	testQuantity2	testRating2	testPrice2
testProduct3	testCategory3	testDescription3	testQuantity3	testRating3	testPrice3
testProduct4	testCategory4	testDescription4	testQuantity4	testRating4	testPrice4

PURCHASE HISTORY



Product	Category	Description	Quantity	Rating	Price
testProduct1	testCategory1	testDescription1	testQuantity1	testRating1	testPrice1
testProduct2	testCategory2	testDescription2	testQuantity2	testRating2	testPrice2
testProduct3	testCategory3	testDescription3	testQuantity3	testRating3	testPrice3
testProduct4	testCategory4	testDescription4	testQuantity4	testRating4	testPrice4

Richard Hutcheson, Noah Lambaria, Austin Blanchard, Josh McKone

GRASP Within BearMarket

(General Responsibility Assignment Software Principles)

1. Information Expert

- all of BearMarket's classes implement the Information Expert principle by ensuring each class is assigned responsibilities based on the date each class possesses
- ex: the ReadProductFile class knows of the product file, so it reads in all the products
- ex: the ProductTable class contains a list of all of the Products, so it populates the table
- ex: the CreateAccount class contains a user's new account details so it handles creating the account

2. Creator

- One of Bear Market's largest Creators is the MainScreen class. Depending on the user's input when deciding which menu button to click, triggers a specific object to be created. An example is if the user selects the Purchase History button, a Purchase History class object is created. Another example is the LoginScreen class, if the user logs in with an existing account, an Account class is created and filled with data stored from the file. Similarly, the CreateAccount class creates a new Account object but rather from file reading, directly from user input.

3. Controller

- The MainScreen class acts as a controller because it delegates out services to its respective classes. Examples include assigning the ProductTable class to generate the table for the main screen, and the ProductTable delegates cell rendering and mouse listening to the JTableButtonRenderer and JTableButtonMouseListener. Similarly the LoginScreen can delegate creating a new account to the CreateAccount class.

4. Low Coupling

- Low coupling is demonstrated through our object independence and isolation. Examples of low coupling include the PurchaseHistory and CurrentlySelling classes which are designed in a way that changes can be easily made to those classes without breaking or requiring adjustments in other classes.

5. High Cohesion

- High Cohesion is a significant principle utilized in BearMarket. Each class seeks to keep objects appropriately focused, manageable and understandable. For example, ReadProductFile does exactly as its name suggests, it's entire responsibility is to read the product tsv in and assign data into new Product objects. Further examples of cohesion include CreateAccount only creating a new Account object, PurchaseItem only handling item purchasing, and the EditAccount class only handling Account editing.

6. Indirection

- Exercising Indirection allows for the process of Low Coupling due to mediation of immediate objects and other components or services, preventing direct coupling. An example of this is the ProductTable mediating between the MainScreen and ReadProductFile. Because the ProductTable acts as the middle-man between MainScreen and ReadProductFile, the coupling between the two files is lowered.

7. Polymorphism

- One example of polymorphism in our software is the ProductTable class inheriting from MainScreen, enabling the ProductTable class access to the protected Swing variables in the MainScreen class. This allows the Product Table class to handle the creation and set up of the product class. Additionally, polymorphism is used extensively through the use of the Swing library. An example being how Jbuttons, JMenuItems, and more, all possess functionality from the ActionListener class.

8. Pure Fabrication

- We do not want to violate high cohesion and low coupling so we implement artificial classes that do not represent an item in the problem domain. An example of this is LoginButton which exists only to maintain cohesion and coupling between LoginMenu and the account database. LoginButton's simple task is merely to ensure an account exists for the information provided and return a boolean.

9. Protected Variation

- Classes that have no reason to be aware of each other or couple, have no direct interaction between each other. We design the classes in a way to reduce instability by assigning responsibilities appropriately enabling for ease of production. By doing so increases the stability of the program as it changes or evolves.

Test Coverage Plan

- JUNIT testing will be used in order to validate specific scenarios that the user might encounter, ensuring that there will be limited issues.

1. Login Screen Testing
    a. Assert that both username and password are editable
    b. Check that the information provided links to an account
    c. Validate an invalid account login
    d. Validate an authorized login
    e. Validate that the buttons work.

2. Create Account Testing
    a. Assert that the user can edit each JTextField
    b. Validate that each JTextField is unable to be empty for an account to be created
    c. Validate that the username hasn't already been taken
    d. Assert true that the account has been created if it meets the specific criteria
    e. Assert that the "accountList" file can be opened
    f. Update the "accountList" file

3. Edit Account Testing
    a. Assert that the user can edit every JTextField
    b. Validate that every JTextField is unable to be empty in order for the information to be updated
    c. Assert that the specific account's file can be opened. ("username".csv)
    d. Verify that the specific accounts file has been updated with the new information

4. Currently Selling Testing
    a. Validate removing a listing
    b. Assert true that the specific listing was deleted

5. Purchase Item Testing
    a. Validate a denied purchase
    b. Validate a confirmed purchase
    c. Assert true that the item was purchased, and the transaction is complete
    d.
6. Delete Item Testing
    a. Validate that logged-in user has a file
    b. Assert true that the csv file has been opened
    c. Assert that the users item list has been updated with the item being removed
    d. Validate that the main screen has been updated successfully.

7. Create Post Testing
    a. Validate that post has been written to file based on user input
    b. Assert that the item has been created
    c. Assert true that the post is contained within the marketplace table

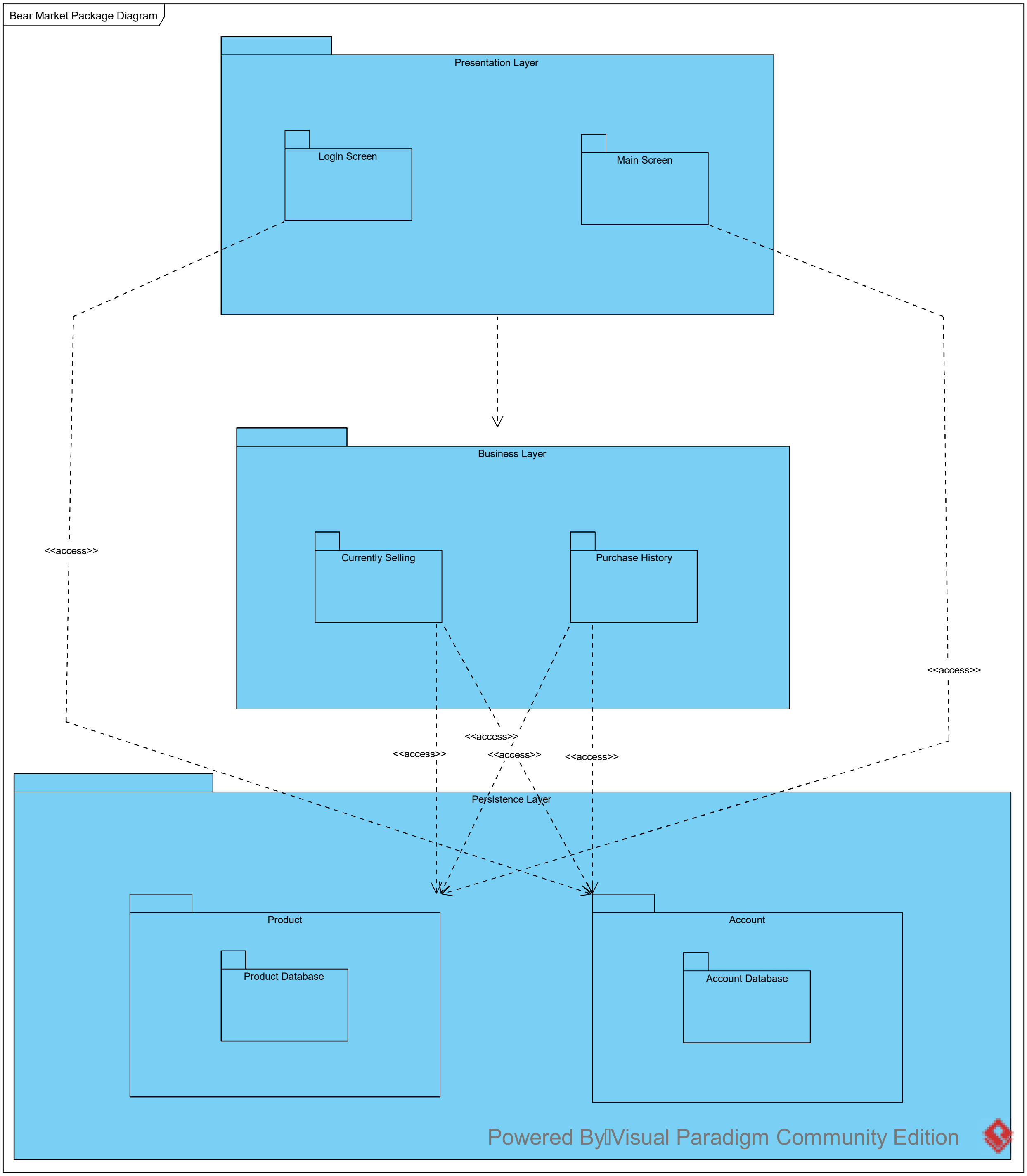
8. Edit Post Testing
    a. Validate that correct item id has been rewritten within the account file
    b. Check if edits are displayed within the main market and are accurate
    c. Validate that confirm button works and is clickable.

