BILBOARD

Team-T4 BugBunny

Use Cases & Non-Functional Requirements & Tech. Stack

DILARA MANDIRACI
BURAK DEMIREL
YUSUF TORAMAN
SILA ÖZEL
EREN HAYRETTIN ARIM

Fall 2023

1. Introduction

This project is a web application that can be accessed only by Bilkent staff for four main operations: buying or selling second-hand products, donating and borrowing things, and posting lost item notices. Additionally, we added a complaint system so that the students could write their complaints freely, and their complaints would not be left unseen.

The students can sell their secondhand products to other students, make donations if they think someone might need that item, post lost notices without having to worry about what to do when they see a lost ID on the road, borrow things that they need for a short time, and finally they can share their complaints.

As can be seen from the above operations, our university does not have an official media for these purposes. With our web application, we will provide secure media for Bilkent students to do these exchanges with peace of mind.

2. Non-Functional Requirements (NFRs)

2.1. Usability

To make our application easy to use, we tried to create a simple user interface design. Here are the things that we have planned:

- As the user enters our application, they will encounter an About page where we inform the user about our application's goals and modules. The user can use the buttons on the navigation bar to navigate to login, register, or about the page.
- Our application will be a responsive web application that can be used on many devices.

2.2. Security

The main purpose of this application is to create a secure media for Bilkent students to exchange items. Also, we are storing some information about the students (such as their mail address, password, etc.) in a secure way so there won't be any data leaks.

- We will store the user's password encrypted.
- The user can access second hand sales, donation, borrow, lost and found modules, and complaint system only if they are authenticated.
- Only Bilkent staff can use this application. So, they must register with their Bilkent mail address. A verification link will be sent to verify the user's mail address.
- Each user can only have one account so this will reduce the frauds.

- In our complaint system, we will use tools that check inappropriate words so that the users cannot misuse this system.

2.3. Maintainability

We have decided to implement our frontend and backend in a way that will make our application more maintainable and easy to debug and extend.

- We will have a decoupled frontend and backend code. So, they will only communicate through data. That way, a change in the backend code will not require much change in the frontend code and vice versa. Also, debugging will be easier that way because the concerns are separated.
- Also, we will use the MVC (Model View Controller) design pattern in this application, which will separate the backend into more maintainable parts. Debugging will be much easier since each part will have a certain purpose.

2.4. Performance

We will consider our application performance, functions will be implemented in an efficient way to make our program faster.

- JWT-based authentication is faster, the system responds in a predefined time for tokens (e.g., 10000 sec)
- Django has some predefined efficient functions that we can directly use from the library.

2.5. Localization

Our application can only be used by Bilkent members.

- Users can only register with ‘Bilkent’ extended emails: @bilkent.edu.tr or @bilkent.ug.edu.tr. It means all features are only targeted at Bilkent University members and students.
- BilBoard has a complaint system that is designed to be used to address the problems in Bilkent University.

2.6. Availability

We have no time restrictions on the use of our application. Users can use the application 24/7. We will notify users if we plan to perform maintenance on the application.

3. Tech Stack

Backend: Django, Django REST, Python

Instead of creating everything from scratch for web application creation, we researched several frameworks and decided to use Django. The Django framework has built-in functions to develop applications such as User models and views. Also, there are many resources about this so that we can learn from different resources like YouTube videos and Udemy courses. In addition, Django has good documentation that we can find answers to our questions quickly, so we decided to go with this framework. Lastly, it implements many operations securely.

