CS-319

Deliverable 1

Team 06

Group Members

Muhammad Rowaha - 22101023

Maher Athar Ilyas - 22001298

Mehshid Atiq - 22101335

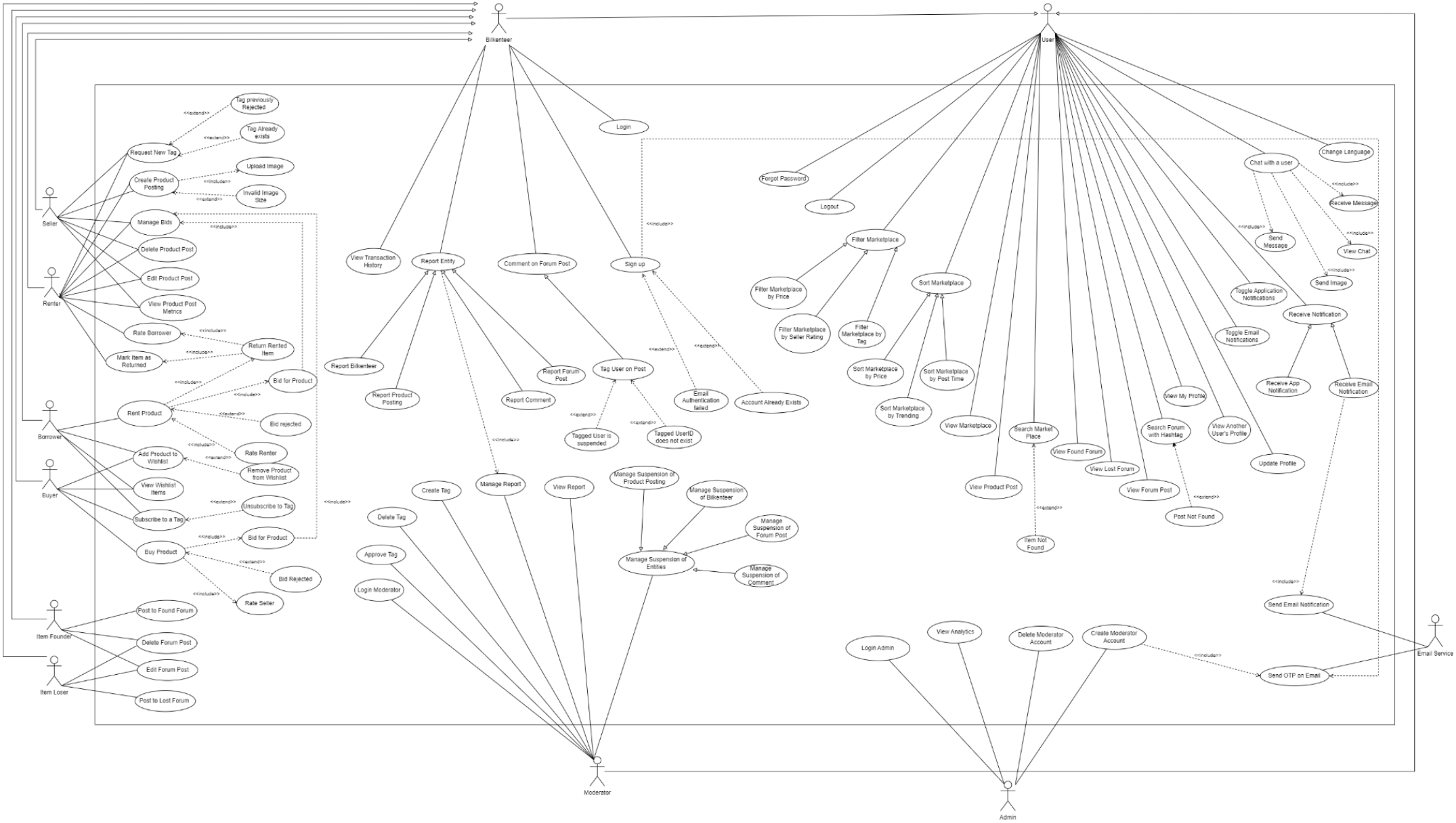
Ghulam Ahmed - 22101001

Ismail Özgenç - 22001648

Date: 12/11/2023

1. Use Case Diagram

https://drive.google.com/file/d/1yijxUhPudVKwzDOU4M9Bpc6fWaDsb2ml/view?usp=sharing