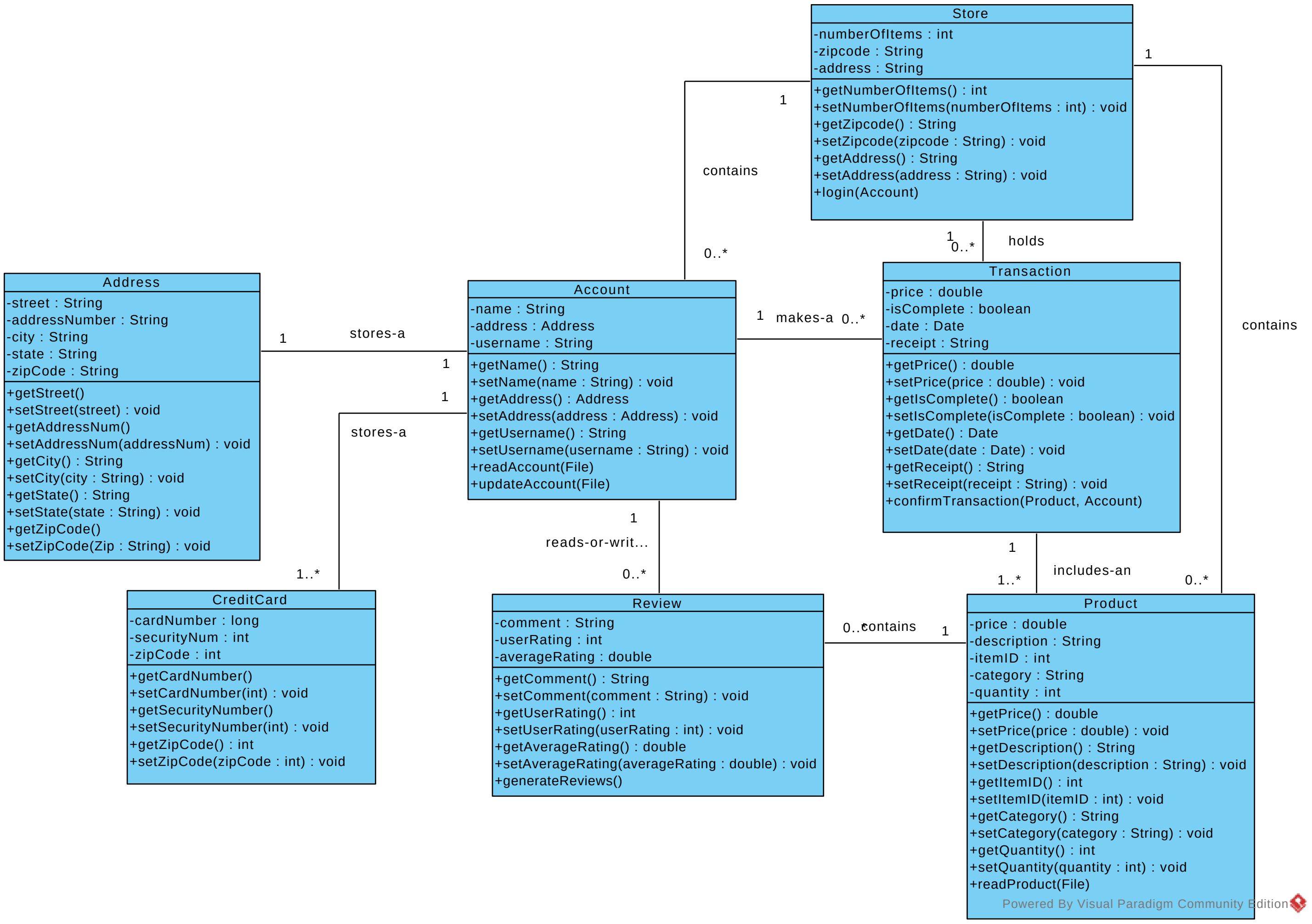
9. Main Screen Testing
    a. Validate that menu buttons work properly
    b. Testing sorting by each category
    c. Ensure that Description button works
    d. Validate that data is properly presented in table
    e. Assert that the data is read in properly from product list tsv

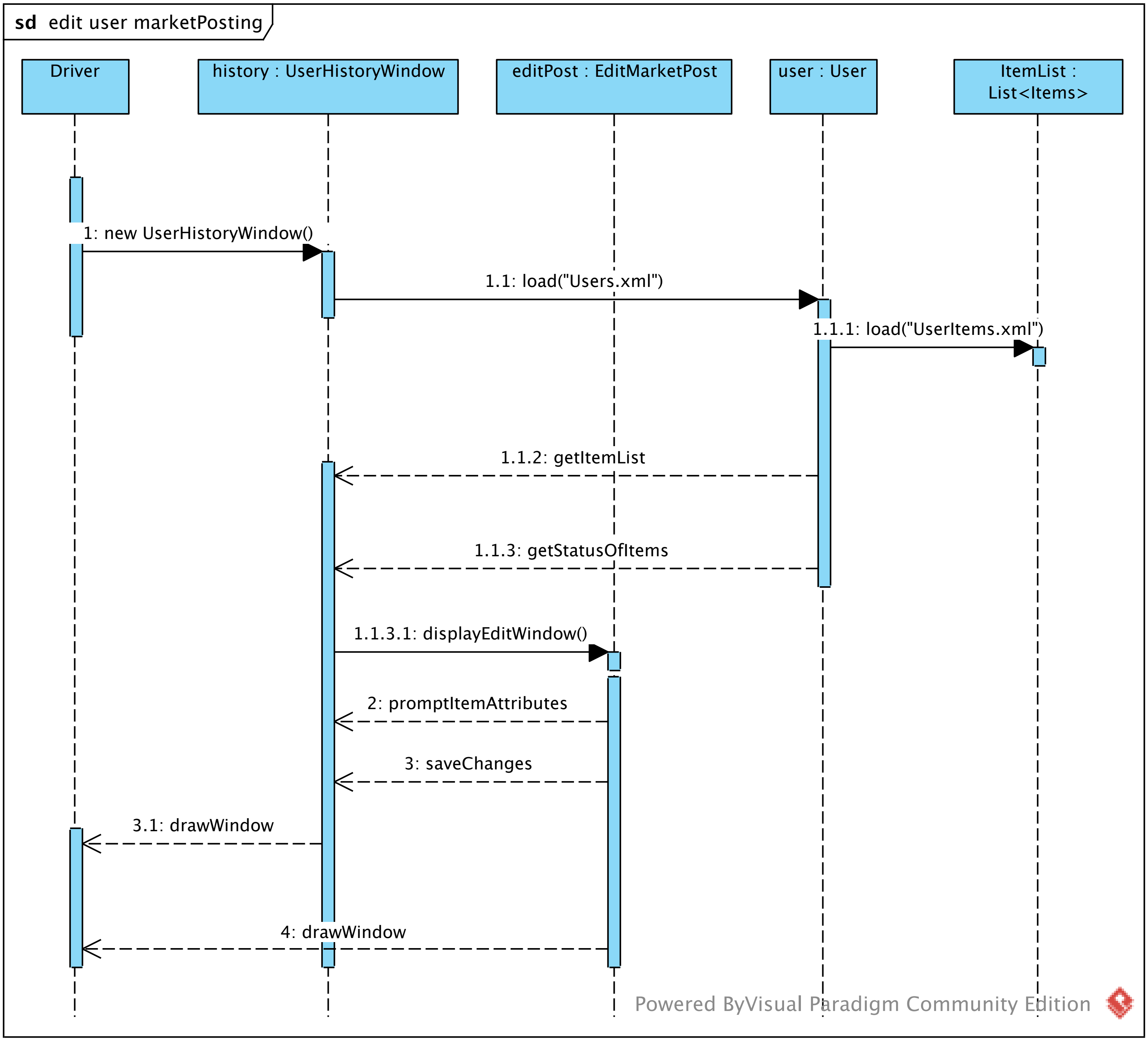
10. Purchase Item History Testing
    a. Validate that the items displayed match the items the user has purchased
    b. Assert that the account's file has been opened (which is where the items will be stored)