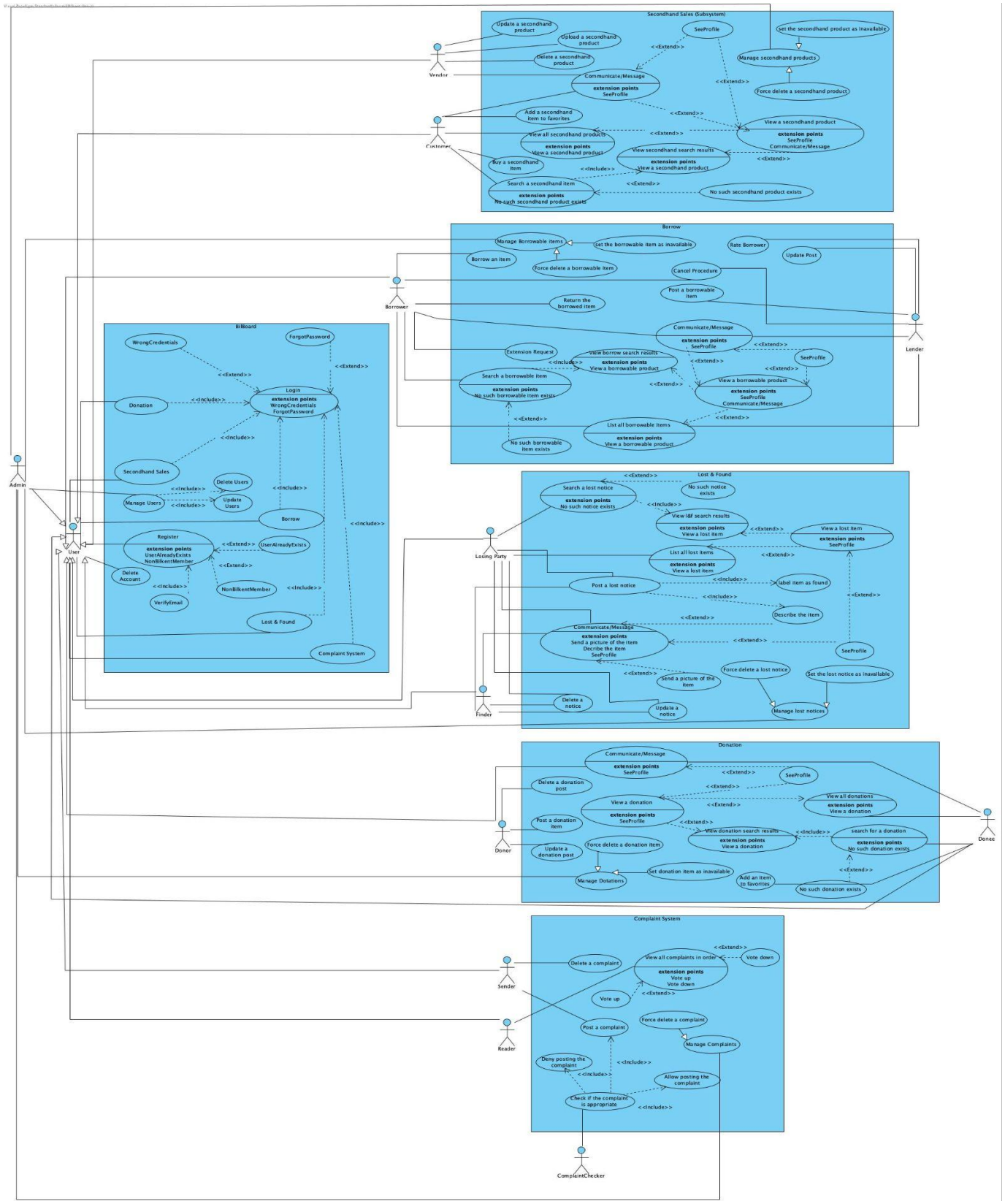
Frontend: React.js, JavaScript, HTML CSS, Bootstrap

React JS is an efficient frontend framework with a component-based structure that makes managing frontend items easy. We created our page designs with Bootstrap Studio and then used those designs to implement the user interface. While implementing the user interface, we used the Bootstrap CSS framework to make our React components look nicer.

Database: PostgreSQL

We decided to use PostgreSQL as our database since, as a result of our research, we found that the most compatible database with Django is PostgreSQL because Django officially supports it. So, we all agreed on using this database. Also, row-column tables are helpful to check our database operations.

4. Use Case Diagram



BilBoard Package

1. **Name:** Login
2. **Participating Actor:** Bilkent Member
3. **Entry Condition:** The user opens the web application and clicks the “Login” button.
4. **Exit Condition:** The user logs in successfully or encounters a problem.
5. **Flow of Events:**
   - **5.1.** The user enters credentials designated as 'required' (email, password) to log in.
     - **5.1.1.** If credentials are incorrect, the use case extends the “Wrong Credentials" case. The related warning message is displayed for the user.
   - **5.2.** If the credentials are correct, the user logs in successfully.
   - **5.3.** If the user cannot remember the credentials, the user uses the forgot password option and changes his/her password again via a verification sent to the user’s specified email.