2. Use Case Details

2.1 - Admin

- Login Admin
    - Participating actor: Admin
    - Entry Condition:
        - Admin has a valid username and password.
    - Exit condition:
        - Admin is logged in and has access to the admin dashboard.
    - Flow of events:
        1. Admin enters username and password.
        2. The system verifies the credentials.
        3. The system grants access to the admin dashboard.
    - Special / quality requirements:
        - Secure login mechanism to prevent unauthorized access. Two-factor authentication is recommended.
- Create Moderator Account
    - Participating actor: Admin
    - Entry Condition:
        - Admin is logged into the admin dashboard.
    - Exit condition:
        - New moderator account is created and active.
    - Flow of events:
        1. Admin navigates to "Create Moderator" section.
        2. Admin enters the required details for the new moderator.
        3. System validates the input and creates a new moderator account.
        4. System displays a success notification to the admin.
    - Special / quality requirements:
        - Admin should be able to set unique permissions and access for each moderator.
- Delete Moderator Account
    - Participating actor: Admin
    - Entry Condition:
        - Admin is logged into the admin dashboard.
        - There are active moderator accounts.
    - Exit condition:
        - Selected moderator account is deactivated and removed.
    - Flow of events:
        1. Admin navigates to the list of active moderators.
        2. Admin selects the moderator account to be deleted.
        3. System prompts for confirmation.
        4. Admin confirms the deletion.
        5. System deactivates and removes the selected moderator account.
    - Special / quality requirements:
        - Admins should receive a prompt to ensure the action is not accidental.

- View Analytics
    - Name: View Analytics
    - Participating actor: Admin
    - Entry Condition:
        - Admin is logged into the admin dashboard.
    - Exit condition:
        - Admin views the desired analytics.
    - The flow of events:
        1. Admin navigates to the "Dashboard" section.
        2. The system displays various analytics such as user activity, sales trends, popular items, etc.
        3. Admin can filter or select specific data points for detailed insights.
    - Special / quality requirements:
        - The system should provide real-time updates and ensure data accuracy. The display should be user-friendly, allowing for graphical representations and easy understanding of trends.

2.2 - Email Service

- Send Email Notification
    - Participating actor: Email Service
    - Entry Condition:
        - A user action requires a notification (e.g., Post approval, Post deletion, User suspension, etc.).
        - Admin or moderator triggers the notification action from the system.
    - Exit condition:
        - A notification email has been sent to the user.
    - The flow of events:
        1. A user action/event takes place that requires a notification.
        2. The system requests the Email Service with the necessary details (user's email, subject, message content).
        3. Email Service processes the request.
        4. Email Service sends an email notification to the specified user’s email.
        5. Notification sent
    - Special / quality requirements:
        - The email notification should be sent within 2 minutes of the request.
        - The content of the email should be clear, precise, and in line with the user action taken.
        - Email Service should provide a delivery status back to the application.
- Send OTP on Email
    - Participating Actor: Email Service
    - Entry Condition:
        - A user has initiated a process that requires email verification.
        - The user's email address is valid and available to the system.
    - Exit Condition:
        - A One-Time Password (OTP) has been sent to the user's email address.
    - Flow of Events:

1. The system triggers the Email Service after determining a need for email verification (e.g. account registration).
2. The Email Service generates a secure, time-sensitive OTP.
3. The Email Service composes an email containing the OTP, with clear instructions for the user on how to use it.
4. The Email Service sends the email to the user's provided email address.
5. The system logs the transaction for security and auditing purposes.

- Special / Quality Requirements:
  - The OTP must be generated using a secure, cryptographically strong algorithm.
  - The email delivery should be prompt to ensure a smooth user experience.
  - The system must handle email delivery failures and provide appropriate feedback to the user or system.
  - The OTP should have a precise expiration time, after which it becomes invalid.

2.3 - User

- Logout
  - Participating Actor: User
  - Entry Condition:
    - User is currently logged into the app.
  - Exit Condition:
    - User is successfully logged out and can no longer access the personalized features.
  - Flow of Events:
    1. User selects the "Logout" option within the app.
    2. The system presents a confirmation prompt to ensure the user intends to log out.
    3. The system terminates the user's active session.
    4. The user is redirected to the app's home page.
  - Special/Quality Requirements:
    - Ensure all active sessions and associated session data are properly terminated upon logout.
    - The logout process should not be susceptible to session fixation or other security vulnerabilities.
    - If the user is inactive for a specified period, implement an automatic logout mechanism to enhance security.
- Forgot Password
  - Participating Actor: User
  - Entry Condition:
    - The user has a registered account but cannot remember their password.
  - Exit Condition:
    - The user successfully resets their password and gains access to their account.

