BearGo

Iteration 1

(Project Vision, Requirements, Fully dressed Use case, Domain Model, SSD, Operation Contract, Wire-frames, Activity Diagram, Use case Diagram)

AGM Islam
Grad Student, Computer Science
agm_islam1@baylor.edu
Baylor University

Maisha Binte Rashid
Grad Student, Computer Science
maisha_rashid1@baylor.edu
Baylor University

Razwan Ahmed Tanvir
Grad Student, Computer Science
razwan_tanvir1@baylor.edu
Baylor University

Swapnil Saha
Grad Student, Computer Science
swapnil_saha1@baylor.edu
Baylor University

Tonni Das
Grad Student, Computer Science
tonni_jui1@baylor.edu
Baylor University

Sep 24, 2022

Contents

1 Project vision:

The vision of the project is to create a platform for people to send items to their known acquaintances in a convenient way.

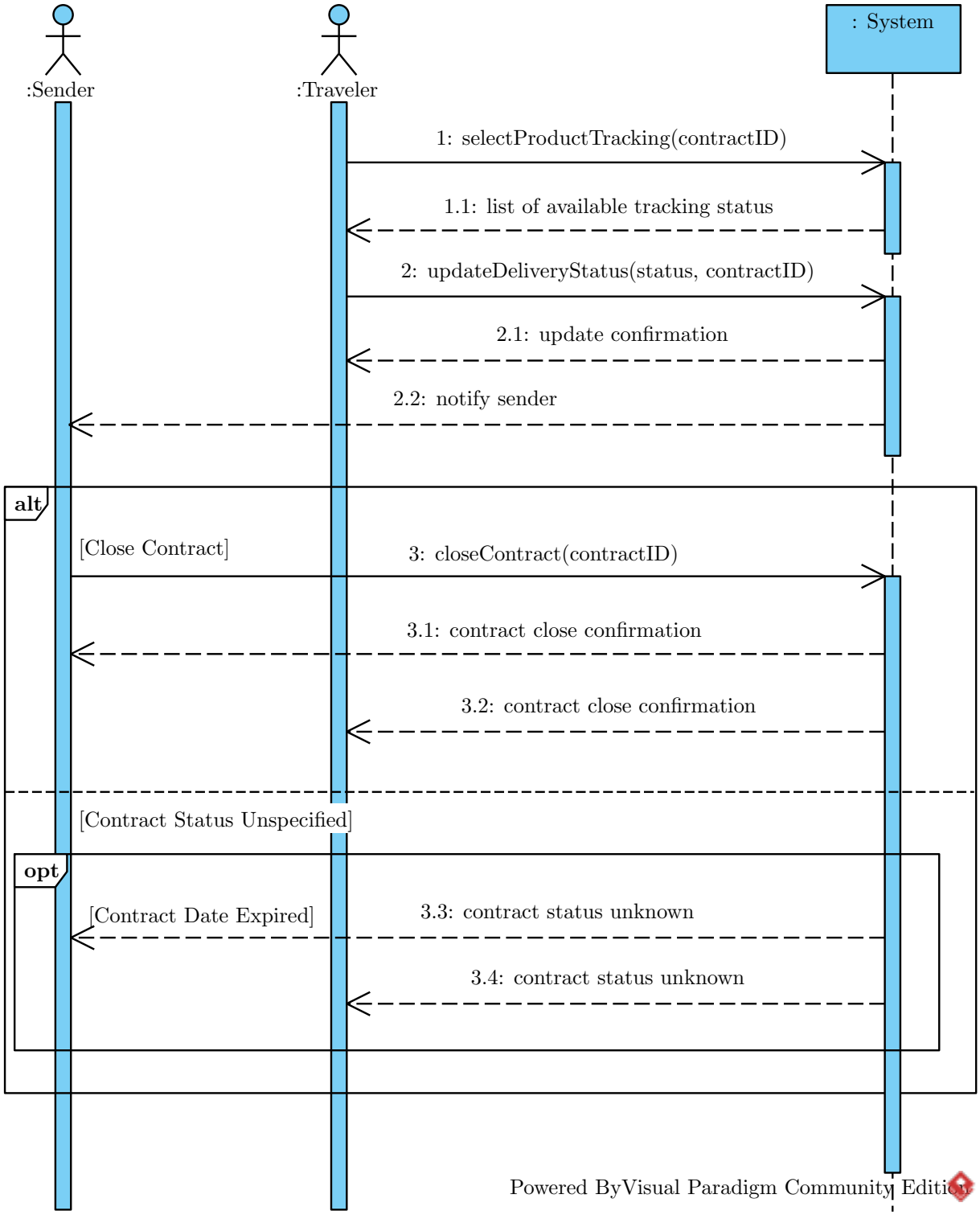
Functional Requirements

1. Product Tracking: Traveler will update the status of different stages of the product delivery. The sender will finally confirm if the product is delivered.
2. Contracts of User: Admin will be able to see any user's activity such as what deals s/he has made so far and what are the status of those deals.
3. Report Review Confirming: Admin can choose to personally talk to reporters of a ProductPost or just notify them that the post was removed.
4. Create ProductPost: Product information posting (routing info, pick up location & travel time) in application feed to find a traveler who can deliver the product.
5. Comment of ProductPost: Comment section under sender's ProductPost in which users can write comments if they have any queries regarding the productPost or other relevant details.
6. Create BlogPost: Registered users can upload their travel stories as blogs in Travel Blogging option.
7. Search ProductPost: Users can search other senders' productPosts based on different filters such as - date, location.
8. Ban User: User can report against another user and admin is responsible to ban a user from our platform.
9. Update ProductPost: Senders can also update the productPost. Senders can change the date, route, and pickup location. But once an invoice has been made or the delivery has passed then sender can not update the productPost anymore.
10. Contract Creation: Sender will create a contract and after agreeing with a suitable traveler, he will confirm the contract. The system will generate an invoice.
11. Moderate ProductPost: Admin will decide to remove/keep the productPost if the productPost is reported by the users.
12. Send Notification: Real-time web push notification and email notification. Notification will be triggered on different events like: agreement signed, products delivered.
13. Review and Rating: When the contract of a productPost will expire, both traveler and sender will see an option there to post a review and rating(1 to 5) to each other.
14. Peer-to-peer Chat: Any traveler/sender can contact to any particular sender/traveler to discuss the product details or any other travel information.
15. Social Media Share: A sender can share his/her productPost on social media.

2 Fully dressed Use cases and SSDs

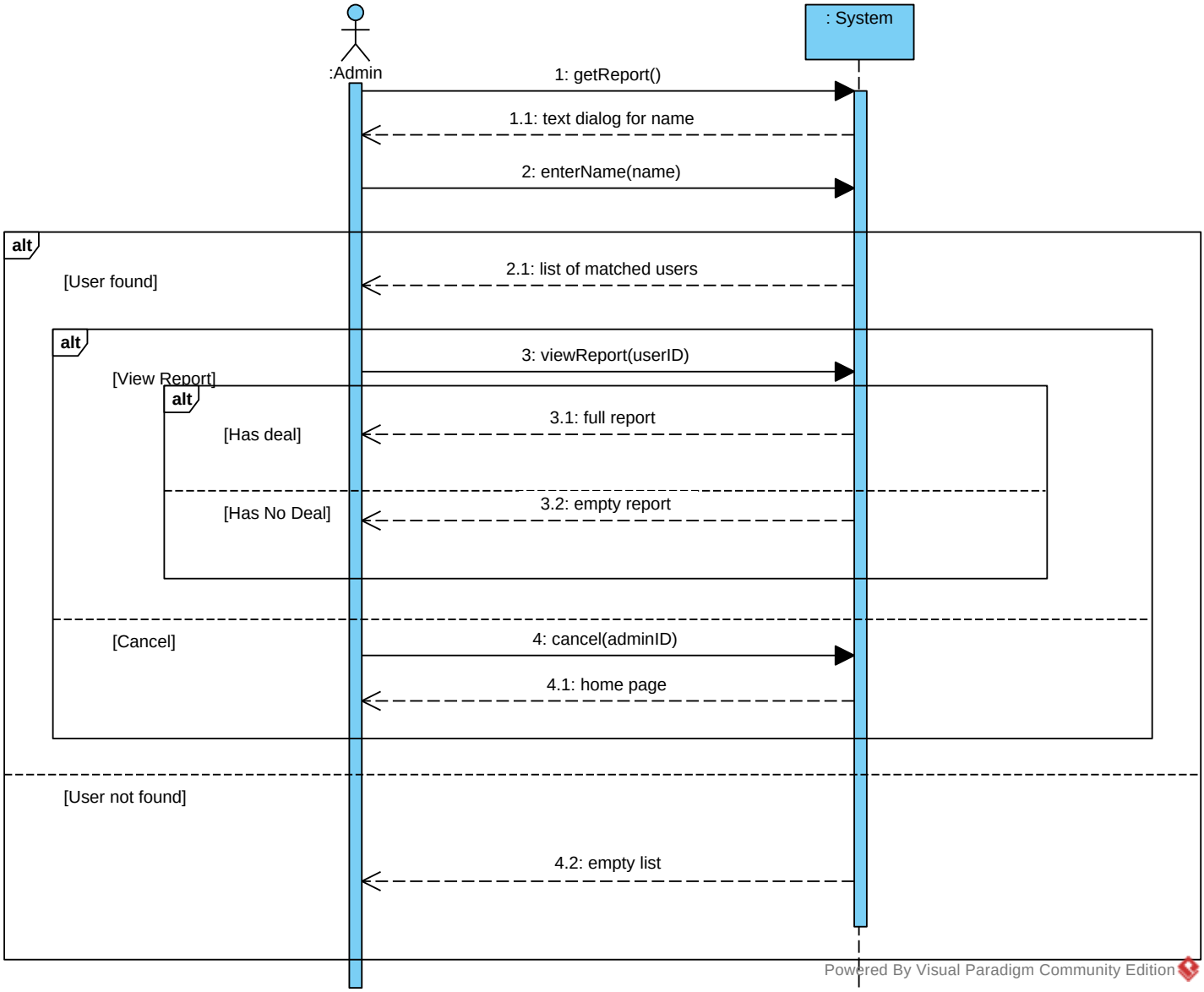
2.1 Use Case: Product Tracking

Use Case: Product Tracking
ID: UC01
Actors: 1. Traveler 2. Sender
Preconditions: 1. A contract (traveler has accepted sender's request to ship the product) has been made between traveler and sender.
Flow of Events: 1. The use case starts when the traveler receives the product. 2. The traveler selects tracking of the product. 3. The system shows the list of the available delivery status. 4. If the traveler selects the status as "Product Received", 4.1. The system updates the status of the product as "Received". 5. If the traveler selects the status as "Product In Transit", 5.1. The system updates the status of the product as "In Transit". 6. If the traveler selects the status as "Product Delivered", 6.1. The system updates the status of the product as "Delivered". 7. If the sender selects the status as "Delivery Confirmed", 7.1. The system records the contract as "Successful"
Postconditions: The customer receives the product and the status of the contract has been updated as "Successful".
Alternative flow: 7.2 If the traveler/sender selects "Product Not Delivered", the system will record the contract as "Unsuccessful".
Postconditions: The status of the contract has been updated as "Unsuccessful".
Alternative flow: 7.3 If the sender does not select or forgets to select the status for the product and the contract expires (exceeds product delivery end date), the system will record the contract as "Unknown".
Postconditions: The status of the contract has been updated as "Unknown".


:Sender
:Traveler
: System
1: selectProductTracking(contractID)
1.1: list of available tracking status
2: updateDeliveryStatus(status, contractID)
2.1: update confirmation
2.2: notify sender
alt
[Close Contract]
3: closeContract(contractID)
3.1: contract close confirmation
3.2: contract close confirmation
[Contract Status Unspecified]
opt
[Contract Date Expired]
3.3: contract status unknown
3.4: contract status unknown
Powered ByVisual Paradigm Community Edition

2.2 Use Case: Contracts of User

Use Case: Contracts of User
ID: UC02
Actors: 1. Admin
Preconditions: 1. Admin is authenticated.
Flow of Events: 1. The use case starts after the admin selects the "Get Report" option. 2. The system asks the admin for a name (name of the person that the admin is looking to see the report). 3. The admin enters the user's name. 4. The system searches for users that match the name. 5. If the system finds some matching users then, 5.1. The system shows a list of matched users with a "View Report" option beside each of these users' names and a "Cancel" option at the bottom of the page. 6. If the admin selects the "View Report" option, 6.1. If the user has made at least one deal with any other user, 6.1.1. The report of all the deals and their status (such as successful, or unsuccessful, or in unknown condition) will be displayed.
Postconditions: The admin will be able to view the report of the user where all his/her deals and statuses are.
Alternative flow: 6.1.2 If the user is new and has not made any deal yet, the system will return a string "No report yet".
Postconditions: The admin will be able to see the string text.
Alternative flow: 6.1.2 If the admin selects the option "Cancel", the system will close the searching page and will take him/her to his/her home page.
Postconditions: The admin will be able to see the home page.
Alternative flow: 5.2 If there is no user that matches, the system will show "No such user".
Postconditions: Does not apply


:Admin
: System
1: getReport()
1.1: text dialog for name
2: enterName(name)
alt
[User found]
2.1: list of matched users
alt
[View Report]
3: viewReport(userID)
alt
[Has deal]
3.1: full report
[Has No Deal]
3.2: empty report
[Cancel]
4: cancel(adminID)
4.1: home page
[User not found]
4.2: empty list
Powered By Visual Paradigm Community Edition

2.3 Use Case: Report Review Confirmation

Use Case: Report Review Confirmation

ID: UC03

Actors:

1. Admin

Preconditions:

1. Admin is authenticated.

Flow of Events:

1. The use case starts after the admin selects the "Notify Reporters" option.
2. The system shows the admin a list of reporters(users) who reported against the productPost with a "Mention", "Chat", and "Get report" option beside each of these users' names and a "Mention all" option at the bottom of the page.
3. The admin can choose to personally send a notification to a reporter or have a peer-to-peer chat with a reporter or review his/her current dealing information.
4. If the admin selects the "Mention" option beside any user,
   4.1. The system will send a notification to that user thanking them for his concern and mentioning that the productPost he/she reported, is removed by the admin from the feed.
5. If the admin selects the "Chat" option beside any user,
   5.1. If there exists previous chat history between the admin and the reporter,
      5.1.1. The conversation will be displayed with a text dialog and a "Send" option.
6. If the admin selects the "Get Report" option beside any reporter,
   6.1. If the user who reported has made at least one deal with any other user,
      6.1.1. The report of all the deals and their status (such as successful, unsuccessful, or in unknown condition) will be displayed.
7. If the admin selects the "Mention all" option from the bottom of the page,
   7.1. The system will send a notification to all users who reported, thanking them for their concern and mentioning that the productPost he/she reported, is removed by the admin from the feed.

Postconditions: Does not apply

Alternative flow:

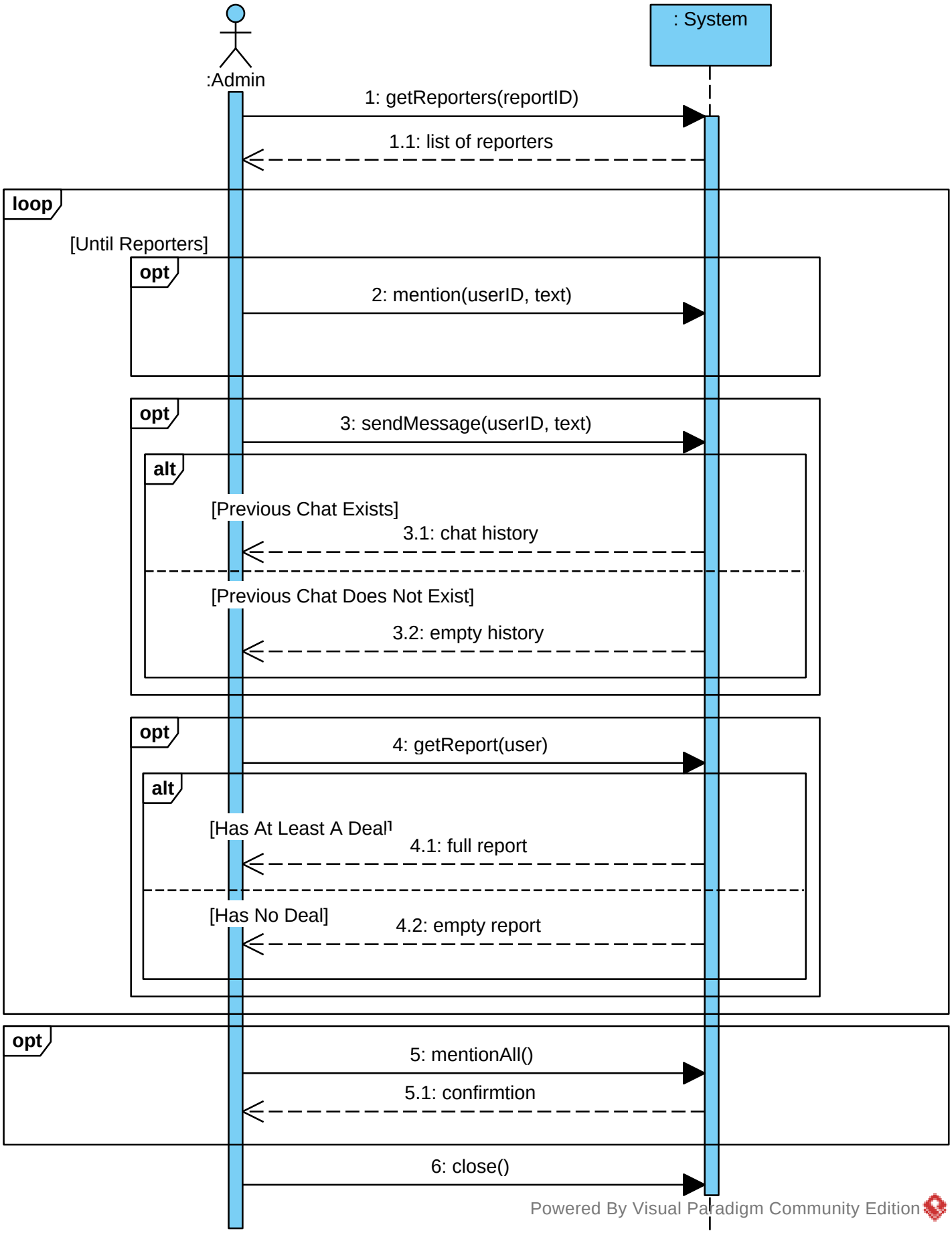
5.1.2 If there is no previous chat history between the users, the system will only show a text dialog and a "Send" option.

Postconditions: The admin will be able to see the sent text.

Alternative flow:

6.1.2 If the user is new and has not made any deal yet, the system will return a string "No report yet".

Postconditions: The admin will be able to see the string text.



