BILBOARD

Team-T4 BugBunny

Activity/Sequence/State/Class Diagrams & Mockup

DILARA MANDIRACI
BURAK DEMIREL
YUSUF TORAMAN
SILA ÖZEL
EREN HAYRETTIN ARIM

Fall 2023

Table of Contents

1. Activity Diagrams

1.1. Adding a Product

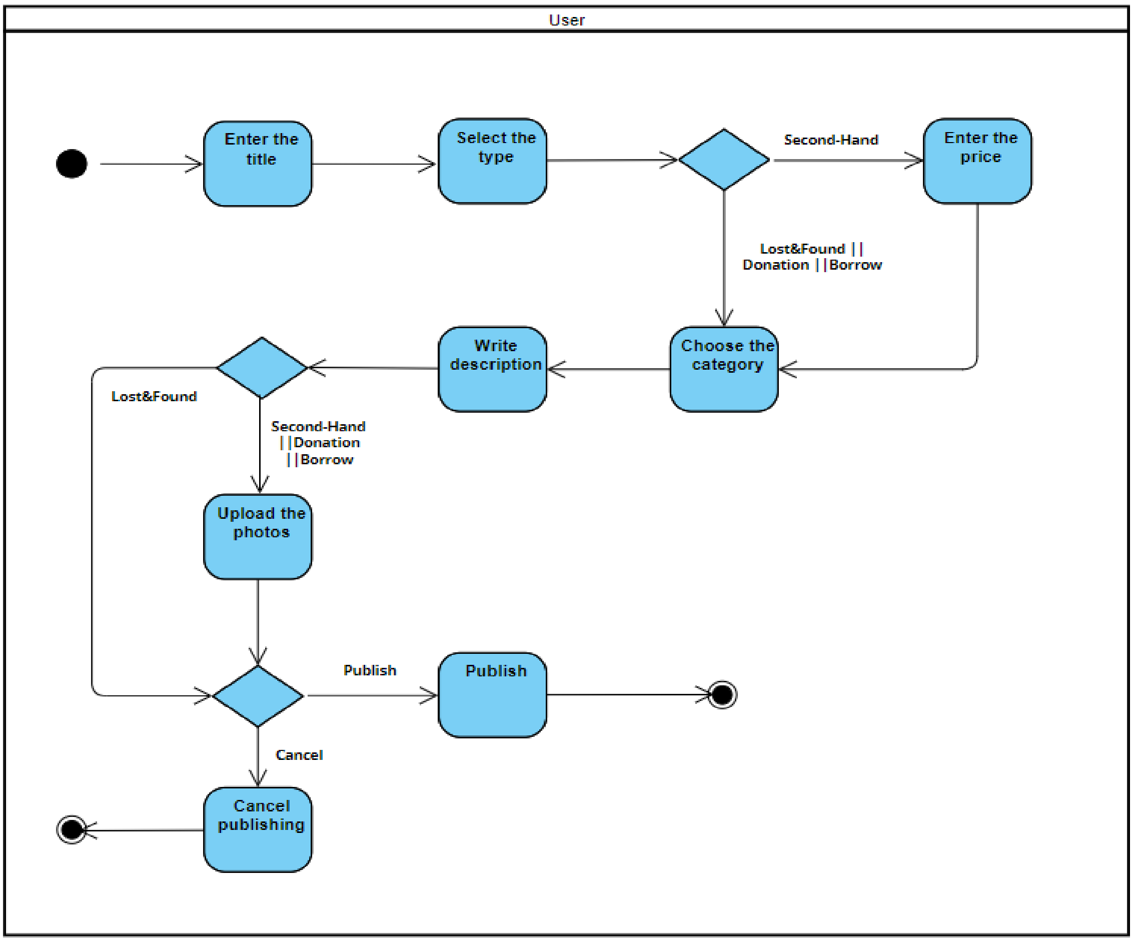

Figure 1: Activity Diagram of Adding a Product

This adding a product diagram illustrates a user's process when creating a post on a platform. Initially, the user provides a post title and selects the content type they're sharing. Depending on the chosen type, they might either enter a price (for second-hand items) or select a category (for lost & found, donations, or borrowed items). The user then crafts a detailed description and uploads relevant photos. Also, uploading photos depends on the type chosen. If the type is lost & found, the photos are not uploaded. After all these steps are completed, the user has the option to publish their post. If they change their mind, there's also an option to cancel the publishing process.

1.2. Updating/Deleting a Product

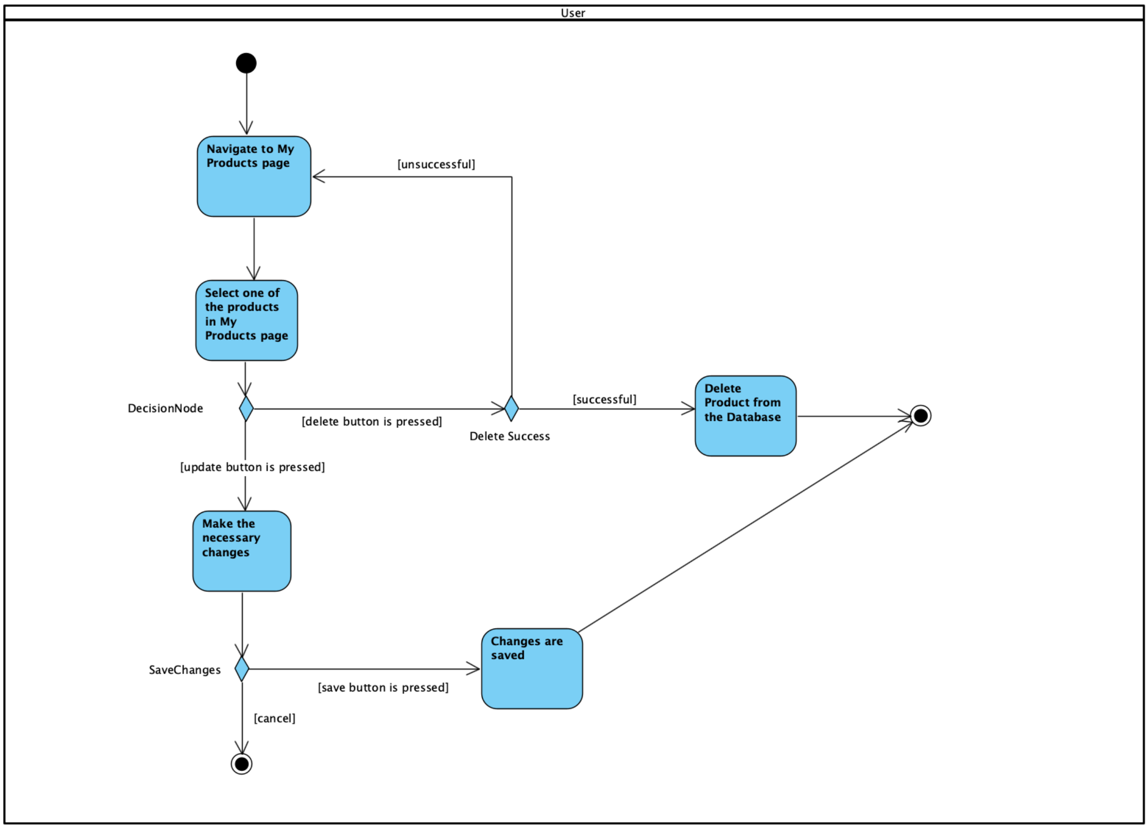
User
Navigate to My Products page
[unsuccessful]
Select one of the products in My Products page
DecisionNode
[delete button is pressed]
Delete Success
[successful]
Delete Product from the Database
[update button is pressed]
Make the necessary changes
SaveChanges
[save button is pressed]
Changes are saved
[cancel]

Figure 2: Activity Diagram of product handling

In our application, the user can delete or update their product posts. To do so, the user should first navigate to the “My Products” page. Then, from that page, they should select the product they want to delete or edit. If the “delete” button is pressed, then the corresponding Controller class on the backend will delete the product record from the database. But if the user presses the “update” button, then they will make the necessary changes, and after they are done with their changes, they will either save the changes to the database by clicking the “save” button, or they can cancel their changes by clicking the “cancel” button.

1.3. Communication Between Users

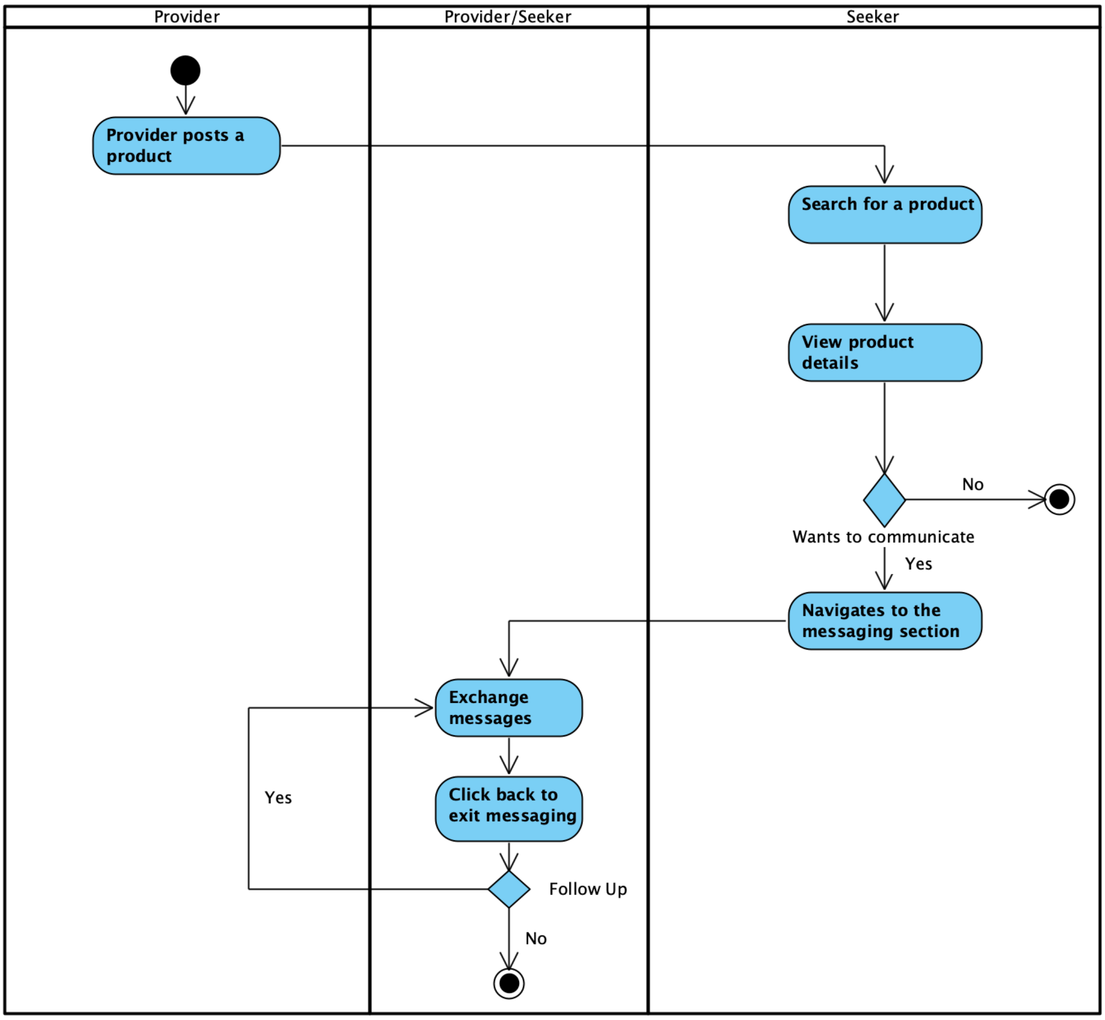

Figure 3: Activity diagram of communication system

For the communication action diagram, we have 2 main actors. One is the provider (lender, donor, or seller), and the other actor is the seeker (borrower, donee, or buyer). Since all these three features will have communication features, we can boil things down to one action diagram. The starting point for the communication will be posting a product by provider since people can only start communication from a specific product. (They cannot search for their names and text them. Since the aim is not texting.) Then the seeker searches for a specific product and when they find a match, they click on a product to see the details. If they want to communicate, they can navigate to the messaging section. After initiating communication with the provider, since they will have a common messaging session, they can text each other.