    - Flow of Events:
        1. The user launches the app and selects the “Forgot Password” option on the login screen.
        2. The user is prompted to enter their registered email address.
        3. The system verifies the provided email address against the user database.
        4. Upon successful verification, the system sends a password reset link or code to the user's registered email.
        5. The user accesses their email and clicks on the reset link or inputs the provided code in the app.
        6. The user is directed to a secure password reset page/form.
        7. The user enters a new password, confirms it by re-entering, and submits the information.
        8. The system updates the user’s account with the new password, ensuring it meets predefined security standards.
        9. The user receives a confirmation message of successful password reset.
    - Special/Quality Requirements:
        - The password reset process should be secure and user-friendly.
        - Employ robust encryption for the transmission of reset links or codes.
- Filter Marketplace by Price
    - Participating Actor: User
    - Entry Condition:
        - User in in the marketplace section of the app.
    - Exit Condition:
        - User successfully filters and views marketplace listings according to the price.
    - Flow of Events:
        1. User searches for a product.
        2. System displays the relevant products.
        3. User selects the filter option to refine marketplace listings
        4. User clicks on the option of ‘filter by price’.
        5. The system displays marketplace listings that match the specified price.
        6. User explores the filtered results.
        7. User can refine the search further or adjust the filtering as needed.
    - Special/Quality Requirements:
        - Allow flexibility in selecting multiple filters to broaden or narrow the search based on user preferences.
        - Design a clear and intuitive user interface for selecting and adjusting price filtering options.
        - Implement real-time updates to reflect changes in the marketplace listings matching the selected filters.
- Filter Marketplace by Seller Ratings
    - Participating Actor: User
    - Entry Condition:
        - User in in the marketplace section of the app.
    - Exit Condition:

        - User successfully filters and views marketplace listings according to the seller ratings.
    - Flow of Events:
        1. User searches for a product.
        2. System displays the relevant products.
        3. User selects the filter option to refine marketplace listings
        4. User clicks on the option of ‘filter by seller ratings’.
        5. The system displays marketplace listings that match the specified ratings.
        6. User explores the filtered results.
        7. User can refine the search further or adjust the filtering as needed.
    - Special/Quality Requirements:
        - Allow flexibility in selecting multiple filters to broaden or narrow the search based on user preferences.
        - Design a clear and intuitive user interface for selecting and adjusting price filtering options.
        - Implement real-time updates to reflect changes in the marketplace listings matching the selected filters.
- Filter Marketplace by Tag
    - Participating Actor: User
    - Entry Condition:
        - User in in the marketplace section of the app.
    - Exit Condition:
        - User successfully filters and views marketplace listings according to the tag
    - Flow of Events:
        1. User searches for a product.
        2. System displays the relevant products.
        3. User selects the filter option to refine marketplace listings
        4. User clicks on the option of ‘filter by tag’.
        5. The system displays marketplace listings that match the specified tag.
        6. User explores the filtered results.
        7. User can refine the search further or adjust the filtering as needed.
    - Special/Quality Requirements:
        - Allow flexibility in selecting multiple filters to broaden or narrow the search based on user preferences.
        - Design a clear and intuitive user interface for selecting and adjusting price filtering options.
        - Implement real-time updates to reflect changes in the marketplace listings matching the selected filters.
- Sort Marketplace by Price
    - Participating Actor: User
    - Entry Condition:
        - User in in the marketplace section of the app.
    - Exit Condition:
        - User successfully sorts and views marketplace listings according to the price.
    - Flow of Events:
        1. User searches for a product.

2. System displays the relevant products.
3. User selects the sort option to refine marketplace listings
4. User clicks on the ‘sort by price’ option and selects between ascending or descending prices.
5. The system displays marketplace listings that match the specified sorting order.
6. User explores the sorted results.
7. User can refine the search further or adjust the sorting as needed.