2.4 Use Case: Create ProductPost

Use Case: Create ProductPost

ID: UC04

Actors:

1. Sender

Preconditions:

1. The sender is authenticated.

Flow of events:

1. The use case starts when a sender selects "Create ProductPost".
   1.1. The system will return a form for productPost creation along with "Upload Image", "Upload ProductPost", and "Cancel" options.
2. The sender will fill up the form with product details (dimension, weight, type, value) and itinerary details (source and destination addresses).
3. System will check the character limit.
4. If the sender selects "Upload Image",
   4.1. The system will return a dialog to select image(s).
   4.2. The sender will select image files from his local disk.
   4.3. System will validate the image size.
   4.4. The system will show the names of selected images.
5. If the sender selects "Upload ProductPost",
   5.1. The system will verify the productPost details (missing fields, incorrect addresses) and image files (file type, file size).
   5.2. If the productPost details are verified,
      5.2.1. The system will record the productPost.

Postconditions: The productPost will be visible on the homepage.

Alternative flow:

3.1 If the character limit exceeds,
   3.1.1 System will show a warning message.

Postconditions: The "Upload ProductPost" button will be disabled.

Alternative flow:

4.3.1 If the image size exceeds,
   4.3.1.1 System will show a warning message.

Postconditions: The image will be removed.

Alternative flow:

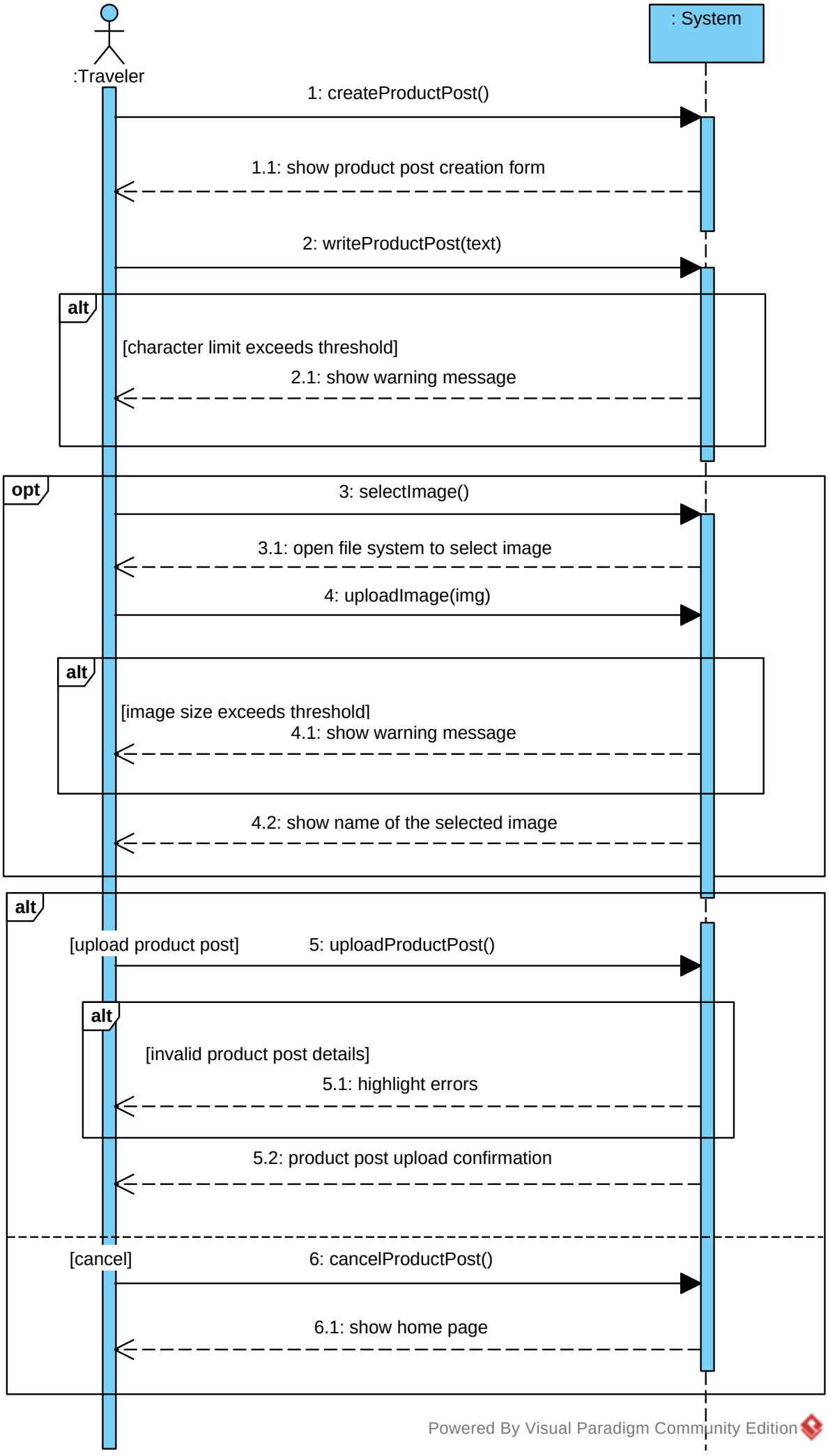
5.2.2 If the productPost details are not verified,
   5.2.2.1 the system will highlight the errors in the same product information form.

Postconditions: The productPost will not be recorded.

Alternative flow:

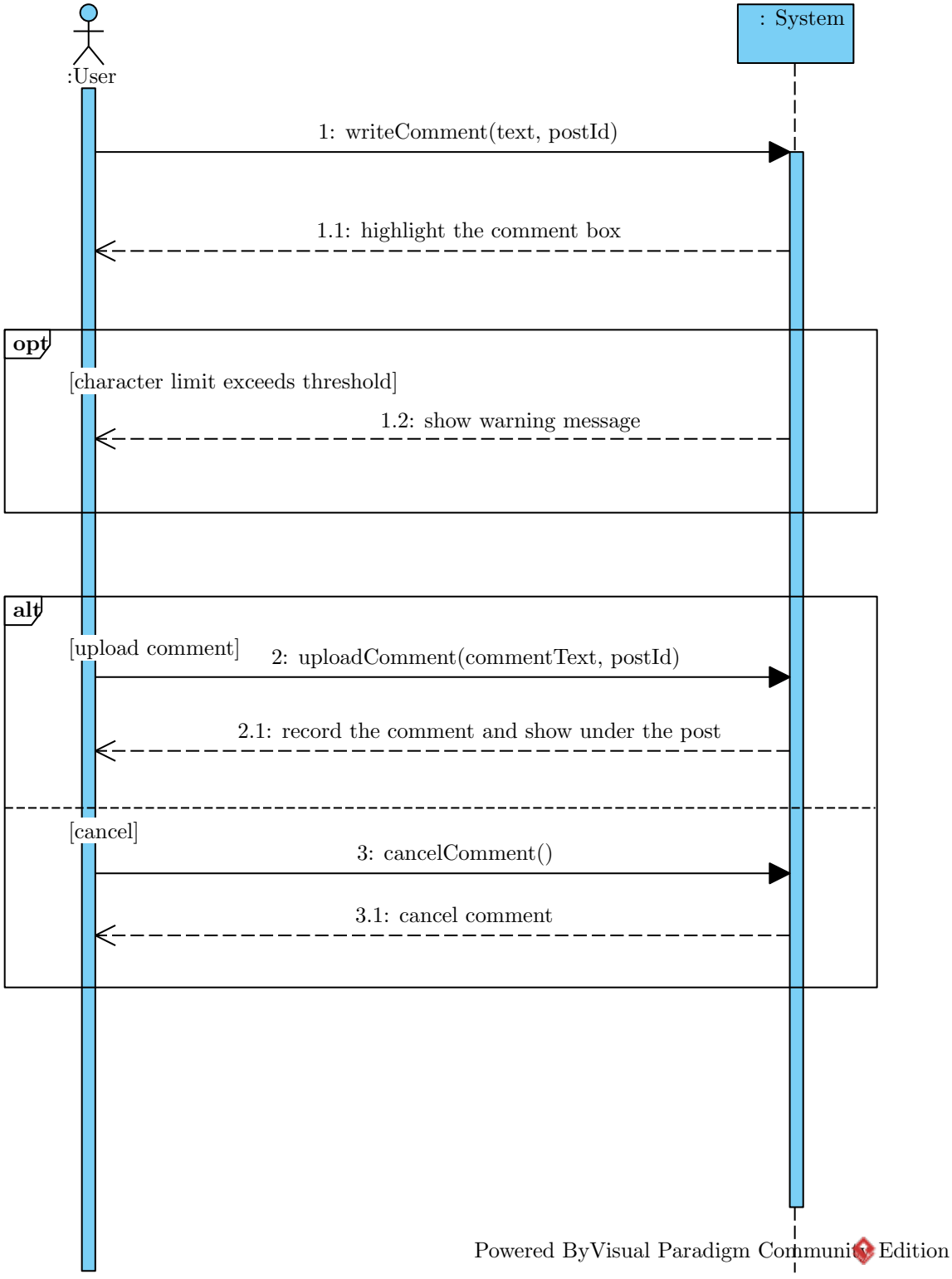
5.3 If the sender clicks the "Cancel" button,
   5.3.1 the system will discard the productPost and take the user to the homepage.

Postconditions: The homepage will remain the same.



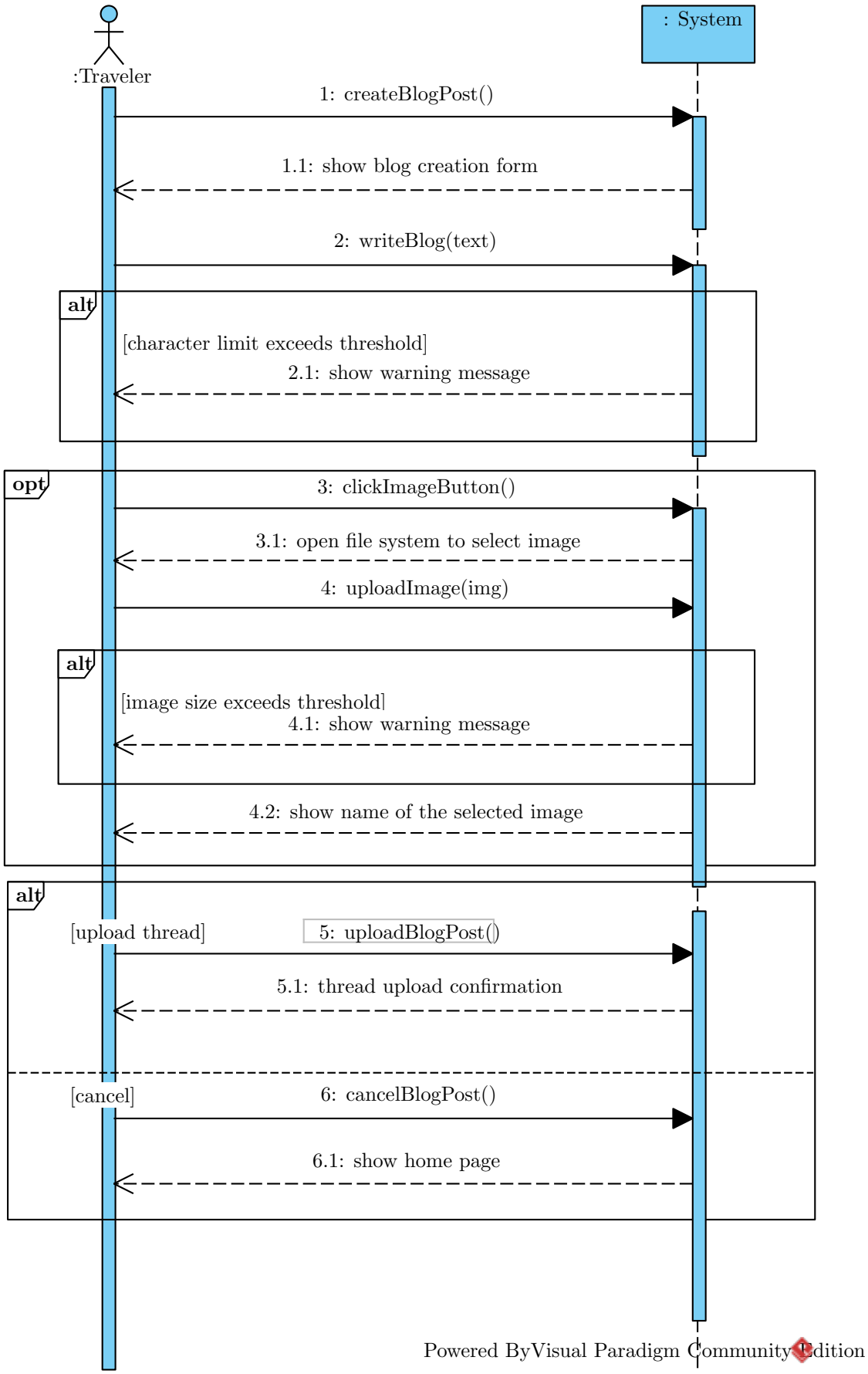
2.5 Use Case: Comment of ProductPost

Use Case: Comment of ProductPost
ID: UC05
Actors: 1. User.
Preconditions: 1. The user is authenticated.
Flow of Events: 1. The use case starts after the user starts writing his/her comment on the comment box below a product post or travel blog post. 2. System will highlight the comment box. 3. System will check the text limit for the comment. 4. User will select 'Comment'
Postconditions: The comment will be visible under the post
Alternative flow: 3.1 If the character limit exceeds, 3.1.1 System will show a warning message.
Postconditions: Comment box will be marked red.
Alternative flow: 4.1 If the customer select 'Cancel' 4.1.1 System will cancel the comment.
Postconditions: Comment text box will be emptied


:User
: System
1: writeComment(text, postId)
1.1: highlight the comment box
opt
[character limit exceeds threshold]
1.2: show warning message
alt
[upload comment]
2: uploadComment(commentText, postId)
2.1: record the comment and show under the post
[cancel]
3: cancelComment()
3.1: cancel comment
Powered ByVisual Paradigm Community Edition

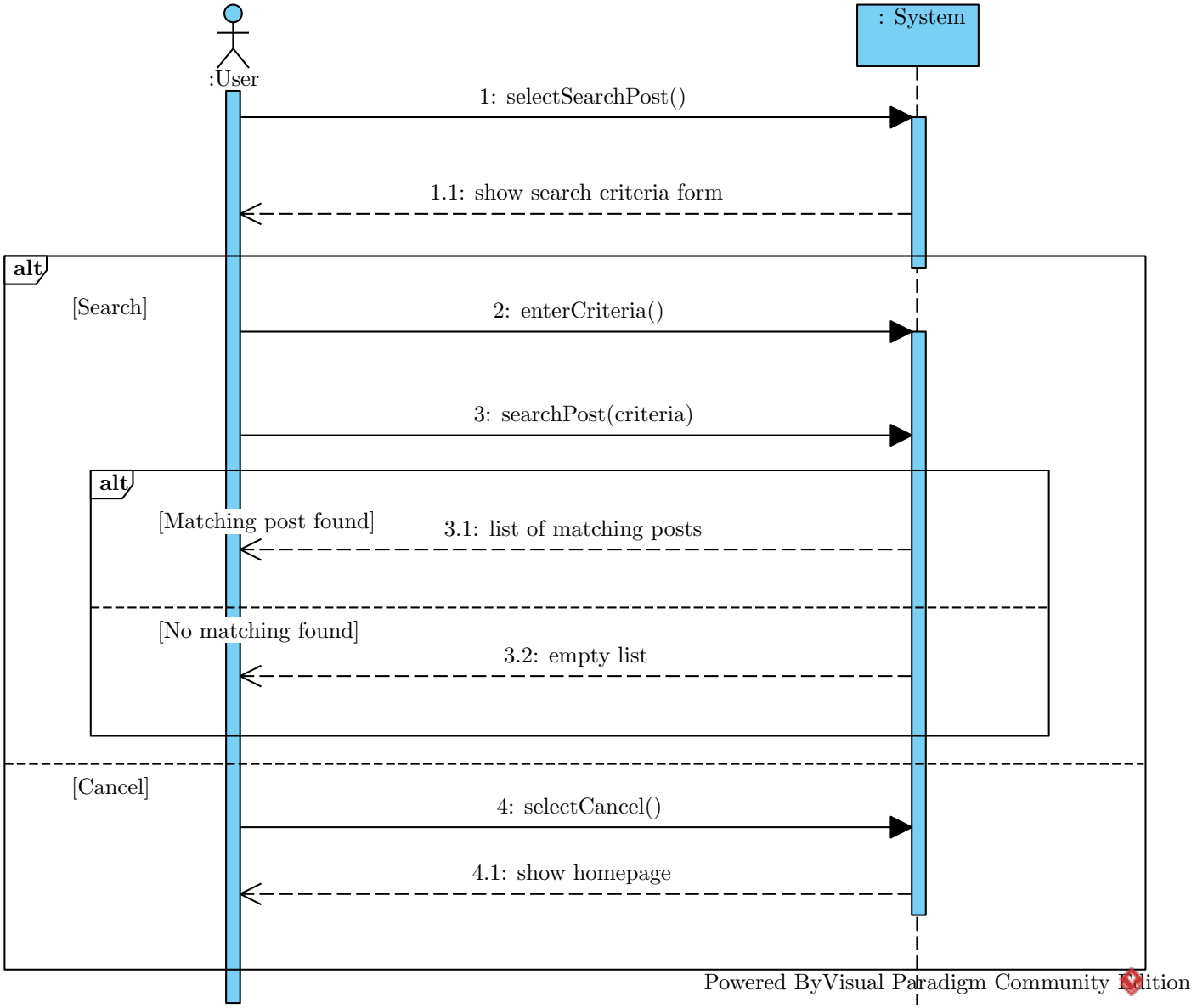
2.6 Use Case: Create BlogPost

Use Case: Create BlogPost
ID: UC06
Actors: 1. Traveler
Preconditions: 1. The traveler is authenticated.
Flow of events: 1. The use case starts when a travelr selects "Create Blog". 1.1. The system will return a form for blog creation along with "Upload Image", "Upload Blog", and "Cancel" options. 2. The traveler will fill up the form with text and images. 3. System will check the character limit. 4. If the sender selects "Upload Image", 4.1. The system will return a dialog to select image(s). 4.2. The traveler will select image files from his local disk. 4.3. System will validate the image size. 4.4. The system will show the names of selected images. 5. If the traveler selects "Upload Blog", 5.1. The system will record the blog.
Postconditions: The blog will be visible on the homepage.
Alternative flow: 3.1 If the character limit exceeds, 3.1.1 System will show a warning message.
Postconditions: The "Upload Blog" button will be disabled.
Alternative flow: 4.3.1 If the image size exceeds, 4.3.1.1 System will show a warning message.
Postconditions: The image will be removed.
Alternative flow: 5.2 If the sender clicks the "Cancel" button, 5.2.1 the system will discard the blog and take the user to the homepage.
Postconditions: The homepage will remain the same.