1.4. Search/Find a Product

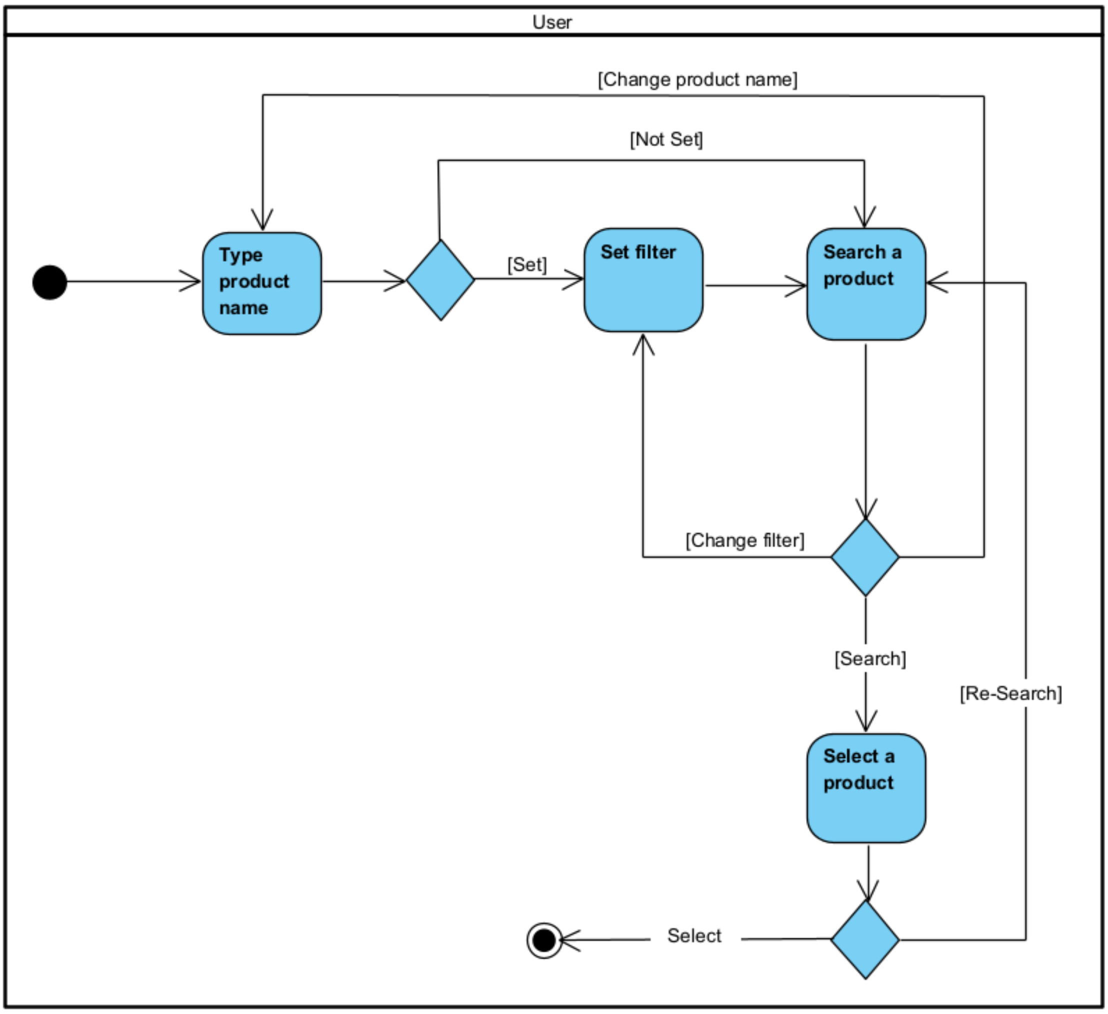

Figure 4: Activity diagram of product search

The given activity diagram represents the logic of searching and finding products. At the beginning, the User types the name of the product. Before searching, the User can also set some filters to enhance the query. If the User chooses to set some, then offered filters can be selected before searching. Otherwise, the User directly searches with the provided keyword. At any time before finalizing the search action, the User can freely change the typed keyword or re-set the filters. After completing the search action, if the user wants another search, then it can do it. If the User finds the intended product, then it selects this product, and the activity finalizes.

1.5. Complaint Updating/Deleting

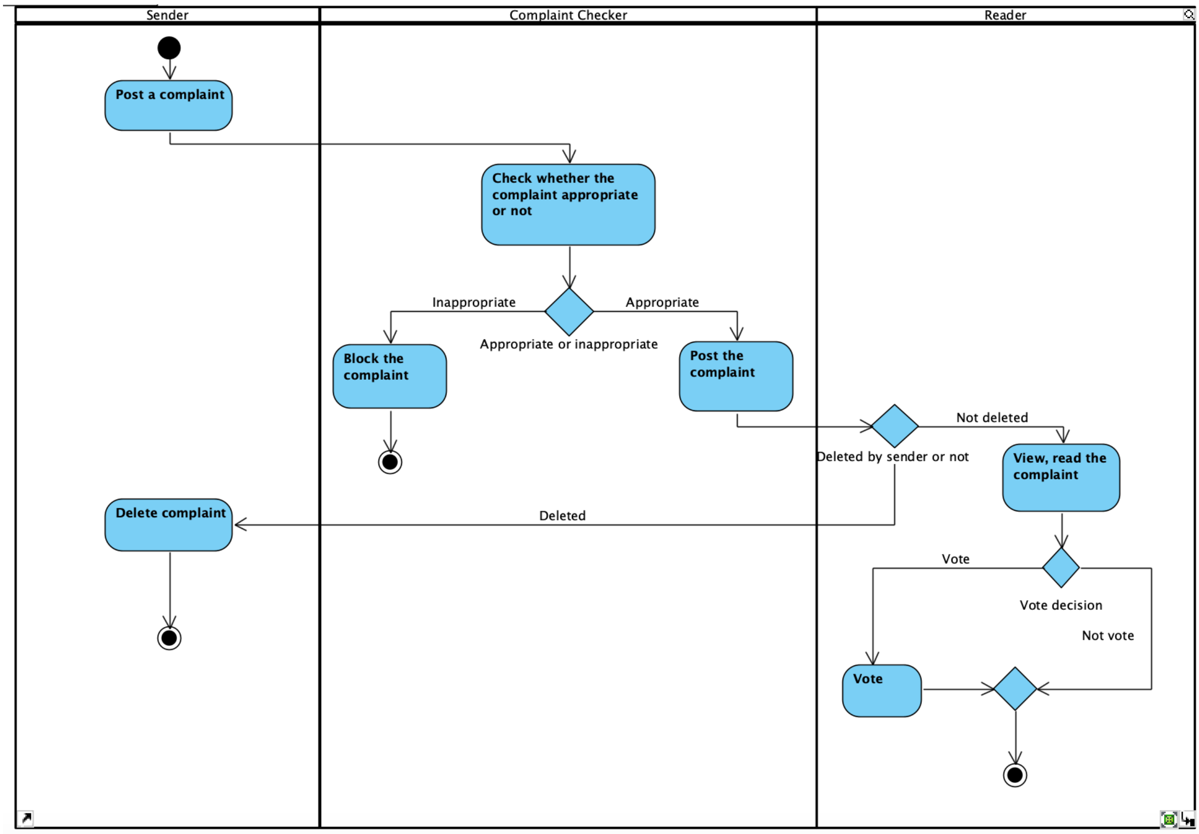

Figure 5: Activity diagram of complaint handling

This activity diagram reflects the complaint module's scenario. The complaint module includes scenarios such as post, delete and view. As we can see from the diagram, the starting point is that a complaint has been posted by the sender. This is because the reader cannot see an unposted complaint. The posted complaint goes to the complaint checker, where it encounters a decision node. If the Complaint Checker decides that the decision is inappropriate, the complaint is blocked and the activity reaches the finish point. Else, seen as decision appropriate, the complaint is posted and starts appearing in the system. At this point, the complaint checker is finished and can start performing reader activities, as we can see in the vertical partition. If the posted complaint has been deleted, the reader will no longer be able to see it and the activity reaches its finishing point. Otherwise, if it is not deleted, the reader can see this complaint and reach the finish point if he wishes, as we can extract from the MergeNode. However, if the user wants to use the vote option, then the user can do the vote activity and reach the finish point as a result of this activity. In general, the complaint module's activity diagram represents the activities related to the complaint and shows which activities to proceed within various conditions.

2. Sequence Diagrams

2.1. Register

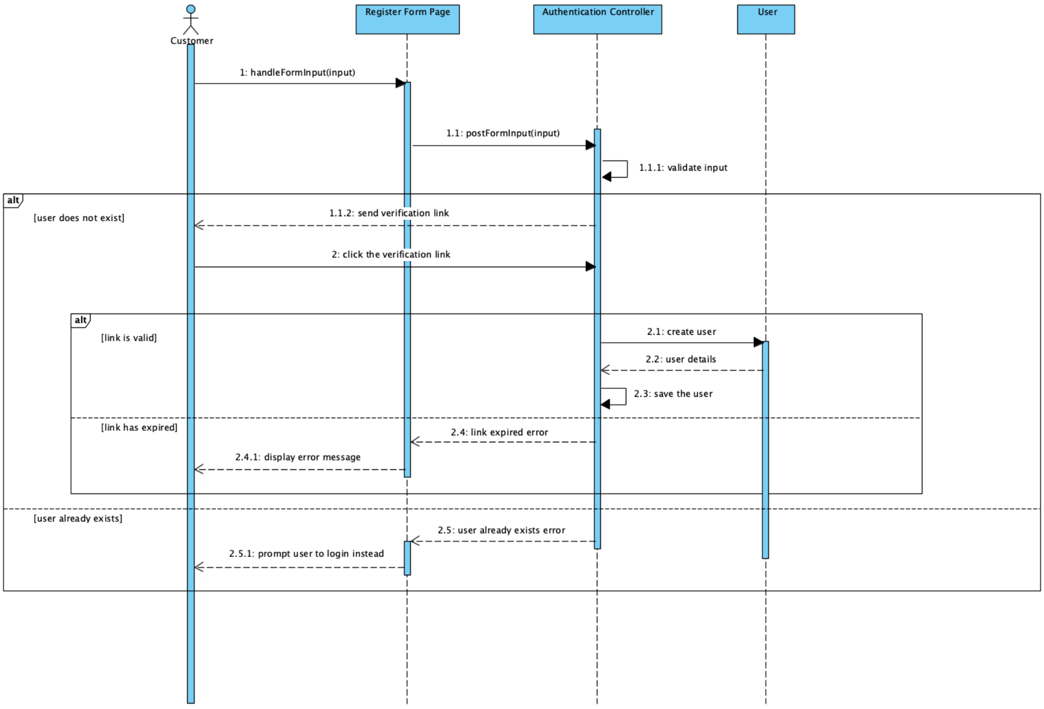
Customer
Register Form Page
Authentication Controller
User
1: handleFormInput(input)
1.1: postFormInput(input)
1.1.1: validate input
alt
[user does not exist]
1.1.2: send verification link
2: click the verification link
alt
[link is valid]
2.1: create user
2.2: user details
2.3: save the user
[link has expired]
2.4: link expired error
2.4.1: display error message
[user already exists]
2.5: user already exists error
2.5.1: prompt user to login instead

Figure 6: Sequence diagram of registration to the system

If a new customer is going to use our application, they must register first. To do so, they need to fill in the necessary areas in the registration form. Then, on the frontend we will handle the input data. After clicking the submission button, the provided information will be sent to the backend controller. In the backend, the input will be validated (checks whether the user already exists). If the user had already registered before, the backend code will return a user already exists error to the frontend, and then the user is prompted to login instead. If the user doesn't exist, then a verification code will be sent to the user's email. Then the user should enter this code, and in the backend the code will be checked. If it is the correct code, then the user will be saved to the database and directed to the user page. But if the code is not correct, the user will see an error message and they will not be registered (they should try to register again).