- Special/Quality Requirements:
  - User can also apply sorting order on the marketplace without searching for a specific product, getting results from all the marketplace listings.
  - Allow flexibility in selecting multiple sortings to broaden or narrow the search based on user preferences.
  - Design a clear and intuitive user interface for selecting and adjusting price sorting options.
  - Implement real-time updates to reflect changes in the marketplace listings matching the selected sortings.

- Sort Marketplace by Trending
  - Participating Actor: User
  - Entry Condition:
    - User is in the marketplace section of the app.
  - Exit Condition:
    - User successfully sorts and views marketplace listings according to the price.
  - Flow of Events:
    1. User searches for a product.
    2. System displays the relevant products.
    3. User selects the sort option to refine marketplace listings.
    4. User clicks on the option of ‘sort by trending’.
    5. The system displays marketplace listings that match the specified sorting order according to the number of views received by the product posts.
    6. User explores the sorted results.
    7. User can refine the search further or adjust the sorting as needed.
  - Special/Quality Requirements:
    - User can also apply sorting orders on marketplace without searching for a specific product, getting results from all the marketplace listings.
    - Allow flexibility in selecting multiple sortings to broaden or narrow the search based on user preferences.
    - Design a clear and intuitive user interface for selecting and adjusting price sorting options.
    - Implement real-time updates to reflect changes in the marketplace listings matching the selected sortings.
- Sort Marketplace by Time
  - Participating Actor: User
  - Entry Condition:
    - User is in the marketplace section of the app.
  - Exit Condition:

        - User successfully sorts and views marketplace listings according to the time of their posting.
    - Flow of Events:
        1. User searches for a product.
        2. System displays the relevant products.
        3. User selects the sort option to refine marketplace listings.
        4. User clicks on the option of ‘sort by time’ and further selects between newest to oldest or its reverse sorting order.
        5. The system displays marketplace listings that match the specified sorting order,
        6. User explores the sorted results.
        7. User can refine the search further or adjust the sorting as needed.
    - Special/Quality Requirements:
        - User can also apply sorting order on marketplace without searching for a specific product, getting results from all the marketplace listings.
        - Allow flexibility in selecting multiple sortings to broaden or narrow the search based on user preferences.
        - Design a clear and intuitive user interface for selecting and adjusting price sorting options.
        - Implement real-time updates to reflect any changes in the marketplace listings matching the selected sortings
- View Marketplace
    - Participating Actor: User
    - Entry Condition:
        - User launches the application.
    - Exit Condition:
        - User successfully views the marketplace listings.
    - Flow of Events:
        1. User clicks on the marketplace button on the menu bar within the app.
        2. System shows the most recent posts to the user.
        3. User can interact with the posts and filter/sort them.
    - Special/Quality Requirements:
        - Ensure that the list of most recent marketplace listings is updated in real-time to reflect the latest uploads.
        - Optimize the system for efficiently loading recent listings, providing a seamless browsing experience.
        - Design a straightforward and user-friendly interface for users.
- Search Marketplace
    - Participating Actor: User
    - Entry Condition:
        - User is in the marketplace section of the app.
    - Exit Condition:
        - User successfully searches and discovers relevant product listings on the marketplace.
    - Flow of Events:
        1. User navigates to the "Search" section within the marketplace.
        2. User types the name of the product they want, this may include keywords that match the tags associated with each product post.
        3. System triggers the search function based on the selected tags.

4. The app displays a list of products matching the selected tags. Users can scroll through these results.
5. User clicks or taps on a specific product from the search results to view more details.

- Special/Quality Requirements:
  - System will notify the user if no relevant products are found.
  - Implement real-time updates to reflect any changes in the product status or details.
  - Optimize the tag-based search functionality for efficiency and accuracy.

- View Product Post
  - Activity: View Product Post from Market Place
  - Participating Actor: User
  - Entry Condition:
    - User is currently in the marketplace section of the app.
  - Exit Condition:
    - User successfully views the details of a product post on the marketplace.
  - Flow of Events:
    1. User clicks or taps on a specific product post to view more details.
    2. The system displays detailed information about the product, including the title, description, price and images, etc.
    3. User can return to the previous screen or navigate back to the main marketplace interface.
  - Special/Quality Requirements:
    - If the product is still available, implement real-time updates to reflect any product status or details changes.
    - Ensure the messaging system is secure and allows users to communicate safely about the product.