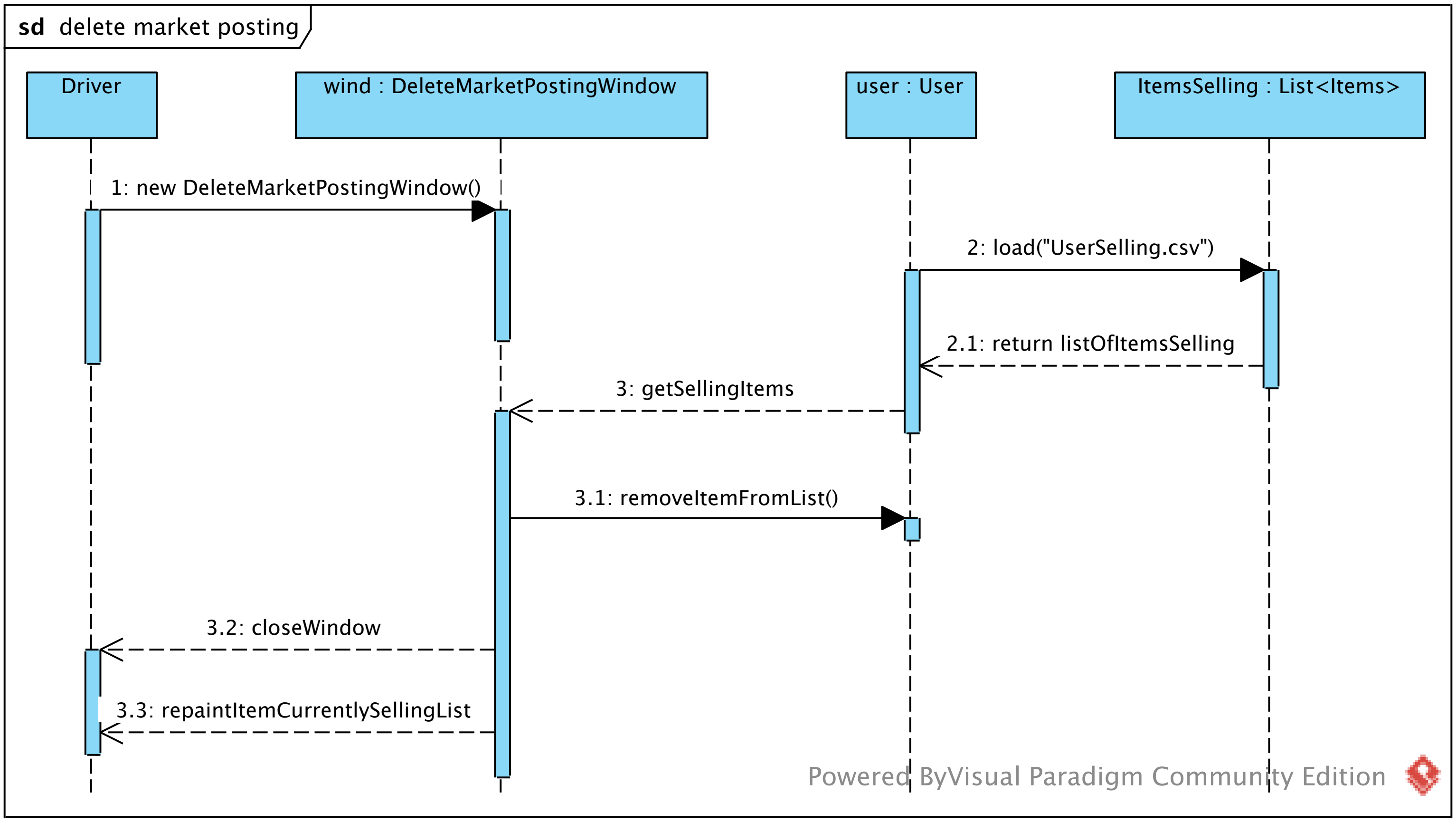
11. Featured Items Testing
    a. Assert that the product file has been opened
    b. Verify that the products have been randomly selected

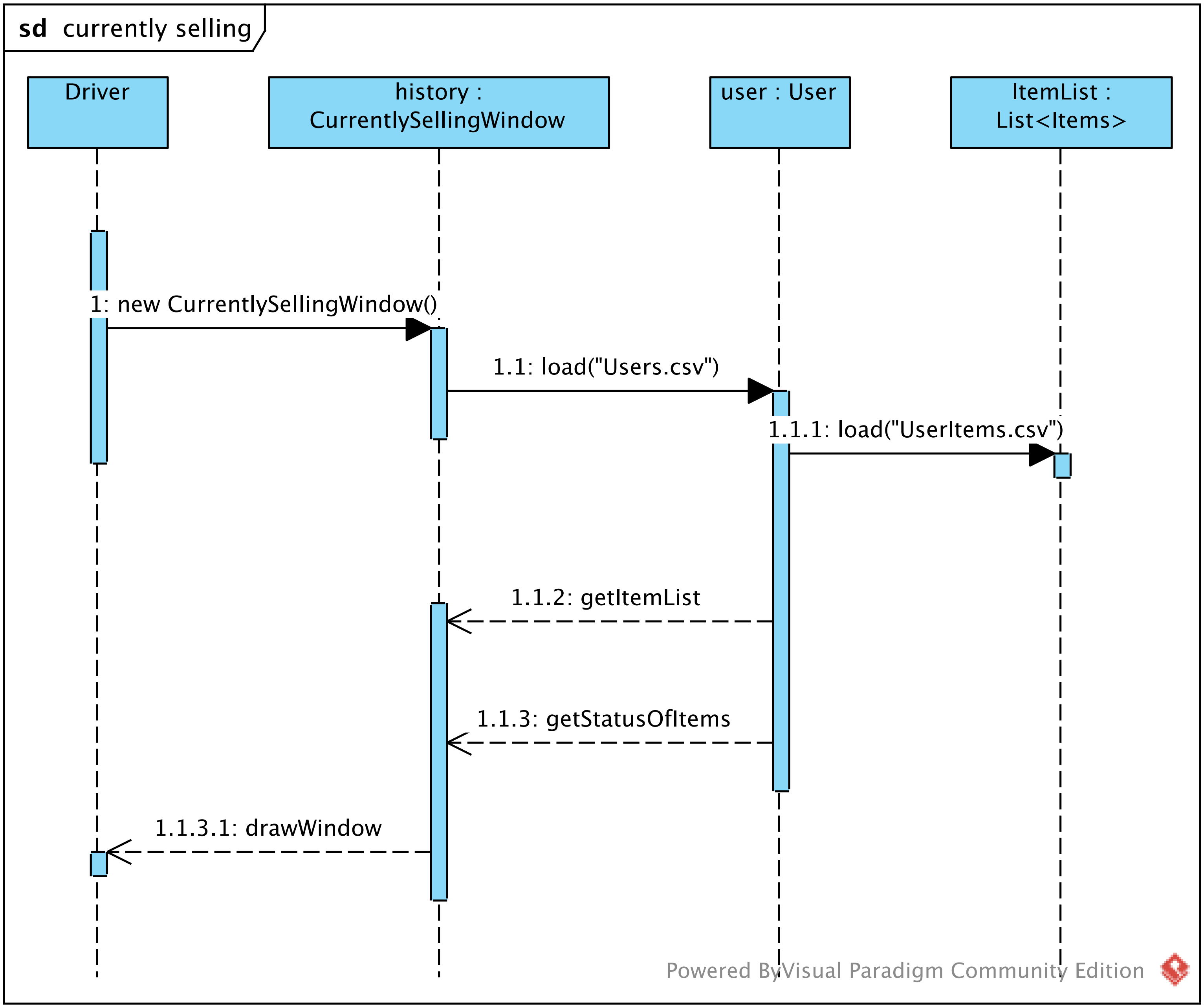
12. Reviews Testing
    a. Verify that every product has at least one review
    b. Ensure that the reviews are randomly generated
    c. Assert that the reviews rating is between 0 and 10