2.2. Login

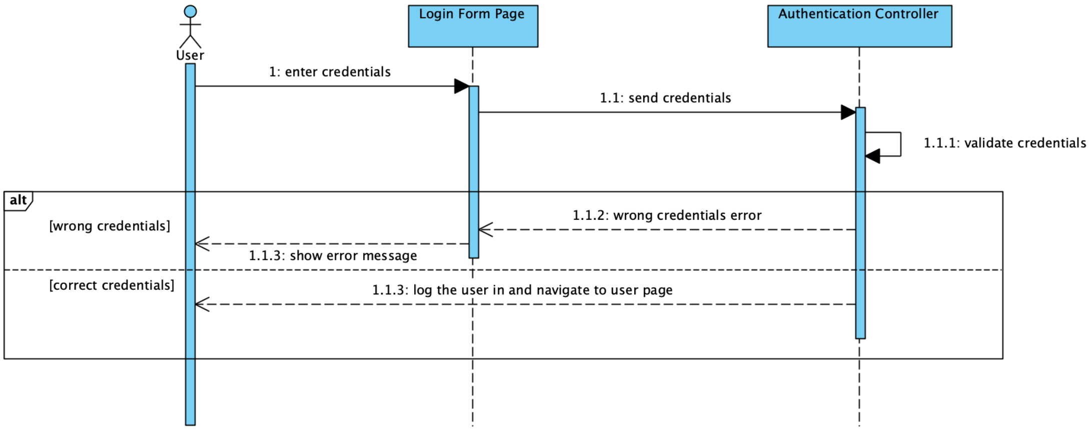
User
Login Form Page
Authentication Controller
1: enter credentials
1.1: send credentials
1.1.1: validate credentials
alt
[wrong credentials]
1.1.2: wrong credentials error
1.1.3: show error message
[correct credentials]
1.1.3: log the user in and navigate to user page

Figure 7: Sequence diagram of Login

In order to login, the user should enter their credentials (email address and password). After clicking the “login” button, the credentials will be sent to the backend. The credentials will be checked, and if they are not correct, then the user will see an error message. If they are correct, then the user will be able to login and see the user page.

2.3. Search a Product

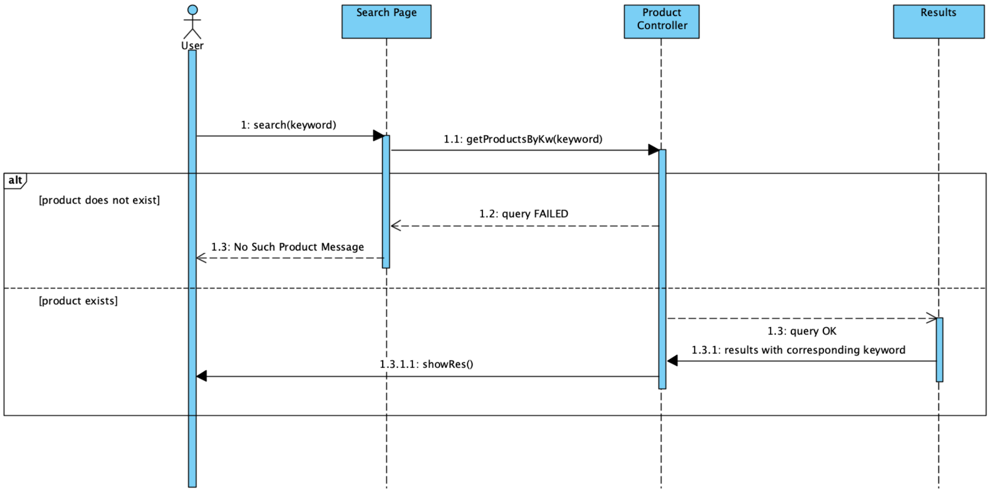

Figure 8: Sequence diagram of product search

The user can search for a product by searching keywords. Then from the frontend page, we will send a GET request and try to get all products with the specified keywords. If the query is successful and the controller is able to fetch the entries from the database, then on the frontend, the user will see the results. If the query fails, however, the user will see an error message (something like not found).

2.4. Upload a Product

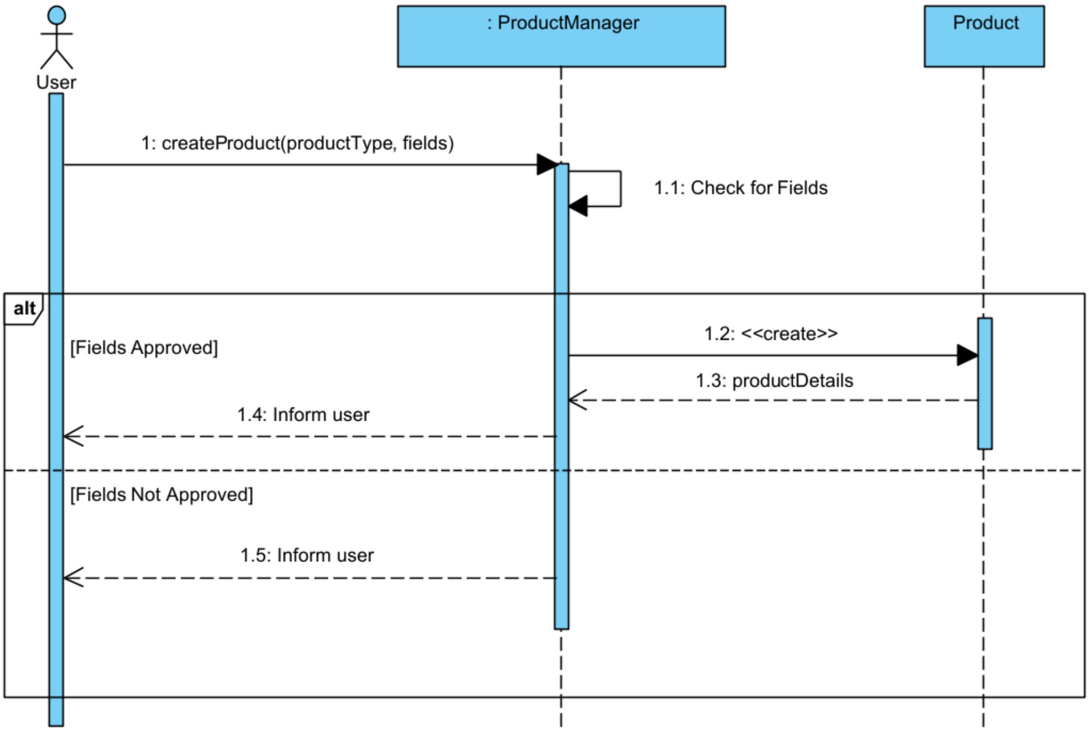

Figure 9: Sequence diagram of uploading a product to the system

This sequence diagram represents the scenario where the user creates a product. Firstly, the User sends the required data to the ProductManager for creating a product. Then ProductManager checks the validity of the given fields. If given fields are validated, then ProductManager creates a product according to the given product type. After retrieving the newly created product's data, finally informs the user by returning the product with a success message. However, if the given fields are not validated, then ProductManager immediately informs the user by sending a failure message with the necessary explanation.

2.5. Handle with a Product

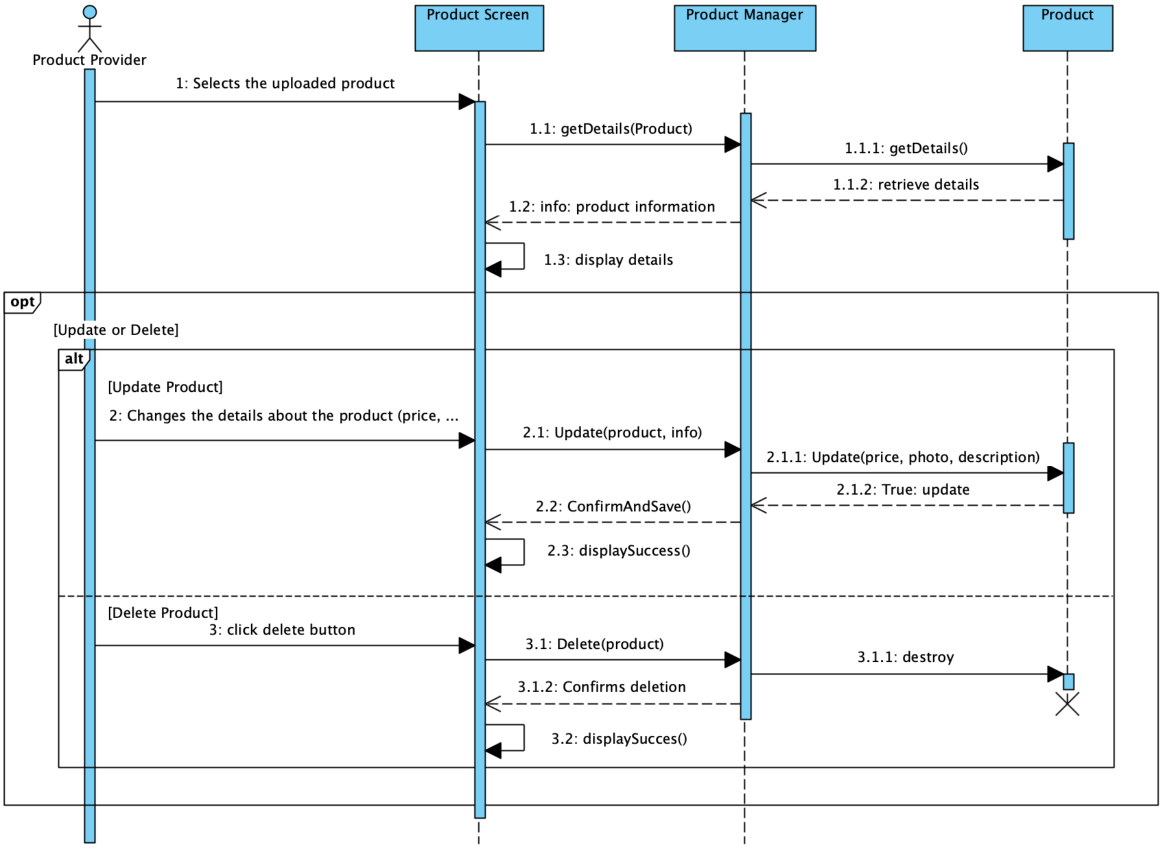

Figure 10: Sequence diagram of product handling process

The product provider will select the product that they previously uploaded. Then, the product manager will retrieve the details of the product, which means when they click on a specific product, they will see the details of it. On the details page, there will be a button. When they click that button, the provider can either update the product details (price, photo, or product description) or, if they wish, they can delete the product. The actor will be only the provider, since only the product provider will be authorized to change their product.