:Traveler
: System
1: createBlogPost()
1.1: show blog creation form
2: writeBlog(text)
alt
[character limit exceeds threshold]
2.1: show warning message
opt
3: clickImageButton()
3.1: open file system to select image
4: uploadImage(img)
alt
[image size exceeds threshold]
4.1: show warning message
4.2: show name of the selected image
alt
[upload thread]
5: uploadBlogPost()
5.1: thread upload confirmation
[cancel]
6: cancelBlogPost()
6.1: show home page
Powered ByVisual Paradigm Community Edition

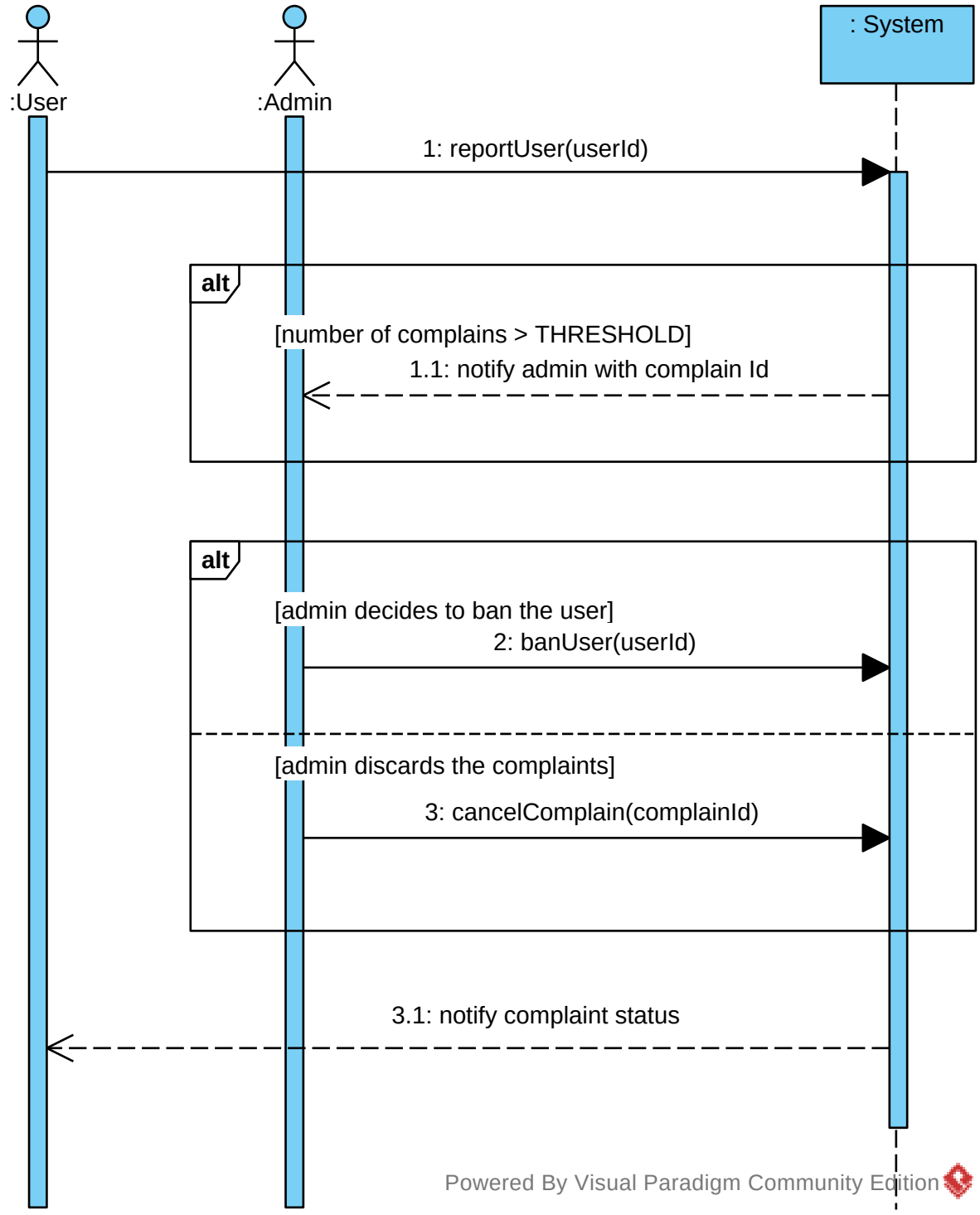
2.7 Use Case: Search ProductPost

Use Case: Search ProductPost
ID: UC07
Actors: 1. User
Preconditions: The user is authenticated.
Flow of Events: 1. The use case starts when a user selects “Search ProductPost”. 2. The system will ask the user for search criteria. 3. The user enters the requested criteria. 4. The system searches for productPosts that match the users’ criteria. 5. If the system finds some matching productPosts then, 5.1 The system displays a list of matching productPosts.
Postconditions: The user will be redirected to the home page.
Alternative flow: 2.1 If the user selects “Cancel” option, 2.1.1 The system will redirect the user to the homepage.
Postconditions: The user will see the homepage.
Alternative flow: 5.2 If the system cannot find any matching productPost 5.2.1 The system will tell that no productPost was found.
Postconditions: The user will see an empty list.


:User
: System
1: selectSearchPost()
1.1: show search criteria form
alt
[Search]
2: enterCriteria()
3: searchPost(criteria)
alt
[Matching post found]
3.1: list of matching posts
[No matching found]
3.2: empty list
[Cancel]
4: selectCancel()
4.1: show homepage
Powered ByVisual Paradigm Community Edition

2.8 Use Case: Ban User

Use Case: Ban User
ID: UC08
Actors: 1. User 2. Admin
Preconditions: 1. The user is identified and authenticated.
Flow of Events: 1. The use case starts when a user reports another user. 2. If the number of complaints is greater than a threshold value, 2.1. System will notify the admin to review the complaints. 2.2. Admin will review the complaints. 2.3. Admin will ban the user.
Postconditions: The banned user will not be able to use our system anymore.
Alternative flow: 2.4 Admin may discard all the complaints against that user.
Postconditions: Does not apply.


:User
:Admin
: System
1: reportUser(userId)
alt
[number of complains > THRESHOLD]
1.1: notify admin with complain Id
alt
[admin decides to ban the user]
2: banUser(userId)
[admin discards the complaints]
3: cancelComplain(complainId)
3.1: notify complaint status
Powered By Visual Paradigm Community Edition

2.9 Use Case: Update ProductPost

Use Case: Update ProductPost

ID: UC09

Actors:

1. Sender

Preconditions:

1. There exists a ProductPost of the sender about the details of the products sender wants to send.

Flow of Events:

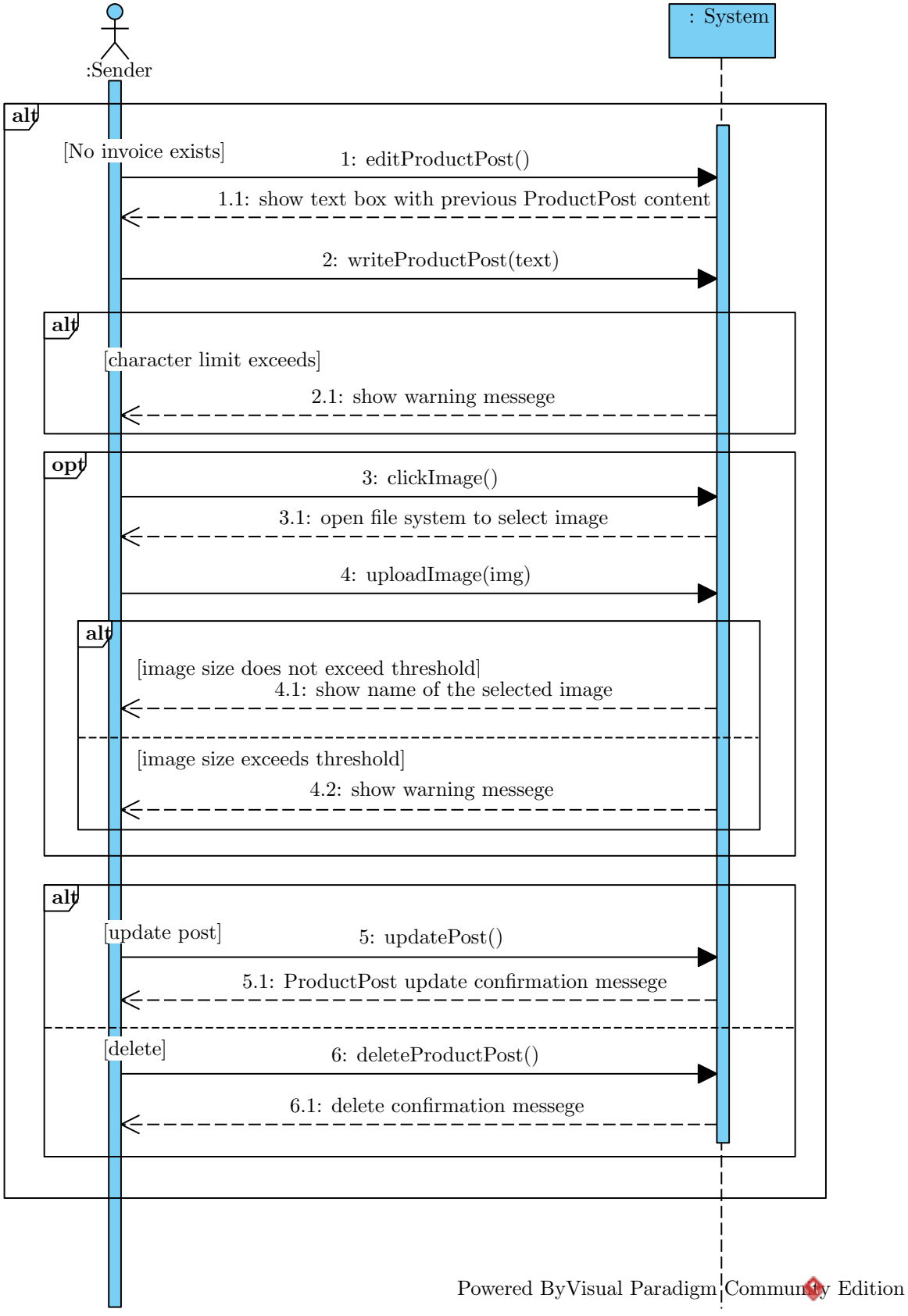
1. If no invoice has been made or traveler hasn't picked up the products yet,
   1.1. Sender will be able to click the edit button.
   1.2. The system will return a window containing previous texts, details, and image files of the ProductPost.
   1.3. Sender can change any detail.
2. System will check the character limit.
3. If the sender selects "Upload Image",
   3.1. The system will return a dialog box to select image(s).
   3.2. The traveler will select image files from his local disk.
   3.3. System will validate the image size.
   3.4. The system will show the names of selected images.
4. If the sender clicks 'Update',
   4.1. System will update the post with the new information.
5. If the user clicks 'delete',
   5.1. System will delete the ProductPost and it will not be in the homepage anymore.

Postconditions: System will show the updated post on the homepage.

Alternative flow:

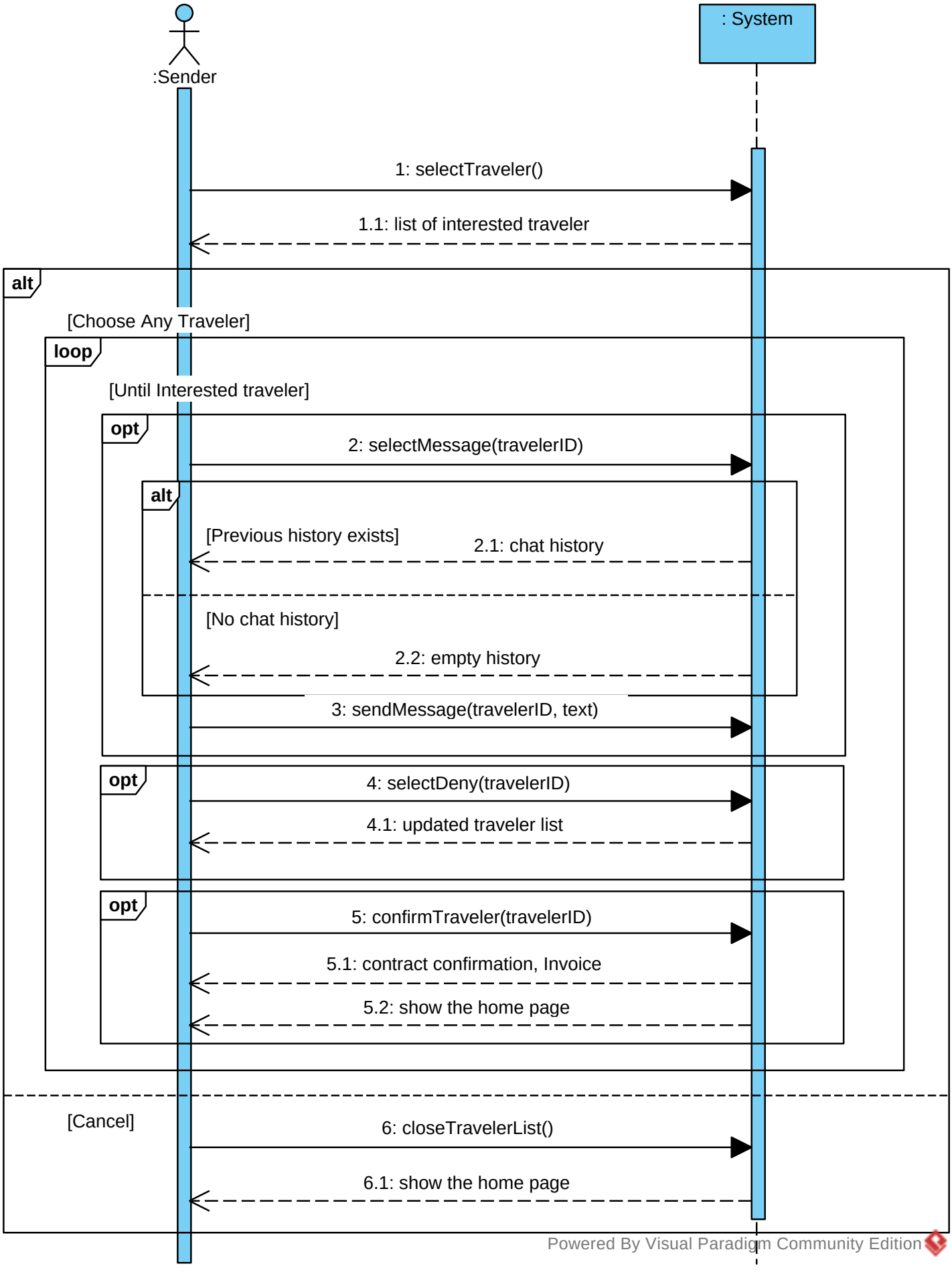
1.4 If an invoice has been made or a traveler has picked up the products,
   1.4.1 The edit button will be disabled.

Postconditions: Does not apply.


:Sender
: System
alt
[No invoice exists]
1: editProductPost()
1.1: show text box with previous ProductPost content
2: writeProductPost(text)
alt
[character limit exceeds]
2.1: show warning messege
opt
3: clickImage()
3.1: open file system to select image
4: uploadImage(img)
alt
[image size does not exceed threshold]
4.1: show name of the selected image
[image size exceeds threshold]
4.2: show warning messege
alt
[update post]
5: updatePost()
5.1: ProductPost update confirmation messege
[delete]
6: deleteProductPost()
6.1: delete confirmation messege
Powered ByVisual Paradigm Community Edition

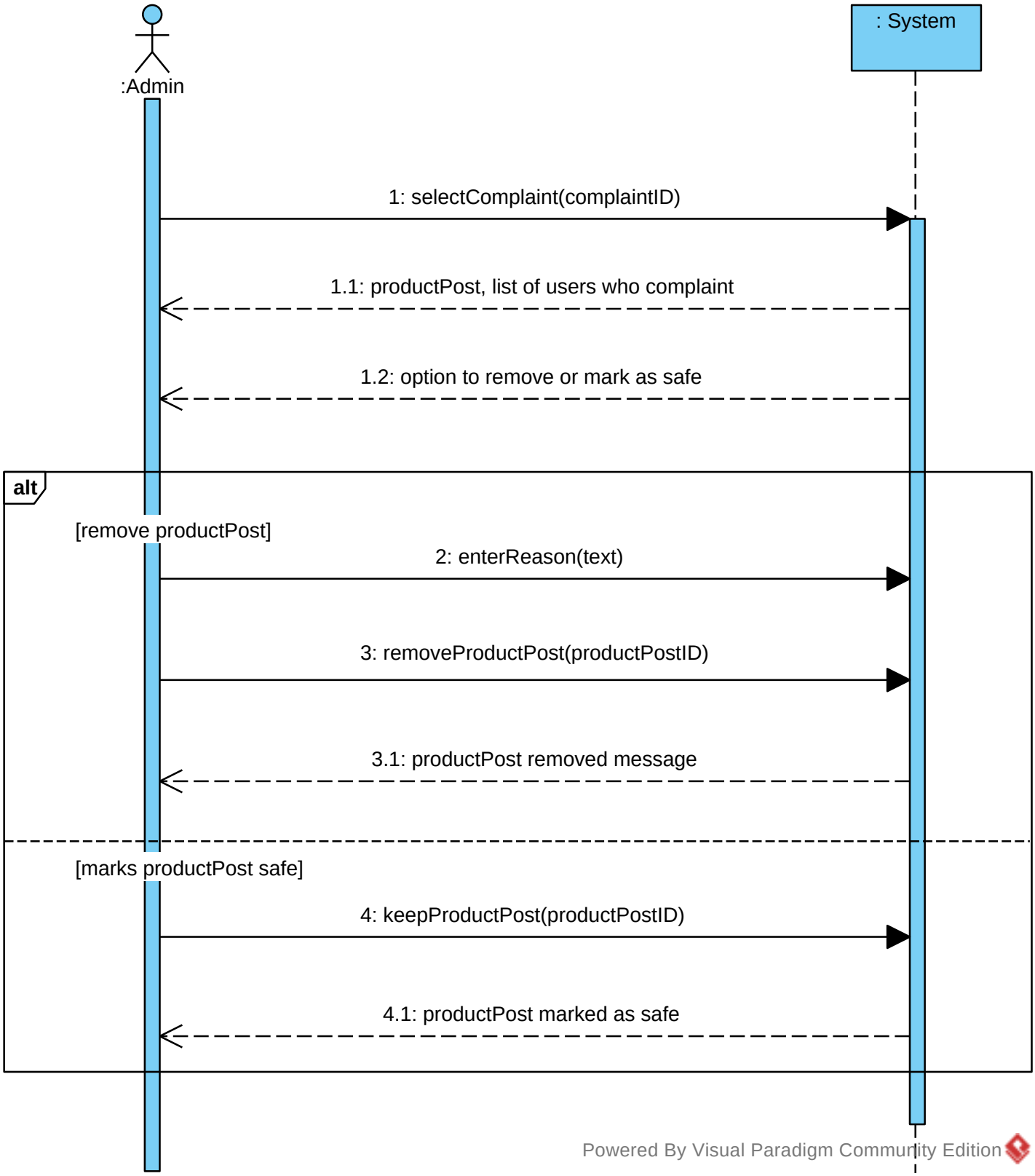
2.10 Use Case: Contract Creation

Use Case: Contract Creation
ID: UC10
Actors: 1. Sender
Preconditions: 1. Sender has some pending requests of the interested Traveler to carry his/her product.
Flow of Events: 1. The use case starts after the Sender Selects the "Select Traveler" option. 2. The sender can choose to select the "Cancel" option ignoring all available travelers. 3. The system returns a list of interested users who agreed to deliver the product within the asked date with "Message", "Deny", and "Confirm" options beside each interested Traveler and a "Cancel" option at the end. 4. If the Sender selects the "Chat" option for having a conversation about the price for delivery and payment method (Users choose their own payment method) beside any user, 4.1. If there exists previous chat history between the Sender and the Traveler, 4.1.1. The conversation will be displayed with a text dialog and a "Send" option. 5. If the conversation went right (they both agreed about delivery price), 5.1. The sender should select the "Confirm" option. 5.2. The system will show the sender his/her home page. 6. The system will add a tag saying "Traveler Confirmed" in the top of the productPost. 7. The system will return an invoice to Traveler.
Postconditions: The Traveler will receive a downloadable invoice document.
Alternative flow: 3.1 If there does not exist any previous chat history between the Sender and the Traveler,, 3.1.1 the system will show an empty chat history with a text dialog and a "Send" option.
Postconditions: Does not apply.
Alternative flow: 4.2 If the did not go right (they both did not agree on the delivery price), 4.2.1 the sender will select the "Deny" option.
Postconditions: Does not apply.