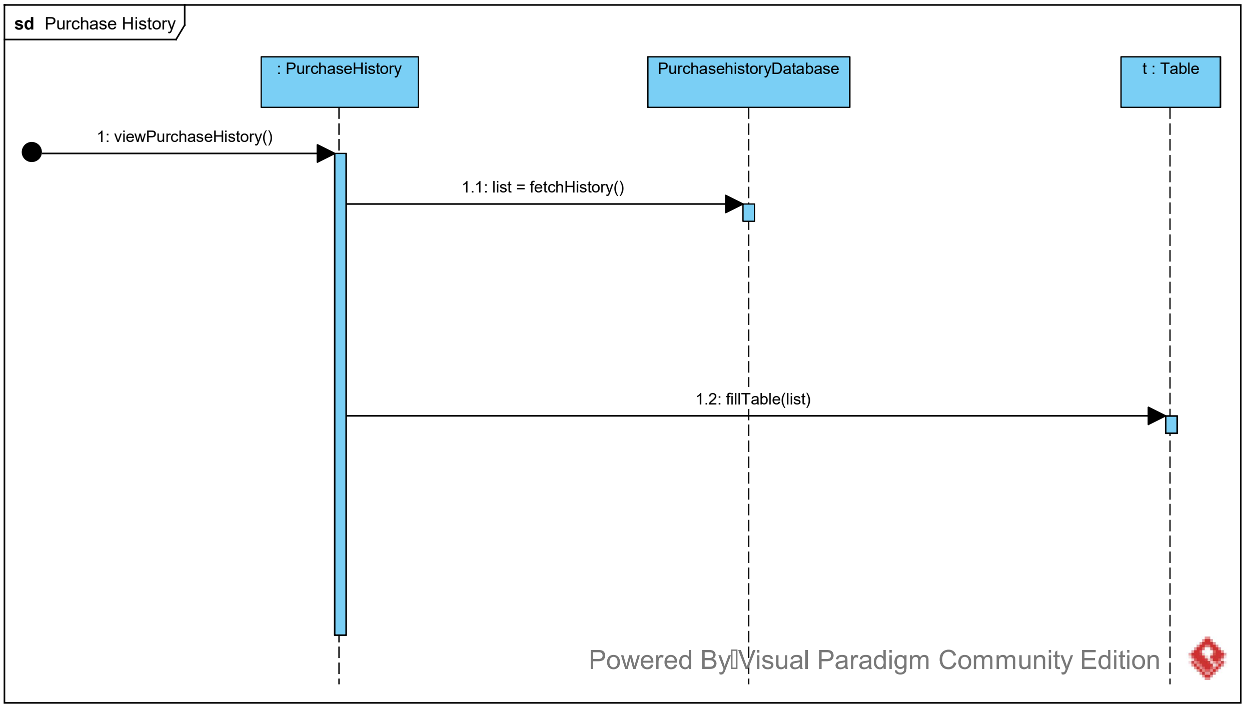

Bear Market Package Diagram
Presentation Layer
Login Screen
Main Screen
Business Layer
Currently Selling
Purchase History
<<access>>
<<access>>
<<access>>
<<access>>
<<access>>
<<access>>
Persistence Layer
Product
Product Database
Account
Account Database
Powered By Visual Paradigm Community Edition

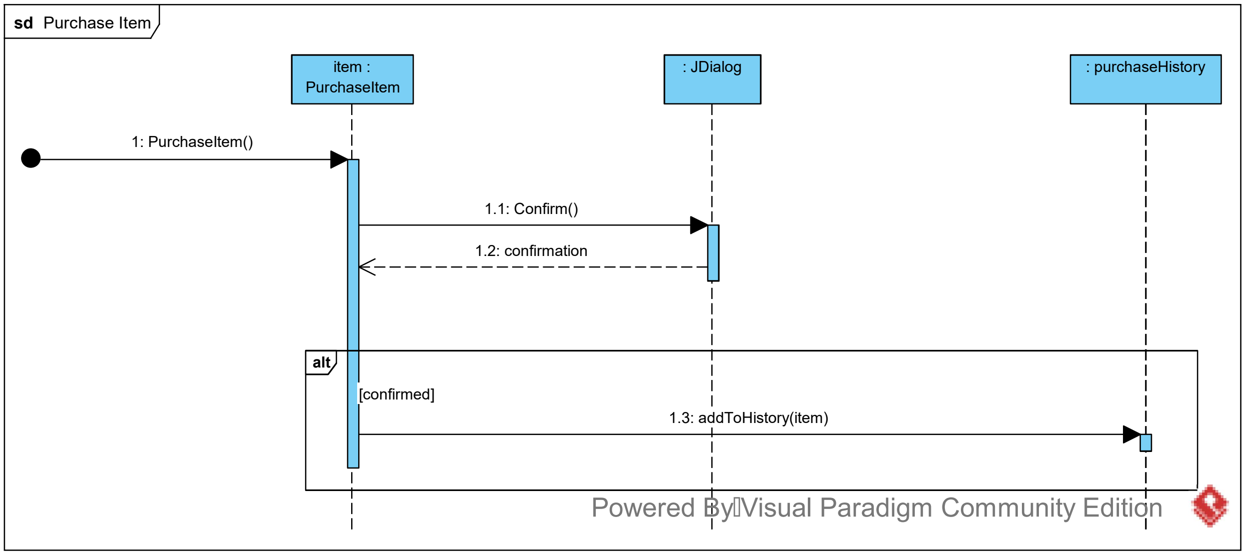

Store
-numberOfItems : int
-zipcode : String
-address : String
+getNumberOfItems() : int
+setNumberOfItems(numberOfItems : int) : void
+getZipcode() : String
+setZipcode(zipcode : String) : void
+getAddress() : String
+setAddress(address : String) : void
+login(Account)
1
1
contains
0..*
1
0..*
holds
contains
Address
-street : String
-addressNumber : String
-city : String
-state : String
-zipCode : String
+getStreet()
+setStreet(street) : void
+getAddressNum()
+setAddressNum(addressNum) : void
+getCity() : String
+setCity(city : String) : void
+getState() : String
+setState(state : String) : void
+getZipCode()
+setZipCode(Zip : String) : void
1
stores-a
1
1
stores-a
1..*
Account
-name : String
-address : Address
-username : String
+getName() : String
+setName(name : String) : void
+getAddress() : Address
+setAddress(address : Address) : void
+getUsername() : String
+setUsername(username : String) : void
+readAccount(File)
+updateAccount(File)
1
makes-a
0..*
Transaction
-price : double
-isComplete : boolean
-date : Date
-receipt : String
+getPrice() : double
+setPrice(price : double) : void
+getIsComplete() : boolean
+setIsComplete(isComplete : boolean) : void
+getDate() : Date
+setDate(date : Date) : void
+getReceipt() : String
+setReceipt(receipt : String) : void
+confirmTransaction(Product, Account)
1
reads-or-writ...
0..*
1
1..*
includes-an
0..*
CreditCard
-cardNumber : long
-securityNum : int
-zipCode : int
+getCardNumber()
+setCardNumber(int) : void
+getSecurityNumber()
+setSecurityNumber(int) : void
+getZipCode() : int
+setZipCode(zipCode : int) : void
Review
-comment : String
-userRating : int
-averageRating : double
+getComment() : String
+setComment(comment : String) : void
+getUserRating() : int
+setUserRating(userRating : int) : void
+getAverageRating() : double
+setAverageRating(averageRating : double) : void
+generateReviews()
0..*
contains
1
Product
-price : double
-description : String
-itemID : int
-category : String
-quantity : int
+getPrice() : double
+setPrice(price : double) : void
+getDescription() : String
+setDescription(description : String) : void
+getItemID() : int
+setItemID(itemID : int) : void
+getCategory() : String
+setCategory(category : String) : void
+getQuantity() : int
+setQuantity(quantity : int) : void
+readProduct(File)
Powered By Visual Paradigm Community Edition