- View Forum Post
  - Participating Actor: User
  - Entry Condition:
    - User is currently in the Forums section of the app.
  - Exit Condition:
    - User successfully views the details of a forum post on the forum..
  - Flow of Events:
    1. User selects from the lost or found forum to view posts from.
    2. User clicks or taps on a specific forum post to view more details.
    3. The system displays detailed information about the post, including the title, description, and images etc.
    4. User can return to the previous screen or navigate back to the main forum.
  - Special/Quality Requirements:
    - Ensure a good UI so that user can easily interact with the post e.g. comment on, share and save post.
    - Implement real-time updates to reflect any changes in the product status or comments.

        - Ensure the messaging system is secure and allows users to communicate safely about the product.
- View Lost Forum
    - Participating Actor: User
    - Entry Condition:
        - User launches the application.
    - Exit Condition:
        - User successfully views the Lost forum.
    - Flow of Events:
        1. User clicks on the lost and found forum button on the menu bar within the app.
        2. System takes the user to the forums section within the app.
        3. User selects the Lost Forum button to view only the lost forum posts.
        4. System shows the most recent posts to the user.
    - Special/Quality Requirements:
        - Implement real-time updates to show new Lost and Found posts without requiring manual refresh if possible.
        - Provide users with the ability to filter and sort the forum.
- View Found Forum
    - Participating Actor: User
    - Entry Condition:
        - User is logged into the app.
    - Exit Condition:
        - User successfully views the Found forum.
    - Flow of Events:
        1. User clicks on the lost and found forum button on the menu bar within the app.
        2. System takes the user to the forums section within the app.
        3. User selects the Found Forum button to view only the Found Forum posts.
        4. System shows the most recent posts to the user.
        5. User can interact with the posts and filter/sort them.
    - Special/Quality Requirements:
        - Implement real-time updates to show new Lost and Found posts without requiring a manual refresh.
        - Provide users with the ability to filter and sort the forum.
- Search Forum with hashtag
    - Participating Actor: User
    - Entry Condition:
        - User is in the forums section of the app.
    - Exit Condition:
        - User successfully searches and discovers relevant forum posts on the marketplace.
    - Flow of Events:
        1. User navigates to the "Search" section within the forums.
        2. User types a hashtag relevant to their search, this may include keywords that match the tags associated with each forum post.
        3. System triggers the search function based on the selected tags.

4. The app displays a list of products matching the selected tags. Users can scroll through these results.
5. User clicks or taps on a specific post from the search results to view more details.

- Special/Quality Requirements:
  - System will notify the user if no relevant posts are found.
  - Implement real-time updates to reflect any changes in the forum posts' status or details.
  - Optimize the tag-based search functionality for efficiency and accuracy.

- View My Profile
  - Participating actor: User
  - Entry Condition:
    - User is logged into their account.
  - Exit condition:
    - User's profile information is displayed.
  - Flow of events:
    1. User selects the 'Profile' option from the menu.
    2. The system displays the user's current profile information.
  - Special/Quality Requirements:
    - Ensure that only the information marked as public or shared by the user is displayed to others.
    - Display the most up-to-date information about the user by fetching real-time data from the server.
- View Another User's Profile
  - Participating actor: User
  - Entry Condition:
    - User is logged into their account.
  - Exit condition:
    - Other user's profile information is displayed.
  - Flow of events:
    1. User initiates a search for the specific user whose profile they want to view.
    2. User identifies the desired user from the search results and clicks on their profile.
    3. The system displays the profile details of the selected user, including username, bio, profile picture, and other relevant information.
    4. User may choose to explore the forum posts and posted products shared by the selected user.
    5. User has the option to interact witht the other user by sending a message.
    6. User may choose to explore more or return back to the main page.
  - Special / quality requirements:
    - Ensure a clear and user-friendly interface for viewing other user profiles.
    - Ensure that user can alternatively access other user's profile by clicking on their profile name or image anywhere in the app.
    - Implement privacy settings to control the visibility of certain profile details.