:Sender
: System
1: selectTraveler()
1.1: list of interested traveler
alt
[Choose Any Traveler]
loop
[Until Interested traveler]
opt
2: selectMessage(travelerID)
alt
[Previous history exists]
2.1: chat history
[No chat history]
2.2: empty history
3: sendMessage(travelerID, text)
opt
4: selectDeny(travelerID)
4.1: updated traveler list
opt
5: confirmTraveler(travelerID)
5.1: contract confirmation, Invoice
5.2: show the home page
[Cancel]
6: closeTravelerList()
6.1: show the home page
Powered By Visual Paradigm Community Edition

2.11 Use Case: Moderate ProductPost

Use Case: Moderate ProductPost
ID: UC11
Actors: 1. Admin
Preconditions: 1. The use case starts when the number of complaints for a productPost is greater than the threshold.
Flow of Events: 1. The use case starts when number of complaints for a productPost is greater than the threshold. 2. If the admin selects the productPost complaint, 2.1 System will show the number of complaints by the users. 2.2 System will show the list of users who complaint. 2.3 System will show the option to remove or keep the ProductPost to the traveler. 3. If the admin removes the ProductPost, 3.1 System will show a form to input the reason for the removal. 3.2 Admin inputs the reason. 3.3 System will record the productPost as removed.
Postconditions: The productPost will be removed from the system.
Alternative flow: 4. If the admin keeps the ProductPost, 4.1 System will not show the productPost in the complaint list. 4.1 System will record the productPost as safe.
Postconditions: The productPost will be removed from the complaints list.


:Admin
: System
1: selectComplaint(complaintID)
1.1: productPost, list of users who complaint
1.2: option to remove or mark as safe
alt
[remove productPost]
2: enterReason(text)
3: removeProductPost(productPostID)
3.1: productPost removed message
[marks productPost safe]
4: keepProductPost(productPostID)
4.1: productPost marked as safe
Powered By Visual Paradigm Community Edition

2.12 Use Case: Send Notification

Use Case: Send Notification

ID: UC12

Actors:

1. Traveler
2. Sender
3. Admin

Preconditions:

1. Sender and traveler are registered in the system.

Flow of Events:

1. The use case starts when users are logged into the system.
2. If the traveler or sends a message.
   - 2.1 System records a notification.
   - 2.2 System will increase the notification counter.
   - 2.3 System will display the notification of one new message to the sender or traveler.
3. If a ProductPost is reported by the traveler,
   - 3.1 If the report count is greater than the threshold,
     - 3.1.1 System records a notification.
     - 3.1.2 System will increase the notification counter.
     - 3.1.3 System will display the notification of one reported productPost to the admin.
4. If a contract is signed by the traveler,
   - 4.1 System records a notification.
   - 4.2 System will increase the notification counter.
   - 4.3 System will display the notification of contract finalization to the sender.
5. If the traveler or sender selects the notification,
   - 5.1 System will display the details of the notification.
   - 5.2 System will decrease the number of notifications for the user.

Postconditions: Traveler or sender will see a notification with the number of notification and will be able to see the details of the notification..

Alternative flow:

2.4 If the sender or traveler is not logged into the system,

- 2.4.1 System will not display the notification.

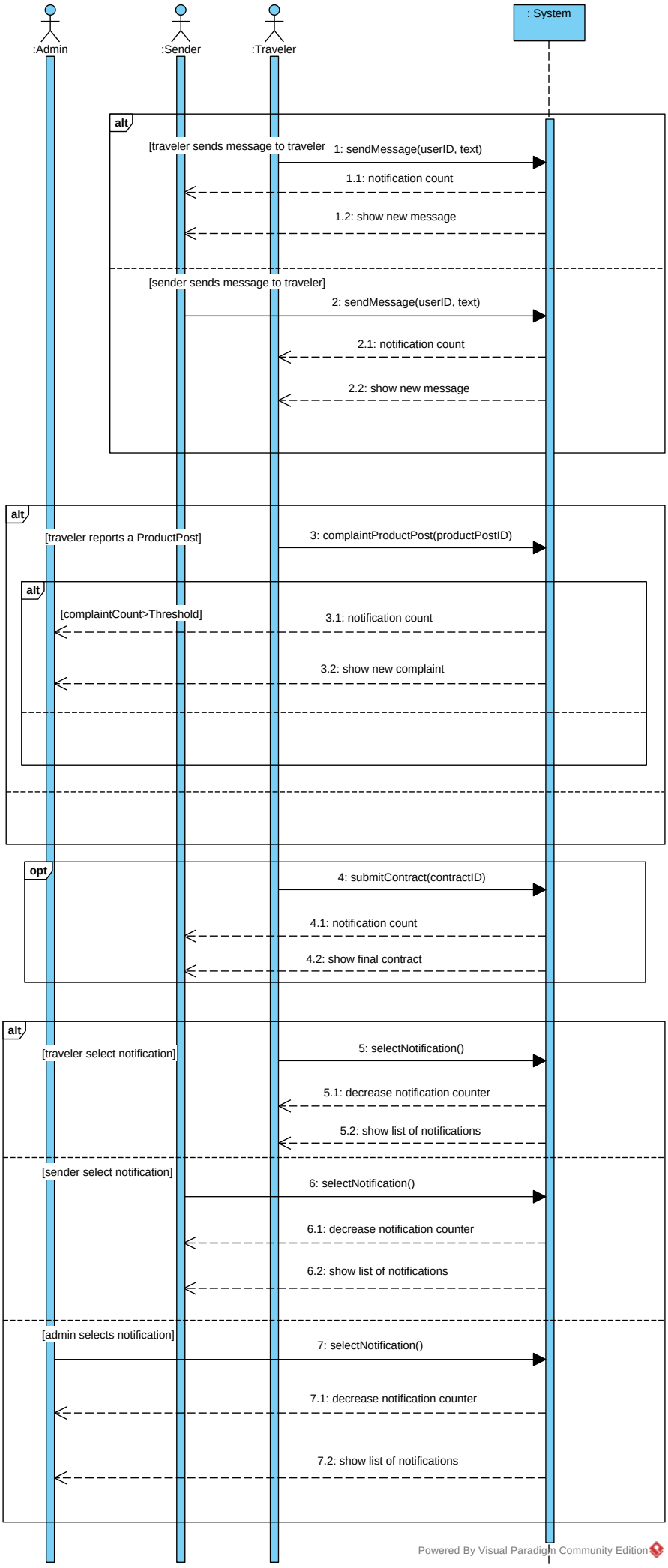
Postconditions: Sender or traveler will see the notification after logging into the system..

Alternative flow:

5.3 If the number of notifications is zero,

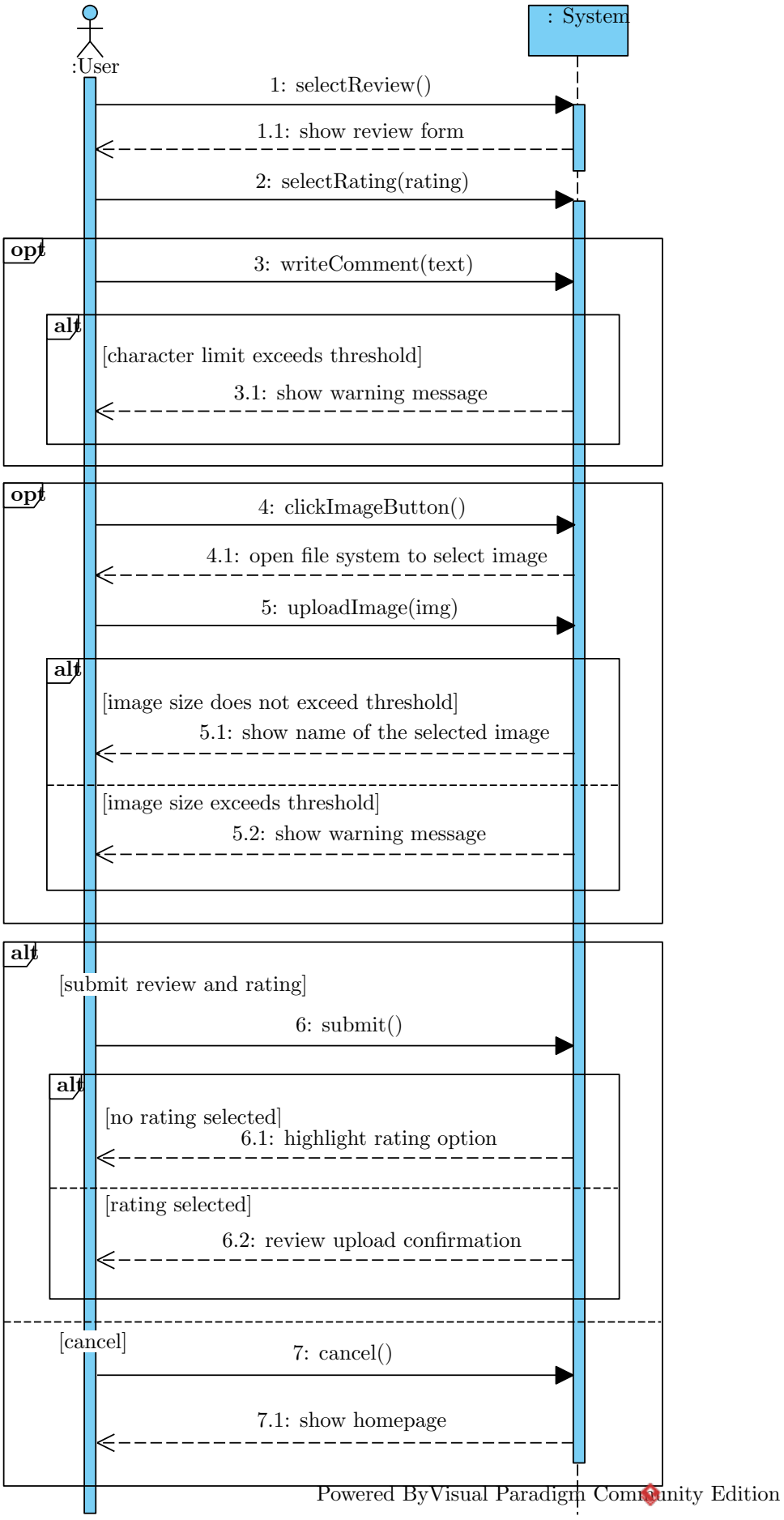
- 5.3.1 System will show that, no new notification found.

Postconditions: System will display an empty notification list.



2.13 Use Case: Review and Rating

Use Case: Review and Rating
ID: UC13
Actors: 1. User
Preconditions: 1. Current date exceeds the contract end date.
Flow of Events: 1. The use case starts when the user selects "Review". 1.1. The system will return a form for writing the review along with the rating (1 to 5) option, the "Upload Image", "Cancel", and the "Submit" options. 2. The user will fill up the rating option with any rating between 1 to 5. 3. The user will fill up the form with comments. 4. System will check the character limit. 5. If the user selects "Upload Image", 5.1. The system will return a dialog to select image(s). 5.2. The reviewer will select image files from his local disk. 5.3. System will validate the image size 5.4. The system will show the names of selected images. 6. If the user selects the "Submit" button, 6.1. The system will verify the rating details (at least a rating between 1 to 5 has to be given). 6.2. If the review and rating details are verified, 6.2.1. The system will record that review (may be empty) and rating (has to be given).
Postconditions: The overall rating of the user will be updated.
Alternative flow: 4.1 If the character limit exceeds 4.1.1 System will show a warning message.
Postconditions: The "Submit" option will be disabled.
Alternative flow: 5.3.1 If the image size exceeds 5.3.1.1 System will show a warning message.
Postconditions: System will remove the image.
Alternative flow: 6.2.2 If the reviewer does not give a rating, 6.2.2.1 the system will highlight the rating option.
Postconditions: The user will see that the review is uploaded.
Alternative flow: 6.3 If the user selects "Cancel", 6.3.1 the system will close the form.
Postconditions: The user will be redirected to the homepage.


:User
: System
1: selectReview()
1.1: show review form
2: selectRating(rating)
opt
3: writeComment(text)
alt
[character limit exceeds threshold]
3.1: show warning message
opt
4: clickImageButton()
4.1: open file system to select image
5: uploadImage(img)
alt
[image size does not exceed threshold]
5.1: show name of the selected image
[image size exceeds threshold]
5.2: show warning message
alt
[submit review and rating]
6: submit()
alt
[no rating selected]
6.1: highlight rating option
[rating selected]
6.2: review upload confirmation
[cancel]
7: cancel()
7.1: show homepage
Powered ByVisual Paradigm Community Edition

2.14 Use Case: Peer-to-peer Chat

Use Case: Peer-to-peer Chat

ID: UC14

Actors:

1. User

Preconditions:

1. The user is authenticated.

Flow of Events:

1. The use case starts after the user selects the "Chat" option.
2. The system asks the user for a name (name of the person that the user is looking for).
3. The user enters the user's name.
4. The system searches for users that match the name.
5. If the system finds some matching users then,
   5.1. The system shows a list of matched users with a "Message" option beside each of these users' names.
6. If the user selects the "Message" option,
   6.1. If there exists previous chat history between them,
      6.1.1. The conversation will be displayed with a text dialog and a "Send" option.
7. If the user writes something in the text dialog and selects "Send",
   7.1. System will send the text to the other user.

Postconditions: The user will see that the conversation between them is updated with the latest text of the user.

Alternative flow:

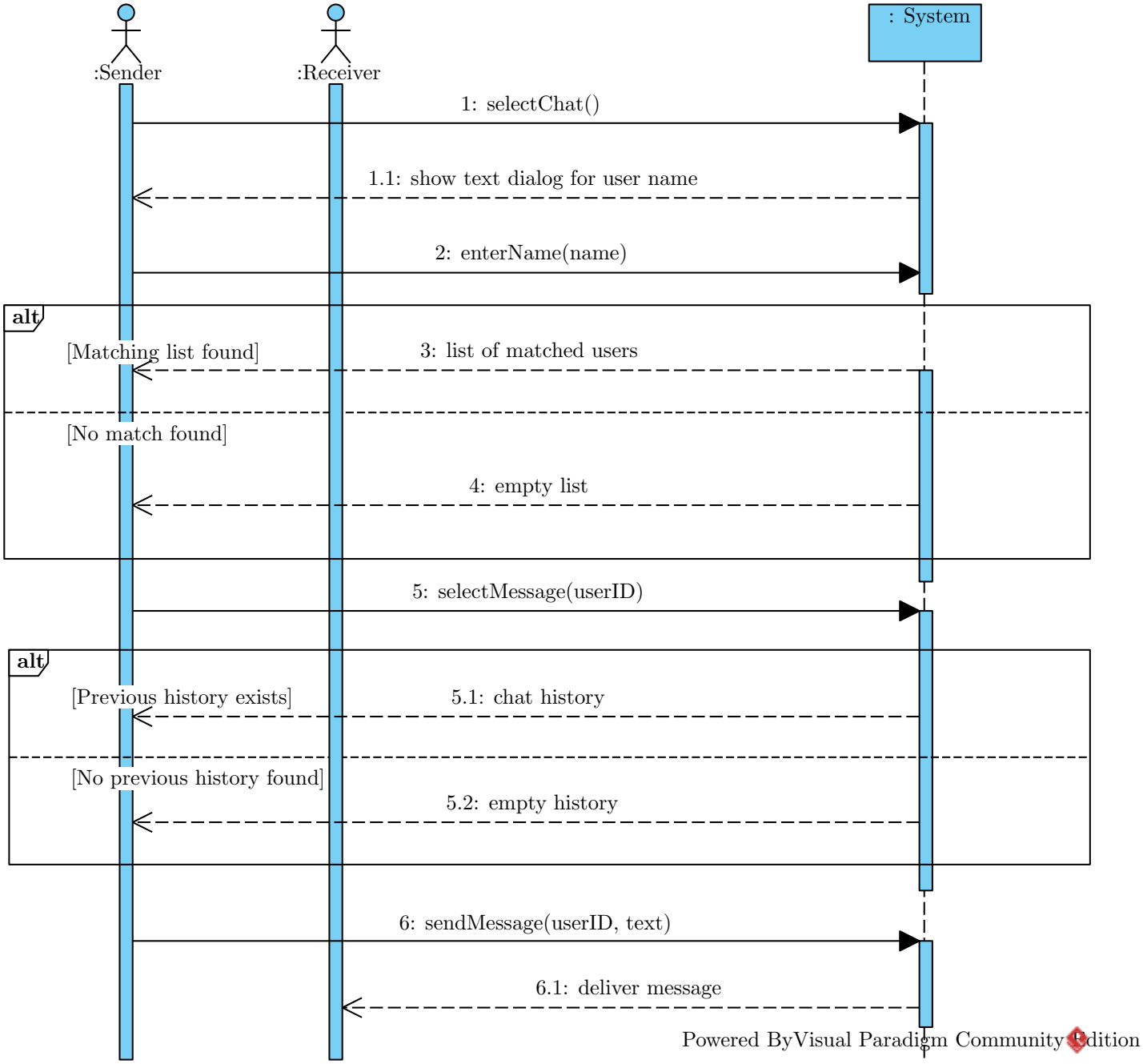
5.2 If the system cannot find any matching user,
   5.2.1 the system will tell that no matching user could be found.

Postconditions: The user will see an empty list.