1. **Name:** Register
2. **Participating Actor:** Bilkent Member
3. **Entry Condition:** The user opens the web application and clicks on the register button to create a new account.
4. **Exit Condition:** The user successfully created an account, navigates to another page, or cancels the registration.
5. **Flow of Events:**
   - **5.1.** User provides necessary details, such as email, password, and any other required personal information.
   - **5.2.** Else if the email is already registered:
     - **5.2.1.** The system displays a "User already exists" message and directs the user to the login screen.
     - **5.2.2.** Else if the extension of the email is not correct, indicates that the user is not from Bilkent.
     - **5.2.3.** The system displays a “Your email address should include Bilkent extension” message.
   - **5.3.** Else, a verification email is sent to the user's email address for confirmation.
   - **5.4.** The user's account is created.

1. **Name:** Secondhand Sales
2. **Participating Actor:** Customer, Vendor
3. **Entry Condition:** After logging in, the user goes to this page by clicking the Secondhand sales button**.**
4. **Exit Condition:** Navigating another page, or leaving the web application.
5. **Flow of Events:**
   - **5.1.** Secondhand sales subsystem has its own use cases, and it is explained in the previous sections.
     - **5.1.1.** If the user loses their access to the page after a predefined time, an error message is displayed.

1. **Name:** Complaint System
2. **Participating Actor:** Sender, Reader, Complaint Checker
3. **Entry Condition:** After logging in, the user goes to this page by clicking the Complaint button**.**
4. **Exit Condition:** Navigating another page, or leaving the web application.
5. **Flow of Events:**
   - **5.1.** The Complaint System subsystem has its own use cases, and it is explained in the previous sections.
     - **5.1.1.** If the user loses their access to the page after a predefined time, an error message is displayed.

1. **Name:** Lost & Found
2. **Participating Actor:** Losing party, Finder
3. **Entry Condition:** After logging in, the user goes to this page by clicking the Lost & Found button.
4. **Exit Condition:** Navigating another page, or leaves the web application.
5. **Flow of Events:**
   - **5.1.** The Lost & Found subsystem has its own use cases, and it is explained in the previous sections.
     - **5.1.1.** If the user loses their access to the page after a predefined time, an error message is displayed.

1. **Name:** Borrow
2. **Participating Actor:** Borrower, Lender
3. **Entry Condition:** After logging in, the user goes to this page by clicking the Borrow button**.**
4. **Exit Condition:** Navigating another page, or leaving the web application.
5. **Flow of Events:**
   5.1. The Borrow subsystem has its own use cases, and it is explained in the previous sections.
      5.1.1. If the user loses their access to the page after a predefined time, an error message is displayed.

1. **Name:** Donation
2. **Participating Actor:** Donator, Donee
3. **Entry Condition:** After logging in, the user goes to this page by clicking the Donation button**.**
4. **Exit Condition:** Navigating another page, or leaving the web application.
5. **Flow of Events:**
   5.1. The donation module acts between the donation and the Donor. It has a product functionality similar to Second-Hand sales and Borrow modules. It has basic features such as User View, Update, Upload, and Post.
      5.1.1. If the user loses their access to the page after a predefined time, an error message is displayed.

Secondhand Sale Package

1. **Name:** Upload a Secondhand Product
2. **Participating Actor:** Vendor
3. **Entry Condition:** Clicking the add product button.
4. **Exit Condition:** Clicking the “Publish” or “Back” button, navigating another page.
5. **Flow of Events:**
   - **5.1.** Vendor decides to upload a new product.
   - **5.2.** The vendor clicks the add product button.
   - **5.3.** The vendor provides the necessary details such as product name, description, price, images, etc.
   - **5.4.** The vendor will decide to publish the product or cancel uploading.
     - **5.4.1.** If the vendor clicks the publish button, the system adds the new product.
     - **5.4.2.** If the vendor clicks the back button, uploading will be canceled.

1. **Name**: Update a Secondhand Product
2. **Participating Actor**: Vendor
3. **Entry Condition**: Navigate to the profile, select the product, and click on the edit button.
4. **Exit Condition**: Clicking the “Save” or “Cancel” button or navigating to another page.
5. **Flow of Events**:
   - **5.1.** Vendor selects the product they wish to update and clicks the edit button.
   - **5.2.** Vendor modifies the necessary product details.
   - **5.3.** Vendor will decide to save or cancel changes
     - **5.3.1.** If the vendor cancels the changes, the system does not update the product details.
     - **5.3.2.** Else the vendor saves the changes, the system updates the product details.