        - Provide options for users to interact with the viewed profile, such as sending messages or following.
        - Respect and comply with privacy and data protection regulations during the profile viewing process.
- Update Profile
    - Participating actor: User
    - Entry Condition:
        - User is logged into their account and viewing their profile.
    - Exit condition:
        - User's profile is updated.
    - Flow of events:
        1. User selects 'Edit Profile' from their profile page.
        2. User makes changes to their profile information.
        3. User submits the changes.
        4. The system validates and saves the updated information.
    - Special / quality requirements:
        - Profile updates should occur in real time.
        - The system should inform the user of successful or unsuccessful updates.
- Toggle Email Notifications
    - Participating actor: User
    - Entry Condition:
        - User is logged into their account and has accessed the notification settings.
    - Exit condition:
        - User's email notification settings are updated.
    - Flow of events:
        1. User navigates to 'Notification Settings' in their account.
        2. User toggles the email notification option to turn on or off the email notifications to their Bilkent email.
        3. The system updates the user's preference.
    - Special / quality requirements:
        - Changes should be saved immediately and feedback provided to the user.
- Toggle Application Notifications
    - Participating actor: User
    - Entry Condition:
        - User is logged into their account and has accessed the notification settings.
    - Exit condition:
        - User's application notification settings are updated.
    - Flow of events:
        1. User navigates to 'Notification Settings' in their account.
        2. User toggles the application notification option to turn on or off the application notifications.
        3. The system updates the user's preference.
    - Special / quality requirements:
        - Changes should be saved immediately and feedback provided to the user.

- Receive App Notification
    - Participating actor: User
    - Entry Condition:
        - User is logged into the app and has in-app notifications enabled, and a notification-triggering event occurs.
    - Exit condition:
        - User receives a new in-app notification.
    - Flow of events:
        1. A triggering event occurs (e.g., a friend request or a new message).
        2. The system generates a notification.
        3. The user receives a notification in the app's interface.
    - Special / quality requirements:
        - Notifications must be timely and relevant to the user's actions.
- Receive Email Notification
    - Participating actor: User
    - Entry Condition:
        - User has enabled email notifications to their Bilkent email and a notification-triggering event occurs.
    - Exit condition:
        - User receives a new email notification.
    - Flow of events:
        1. A triggering event occurs (e.g., a friend request or a new message).
        2. The system sends an email to the user's registered email address.
    - Special / quality requirements:
        - Emails should be formatted correctly and contain relevant information.
- View Chat
    - Participating actor: User
    - Entry Condition:
        - User is logged into their account.
        - User selects a conversation from their chat list.
    - Exit condition:
        - User views the conversation history.
    - Flow of events:
        1. User navigates to the 'Chats' section.
        2. User selects a conversation.
        3. The system displays the conversation's message history.
    - Special / quality requirements:
        - The chat interface should load recent messages efficiently.
- Send Message
    - Participating actor: User
    - Entry Condition:
        - User has opened a conversation with another user.
    - Exit condition:
        - The message is sent and appears in the conversation.
    - Flow of events:
        1. User types a message in the message input field.
        2. User clicks 'Send' button.
        3. The system sends the message and displays it in the conversation.
    - Special / quality requirements:

        - The system should ensure message delivery with confirmation.
- Receive Message
    - Participating actor: User
    - Entry Condition:
        - User is logged in and the other user sends a message.
    - Exit condition:
        - User receives the message.
    - Flow of events:
        1. Another user sends a message to the user.
        2. The system notifies the user of the new message.
        3. User views the new message in the conversation.
    - Special / quality requirements:
        - The system should notify the user of new messages promptly.
- Send Image
    - Participating actor: User
    - Entry Condition:
        - User is engaged in a conversation and has an image to send.
    - Exit condition:
        - The image is sent and appears in the conversation.
    - Flow of events:
        1. User selects the 'Attach Image' option within the conversation.
        2. User chooses an image file to send.
        3. User confirms the action and sends the image.
        4. The system uploads the image and displays it in the conversation.
    - Special / quality requirements:
        - The system must ensure the secure transmission of images.
        - Supported image formats: JPG, PNG.
- Change Language
    - Name: Change Language
    - Participating actor: User
    - Entry Condition:
        - User is logged in and on any page where language options are available.
    - Exit condition:
        - User has selected a new language, and the system has applied it to the user interface.
    - Flow of events:
        1. User navigates to the language selection option in the settings or the page header/footer.
        2. User selects the desired language from English and Turkish.
        3. The system updates the user interface to display the text in the selected language.
    - Special / quality requirements:
        - The language change should be persistent for the user across sessions.
        - The system should remember and apply the user's choice for future sessions.