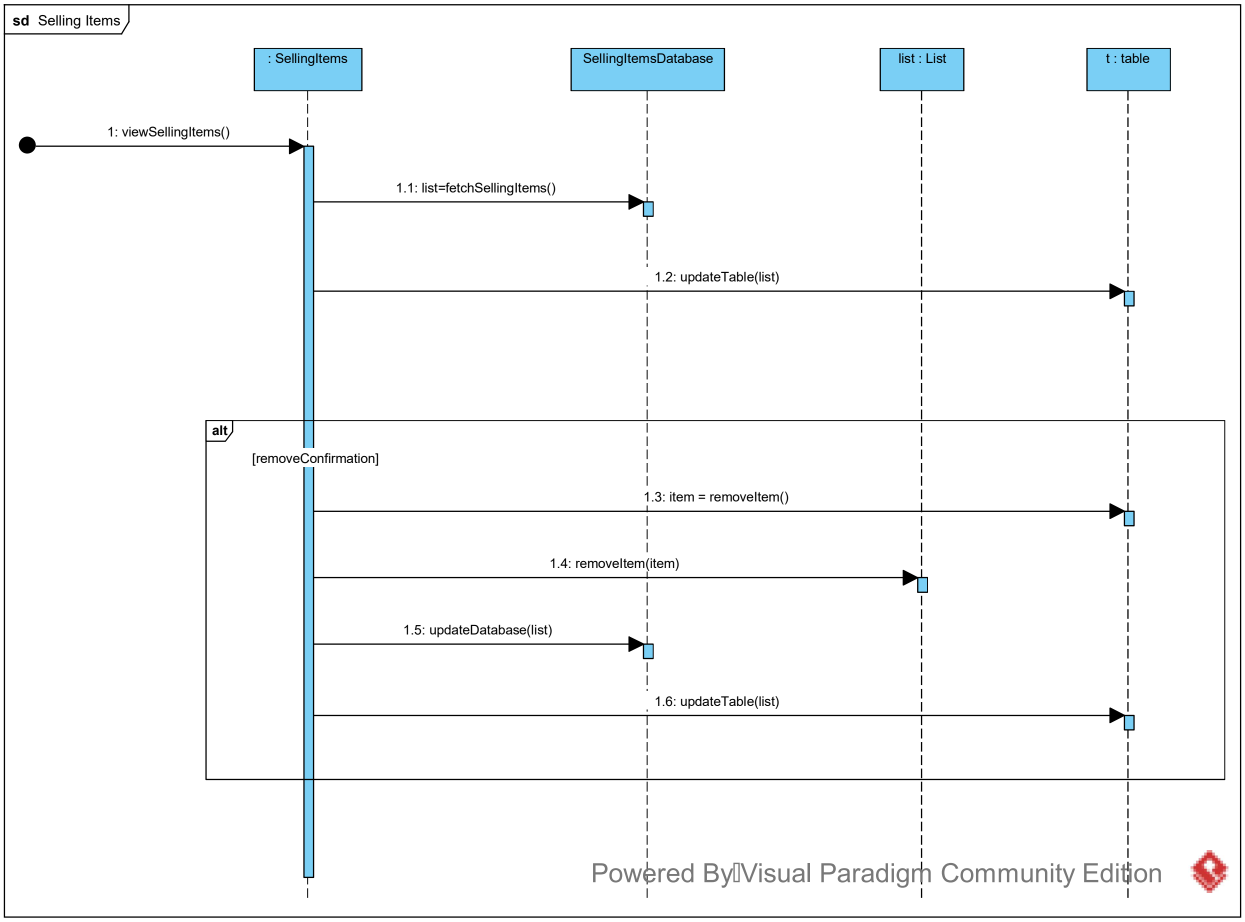

sd edit user marketPosting
Driver
history : UserHistoryWindow
editPost : EditMarketPost
user : User
ItemList : List<Items>
1: new UserHistoryWindow()
1.1: load("Users.xml")
1.1.1: load("UserItems.xml")
1.1.2: getItemList
1.1.3: getStatusOfItems
1.1.3.1: displayEditWindow()
2: promptItemAttributes
3: saveChanges
3.1: drawWindow
4: drawWindow
Powered ByVisual Paradigm Community Edition

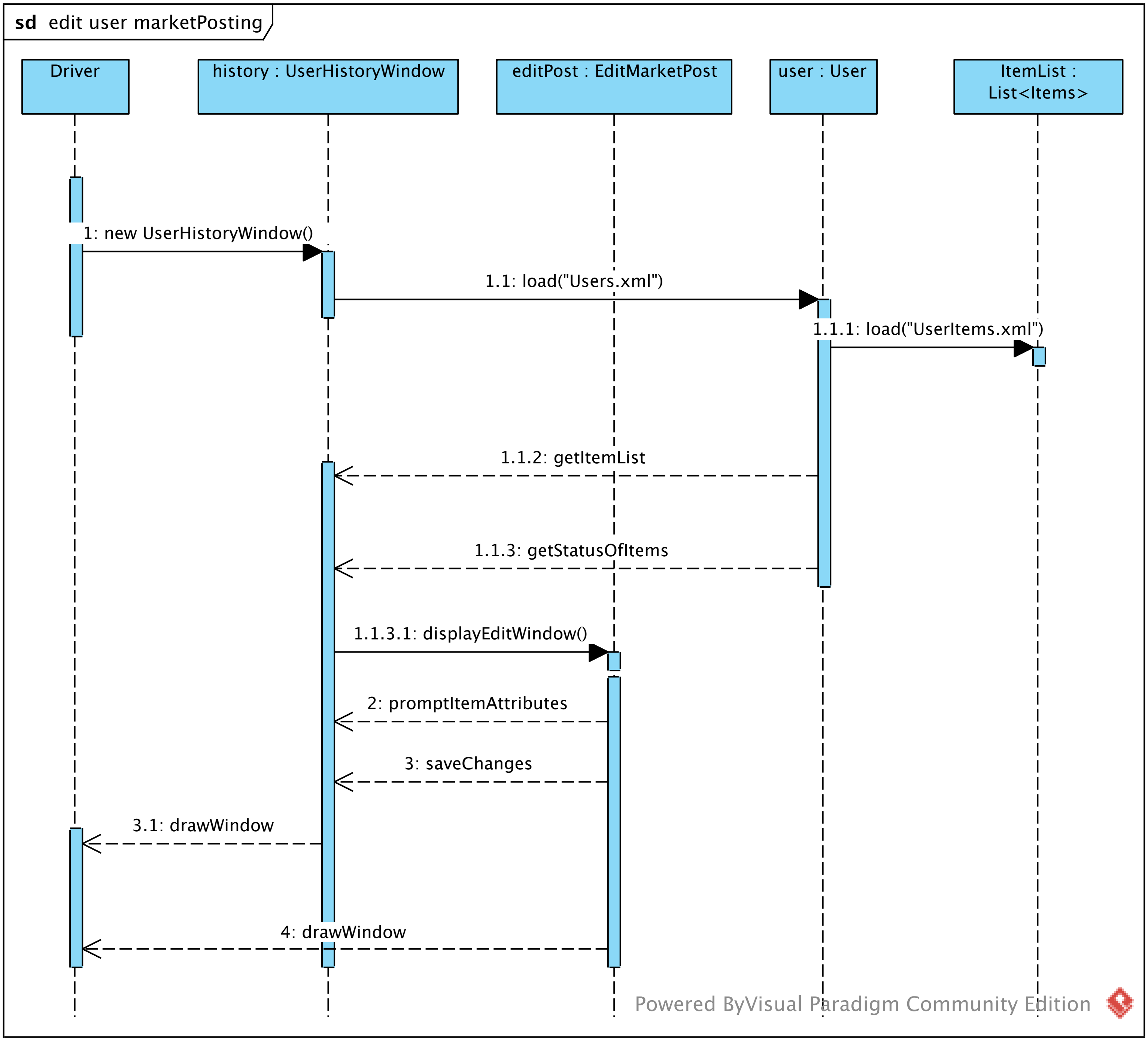

sd delete market posting
Driver
wind : DeleteMarketPostingWindow
user : User
ItemsSelling : List<Items>
1: new DeleteMarketPostingWindow()
2: load("UserSelling.csv")
2.1: return listOfItemsSelling
3: getSellingItems
3.1: removeItemFromList()
3.2: closeWindow
3.3: repaintItemCurrentlySellingList
Powered ByVisual Paradigm Community Edition