1. **Name:** Delete a Secondhand Product
2. **Participating Actor:** Vendor
3. **Entry Condition**: Entering the product page and clicking the “Delete” button
4. **Exit Condition:** Navigation to another page or removing the product successfully.
5. **Flow of Events:**
   - **5.1.** Vendor decides to remove a listed product from the platform.
   - **5.2.** The vendor selects the product they wish to delete.
   - **5.3.** Vendor clicks the delete button.

5.4. Confirmation will be received from vendors to delete the product by clicking the confirm button.

5.4.1. If vendors confirm their intention, the selected product will be removed from the platform.

5.4.2. Else deletion will be canceled

1. **Name:** See Profile
2. **Participating Actor:** Vendor, Customer
3. **Entry Condition:** Clicking the profile image.
4. **Exit Condition:** Navigation to another page.
5. **Flow of Events:**
   - **5.1.** Users (either vendor or customer) want to view their profile.
   - **5.2.** The user clicks the profile image to view their profile.
   - **5.3.** The system displays the user's profile details.

1. **Name:** Communicate/Message
2. **Participating Actor:** Vendor, Customer
3. **Entry Condition:** Navigating to the messaging page.
4. **Exit Condition:** Navigating to another page or message is successfully sent.
5. **Flow of Events:**
   - **5.1.** User (either vendor or customer) wants to send a message to the other (either vendor or customer) user.
   - **5.2.** The user clicks on the send message button.
   - **5.3.** The user is navigated to the message screen.
     - **5.3.1.** If the user navigates to another page, the process will be concluded.
   - **5.4.** Else, the user composes the message and sends it.
   - **5.5.** The system notifies the recipient of the new message.

1. **Name:** Add an Item to Favorites
2. **Participating Actor:** Customer
3. **Entry Condition:** Entering the product page and clicking the related button.
4. **Exit Condition:** The product is added to the customer's favorites.
5. **Flow of Events:**
   - **5.1.** Customer is interested in a product and wishes to save it for later.
   - **5.2.** Customer selects a product and enters the product page.
   - **5.3.** Customers click the save button to add the product to their favorites.
   - **5.4.** The product will be added to the customer's favorites list.

1. **Name:** View All Products
2. **Participating Actor:** Customer
3. **Entry Condition:** Entering to the “Secondhand” page.
4. **Exit Condition:** Navigating to another page.
5. **Flow of Events:**
   - **5.1.** Customer enters the second-hand page.
   - **5.2.** The system fetches the list of all products from the database.
     - 5.2.1. If there is no item in the system, an informational message is shown.
   - **5.3.** Else, the system displays the products in a list or grid format, showing essential details like product name, image, and price.
   - **5.4.** Customers can scroll through and navigate the list.

1. **Name:** View a Secondhand Product
2. **Participating Actor:** Customer
3. **Entry Condition:** Clicking on a product.
4. **Exit Condition:** Navigating to another page or clicking the back button.
5. **Flow of Events:**
   - **5.1.** Customer selects a specific product from the list or search results.
   - **5.2.** System fetches detailed information about the product from the database.
   - **5.3.** The system displays the product details, including product name, description, images, price, and other relevant information.
     - **5.3.1.** Customers choose to take further actions such as adding to favorites, communicating with the vendor, or purchasing the product.

1. **Name:** Buy a Secondhand Product
2. **Participating Actor:** Customer
3. **Entry Condition:** Sending a message to the vendor.
4. **Exit Condition:** Reaching an agreement with the vendor or not.
5. **Flow of Events:**
   - **5.1.** Customer decides to purchase a product.
   - **5.2.** Customer sends a message to the vendor to deal with.
   - **5.3.** Vendor approves customer’s request.
     - 5.3.1. If the vendor approves the customer’s request, the customer will buy the item.
     - 5.3.2. If the vendor rejects the customer’s request, the purchase will be canceled.

1. **Name:** Search for a Secondhand Product
2. **Participating Actor:** Customer
3. **Entry Condition:** Searching for a product with a name or filter.
4. **Exit Condition:** Search results are displayed to the customer.
5. **Flow of Events:**
   - **5.1.** Customer inputs search criteria.
   - **5.2.** System searches the database for matching products.
     - **5.2.1.** If the item is not found, No Such Item is found, exception is handled. Then, the system suggests related products to customers.
     - **5.2.2.** Else, search results are displayed to the customer.