2.6. Post a Complaint

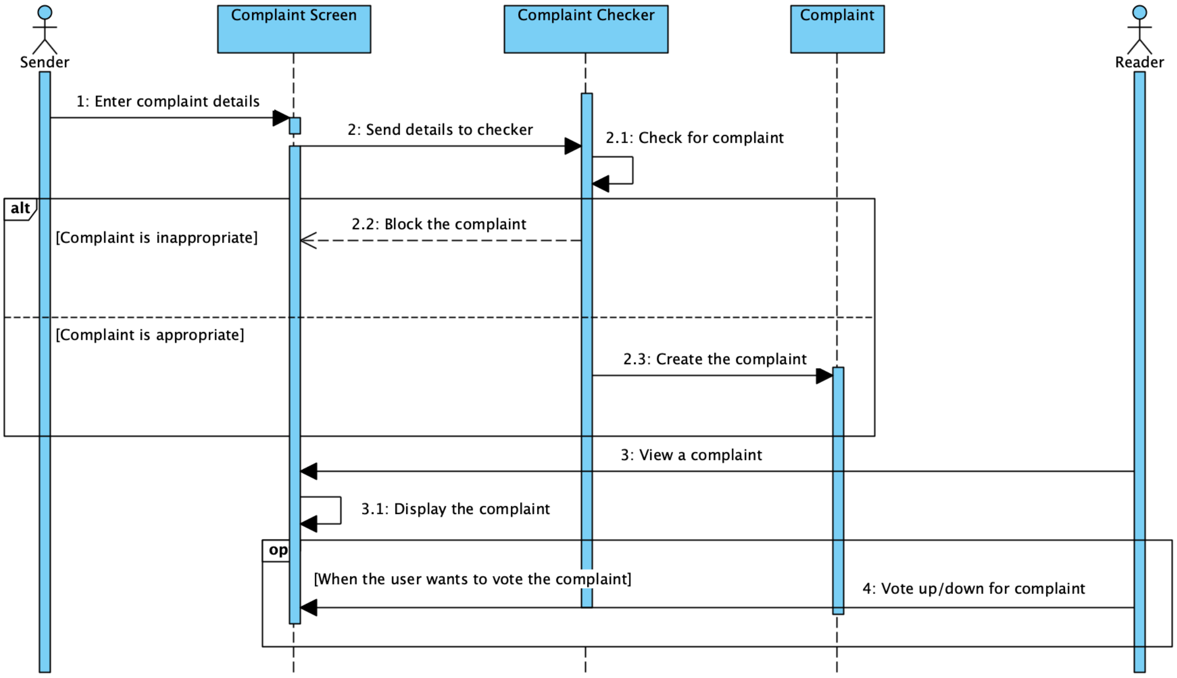
Sender
Complaint Screen
Complaint Checker
Complaint
Reader
1: Enter complaint details
2: Send details to checker
2.1: Check for complaint
alt
[Complaint is inappropriate]
2.2: Block the complaint
[Complaint is appropriate]
2.3: Create the complaint
3: View a complaint
3.1: Display the complaint
op
[When the user wants to vote the complaint]
4: Vote up/down for complaint

Figure 11: Sequence diagram of posting a complaint

The diagram above contains a sequence diagram for the PostComplaint scenario. As can be seen from the diagram, the Sender Actor first posts a complaint. Then, this post goes to the compliant checker lifeline and the compliant checker checks it internally. At this point, the next job is to block the complaint if it is inappropriate. Otherwise, post the complaint to the server lifeline. Finally, it is time for the Reader actor to display the complaint. To do this, the reader sends a request to the server and displays it. After viewing, it can perform the vote up/down operation, which is an optional event, as stated in the sequence diagram. All sequences of the Complaint post scenario are like this.

2.7. Display and Delete a Previously Posted Complaint

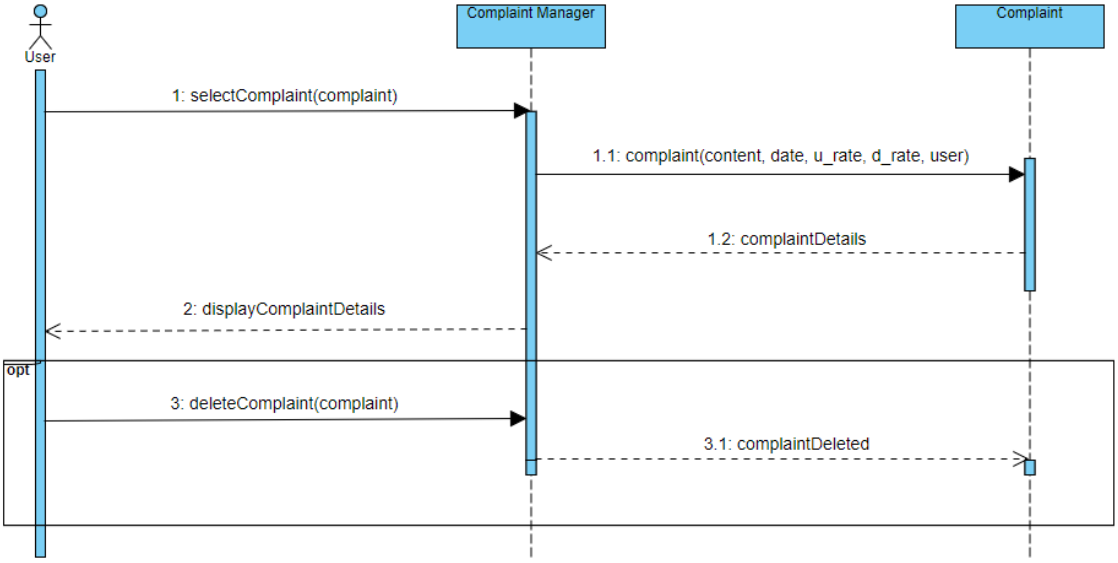
User
Complaint Manager
Complaint
1: selectComplaint(complaint)
1.1: complaint(content, date, u_rate, d_rate, user)
1.2: complaintDetails
2: displayComplaintDetails
opt
3: deleteComplaint(complaint)
3.1: complaintDeleted

Figure 12: Sequence diagram of display and deletion of a complaint

This sequence diagram presents a user's interaction with the "Complaint Manager" system. The user starts by choosing a particular complaint. Once selected, the Complaint Manager retrieves the specific details associated with that complaint. These details are shown to the user. The user has an optional deletion choice. If they decide to delete, the Complaint Manager processes the deletion.

3. State Diagrams

3.1. Login

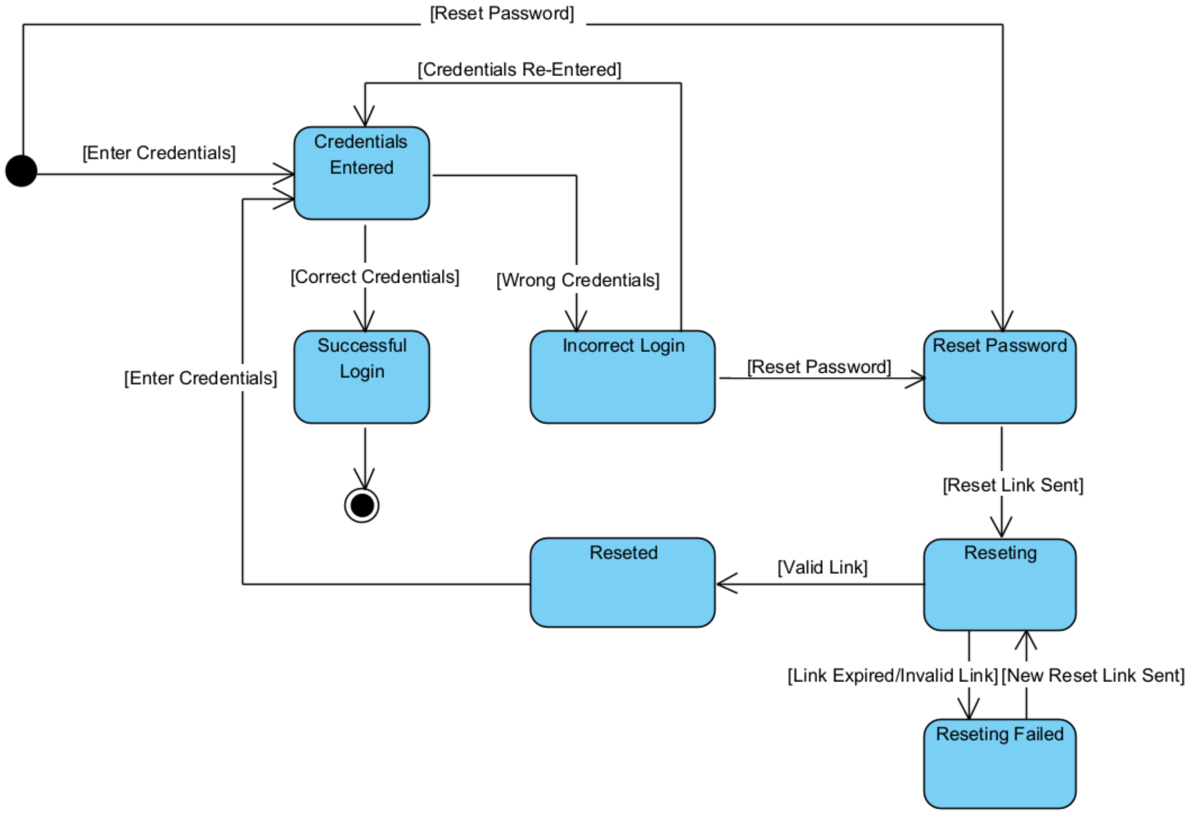

Figure 13: State diagram of login to the system

The given diagram represents the status of a login request. In the beginning, after providing credentials, the system checks the validity of them. If the credentials are correct, then the User with the given credentials is authorized. If credentials are wrong, then the system does not authorize the given credentials, and if again there are provided credentials, the system evaluates it. If there is a situation about a forgotten password, then the system sends a reset link to the given email address and waits for resetting. If the accessed reset link is expired or invalid, then the system resends another reset link and again waits for resetting. If the accessed reset link is a valid one, then the password is reset. If the user provides credentials at this stage, again the system evaluates it.

3.2. Register

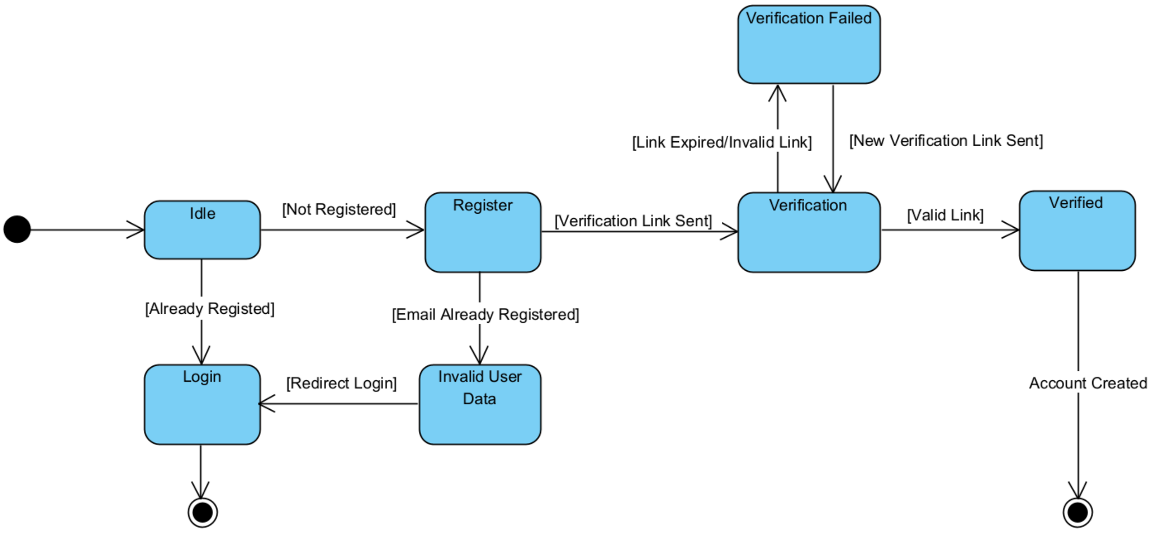

Figure 14: State diagram of registration of a user

The given diagram represents the states of a register request. In the beginning, the system waits for an action. If the user has an account, then the system enters the login state. Otherwise, the user enters the register state. If the provided user email is linked with another account, then the system enters the login state directly. Otherwise, a verification link is sent to the given mail address. If the accessed verification link is expired or invalid, then the system resends another reset link and again waits for verification. If the accessed verification link is a valid one, then the system verifies the newly created account.