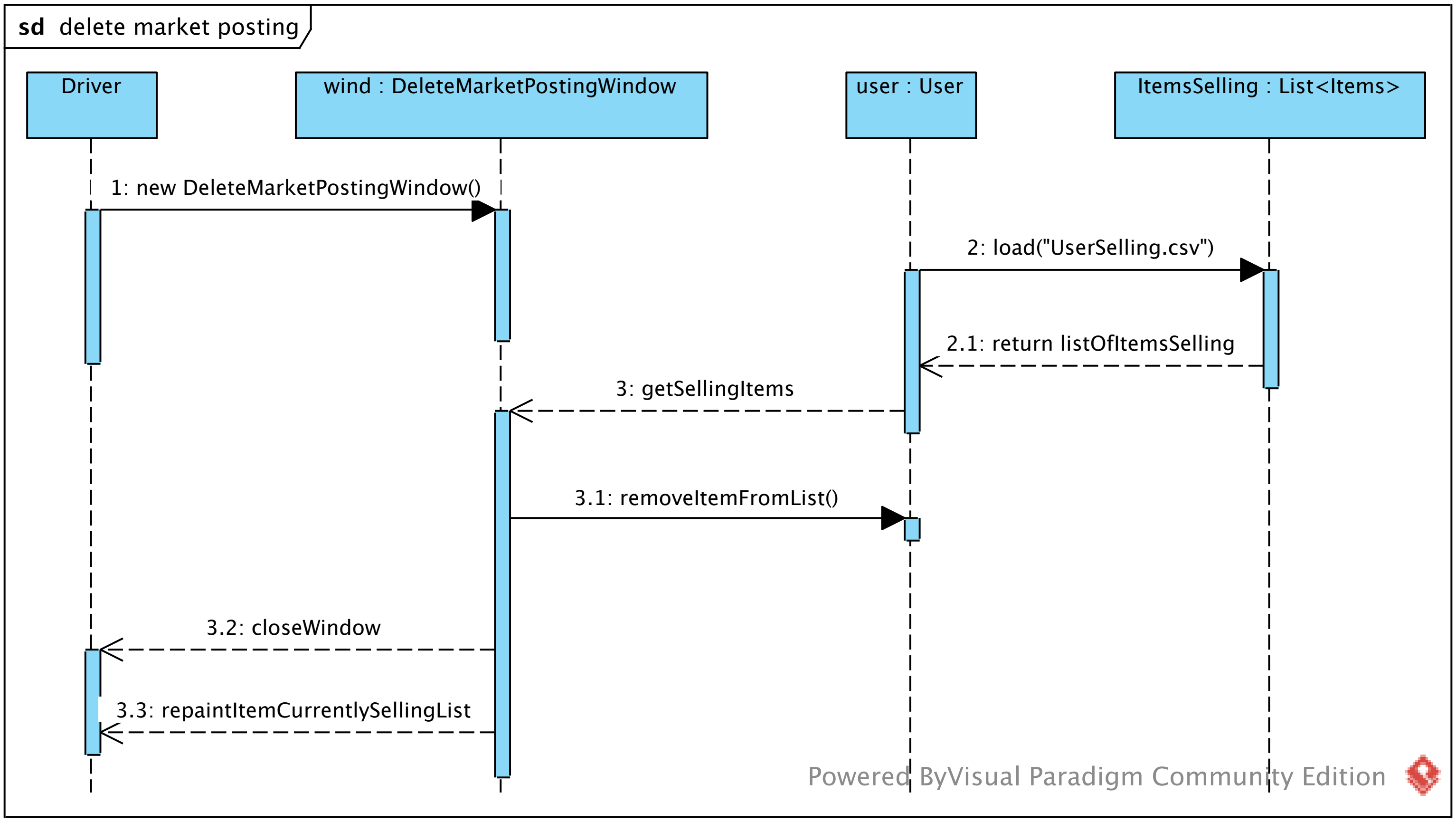

sd currently selling
Driver
history : CurrentlySellingWindow
user : User
ItemList : List<Items>
1: new CurrentlySellingWindow()
1.1: load("Users.csv")
1.1.1: load("UserItems.csv")
1.1.2: getItemList
1.1.3: getStatusOfItems
1.1.3.1: drawWindow
Powered By Visual Paradigm Community Edition

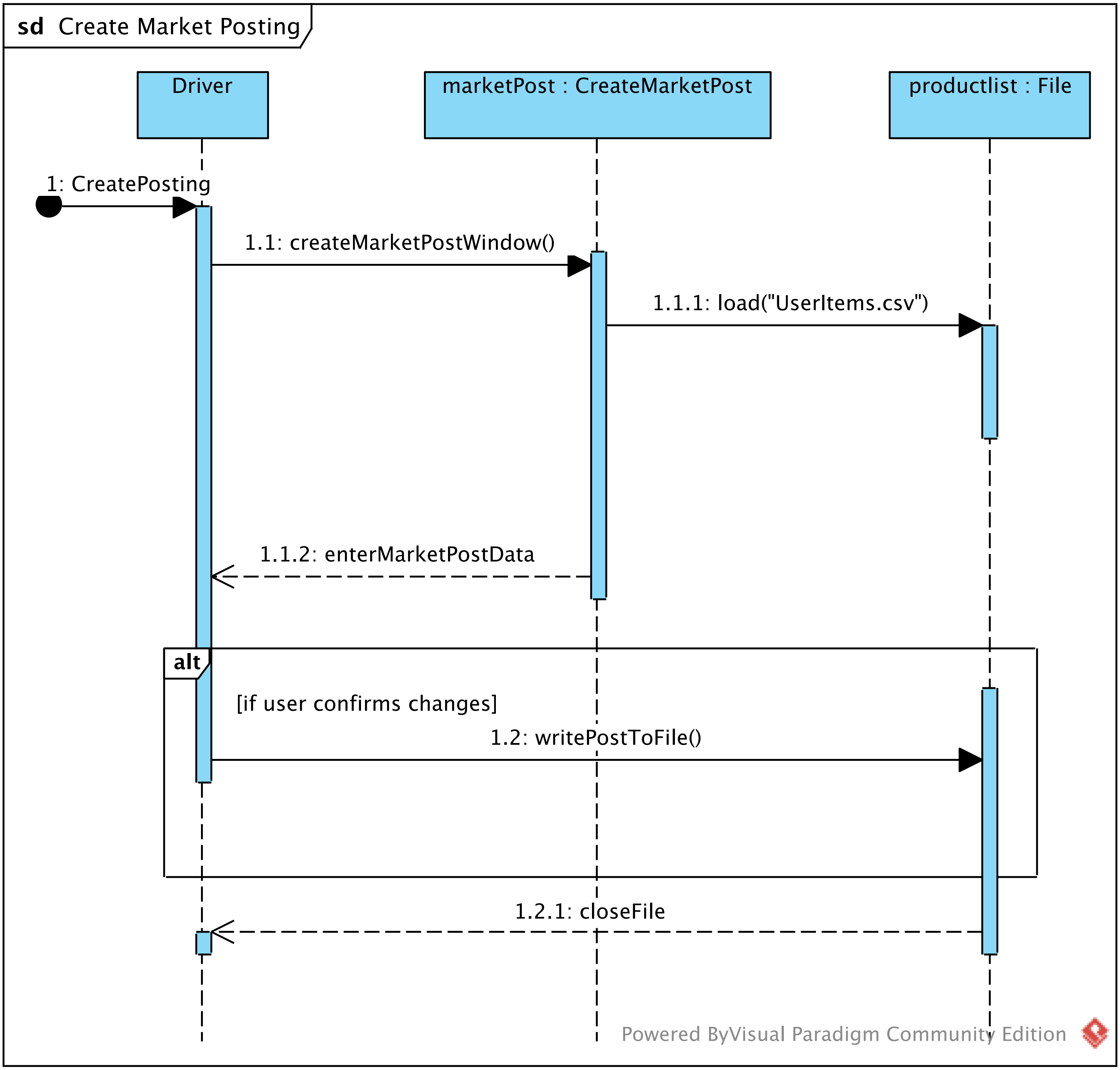

sd Purchase History
: PurchaseHistory
PurchasehistoryDatabase
t : Table
1: viewPurchaseHistory()
1.1: list = fetchHistory()
1.2: fillTable(list)
Powered By Visual Paradigm Community Edition