Alternative flow:

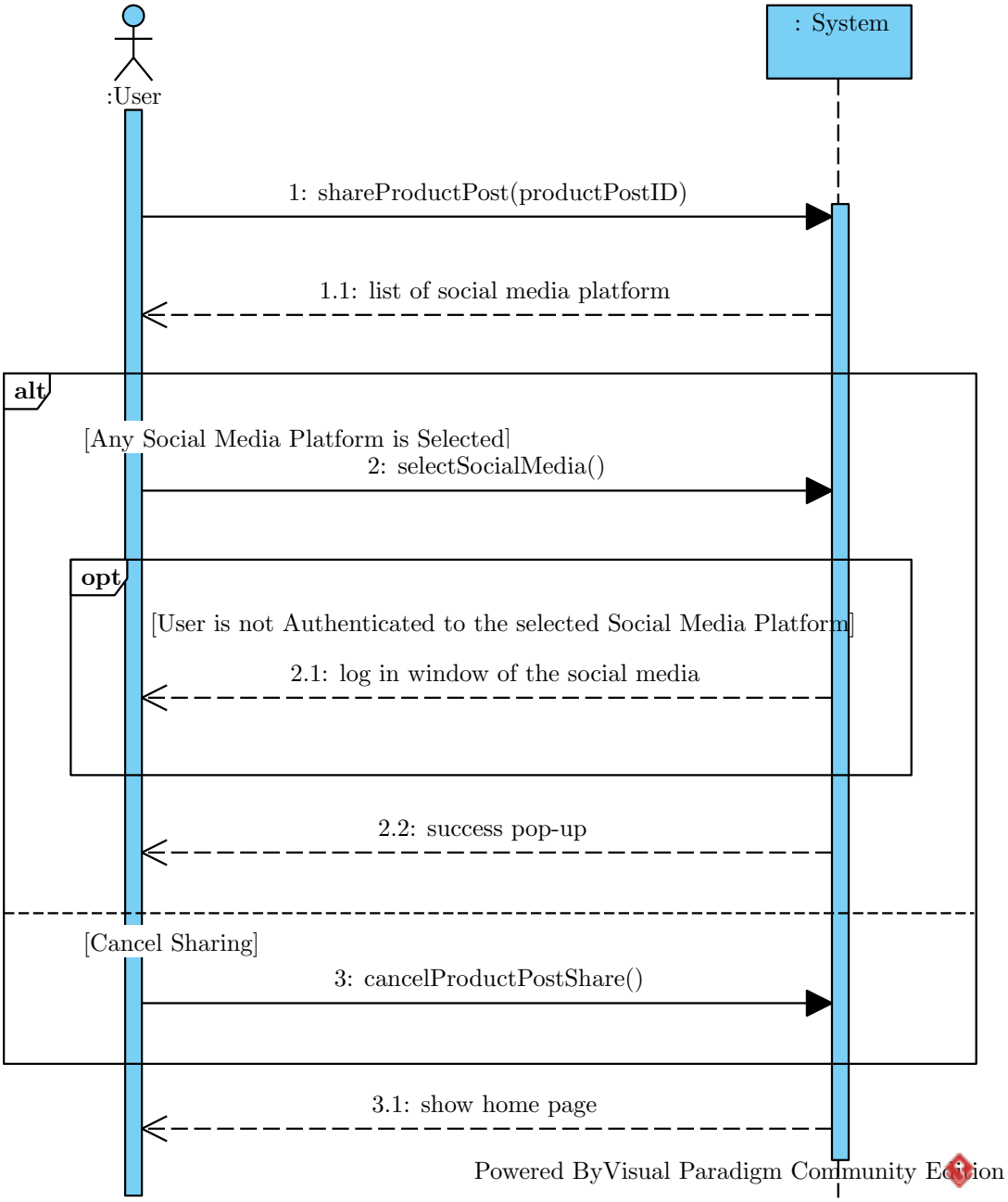
6.1.2 If there is no previous chat history between the users,
   6.1.2.1 the system will only show a text dialog and a "Send" option.

Postconditions: The user will see a conversation with only the recently sent text message.

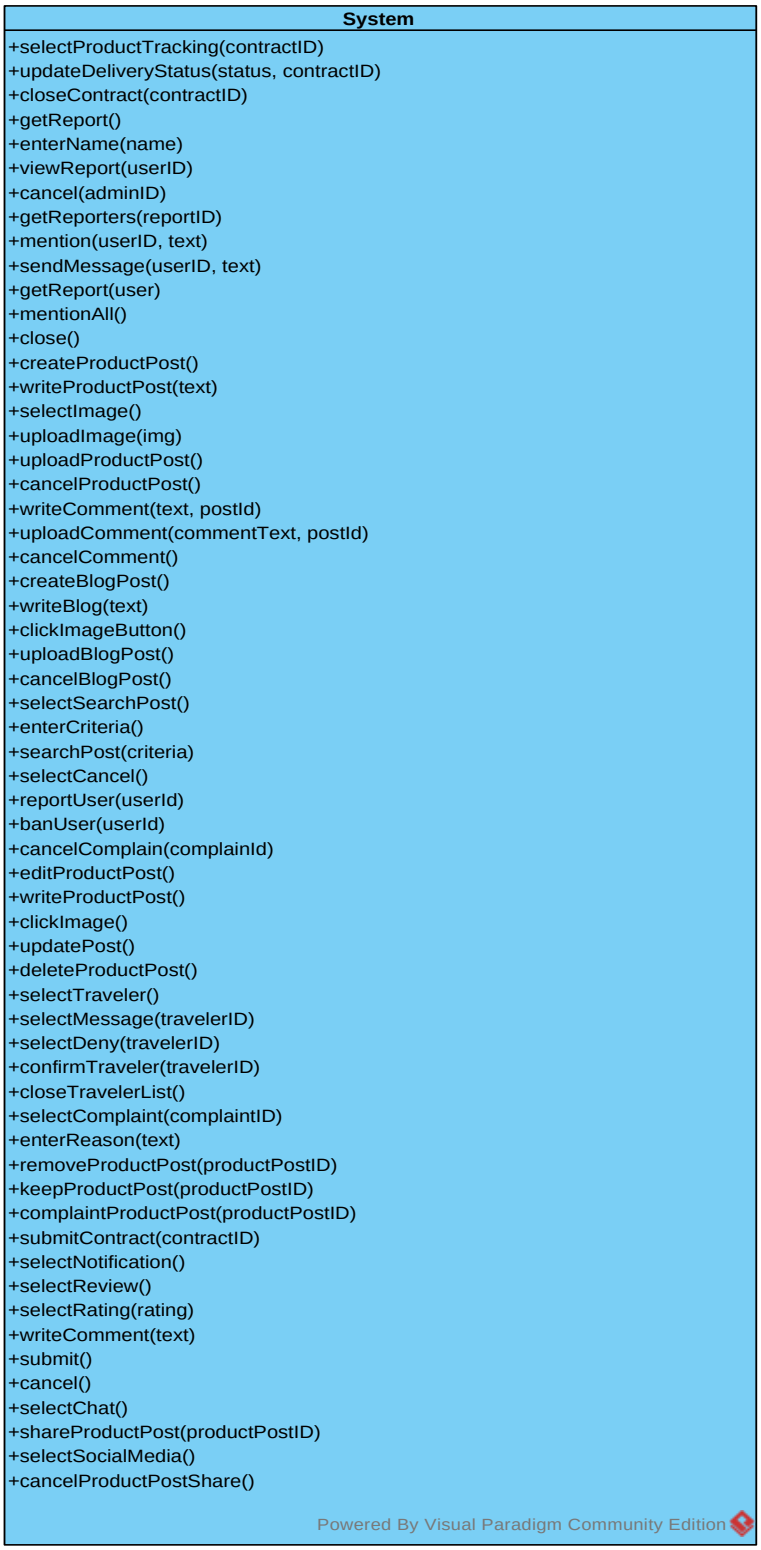

:Sender
:Receiver
: System
1: selectChat()
1.1: show text dialog for user name
2: enterName(name)
alt
[Matching list found]
3: list of matched users
[No match found]
4: empty list
5: selectMessage(userID)
alt
[Previous history exists]
5.1: chat history
[No previous history found]
5.2: empty history
6: sendMessage(userID, text)
6.1: deliver message
Powered ByVisual Paradigm Community Edition

2.15 Use Case: Social Media Share

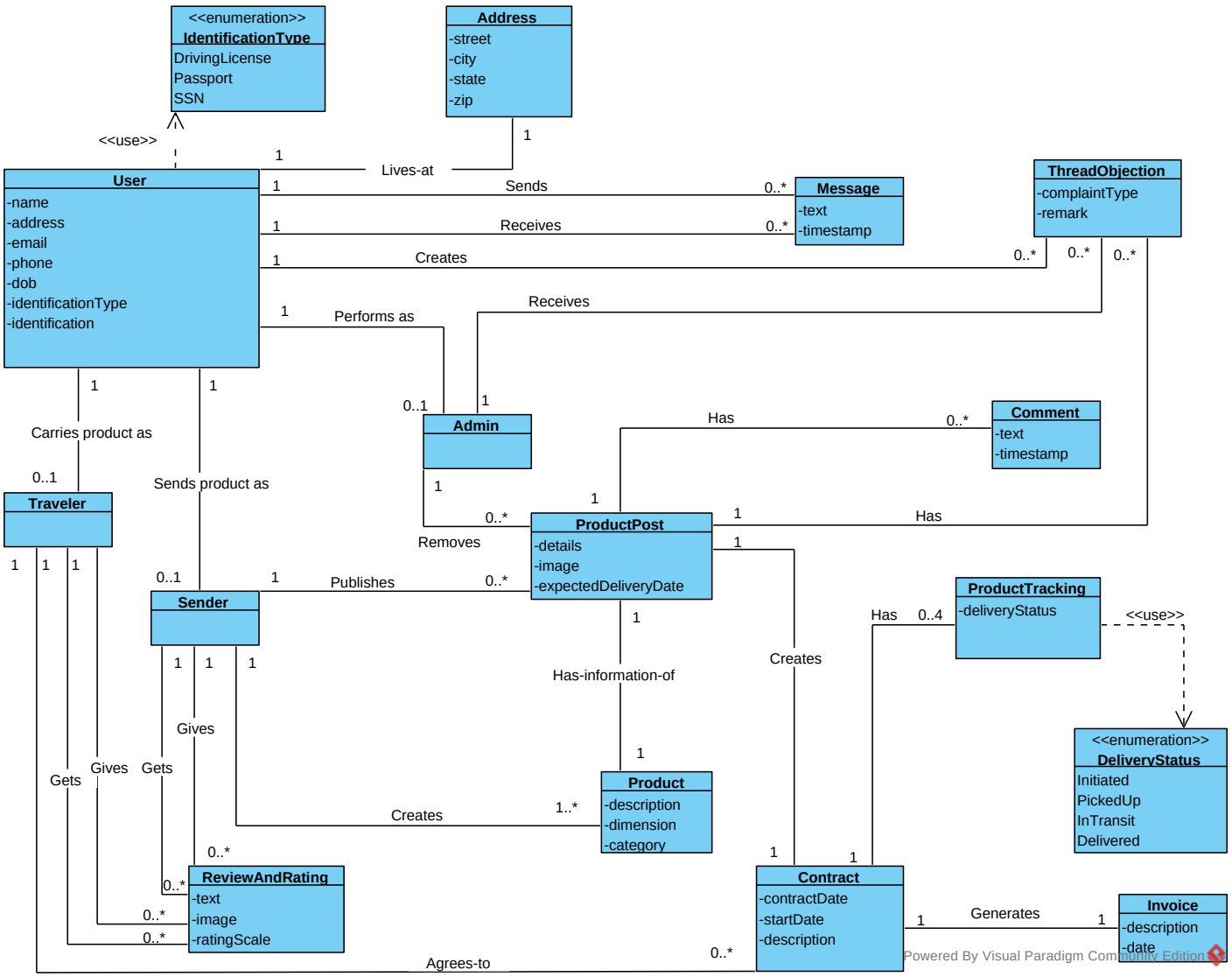
Use Case: Social Media Share
ID: UC15
Actors: 1. User.
Preconditions: 1. There exists at least one productPost and user is authenticated.
Flow of Events: 1. The use case starts after any user selects the "Share ProductPost" option under any particular productPost. 2. The system shows a window to the user which will contain a list of social media. 3. The user will select one of the social media. 4. If the system is able to share the productPost to that social media platform, 4.1. The productPost will be shared to that social media platform.
Postconditions: The system will show a pop-up that the productPost has been shared.
Alternative flow: 3.1 If the user selects "Cancel" option, 3.1.1 The system will close the window of the social media platform's list.
Postconditions: The user will be able to see the homepage.
Alternative flow: 4.2 If the system is not able to share that productPost to that social media platform, 4.2.1 The system will open a new browser tab for the selected social media platform. 4.2.2 When the user will log in to that platform, the productPost will be shared.
Postconditions: Does not apply.


:User
: System
1: shareProductPost(productPostID)
1.1: list of social media platform
alt
[Any Social Media Platform is Selected]
2: selectSocialMedia()
opt
[User is not Authenticated to the selected Social Media Platform]
2.1: log in window of the social media
2.2: success pop-up
[Cancel Sharing]
3: cancelProductPostShare()
3.1: show home page
Powered ByVisual Paradigm Community Edition

3 System Operation



4 Domain Model



5 Operation Contracts

Contract CO1: selectProductTracking

Operation: selectProductTracking(contractID: ContractID)
Cross References: Use Cases: Product Tracking
Pre-conditions: There exists a contract between traveler and sender.
Post-conditions:

- A ProductStatus instance *status* was created.
- *status* is associated with ProductStatus.
- *contractID* was associated with Contract.
- *contractID* was associated User.
- *contractID* was associated with ProductStatus.

Contract CO2: updateDeliveryStatus

Operation: updateDeliveryStatus(status: ProductStatus, contractID: ContractID)
Cross References: Use Cases: Product Tracking
Pre-conditions: There exists a contract between traveler and sender.
Post-conditions:

- *contractID* was associated with Contract.
- *contractID* was associated with ProductStatus.
- *status.status* was set to status.

Contract CO3: closeContract

Operation: closeContract(contractID: ContractID)
Cross References: Use Cases: Product Tracking
Pre-conditions: There exists a contract between traveler and sender.
Post-conditions:

- *contractID* was associated with Contract.
- *contract.state* was set to a String "Closed".

Contract CO4: getReport

Operation: getReport()
Cross References: Use Cases: Report of User
Pre-conditions: Admin is authenticated and logged in to the system.
Post-conditions:

- A Report instance *rep* was created.
- *rep* was associated with the User.
- *rep* was associated with the Admin.

Contract CO5: enterName

Operation: enterName(name: String)
Cross References: Use Cases: Report of User
Pre-conditions: Admin is authenticated and logged in to the system.
Post-conditions:

- A Name instance n was created.
- n was associated with the User.
- *n.name* was set to name.

Contract CO6: viewReport

Operation: viewReport(userID: UserID)
Cross References: Use Cases: Report of User
Pre-conditions: Admin is authenticated and logged in to the system.
Post-conditions:

- *rep* was associated with the User, based on userID match.
- *rep* was associated with the Admin.

Contract CO7: cancel

Operation: cancel()
Cross References: Use Cases: Report of User
Pre-conditions: Admin is authenticated and logged in to the system.
Post-conditions:

- *adminID* was associated with Home.
- *homeID* was associated with Admin.

Contract CO8: getReporters

Operation: getReporters(reportID: ReportID)
Cross References: Use Cases: Report Review Confirming
Pre-conditions: Admin is authenticated and logged in to the system.
Post-conditions:

- *reportID* was associated with Report.
- *reportID* was associated with User.
- *userID* was associated with Report.
- A list of user was associated with Admin.

Contract CO9: mention

Operation: mention(userID: UserID, text: String)
Cross References: Use Cases: Report Review Confirming
Pre-conditions: Admin is authenticated and logged in to the system.
Post-conditions:

- A Notification instance *not* was created.
- *not.text* was set to text.
- *not* was associated with Admin.
- *not* was associated with a User, based on userID match.

Contract CO10: mentionAll

Operation: mentionAll()
Cross References: Use Cases: Report Review Confirming
Pre-conditions: Admin is authenticated and logged in to the system.
Post-conditions:

- A Notification instance *not* was created.
- *not.text* was set to text.
- *not* was associated with Admin.
- *not* was associated with a list of User instances user, based on user matching ReportID.

Contract C11: uploadProductPost

Operation: uploadProductPost()
Cross References: Use Cases: Create ProductPost
Pre-conditions: Sender is registered and logged in to the system.
Post-conditions:

- A *ProductPost* instance *productPost* will be created.
- *productPost* will be associated with the *Sender*.
- *productPost.details* will be set to *text* from the input form.
- *productPost.image* will be set to *image* from the uploaded image.

Contract CO12: submitReviewRating

Operation: submitReviewRating(text: String, rating: Integer)
Cross References: Use Cases: Review and Rating
Pre-conditions: Current date exceeds the contract end date.
Post-conditions:

- A *ReviewAndRating* instance *reviewAndRating* will be created.
- *reviewAndRating* will be associated with the user.
- *reviewAndRating.ratingScale* will be set to the selected rating.
- *reviewAndRating.text* will be set to the given review.

Contract CO13: sendMessage

Operation: sendMessage(userID: UserID, text: String)
Cross References: Use Cases: Peer-to-peer chat
Pre-conditions: User is registered and logged in to the system
Post-conditions:

- A *Message* instance *message* will be created.
- *message* will be associated with the user.
- *message.text* will be set to the text from the chat.
- *message.timestamp* will be set to the current time.

Contract CO14: updateProductPost

Operation: updateProductPost()
Cross References: Use Cases: Sender's ProductPost Update
Pre-conditions: Sender is able to click the 'Edit' button.
Post-conditions:

- The existing *ProductPost* instance *productPost* will be updated.
- Previous *productPost.details* will be replaced by new *text* from the input form.
- Previous *productPost.image* will be replaced by the new *image* from the uploaded image.

Contract CO15: selectDeny

Operation: selectDeny(travelerID: TravelerID)
Cross References: Use Cases: Contact Creation
Pre-conditions: Sender has some pending requests of the interested traveler to carry his/her product.
Post-conditions:

- A *Traveler* instance *traveler* will be created.
- *Traveler.ID* will be set to *travelerID*.
- *Traveler* will be sent to the system and the updated traveler list after deleting the traveler will be shown.

Contract CO16: confirmTraveler

Operation: confirmTraveler(travelerID: TravelerID)
Cross References: Use Cases: Contact Creation
Pre-conditions: Sender has some pending requests of the interested traveler to carry his/her product.
Post-conditions:

- A $Traveler$ instance $traveler$ will be created.
- $Traveler.ID$ will be set to $travelerID$.
- $Traveler$ will be sent to the system.
- A contract will be confirmed with the between the sender and traveler.
- An invoice will be generated and will be sent to sender and traveler.
- Sender will be redirected to the homepage.

Contract CO17: complaintProductPost

Operation: complaintProductPost(productPostID: ProductPostID)
Cross References: Use Cases: Send Notification
Pre-conditions: Sender and traveler are registered in the system.
Post-conditions:

- A $notification$ instance will be created.
- $Notification.qty$ will be increased.
- An association between $Notification$ and $User$ will be created.

Contract CO18: submitContract

Operation: submitContract(contractID: contractID)
Cross References: Use Cases: Send Notification
Pre-conditions: Sender and traveler are registered in the system.
Post-conditions:

- A $notification$ instance will be created.
- $Notification.qty$ will be increased.
- An association between $Notification$ and $User$ will be created.

Contract CO19: selectNotification

Operation: selectNotification()
Cross References: Use Cases: Send Notification
Pre-conditions: Sender and traveler are registered in the system.
Post-conditions:

- $Notification.qty$ will be increased.

Contract CO20: enterReason

Operation: enterReason(text: String)
Cross References: Use Cases: Moderate ProductPost
Pre-conditions: Number of complaints for a productPost is greater than the threshold.
Post-conditions:

- Provided reason will be visible.