3.3. Product Post/Update/Delete

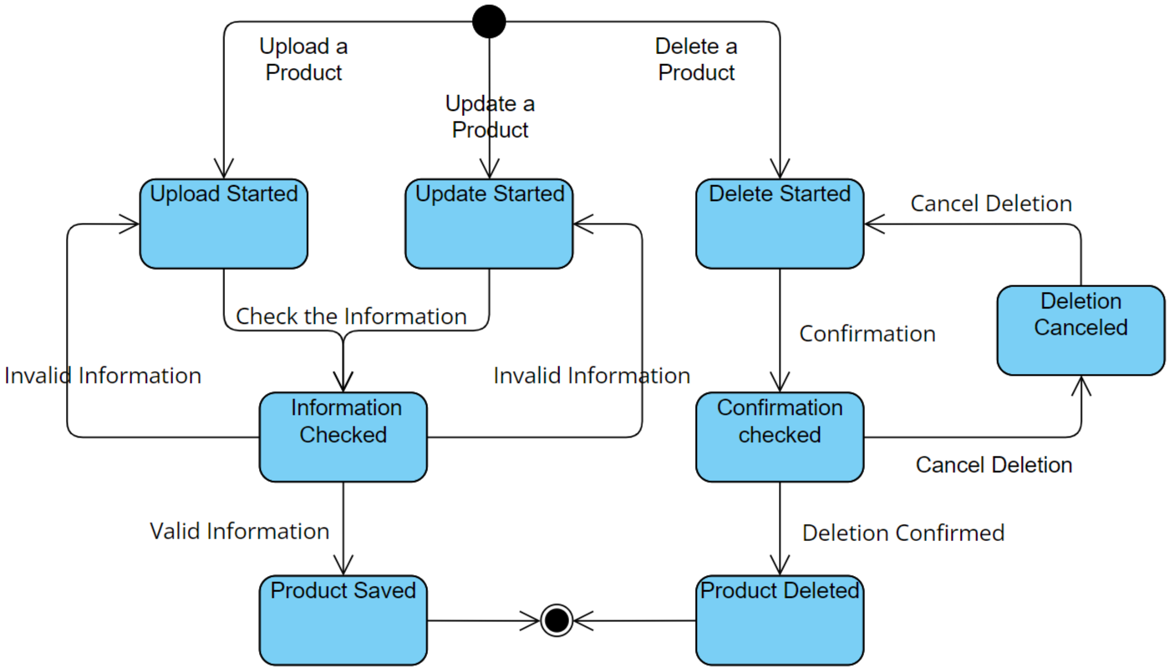

Figure 15: State diagram of product upload, update, and delete process

The state diagram illustrates the workflow of product management, encompassing the processes of uploading, updating, and deleting a product. The flow begins with three distinct starting points: 'Upload Started', 'Update Started', and 'Delete Started', corresponding to the three primary actions. The information is first sent to an 'Information Checker' when uploading or updating a product. If the data provided is invalid, the flow circles back to the respective 'Upload Started' or 'Update Started' states. The product data is saved if the information is valid, signified by the 'Product Saved' state. On the other hand, when deleting a product, a 'Confirmation' phase ensures the user's decision. If the user confirms the deletion, the process leads to the 'Product Deleted' state. However, if the user cancels the deletion at this juncture, the flow reverts to the 'Deletion Canceled' state and terminates the delete operation.

3.4. Complaint Active/Check/Delete

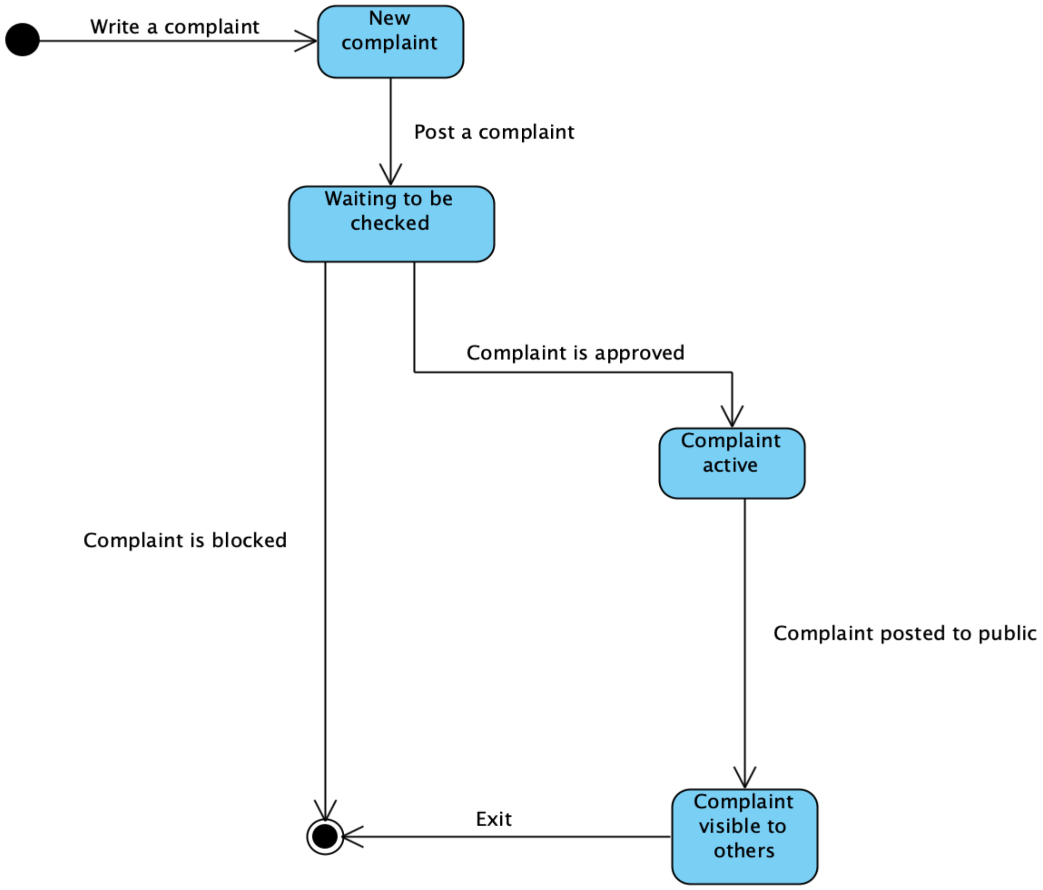

Figure 16: State diagram of complaint handling

The state diagram presented showcases the complaint module of a web application. At the start, a user has the option to "Write a complaint," moving the system to the "New complaint" state. From here, the user can choose to "Post a complaint", which prompts the "Complaint checker" to take a look at the submission. Based on this review, the complaint might be labeled as "Complaint inactive" if it doesn't meet certain criteria, or recognized as "Complaint active" if it aligns with guidelines. In the scenario where a complaint becomes inactive, it eventually gets to a point where it is "Complaint deleted." On the flip side, a greenlit active complaint progresses to the stage of being "posted to the public," making the "Complaint visible to others." This step allows fellow users or members to see and possibly engage with the shared concern, fostering a sense of community interaction. Once all these stages are navigated, the system wraps up its workflow, leading to the "Exit" state.

3.5. State Diagram of Message

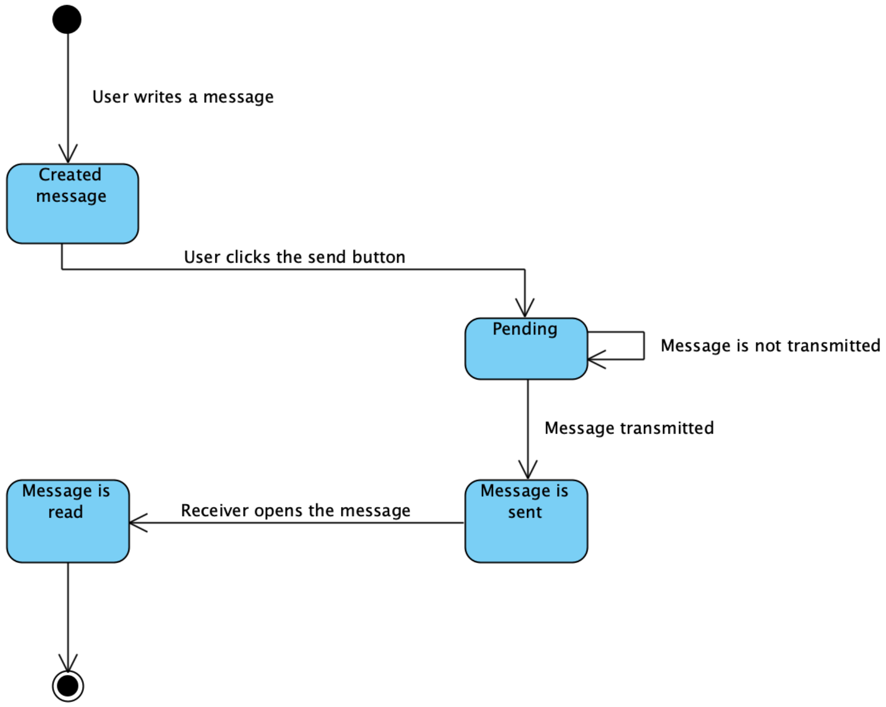

Figure 17: State diagram of the message

The state diagram presented showcases the messaging system in the web application. It starts with the “User writes a message” and directs it to the “Created message” state. When the user clicks the send button, the message is moved to a "Pending" state as it awaits transmission. If transmission fails, it remains pending; otherwise, it progresses to the "Message is sent" state upon successful transmission. The final transition occurs when the receiver opens the message. This changes its state to "Message is read," which completes the messaging process.

3.6. State Diagram of Notification

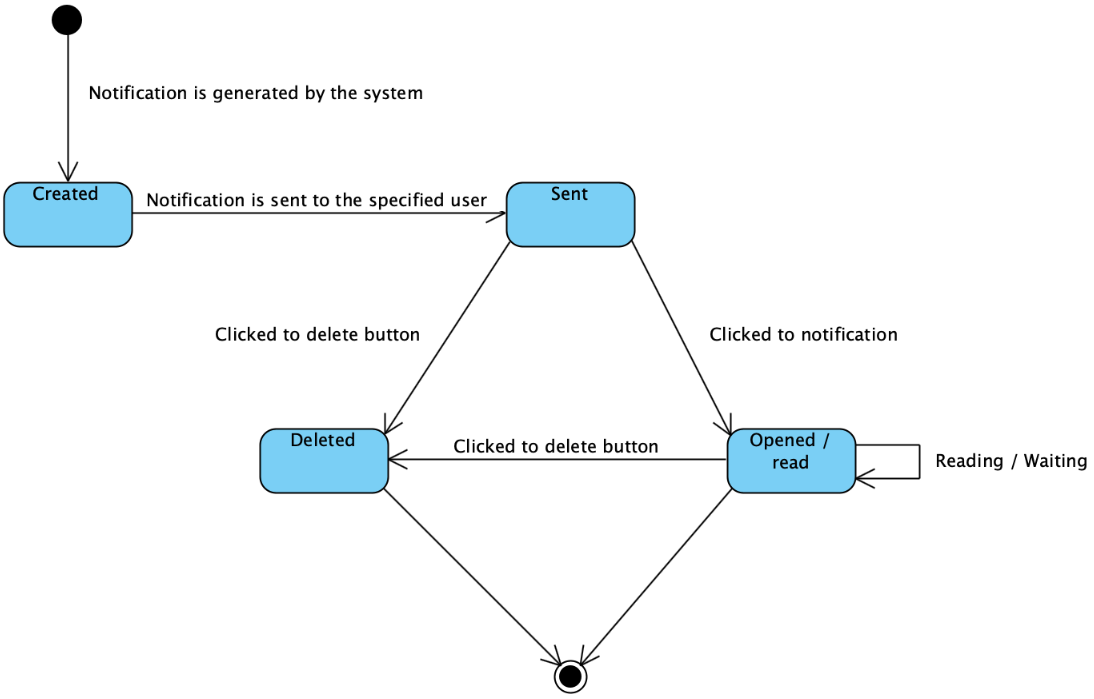

Figure 18: State diagram of notification

At the start, a notification is generated by the system in response to a specific action, such as sending a message or a borrow request. Subsequently, the state diagram transitions to the 'Created' state. Following this, the notification is dispatched to the specified user by the notification system. Once this process is completed, the state diagram moves to the 'Sent' state. In the 'Sent' state, the user has two options: either to click the delete button or to click on the notification to read it in detail. If the notification is opened, the state diagram transitions to the 'Opened' state, where the user can continue to read or wait, eventually leading to the end of the state diagram. Alternatively, if the notification is deleted, the state diagram directly reaches the end after the deletion.