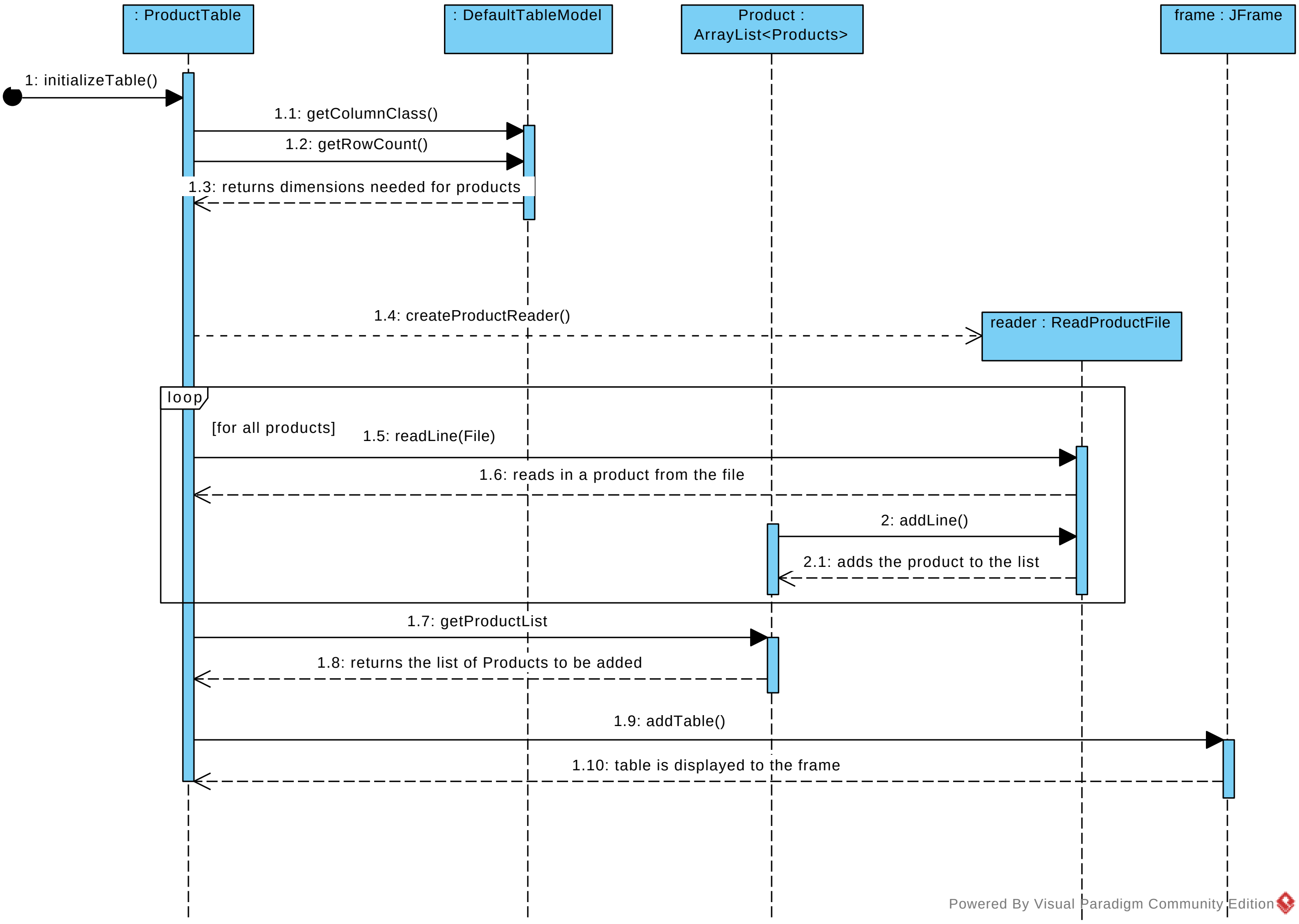

sd Purchase Item
item : PurchaseItem
: JDialog
: purchaseHistory
1: PurchaseItem()
1.1: Confirm()
1.2: confirmation
alt
[confirmed]
1.3: addToHistory(item)
Powered By Visual Paradigm Community Edition

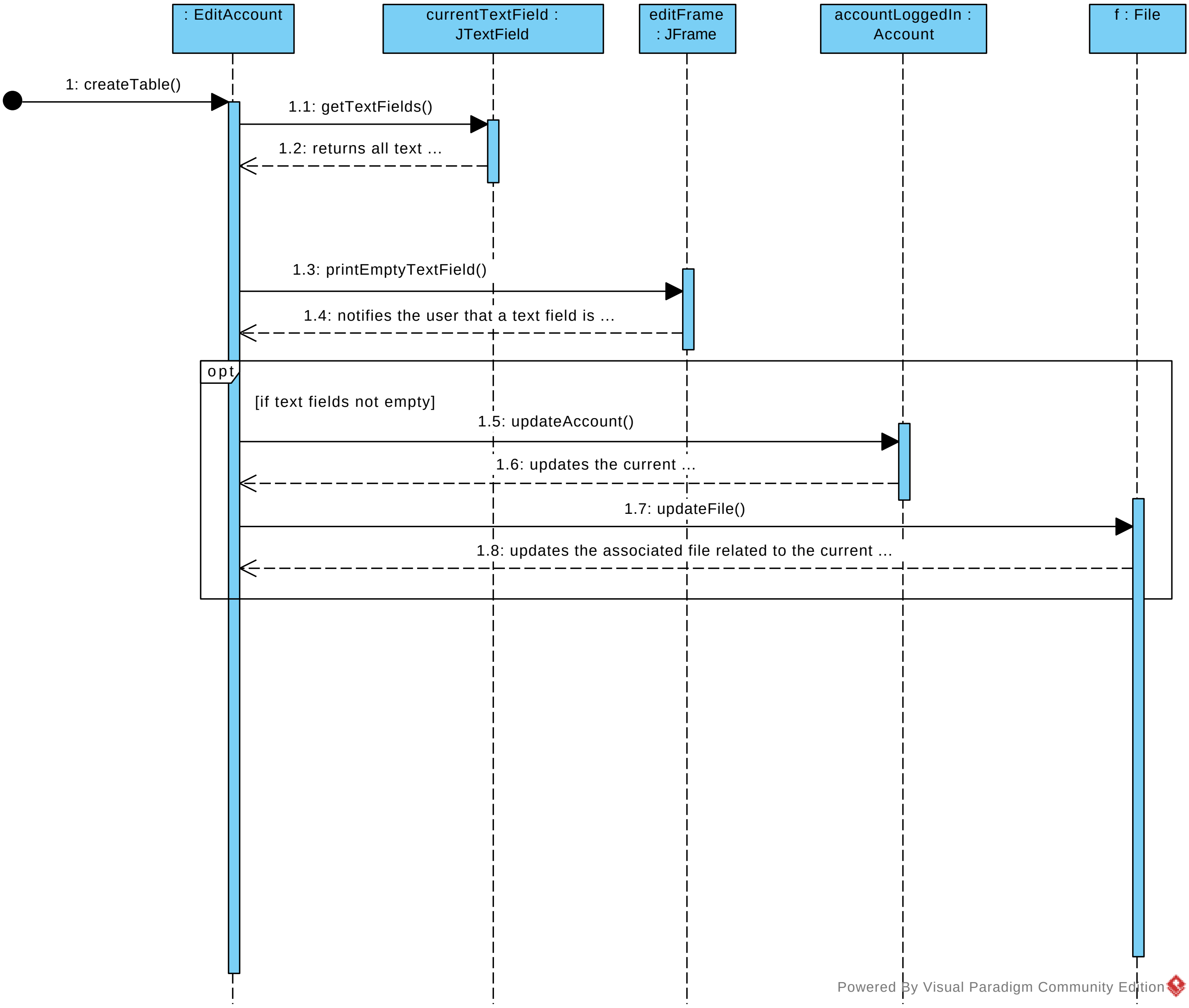

sd Selling Items
: SellingItems
SellingItemsDatabase
list : List
t : table
1: viewSellingItems()
1.1: list=fetchSellingItems()
1.2: updateTable(list)
alt
[removeConfirmation]
1.3: item = removeItem()
1.4: removeItem(item)
1.5: updateDatabase(list)
1.6: updateTable(list)