Donation Package

1. **Name:** View All Donations
2. **Participating Actor:** Donee
3. **Entry Condition:** Entering the “Donation” page.
4. **Exit Condition:** Navigating to another page.
5. **Flow of Events:**
   - **5.1.** Donee enters the donation page from the main page.
   - **5.2.** The system fetches the list of all donations from the database.
     - **5.2.1.** If there is no donation in the system, the No Such Donation scenario is triggered.
     - **5.2.2.** Else, the system displays the donations in a list or grid format, showing essential details like product name, image, and price.
   - **5.3.** Donee can scroll through and navigate the list.

1. **Name:** View a Donation
2. **Participating Actor:** Donee
3. **Entry Condition:** Clicking on a specific donation.
4. **Exit Condition:** Navigating to another page or clicking to the back button.
5. **Flow of Events:**
   - **5.1.** Donee enters the donation page
   - **5.2.** The donee selects a specific donation from the list or search results.
   - **5.3.** The system displays the donation details, including the donation name, description, images, and any other relevant information.
     - **5.3.1.** Donee chooses to take further actions, such as adding to favorites or communicating with the Donor.
     - **5.3.2.** Donee may turn back to the previous page.

1. **Name:** Search for a Donation
2. **Participating Actor:** Donee
3. **Entry Condition:** Searching for a donation with a name or filter.
4. **Exit Condition:** Navigating to another page or canceling the search process.
5. **Flow of Events:**
   - **5.1.** Donee inputs search criteria.
   - **5.2.** System searches for matching donations with related filters.
     - **5.2.1.** If the donation is not found, No Such Donation is found, exception is handled. Then, the system suggests related donations to customers.
     - **5.2.2.** Else, search results are displayed to the customer.

1. **Name:** Communicate/Message
2. **Participating Actor:** Donor/Donee
3. **Entry Condition:** By clicking the related message button.
4. **Exit Condition:** Navigating to another page or canceling the procedure.
5. **Flow Of Events:**
   - **5.1.** Donor/Donee is directed to the related messaging page.
   - **5.2.** The system gathers the data of the previous messages, if there are any, from the system.
   - **5.3.** The system displays the messages, if there are any, in an appropriate messaging page.
   - **5.4.** Donor/Donee either sends another message or just checks previous messages, if there are any.
     - **5.4.1.** If Donor/Donee sends a new message then the system updates related segments.

1. **Name:** Update a Donation
2. **Participating Actor:** Donor
3. **Entry Condition:** Clicking to the “update an item” button on the “Donation” page.
4. **Exit Condition:** Navigating to another page or canceling the procedure or publishing the updated version of the donation.
5. **Flow Of Events:**
   - **5.1.** Donor opens the intended donation target.
   - **5.2.** System gathers and displays the data of donation on the information page.
   - **5.3.** The donor modifies the properties of the donation by changing related fields. Then save the changes.

1. **Name:** Delete a Donation
2. **Participating Actor:** Donor
3. **Entry Condition:** Selecting the intended donation.
4. **Exit Condition:** Navigating to another page or canceling the procedure or deleting the donation.
5. **Flow Of Events:**
   - **5.1.** Donor opens the intended donation's page.
   - **5.2.** System gathers and displays the donations on an information page.
   - **5.3.** Donor clicks the delete button to start the deleting procedure.
     - **5.3.1.** If the Donor confirms the deletion, the donation is deleted from the database.
     - **5.3.2.** Else, the donor is directed to the donation back.

1. **Name:** Post a Donation
2. **Participating Actor:** Donor
3. **Entry Condition:** Clicking to "Publish" on the "Donation" page.
4. **Exit Condition:** Navigating to another page or canceling the procedure or publishing the post.
5. **Flow Of Events:**
   - **5.1.** Donors fill up the required fields about donation such as name, image, product health, recommended course(optional), description, etc.
     - **5.1.1.** If required fields are correctly filled by the Donor, the product is uploaded to the system.
     - **5.1.2.** Else, the system throws an exception about the error.
   - **5.2.** The system creates the donation on a database and publishes the donation.

Borrow Package

1. **Name:** View All Loanables
2. **Participating Actor:** Borrower, Lender
3. **Entry Condition:** Entering the "Borrow" page.
4. **Exit Condition:** Navigating to another page.
5. **Flow Of Events:**
   - **5.1.** Borrower/Lender enters the "Borrow" page.
   - **5.2.** The system fetches the list of all donations from the database.
     - **5.2.1.** If there is no donation in the system, the No Active Donation scenario is triggered.