4. Class Diagram

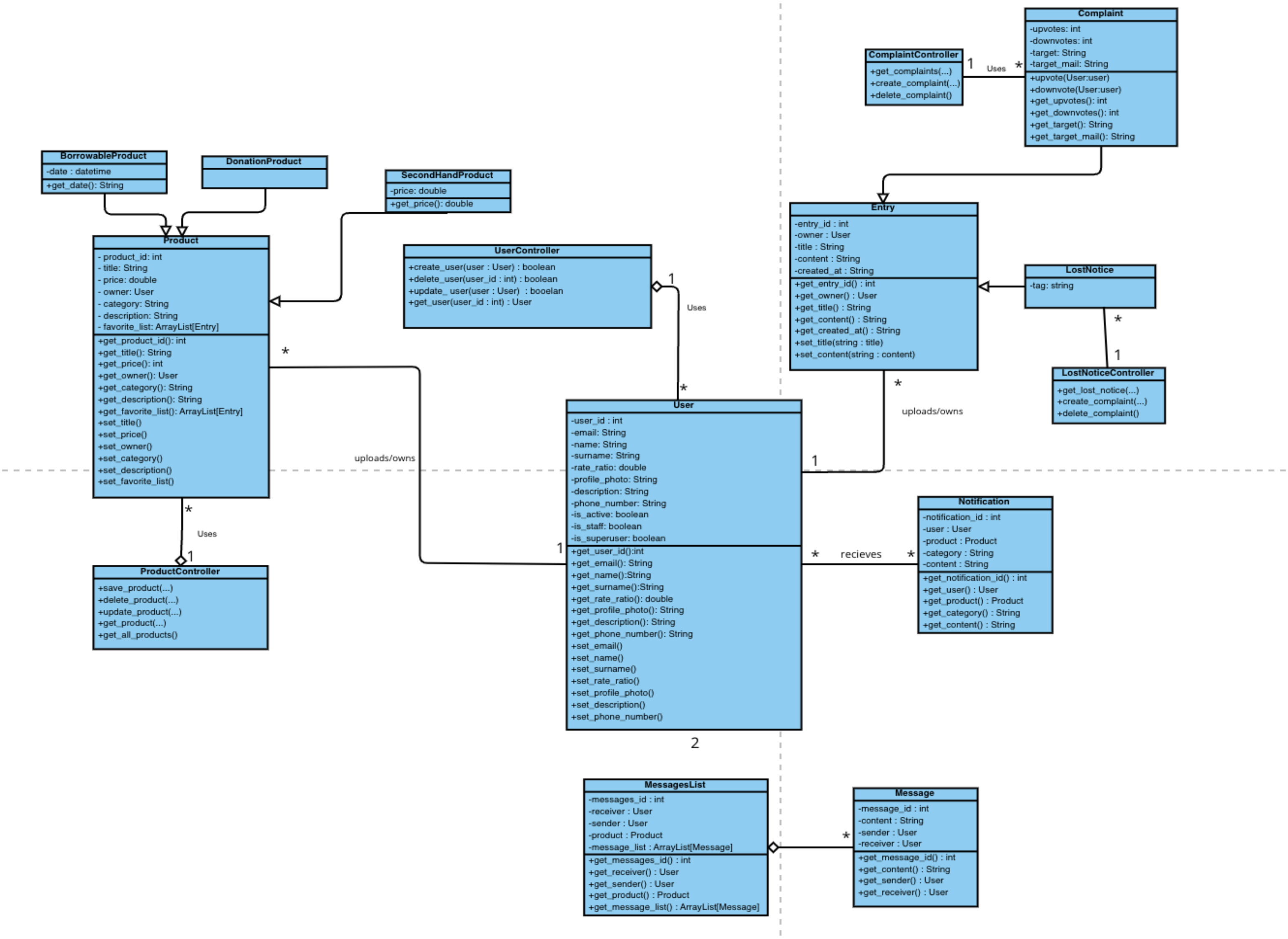

Figure 19: Class diagram of the web application

5. Mockups

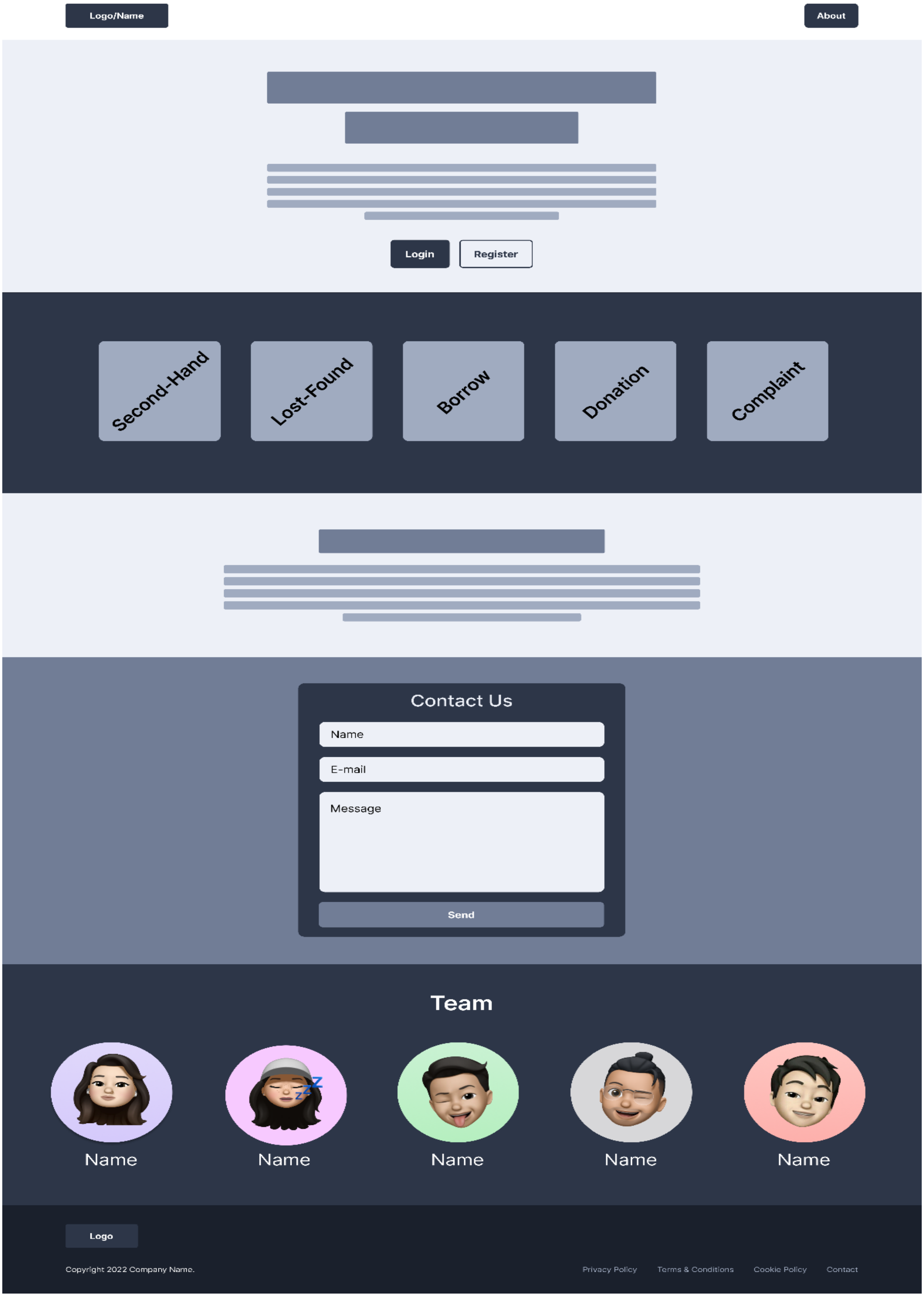

Figure 20: Landing Screen

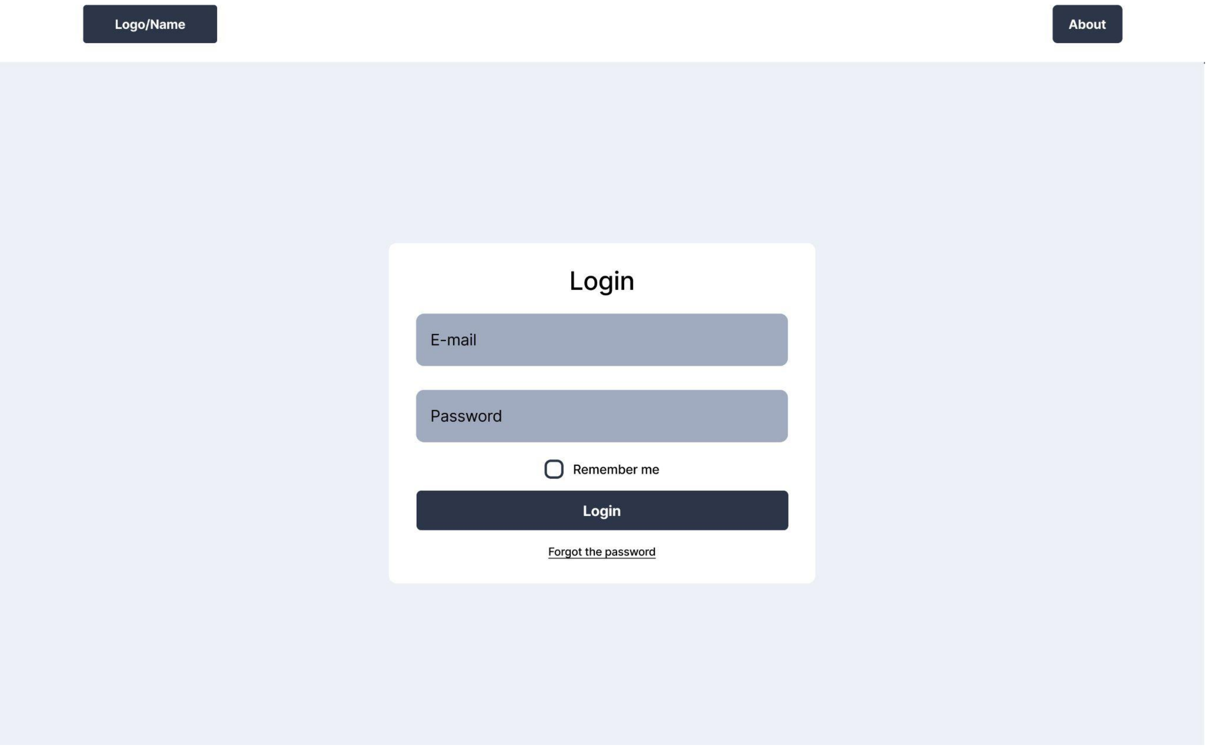

Figure 21: Login screen

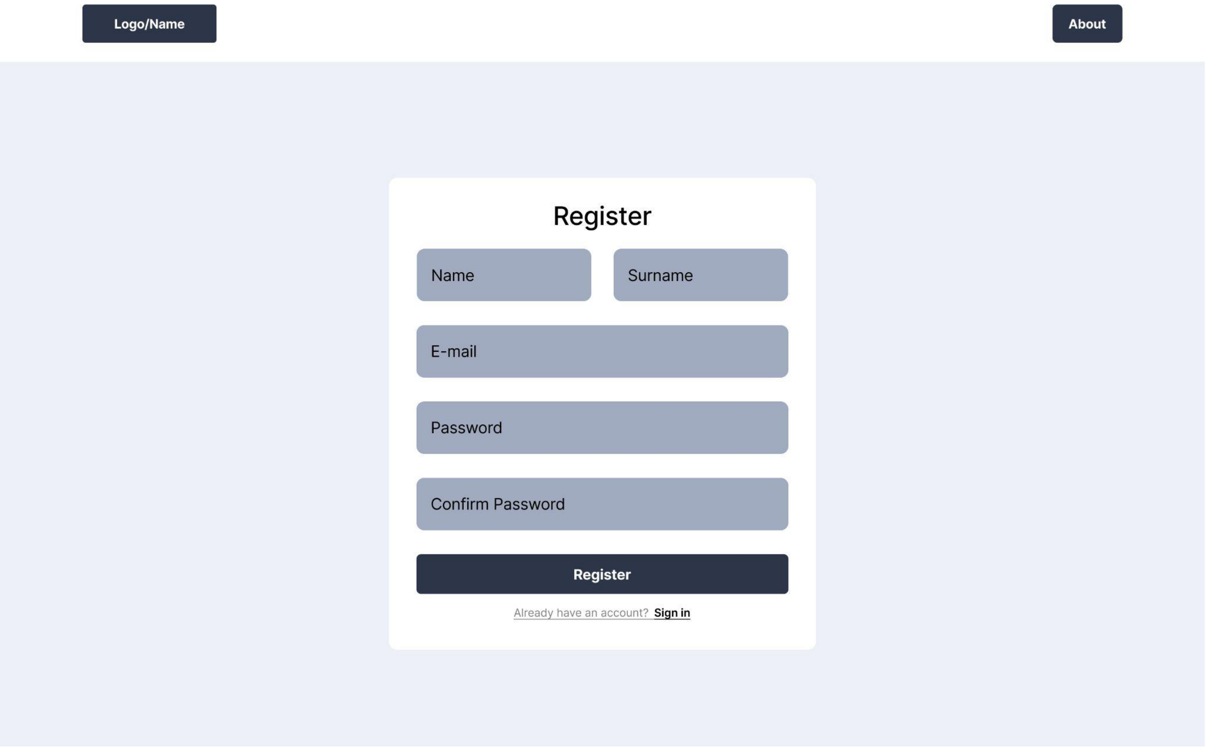