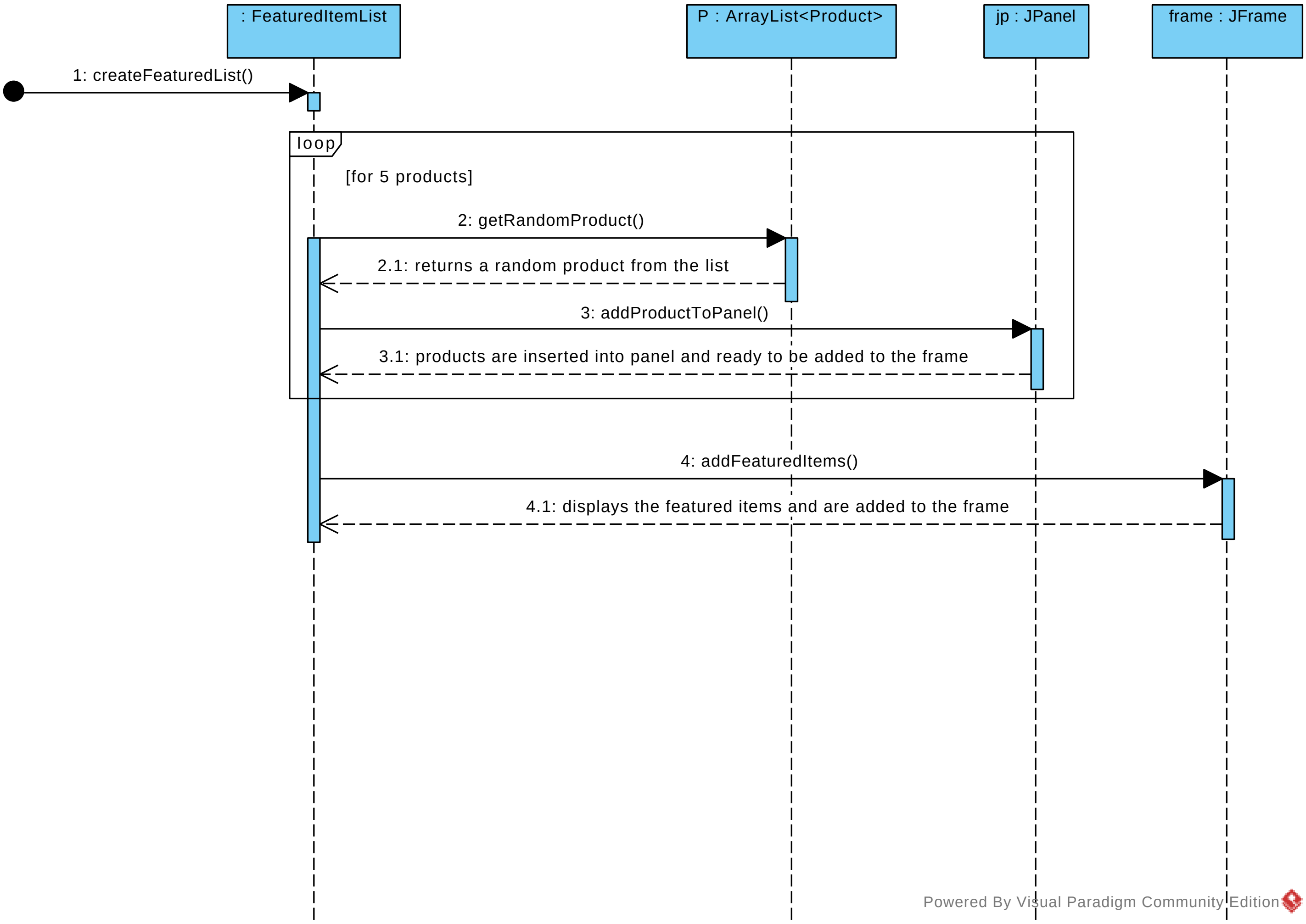

sd edit user marketPosting
Driver
history : UserHistoryWindow
editPost : EditMarketPost
user : User
ItemList : List<Items>
1: new UserHistoryWindow()
1.1: load("Users.xml")
1.1.1: load("UserItems.xml")
1.1.2: getItemList
1.1.3: getStatusOfItems
1.1.3.1: displayEditWindow()
2: promptItemAttributes
3: saveChanges
3.1: drawWindow
4: drawWindow
Powered ByVisual Paradigm Community Edition


sd delete market posting
Driver
wind : DeleteMarketPostingWindow
user : User
ItemsSelling : List<Items>
1: new DeleteMarketPostingWindow()
2: load("UserSelling.csv")
2.1: return listOfItemsSelling
3: getSellingItems
3.1: removeItemFromList()
3.2: closeWindow
3.3: repaintItemCurrentlySellingList
Powered ByVisual Paradigm Community Edition


sd Create Market Posting
Driver
marketPost : CreateMarketPost
productlist : File
1: CreatePosting
1.1: createMarketPostWindow()
1.1.1: load("UserItems.csv")
1.1.2: enterMarketPostData
alt
[if user confirms changes]
1.2: writePostToFile()
1.2.1: closeFile
Powered ByVisual Paradigm Community Edition


: ProductTable
: DefaultTableModel
Product : ArrayList<Products>
frame : JFrame
1: initializeTable()
1.1: getColumnClass()
1.2: getRowCount()
1.3: returns dimensions needed for products
1.4: createProductReader()
reader : ReadProductFile
loop
[for all products]
1.5: readLine(File)
1.6: reads in a product from the file
2: addLine()
2.1: adds the product to the list
1.7: getProductList
1.8: returns the list of Products to be added
1.9: addTable()
1.10: table is displayed to the frame
Powered By Visual Paradigm Community Edition


: EditAccount
currentTextField : JTextField
editFrame : JFrame
accountLoggedIn : Account
f : File
1: createTable()
1.1: getTextFields()
1.2: returns all text ...
1.3: printEmptyTextField()
1.4: notifies the user that a text field is ...
opt
[if text fields not empty]
1.5: updateAccount()
1.6: updates the current ...
1.7: updateFile()
1.8: updates the associated file related to the current ...
Powered By Visual Paradigm Community Edition


: FeaturedItemList
P : ArrayList<Product>
jp : JPanel
frame : JFrame
1: createFeaturedList()
loop
[for 5 products]
2: getRandomProduct()
2.1: returns a random product from the list
3: addProductToPanel()
3.1: products are inserted into panel and ready to be added to the frame
4: addFeaturedItems()
4.1: displays the featured items and are added to the frame
Powered By Visual Paradigm Community Edition

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>csi3471</groupId>
  <artifactId>bearMarket</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>bearMarket</name>
  <url>https://richard-hutch.github.io/BearMarket/</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
      <plugins>
        <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
        </plugin>
        <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_jar_packaging -->
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
        <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
        <plugin>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.7.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
      </plugins>
    </pluginManagement><plugins><plugin><groupId>org.apache.maven.plugins</groupId><artifactId>maven-compiler-plugin</artifactId><configuration><source>14</source><target>14</target></configuration></plugin></plugins>
  </build>
</project>

```