5.2.2. Else, the system displays the donations in a list or grid format, showing essential details like product name, image, and price.

5.3. Borrower/Lender can scroll through and navigate the list.

1. **Name:** Borrow
2. **Participating Actor:** Borrower
3. **Entry Condition:** Selecting a specific loanable.
4. **Exit Condition:** Navigating to another page or clicking to the back button.
5. **Flow Of Events:**
   - **5.1.** Borrower selects and opens the intended loanable's information page.
   - **5.2.** System gathers and displays the data of loans.
   - **5.3.** Borrower examines and sends a borrow request to the lender.
   - **5.4.** A notification is delivered to Lender's account.

1. **Name:** Cancel Procedure
2. **Participating Actor:** Borrower, Lender
3. **Entry Condition:** Selecting a borrowed loanable request.
4. **Exit Condition:** Navigating to another page or canceling the current status (borrowed) of the loanable.
5. **Flow Of Events:**
   - **5.1.** Borrower/Lender opens a borrow request affiliated with its account.
   - **5.2.** System gathers and displays the data of loans.
   - **5.3.** Borrower/Lender clicks the cancel button to cancel the borrowing request.
     - **5.3.1.** If the lender confirms the cancellation, the procedure is done.
     - **5.3.2.** Else, the lender is directed to the previous page.
   - **5.4.** A notification is delivered to Borrower's/Lender's account.

1. **Name:** Return
2. **Participating Actor:** Borrower
3. **Entry Condition:** By selecting the borrow request will be returned.
4. **Exit Condition:** Navigating to another page or canceling the procedure or returning the loanable.
5. **Flow Of Events:**
   - **5.1.** Borrower opens the intended borrow request affiliated with its account.
   - **5.2.** System gathers and displays the data of loans.
   - **5.3.** Borrower clicks the return button to return the loanable.
   - **5.4.** A notification is delivered to Borrower's/Lender's account.

5.4.1. Once Lender approves the return request, the system finalizes the borrowing request.

1. **Name:** Post a Loanable
2. **Participating Actor:** Lender
3. **Entry Condition:** Clicking the “post” button on the “Borrow” page.
4. **Exit Condition:** Navigating to another page or canceling the procedure or publishing the post.
5. **Flow Of Events:**
   - **5.1.** Lenders fill up the required fields about loanable assets such as name, image, anticipated borrowing period, recommended course, and description.
   - **5.2.** System publishes the loans.

1. **Name:** Update a Loanable
2. **Participating Actor:** Lender
3. **Entry Condition:** Selecting the intended loanable.
4. **Exit Condition:** Navigating to another page or canceling the procedure or updating the post.
5. **Flow Of Events:**
   - **5.1.** Lender opens the intended loanable’s page.
   - **5.2.** System gathers and displays the data of loans on an information page.
   - **5.3.** The lender modifies the properties of the loanable by changing related fields. Then save the changes.
     - **5.3.1.** If the user fills some of the fields incorrectly, the system gives an error and then directs the user to the beginning of step 5.3
   - **5.4.** Once modifications are done, the system updates the properties of the loans.

1. **Name:** Rate Borrow
2. **Participating Actor:** Lender
3. **Entry Condition:** Selecting the intended borrowing request that was finalized.
4. **Exit Condition:** Navigating to another page or canceling the procedure or rating the Borrower.
5. **Flow Of Events:**
   - **5.1.** Lender selects and opens the intended borrowing request’s information page.
   - **5.2.** The system gathers and displays the data of the borrowing request on an information page.

5.3. The lender sets a rate from 1 to 5 and saves it.
5.4. A notification is delivered to the Borrower's account.

1. **Name:** Communicate/Message
2. **Participating Actor:** Borrower/Lender
3. **Entry Condition:** By clicking the related message button.
4. **Exit Condition:** Navigating to another page or canceling the procedure.
5. **Flow Of Events:**
   - **5.1.** Borrower/Lender is directed to the related messaging page.
   - **5.2.** System gathers the data of the previous messages, if there are any, from the system.
   - **5.3.** The system displays the messages, if there are any, in an appropriate messaging page.
   - **5.4.** Borrower/Lender either sends another message or just checks previous messages, if there are any.
     - **5.4.1.** If Borrower/Lender sends a new message then the system updates related segments.