Figure 22: Registration screen

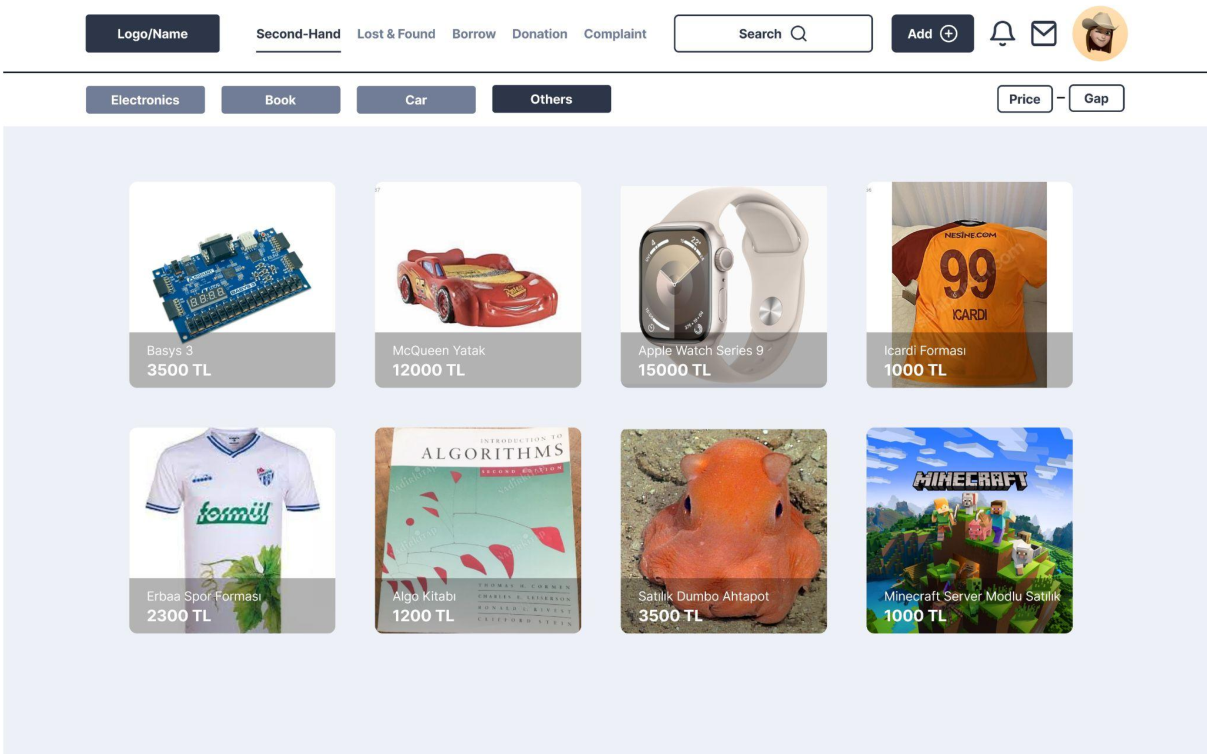

Figure 23: Main screen

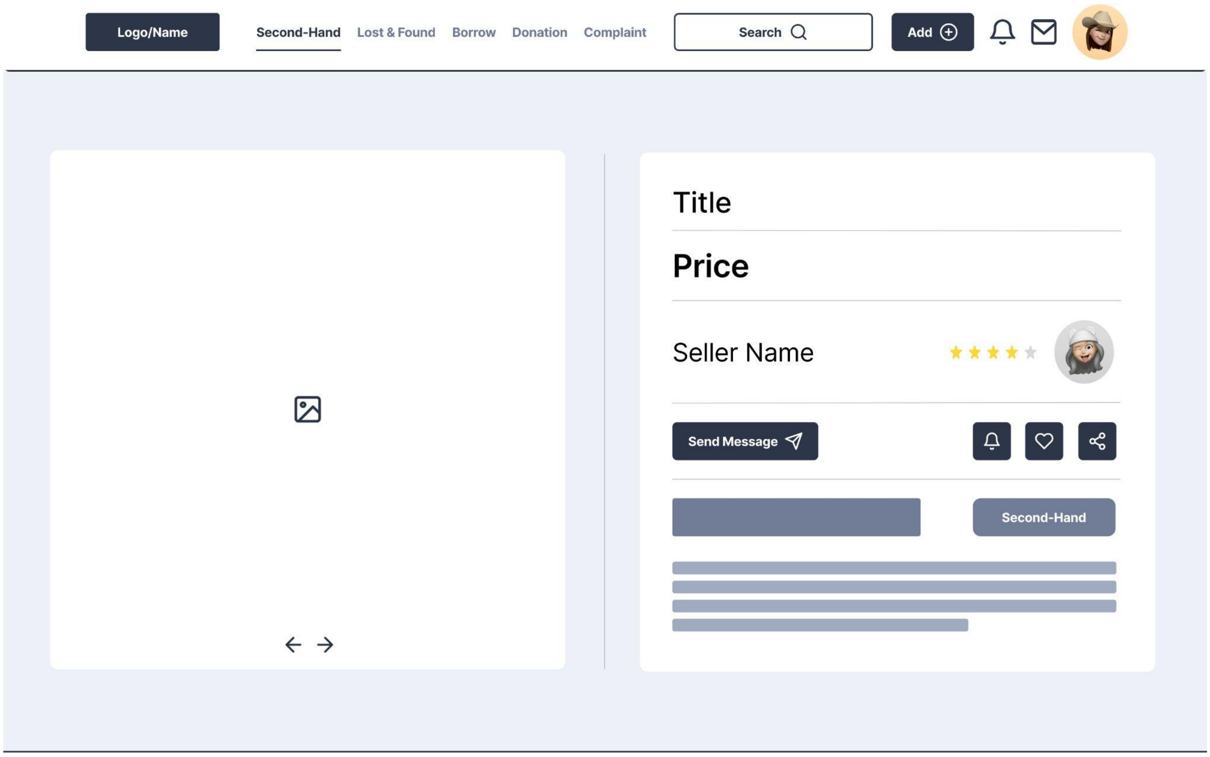

Figure 24: Product Information Page

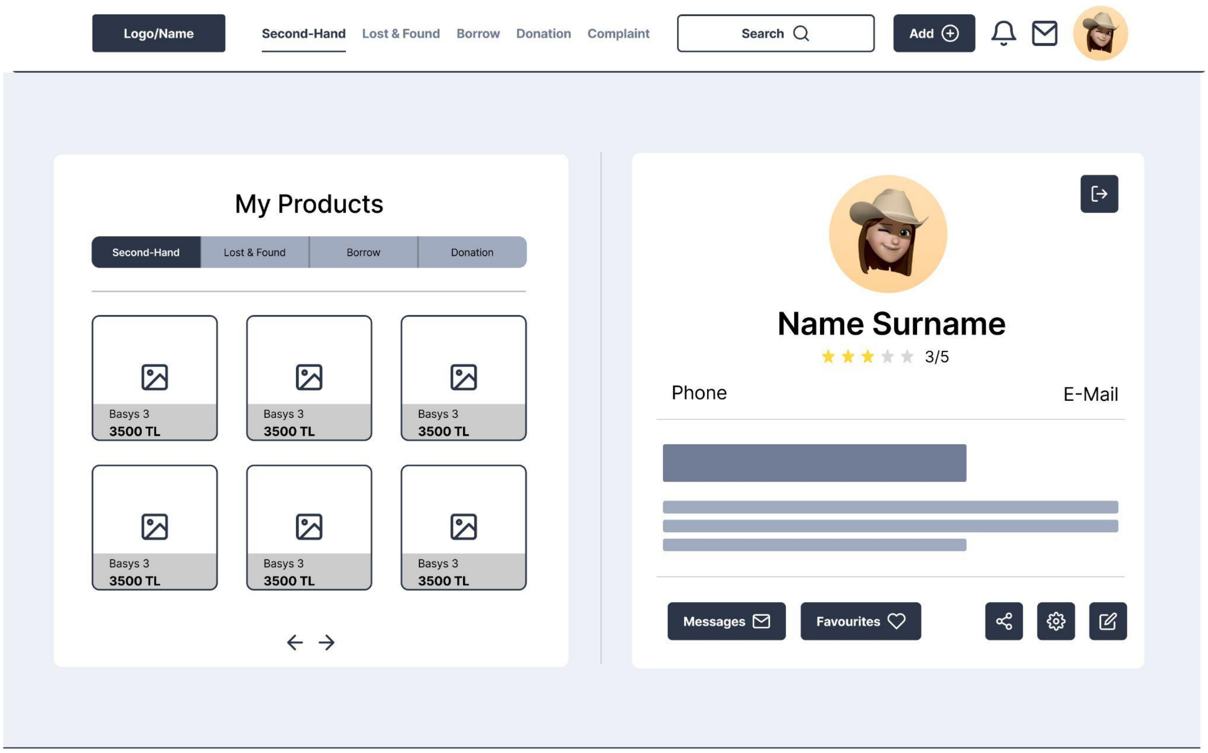

Figure 25: Profile Page

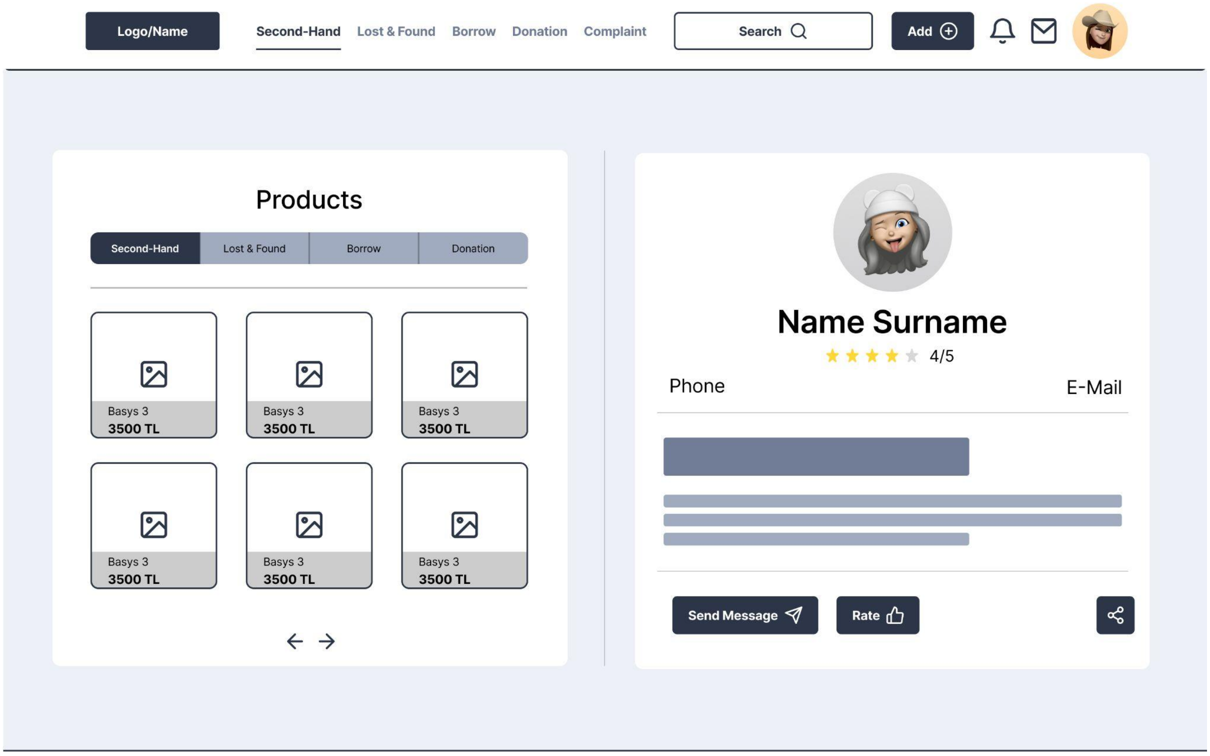

Figure 26: User Information Page