Contract CO21: removeProductPost

Operation: removeProductPost(productPostID: productPostID)
Cross References: Use Cases: Moderate ProductPost
Pre-conditions: Number of complaints for a productPost is greater than the threshold.
Post-conditions:

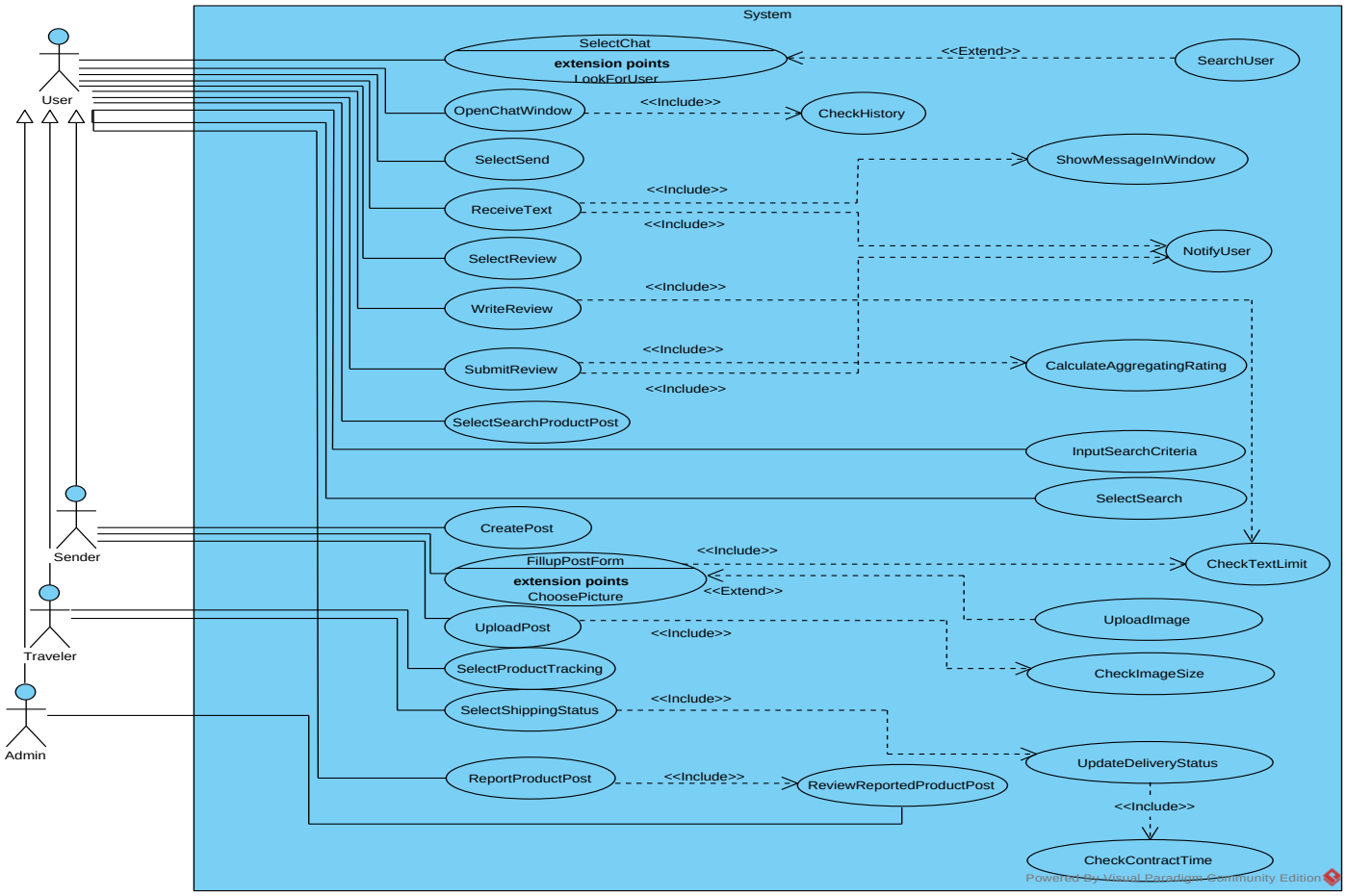
- An association will be created between the user and the productPost.
- *productPost.complaintqty* will be decreased.

Contract CO22: keepProductPost

Operation: keepProductPost(productPostID: productPostID)
Cross References: Use Cases: Moderate ProductPost
Pre-conditions: Number of complaints for a productPost is greater than the threshold.
Post-conditions:

- An association will be created between the *user* and the *productPost*.
- *productPost.complaintqty* will be decreased.

6 Use Case Diagram



7 Wireframes

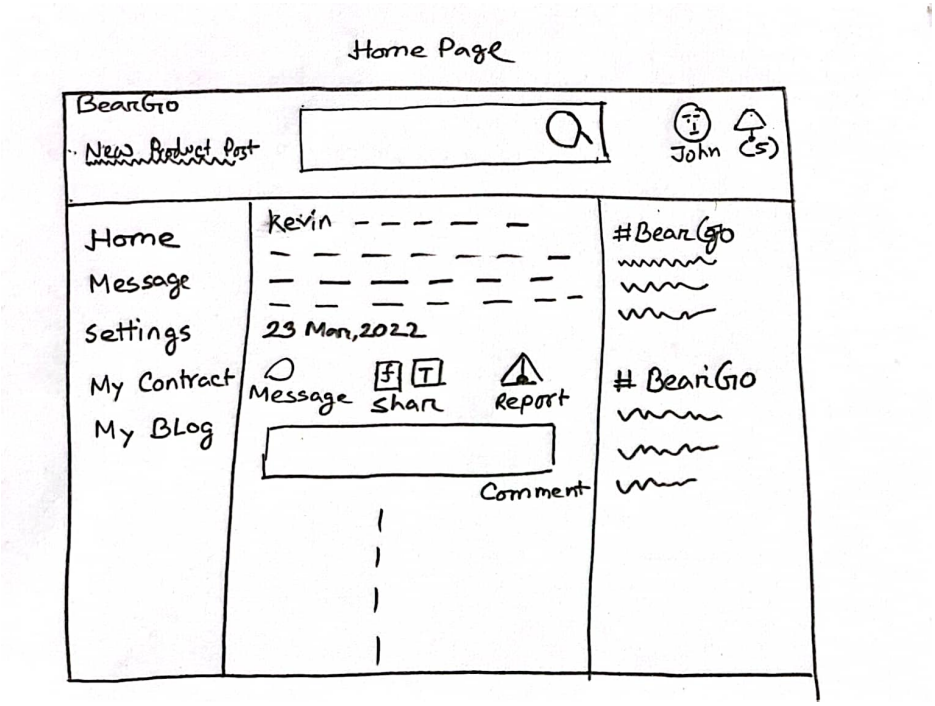


Figure 1: Wireframe design for **Homepage**

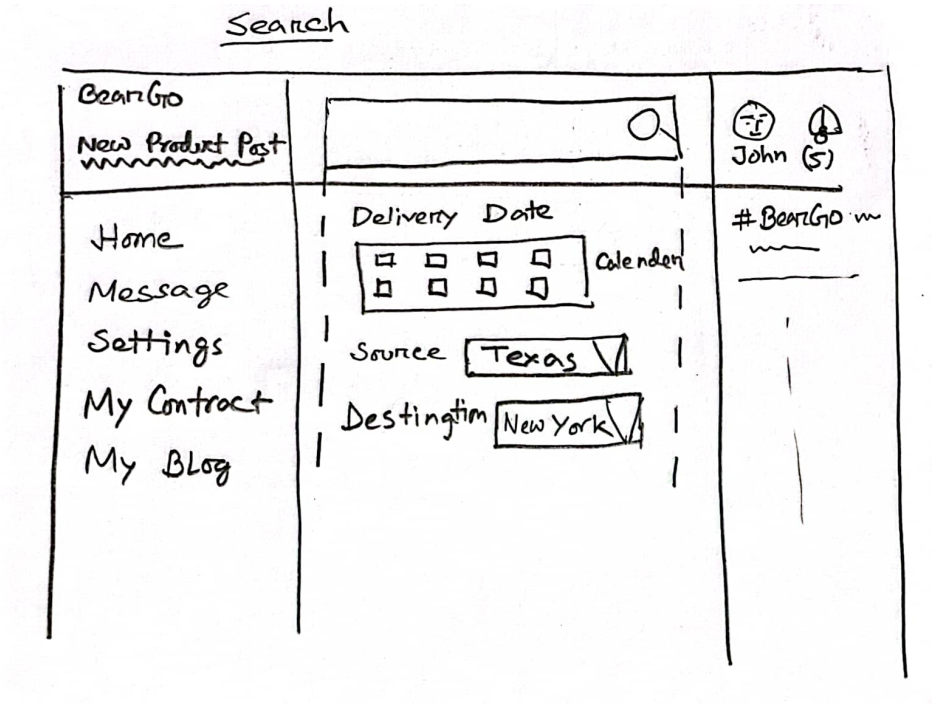


Figure 2: Wireframe design for **Search**

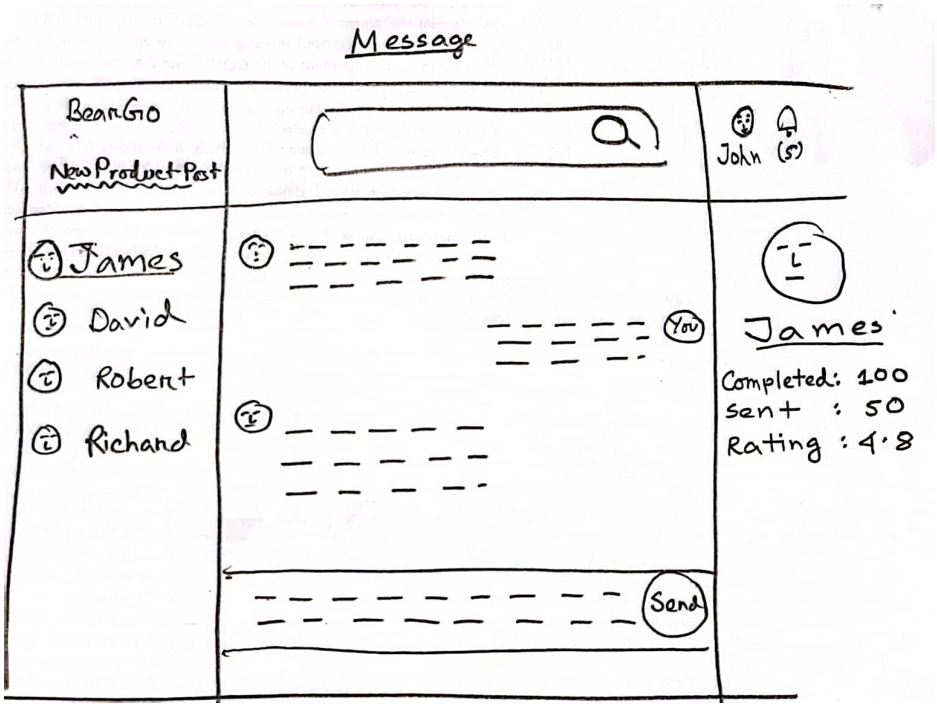


Figure 3: Wireframe design for **Message**

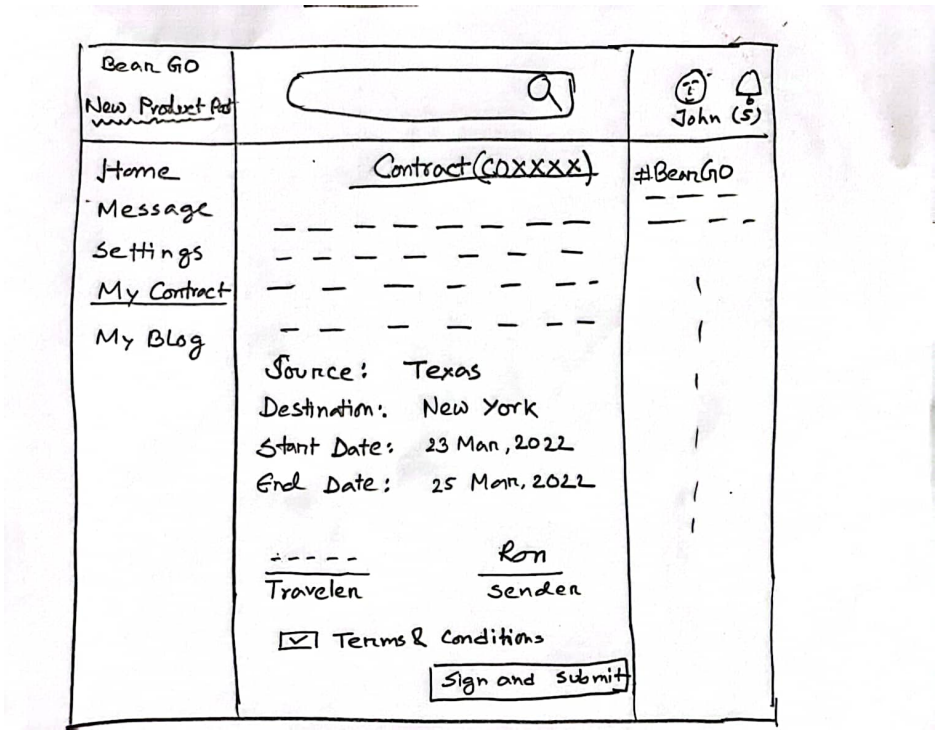


Figure 4: Wireframe design for **Contract**

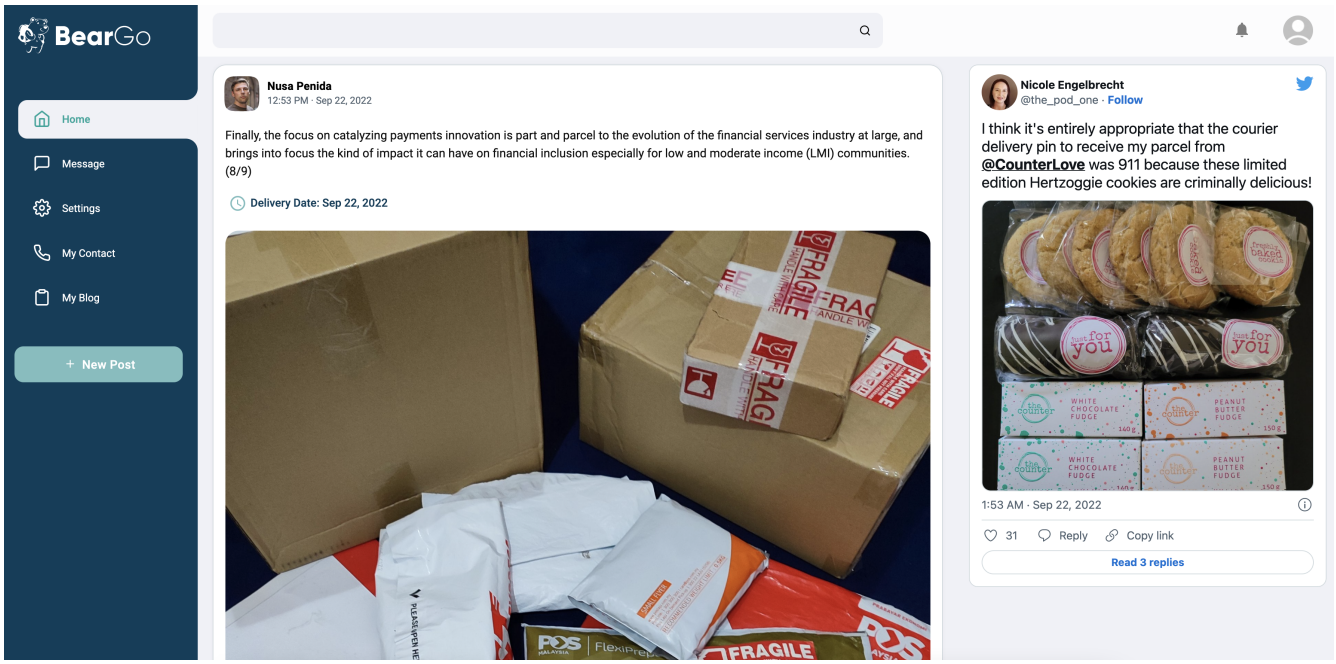
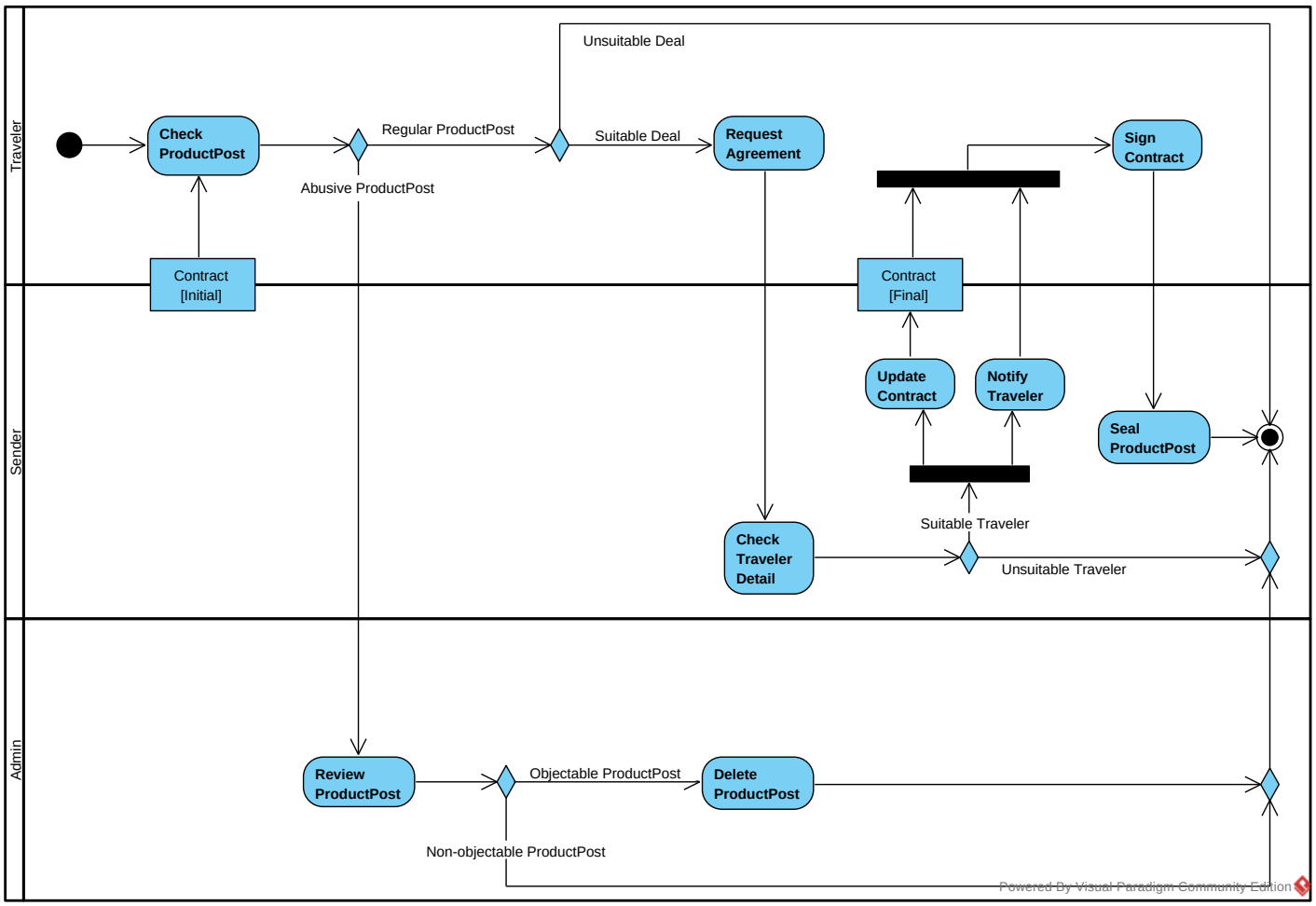