1. **Name:** Search a loanable
2. **Participating Actor:** Borrower
3. **Entry Condition:** Searching for a product with a name or filter.
4. **Exit Condition:** Search results are displayed to the Borrower.
5. **Flow Of Events:**
   - **5.1.** Borrower inputs search criteria.
   - **5.2.** System searches the database for matching products.
     - **5.2.1.** If there is no matching product, the system shows an empty page with an appropriate message on it.
   - **5.3.** Search results are displayed to the customer.

1. **Name:** Delete a loanable
2. **Participating Actor:** Borrower/Lender
3. **Entry Condition:** Selecting the intended loan.
4. **Exit Condition:** Navigating to another page or canceling the procedure or deleting the loan.
5. **Flow Of Events:**
   - **5.1.** Lender opens the intended loanable's page.
   - **5.2.** The system gathers the loanable's data and displays the loanable's data on an information page.
   - **5.3.** The lender clicks the delete button to start the deleting procedure.

Complaint System Package

1. **Name:** Post a Complaint
2. **Participating Actor:** Sender, ComplaintChecker
3. **Entry Condition:** After logging in to the website, the user goes to the complaint module from the navigation bar and then uses the add product (complaint) button.
4. **Exit Condition:** The user moves from the navigation bar to another module, posts the complaint, cancels the post, or leaves the web application or fails to post the complaint.
5. **Flow of Events:**
   - **5.1.** The user clicks the add button.
   - **5.2.** After clicking the Add button, the user selects the complaint category to create a complaint.
   - **5.3.** Creates a complaint by entering the specified properties.
   - **5.4.** ComplaintChecker checks the complaint.
     - **5.4.1.** If the complaint is appropriate, the system creates the complaint.
     - **5.4.2.** Otherwise, the system will not create the complaint.

1. **Name:** Delete a Complaint
2. **Participating Actor:** Sender
3. **Entry Condition:** After logging in to the website, the user goes to the Complaint module.
4. **Exit Condition:** The user navigates to another module, cancels deletion, or deletes the complaint.
5. **Flow of Events:**
   - **5.1.** After entering the user profile, the user goes to the Dashboard section.
   - **5.2.** Find your complaint in the complaint category in the Dashboard section.
   - **5.3.** Press the delete button for the complaint and delete the complaint.

1. **Name:** Check if the complaint is appropriate
2. **Participating Actor:** Complaint Checker
3. **Entry Condition:** Posting a complaint
4. **Exit Condition:** When the check is complete (either allows or denies)
5. **Flow of Events:**
   - **5.1.** The user will create a complaint and try to post it.
   - **5.2.** The ComplaintChecker will automatically check if the complaint is appropriate.
     - **5.2.1.** If the complaint is appropriate it will be posted.
     - **5.2.2.** Otherwise, the ComplaintChecker will deny the post.

1. **Name:** View all complaints in order
2. **Participating Actor:** Reader
3. **Entry Condition:** Entering the Complaint System.
4. **Exit Condition:** User moves from the navigation bar to another module.
5. **Flow of Events:**
   - **5.1.** After entering the complaint system, the user sees the complaints in the vote order.
   - **5.2.** They can choose to upvote and downvote the complaint, which extends the view of all complaints.

1. **Name:** Vote up
2. **Participating Actor:** Reader
3. **Entry Condition:** After reviewing a complaint among the others, when the user clicks the upvote button.
4. **Exit Condition:** The user navigates to another module.
5. **Flow of Events:**
   - **5.1.** When a user sees a complaint that they agree with, they can click on the button to upvote the complaint.
   - **5.2.** If a complaint has more upvotes, it will appear more likely in the top complaints in terms of ordering the complaints since we will order according to the difference between the upvotes and the downvotes.

1. **Name:** Vote down
2. **Participating Actor:** Reader
3. **Entry Condition:** After reviewing a complaint among the others, when the user clicks the downvote button.
4. **Exit Condition:** The user navigates to another module.
5. **Flow of Events:**
   - **5.1.** When a user sees a complaint that they don’t agree with, they can click on the button to downvote the complaint.
     - **5.1.1.** If a complaint has more downvotes, it will appear more likely in the bottom complaints since the ordering will be made according to the difference between the upvote count and the downvote count.