2.4 - Moderator

- Login Moderator
    - Participating actor: Moderator
    - Entry Condition:
        - Moderator has a valid username and password.
    - Exit condition:
        - Moderator is logged in and has access to the moderator dashboard.
    - Flow of events:
        1. Moderator enters username and password.
        2. System verifies the credentials.
        3. System grants access to the moderator dashboard.
    - Special / quality requirements:
        - Secure login mechanism to prevent unauthorized access. Two-factor authentication is recommended.
- View Report
    - Participating Actor: Moderator
    - Entry Condition:
        - Moderator is logged into the website.
    - Exit Condition:
        - Moderator has viewed the details of a report.
    - Flow of Events:
        1. Moderator accesses the reports section from the moderator dashboard.
        2. The system displays a list of user-submitted reports.
        3. Moderator selects a report to view.
        4. The system displays the full details of the report, including the complainant's account, the accused account (if applicable), timestamps, and the nature of the complaint.
    - Special / Quality Requirements:
        - The system must ensure data privacy and security when displaying reports.
        - Reports should be presented clearly and organized for quick assessment by the moderator.
        - The interface should allow moderators to navigate between reports efficiently.
- Manage Report
    - Participating actor: Moderator
    - Entry Condition:
        - Moderator is logged into the website.
        - A Bilkenteer has created a report.
    - Exit condition:
        - Moderator has reviewed and taken necessary action on the report.
        - Report marked as resolved.
    - Flow of events:
        1. Moderator accesses the reports section.
        2. The system displays a list of user reports.
        3. Moderator selects a report to review.

4. The system displays report details.
5. Moderator takes an appropriate action (e.g., suspend, unsuspend, email user).

- Approve Tag
  - Inherits From: Manage Tag
  - Entry Condition:
    - Moderator is logged into the website.
    - There are tags awaiting approval
  - Exit condition:
    - Moderator has approved or rejected the tag.
  - Flow of events:
    1. Moderator accesses the tag approval section.
    2. The system displays a list of tags awaiting approval.
    3. Moderator selects a tag to review.
    4. The system displays tag details.
    5. Moderator decides to approve or reject the tag.
  - Special / quality requirements: The system should provide a brief reason for the tag creation to aid the moderator in decision-making.
- Create Tag
  - Inherits From: Manage Tag
  - Entry Condition:
    - Moderator is logged into the website.
  - Exit condition:
    - Moderator has created a new tag.
  - Flow of events:
    1. Moderator accesses the tag creation section.
    2. The system prompts the moderator to enter tag details.
    3. Moderator submits the new tag details.
    4. The system saves and displays the new tag.
  - Special / quality requirements: The system should ensure that the tag does not already exist.
- Delete Tag
  - Inherits From: Manage Tag
  - Entry Condition:
    - Moderator is logged into the website.
    - There are tags available to be deleted.
  - Exit condition:
    - Moderator has deleted the tag.
  - Flow of events:
    1. Moderator accesses the tag section.
    2. The system displays a list of tags.
    3. Moderator selects a tag for deletion.
    4. The system prompts the moderator to confirm deletion.
    5. Moderator confirms deletion.
  - Special / quality requirements: The system should warn about the potential impact of deleting the tag.
- Manage Suspension of Bilkenteer
  - Participating actor: Moderator
  - Entry Condition:

        - Moderator is logged into the website.
        - A Bilkenteer is reported.
    - Exit condition:
        - Moderator has updated the suspend status of a Bilkenteer.
    - Flow of events:
        1. Moderator views the report
        2. The system displays URL to reported Bilkenteer’s User Profile.
        3. Moderator clicks on the profile URL.
        4. On the Moderator’s view of profile, moderator suspends or unsuspends the Bilkenteer using a button.
        5. The system prompts the moderator to provide a reason for updating status.
        6. Moderator submits the reason and confirms state update.
        7. System updates suspend status of a Bilkenteer.
    - Special / quality requirements: The system should notify the Bilkenteer about the status update and the reason.
- Manage Suspension of Entities
    - Participating Actor: Moderator
    - Entry Condition:
        - Moderator is logged into the website and there was a report related to an entity.
    - Exit Condition:
        - Moderator has suspended one or more entities (product postings, forum posts, comments).
    - Flow of Events:
        1. Moderator views the report.
        2. The system displays URL to the reported entity’s page.
        3. Moderator clicks on the URL to access the entity's page.
        4. On the entity's page, Moderator takes appropriate action to suspend or unsuspend the entity using provided options.
        5. The system prompts the moderator to provide a reason for the action.
        6. Moderator submits the reason and confirms the update.
        7. The system updates the suspension status of the entity.
    - Special / Quality Requirements:
        - The system should maintain a log of all suspensions for audit and review.
        - Suspended entities should be removed or hidden from public view.
- Manage Suspension of Product Posting
    - Participating Actor: Moderator
    - Entry Condition:
        - Product postings are available for review.
    - Exit Condition:
        - Product posting(s) have been suspended or unsuspended.
    - Flow of Events:
        1. Moderator views the report related to a specific product posting.
        2. The system displays the URL to the reported product posting’s page.
        3. Moderator clicks on the URL to access the product posting's page.