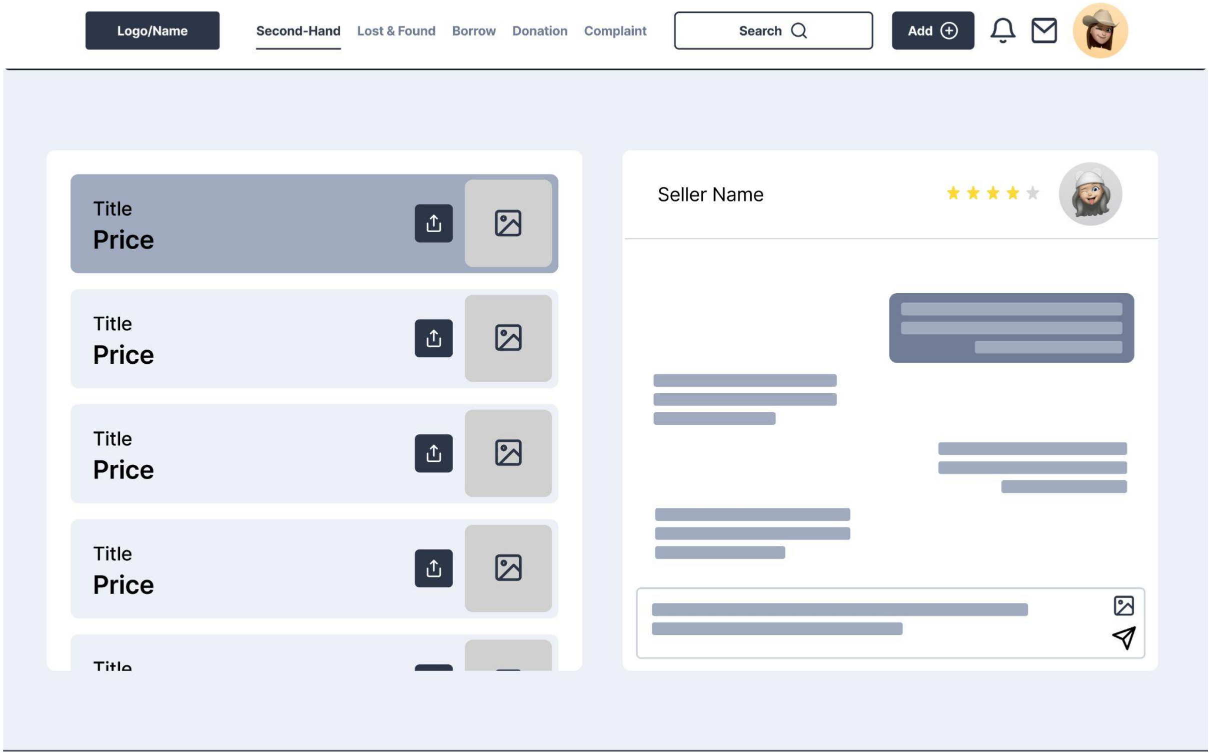

Figure 27: Messages Page

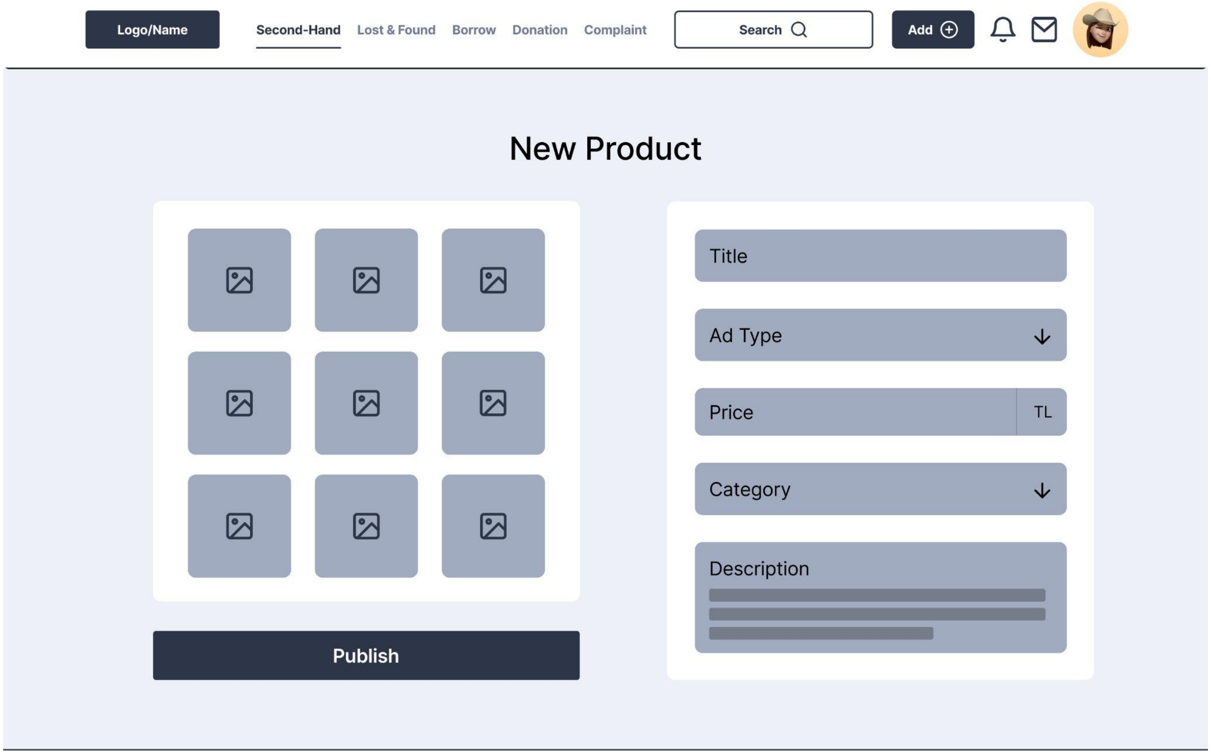

Figure 28: Product Adding Page

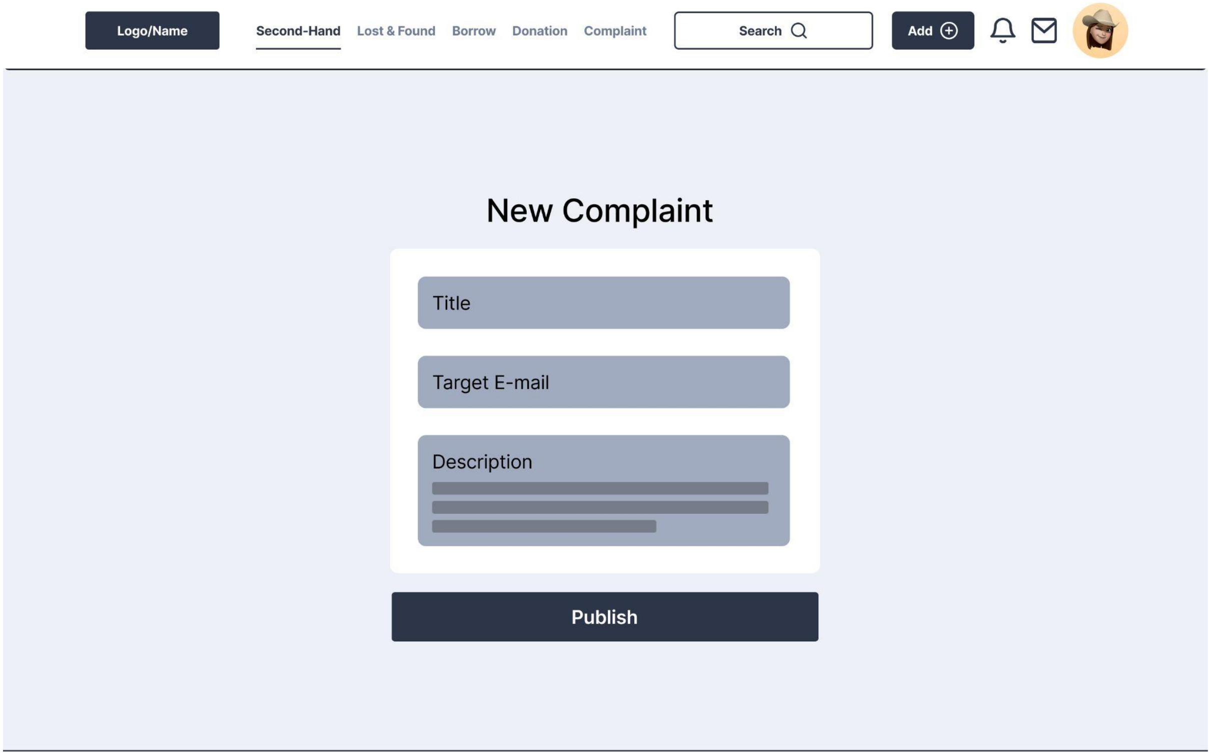

Figure 29: Complaint Adding Page

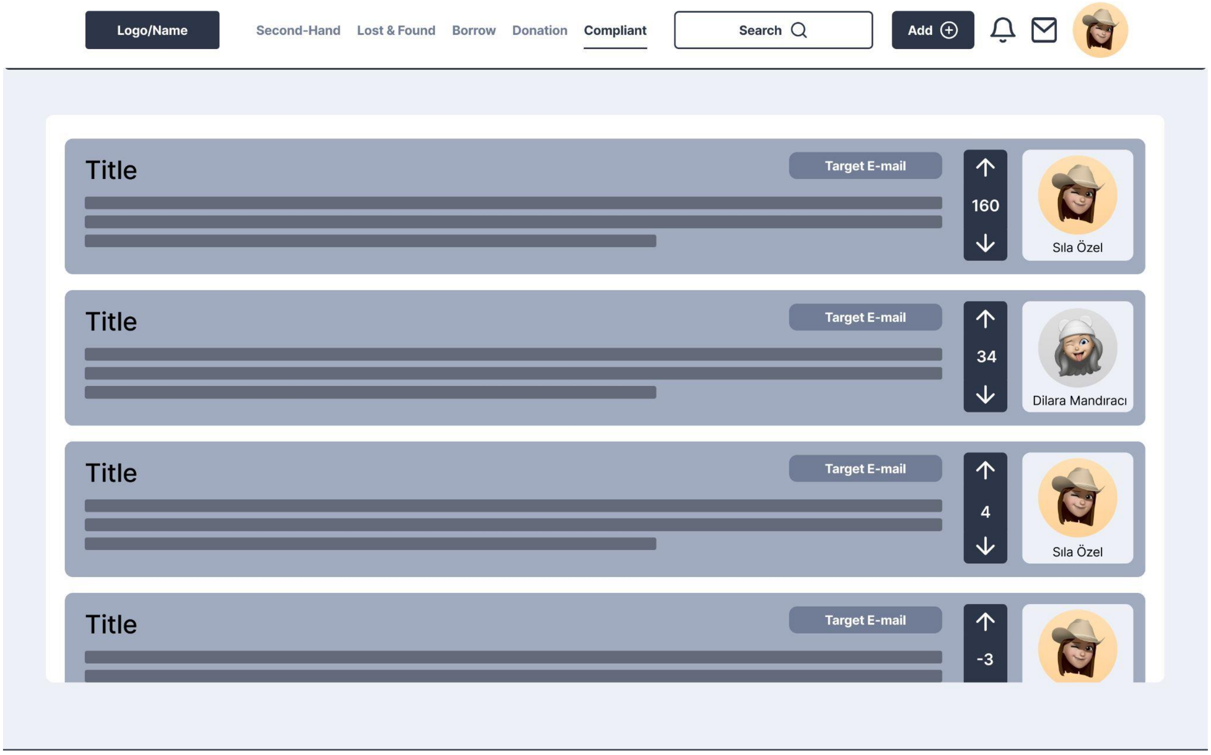

Figure 30: Complaints Screen