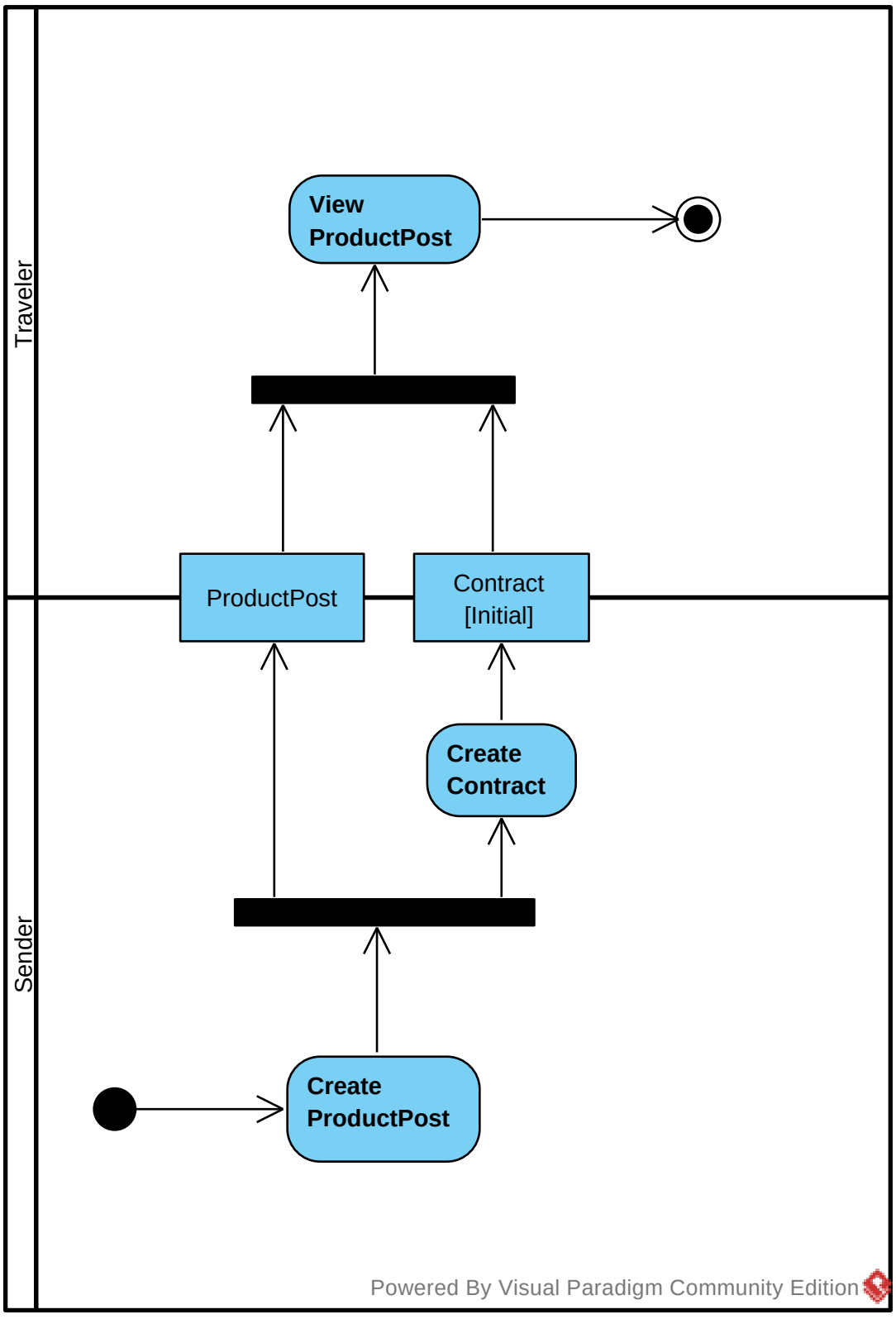
Figure 5: Html design for **Homepage**

8 Activity Diagram

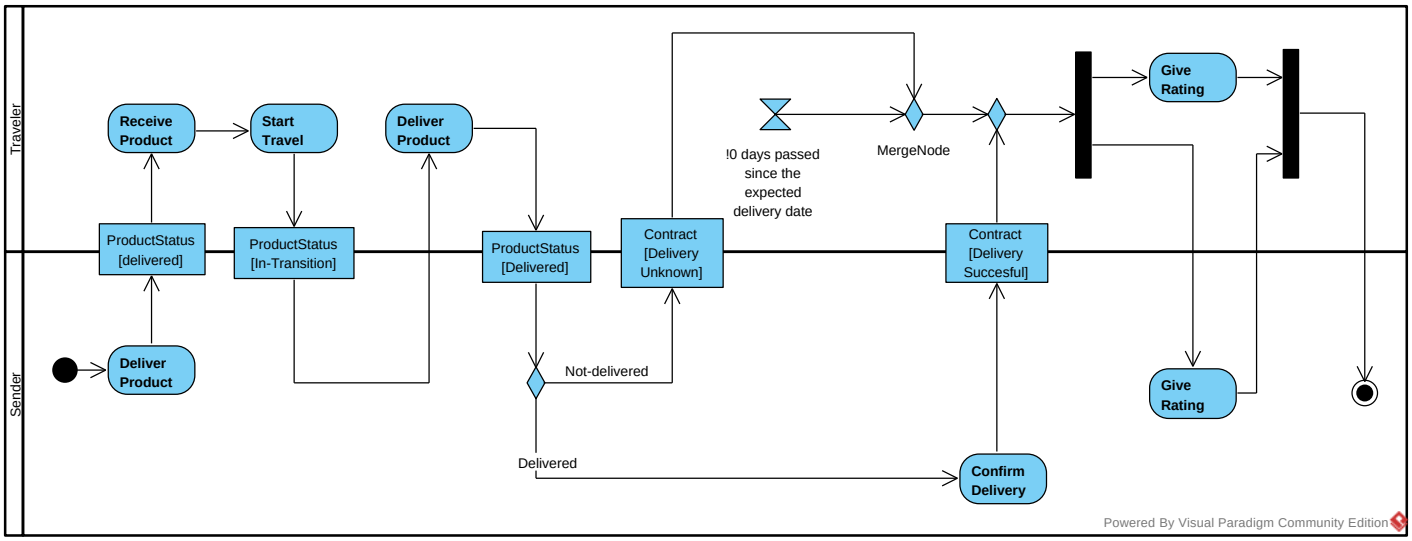
8.1 Activity diagram: Contract Creation



8.2 Activity diagram: Publish Product post



8.3 Activity diagram: Product status update



9 Data Model

We followed JPA persistene,

1. Controller
2. Service
3. Model
4. Repository

Our project link: Github

10 Gantt Chart

Project manager	
Project dates	Aug 26, 2022 - Dec 6, 2022
Completion	0%
Tasks	28
Resources	5

BearGo is the platform for the travelers who willingly delivers any products for someone who wants to ship products.

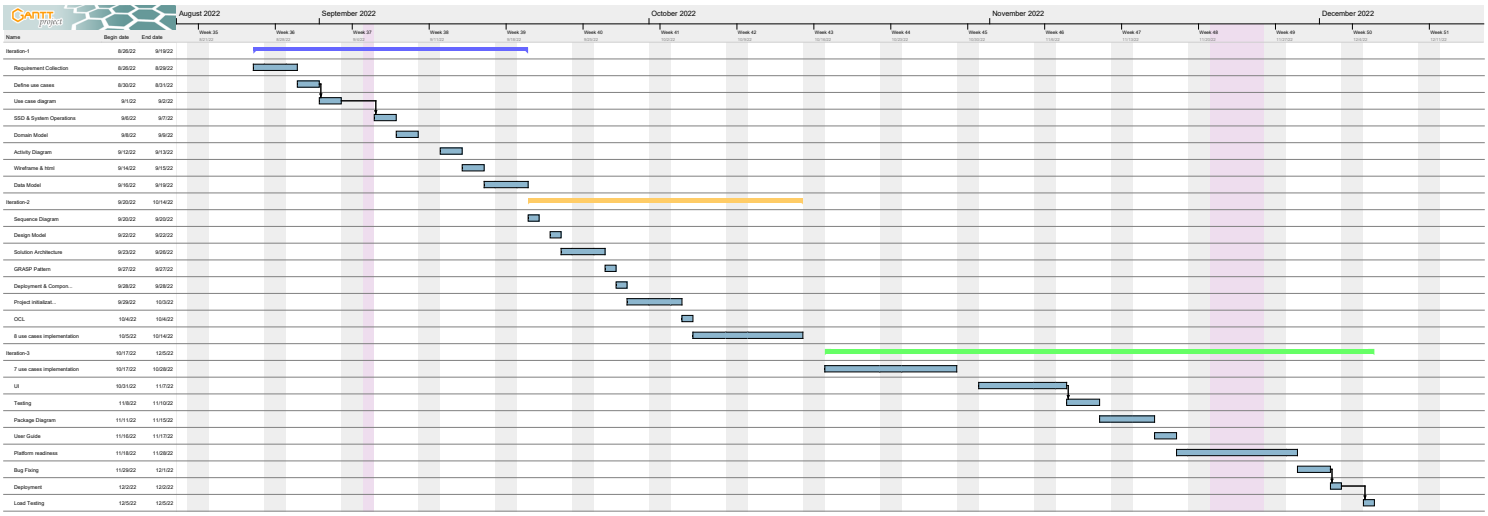
Tasks

Name	Begin date	End date
Iteration-1	8/26/22	9/19/22
Requirement Collection	8/26/22	8/29/22
Define use cases	8/30/22	8/31/22
Use case diagram	9/1/22	9/2/22
SSD & System Operations	9/6/22	9/7/22
Domain Model	9/8/22	9/9/22
Activity Diagram	9/12/22	9/13/22
Wireframe & html	9/14/22	9/15/22
Data Model	9/16/22	9/19/22
Iteration-2	9/20/22	10/14/22
Sequence Diagram	9/20/22	9/20/22
Design Model	9/22/22	9/22/22
Solution Architecture	9/23/22	9/26/22
GRASP Pattern	9/27/22	9/27/22
Deployment & Component Diagram	9/28/22	9/28/22
Project initialization: CRUD Rest API	9/29/22	10/3/22
OCL	10/4/22	10/4/22
8 use cases implementation	10/5/22	10/14/22
Iteration-3	10/17/22	12/5/22
7 use cases implementation	10/17/22	10/28/22
UI	10/31/22	11/7/22
Testing	11/8/22	11/10/22
Package Diagram	11/11/22	11/15/22
User Guide	11/16/22	11/17/22
Platform readiness	11/18/22	11/28/22
Bug Fixing	11/29/22	12/1/22
Deployment	12/2/22	12/2/22
Load Testing	12/5/22	12/5/22

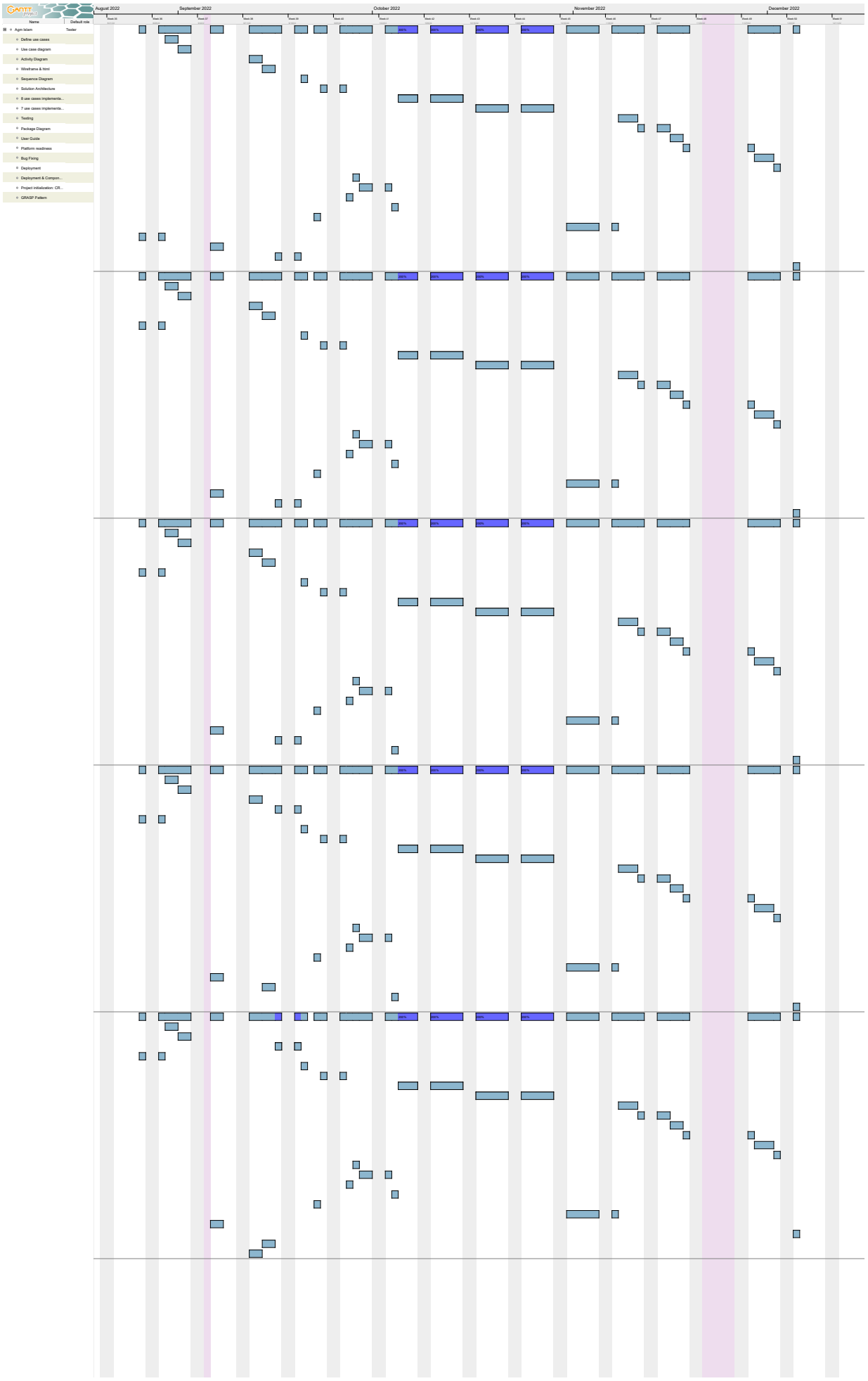
Resources

Name	Default role
Agm Islam	Tester
Tonni Das Jui	Analyst
Swapnil Saha	Developer
Maisha Binte Rashid	Developer
Razwan Ahmed Tanvir	Team Lead

Gantt Chart


August 2022
September 2022
October 2022
November 2022
December 2022
Week 35
Week 36
Week 37
Week 38
Week 39
Week 40
Week 41
Week 42
Week 43
Week 44
Week 45
Week 46
Week 47
Week 48
Week 49
Week 50
Week 51
Name
Begin date
End date
Iteration-1 8/26/22 9/19/22
Requirement Collection 8/26/22 8/29/22
Define use cases 8/30/22 8/31/22
Use case diagram 9/1/22 9/2/22
SSD & System Operations 9/6/22 9/7/22
Domain Model 9/8/22 9/9/22
Activity Diagram 9/12/22 9/13/22
Wireframe & html 9/14/22 9/15/22
Data Model 9/16/22 9/19/22
Iteration-2 9/20/22 10/14/22
Sequence Diagram 9/20/22 9/20/22
Design Model 9/22/22 9/22/22
Solution Architecture 9/23/22 9/26/22
GRASP Pattern 9/27/22 9/27/22
Deployment & Compon... 9/28/22 9/28/22
Project initializat... 9/29/22 10/3/22
OCL 10/4/22 10/4/22
8 use cases implementation 10/5/22 10/14/22
Iteration-3 10/17/22 12/5/22
7 use cases implementation 10/17/22 10/28/22
UI 10/31/22 11/7/22
Testing 11/8/22 11/10/22
Package Diagram 11/11/22 11/15/22
User Guide 11/16/22 11/17/22
Platform readiness 11/18/22 11/28/22
Bug Fixing 11/29/22 12/1/22
Deployment 12/2/22 12/2/22
Load Testing 12/5/22 12/5/22