Lost & Found Package

1. **Name:** Search a lost notice
2. **Participating Actor:** Losing party
3. **Entry Condition:** After entering the “Lost & Found” page, click to the search box.
4. **Exit Condition:** Click an item or navigate to another page or get no such notice response or get all the notices with the corresponding keywords.
5. **Flow Of Events:**
   - **5.1.** Losing party enters the “Lost & Found” page.
   - **5.2.** The losing party searches for an item that they lost.
     - **5.2.1.** If there is no matching product, the system shows an empty page with an appropriate message on it.
   - **5.3.** Filtering mechanism shows the matching notices with keywords/tags.
   - **5.4.** The losing party can scroll through and navigate the list.

1. **Name:** View a lost notice
2. **Participating Actor:** Losing party
3. **Entry Condition:** Clicking a notice while searching.
4. **Exit Condition:** Exiting the view mode or navigating to the message section with the publisher of the notice or navigating to another page.
5. **Flow Of Events:**
   - **5.1.** Losing party clicks on a specific notice.
   - **5.2.** Users can see the publisher and can text them.

1. **Name:** List all lost items
2. **Participating Actor:** Losing party
3. **Entry Condition:** Entering the “Lost & Found” page.
4. **Exit Condition:** Navigating through another page or getting no lost item is found response.
5. **Flow Of Events:**
   - **5.1.** When the user enters the lost & found page, the main screen will show all notices.
     - 5.1.1. If there is no matching product, the system shows an empty page with an appropriate message on it.
   - **5.2.** Users can scroll down and see the list of all lost items.
   - **5.3.** Or, they can search for a specific notice from the search section.

1. **Name:** Post a lost notice
2. **Participating Actor:** Losing party and Finder
3. **Entry Condition:** After entering the “Lost & Found” page, when they click the “Create” button.
4. **Exit Condition:** Navigating through another page, posting the notice or canceling the process by clicking corresponding buttons.
5. **Flow Of Events:**
   - **5.1.** When the user enters the lost & found page, they will see a button on there called something similar, “create a new lost notice.”
   - **5.2.** They can click that button to create their lost post.
     - **5.2.1.** If the user fills some of the fields incorrectly, the system gives an error and then directs the user to the beginning of the step 5.1

1. **Name:** Describe the lost item
2. **Participating Actor:** Losing party
3. **Entry Condition:** When they click the “create” button.
4. **Exit Condition:** Canceling the process or posting a notice.
5. **Flow Of Events:**
   - **5.1.** Inside the create page, they will see a text field that they can describe the item, time, place, etc.
   - **5.2.** They can post the notice or cancel the process and go back to the main page.

1. **Name:** Communicate/Message
2. **Participating Actor:** Losing party / Finder
3. **Entry Condition:** By clicking the related message button.
4. **Exit Condition:** Navigating to another page or canceling the procedure.
5. **Flow Of Events:**
   - **5.1.** Borrower/Lender is directed to the related messaging page.
   - **5.2.** The system gathers the data of the previous messages, if there are any, from the system.
   - **5.3.** The system displays the messages, if there are any, in an appropriate messaging page.
   - **5.4.** Borrower/Lender either sends another message or just checks previous messages if there are any.
     - **5.4.1.** If Borrower/Lender sends a new message then the system updates related segments.

1. **Name:** Send the picture of the lost item
2. **Participating Actor:** Losing party / Finder
3. **Entry Condition:** When they click to the share image part on the communication section.
4. **Exit Condition:** Navigating through another page or sending the message or canceling the process by clicking the corresponding buttons.
5. **Flow Of Events:**
   - **5.1.** When the losing party finds a notice about their lost product, they communicate with the finder.
   - **5.2.** After gathering enough information about the item from each other, both sides can share a photo of the item. For example, the losing party can prove that the item is theirs by sharing a previous photo of the item.

1. **Name:** Delete a lost notice
2. **Participating Actor:** Losing party and Finder
3. **Entry Condition:** Entering the “Lost & Found” page.
4. **Exit Condition:** Navigating through another page
5. **Flow Of Events:**
   - **5.1.** When the lost item is found, the Finder can delete their lost notice.

1. **Name:** Update a lost notice
2. **Participating Actor:** Losing party and Finder
3. **Entry Condition:** entering the “Lost & Found” page.
4. **Exit Condition:** Navigating through another page
5. **Flow Of Events:**
   - **5.1.** When something has changed about the notice or is miswritten, they can update their lost posts.
     - **5.1.1.** If the user fills some of the fields incorrectly, the system gives an error and then directs the user to the beginning of step 5.1