Resources Chart



11 Trello Task Management

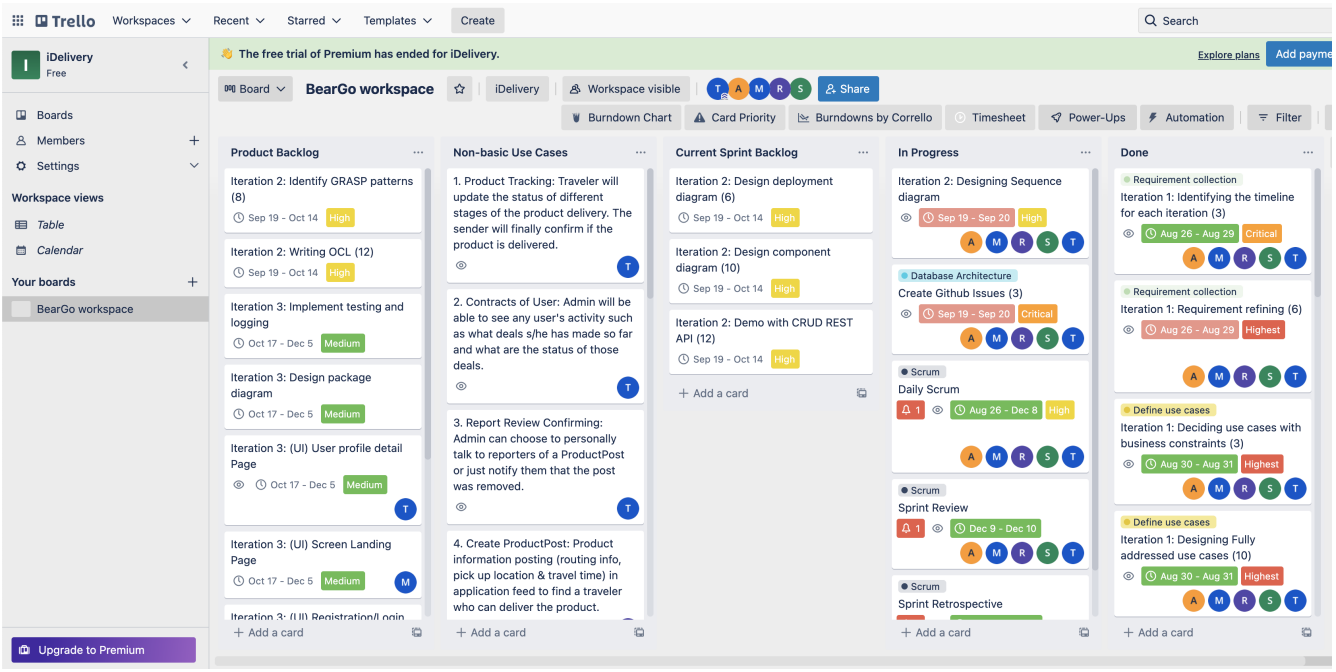


Figure 6: Trello Tasks

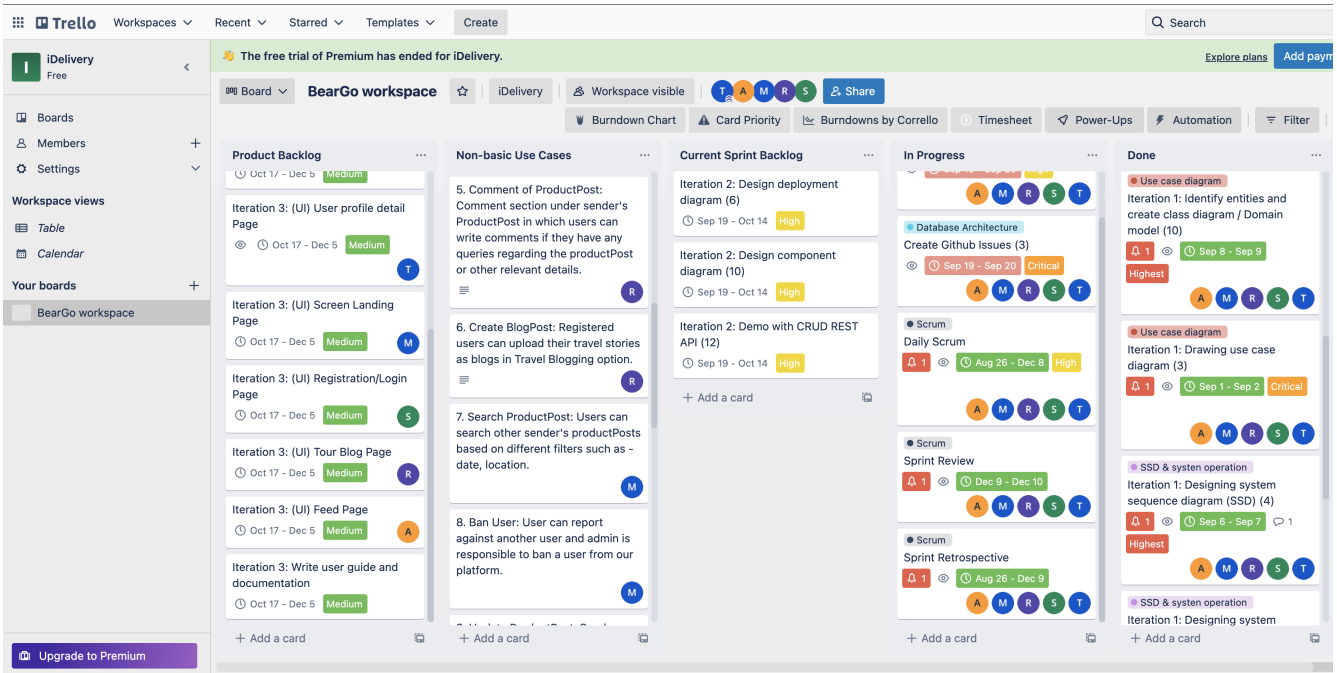


Figure 7: Trello Tasks

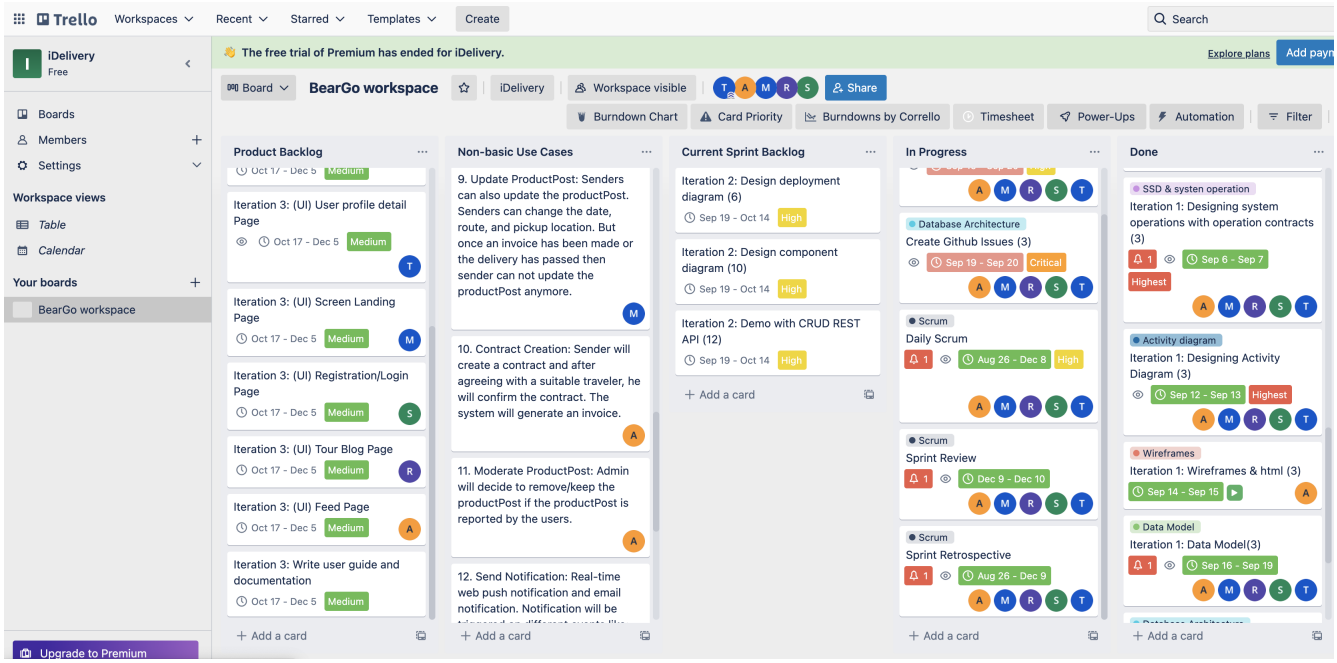


Figure 8: Trello Tasks

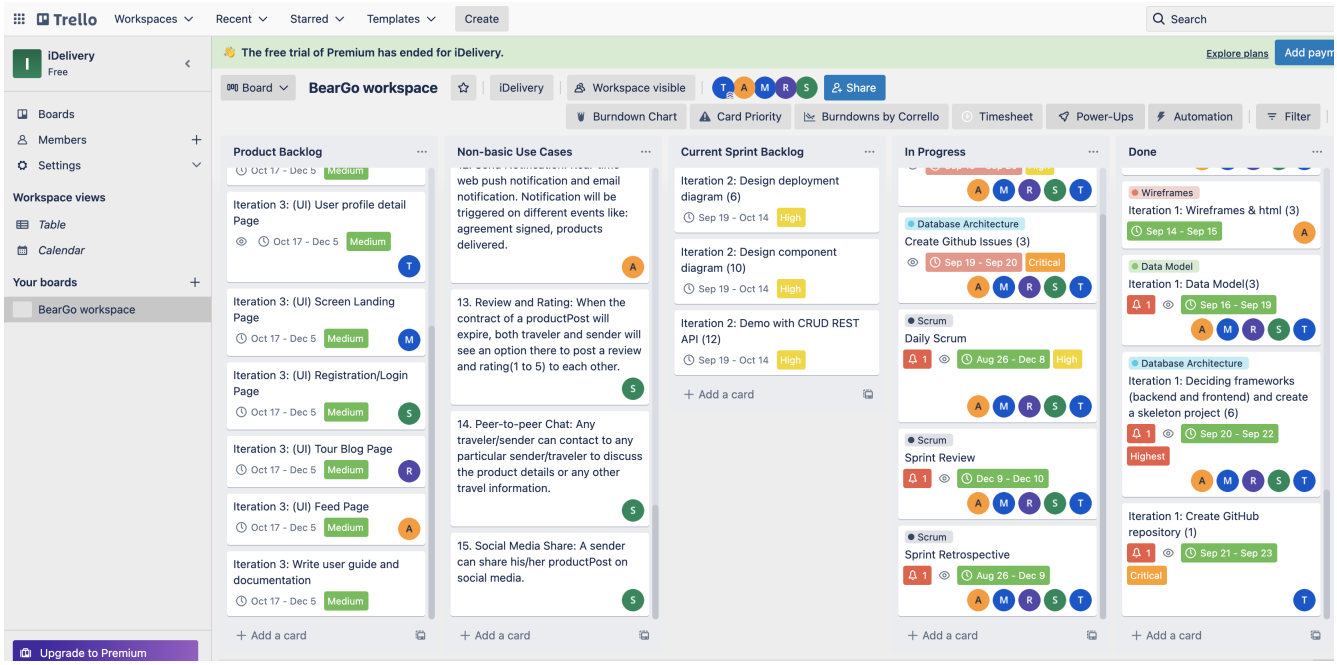


Figure 9: Trello Tasks

12 Burndown Chart

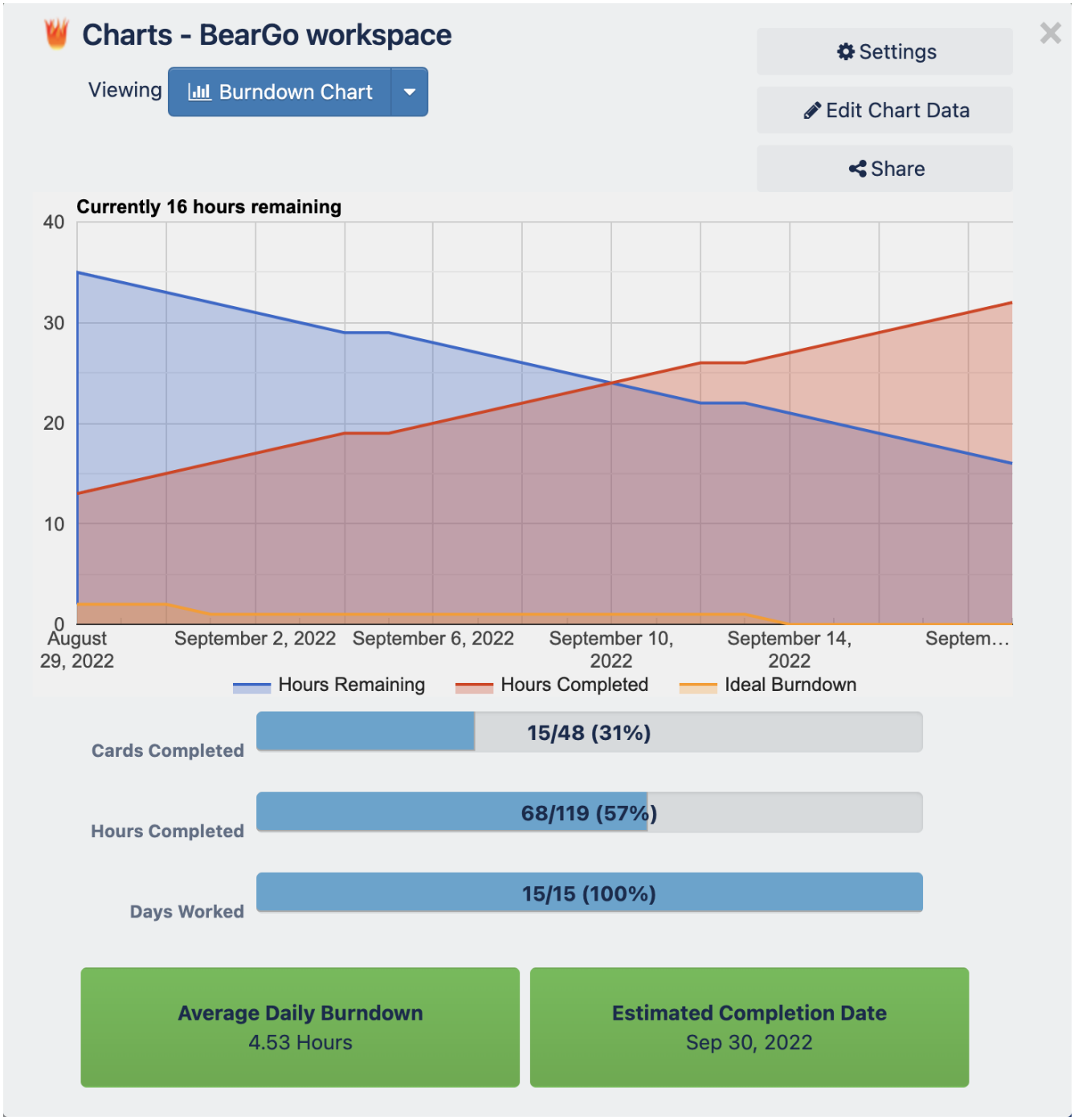


Figure 10: Html design for **Homepage**

13 Summary of Time spent per Task

Timesheet for all Group members:

Overall Iteration 1 Timesheet	
Resource	**Total Time Spent**
Agm Islam	43.98
Maisha Binte Rashid	41.4
Razwan Ahmed Tanvir	42.42
Swapnil Saha	45.71
Tonni Das	50.43
Total	223.94

Figure 11: Total length of the working hour for all group members.

13.1 Trello Timesheet for AGM Islam

Agm Islam	
Task	**Sum of Timespent (h)**
Daily Scrum	5
Iteration 1: Complete task assignment in Trello for current sprint to team members (5)	1
Iteration 1: Create GitHub repository (1)	2
Iteration 1: Data Model(3)	3.92
Iteration 1: Deciding frameworks (backend and frontend) and create a skeleton project (6)	2
Iteration 1: Deciding use cases with business constraints (3)	1
Iteration 1: Designing Activity Diagram (3)	2.66
Iteration 1: Designing Fully addressed use cases (10)	2.5
Iteration 1: Designing system operations with operation contracts (3)	2
Iteration 1: Designing system sequence diagram (SSD) (4)	4
Iteration 1: Drawing use case diagram (3)	3
Iteration 1: Identify entities and create class diagram / Domain model (10)	3.25
Iteration 1: Identifying the timeline for each iteration (3)	0.5
Iteration 1: Prepare Gantt Chart (5)	1
Iteration 1: Requirement refining (6)	1
Iteration 1: Wireframes & html (3)	4
Sprint Retrospective	4
Sprint Review	1.15
Grand Total	**43.98**

13.2 Trello Timesheet for Tonni Das

Tonni Das	
Tasks	**Sum of Timespent (h)**
Daily Scrum	5.5
Iteration 1: Complete task assignment in Trello for current sprint to team	0.67
Iteration 1: Create GitHub repository (1)	4
Iteration 1: Data Model(3)	3.87
Iteration 1: Deciding frameworks (backend and frontend) and create a sk	2
Iteration 1: Deciding use cases with business constraints (3)	2.08
Iteration 1: Designing Activity Diagram (3)	2.85
Iteration 1: Designing Fully addressed use cases (10)	1.83
Iteration 1: Designing system operations with operation contracts (3)	2.42
Iteration 1: Designing system sequence diagram (SSD) (4)	4
Iteration 1: Drawing use case diagram (3)	1.23
Iteration 1: Identify entities and create class diagram / Domain model (1	6.25
Iteration 1: Identifying the timeline for each iteration (3)	0.75
Iteration 1: Prepare Gantt Chart (5)	1
Iteration 1: Requirement refining (6)	0.83
Iteration 1: Wireframes & html (3)	6
Sprint Retrospective	4
Sprint Review	1.15
Grand Total	**50.43**

13.3 Trello Timesheet for Swapnil Saha

Swapnil Saha	
Task	**Sum of Timespent (h)**
Daily Scrum	5
Iteration 1: Complete task assignment in Trello for current sprint to team member	1.25
Iteration 1: Create GitHub repository (1)	2
Iteration 1: Data Model(3)	3.7
Iteration 1: Deciding frameworks (backend and frontend) and create a skeleton pr	1.83
Iteration 1: Deciding use cases with business constraints (3)	1.08
Iteration 1: Designing Activity Diagram (3)	2.85
Iteration 1: Designing Fully addressed use cases (10)	2.5
Iteration 1: Designing system operations with operation contracts (3)	2
Iteration 1: Designing system sequence diagram (SSD) (4)	4.5
Iteration 1: Drawing use case diagram (3)	2.92
Iteration 1: Identify entities and create class diagram / Domain model (10)	3.67
Iteration 1: Identifying the timeline for each iteration (3)	0.42
Iteration 1: Prepare Gantt Chart (5)	1.17
Iteration 1: Requirement refining (6)	0.67
Iteration 1: Wireframes & html (3)	5
Sprint Retrospective	4
Sprint Review	1.15
Grand Total	**45.71**

13.4 Trello Timesheet for Rezwan Ahmed Tanvir

Razwan Ahmed Tanvir	
Task	**Sum of Timespent (h)**
Daily Scrum	5
Iteration 1: Complete task assignment in Trello for current sprint to team members (5)	1.5
Iteration 1: Create GitHub repository (1)	1.5
Iteration 1: Data Model(3)	2.75
Iteration 1: Deciding frameworks (backend and frontend) and create a skeleton project (6)	1.5
Iteration 1: Deciding use cases with business constraints (3)	0.5
Iteration 1: Designing Activity Diagram (3)	3.5
Iteration 1: Designing Fully addressed use cases (10)	2
Iteration 1: Designing system operations with operation contracts (3)	2
Iteration 1: Designing system sequence diagram (SSD) (4)	4.5
Iteration 1: Drawing use case diagram (3)	3.25
Iteration 1: Identify entities and create class diagram / Domain model (10)	3
Iteration 1: Identifying the timeline for each iteration (3)	0.5
Iteration 1: Prepare Gantt Chart (5)	1.02
Iteration 1: Requirement refining (6)	0.5
Iteration 1: Wireframes & html (3)	4.25
Sprint Retrospective	4
Sprint Review	1.15
Grand Total	**42.42**

13.5 Trello Timesheet for Maisha Binte Rashid

Maisha Binte Rashid	
Tasks	**Sum of Timespent (h)**
Daily Scrum	5
Iteration 1: Complete task assignment in Trello for current sprint to team members (5)	1
Iteration 1: Create GitHub repository (1)	1.5
Iteration 1: Data Model(3)	3.17
Iteration 1: Deciding frameworks (backend and frontend) and create a skeleton project (6)	2
Iteration 1: Deciding use cases with business constraints (3)	1.75
Iteration 1: Designing Activity Diagram (3)	3
Iteration 1: Designing Fully addressed use cases (10)	2.5
Iteration 1: Designing system operations with operation contracts (3)	1.75
Iteration 1: Designing system sequence diagram (SSD) (4)	4.33
Iteration 1: Drawing use case diagram (3)	2
Iteration 1: Identify entities and create class diagram / Domain model (10)	1.75
Iteration 1: Identifying the timeline for each iteration (3)	1
Iteration 1: Prepare Gantt Chart (5)	1
Iteration 1: Requirement refining (6)	1
Iteration 1: Wireframes & html (3)	3.5
Sprint Retrospective	4
Sprint Review	1.15
Grand Total	**41.4**