4. The product posting page includes detailed information about the product, the reason for reporting, and options to suspend or unsuspend the posting.
5. Moderator reviews the product posting and decides whether to suspend or unsuspend it.
6. Upon selecting an action, the system prompts the moderator to provide a reason for the decision.
7. Moderator inputs the reason and confirms the suspension or unsuspension.
8. The system updates the status of the product posting accordingly and logs the action for audit purposes.

- Special / Quality Requirements:
  - Suspended products should be removed from the marketplace.

- Manage Suspension of Forum Post
  - Participating Actor: Moderator
  - Entry Condition:
    - Forum posts are available for review.
  - Exit Condition:
    - Forum post(s) have been suspended or unsuspended.
  - Flow of Events:
    1. Moderator receives a report concerning a forum post.
    2. The system presents the URL to the reported forum post.
    3. Moderator navigates to the forum post's page via the provided URL.
    4. On the forum post page, the moderator sees the post content, the report details, and options to suspend or unsuspend the post.
    5. Moderator evaluates the content of the post in the context of the report.
    6. Moderator selects the appropriate action (suspend or unsuspend) and is prompted to enter a rationale for this decision.
    7. After providing the reason, the moderator confirms the action.
    8. The system updates the forum post's status and records the moderator's action.
  - Special / Quality Requirements:
    - Forum posts must be displayed with context to aid in decision-making.
- Manage Suspension of Comment
  - Participating Actor: Moderator
  - Entry Condition:
    - Comments on posts or products are available for review.
  - Exit Condition:
    - Comment(s) have been suspended or unsuspended.
  - Flow of Events:
    1. Moderator views a report related to a specific comment.
    2. The system directs the moderator to the comment's page by providing its URL.
    3. Moderator reviews the comment and the associated context on its page.
    4. The page displays the comment, the context of the comment (the post or product it's related to), and the report details.

5. Moderator decides to suspend or unsuspend the comment based on the review.
6. Upon choosing an action, the system asks the moderator to justify their decision.
7. Moderator submits the reason and confirms the action.
8. The system then updates the comment status and logs the moderator's action.

- Special / Quality Requirements:
  - Comments should be presented with the relevant post or product for proper context.

2.5 - Bilkenteer

- Login
  - Participating Actor: Bilkenteer
  - Entry Condition:
    - Bilkenteer has a registered account with a valid username and password.
  - Exit Condition:
    - Bilkenteer is successfully logged in and gains access to personalized features.
  - Flow of Events:
    1. Bilkenteer launches the app.
    2. Selects the "Login" option.
    3. Bilkenteer enters a valid email and password in the designated fields.
    4. The system validates the entered credentials against the stored user database.
    5. The system grants access to the account if the credentials are correct.
  - Special/Quality Requirements:
    - The login process employs robust encryption to safeguard user credentials during transmission and storage.
    - Passwords are securely hashed and stored.
    - The system provides clear and user-friendly error messages for incorrect login attempts.
- Sign up
  - Participating Actor: Bilkenteer
  - Entry Condition:
    - User does not have an existing account in the app.
  - Exit Condition:
    - User successfully creates an account and gains access to personalized features.
  - Flow of Events:
    1. User selects the "Create Account" or "Sign Up" option within the app.
    2. User provides required information, including username, email, password, and any other necessary details.
    3. The system validates the entered information, ensuring it meets the required criteria (e.g., valid email format, strong password).
    4. System generates an OTP and sends it to user's email address.
    5. The user enters the received OTP.

6. If the information is valid, the system creates a new user account and associates it with the provided credentials.
7. A confirmation message is displayed, informing the user that the account creation was successful.

- Special/Quality Requirements:
  - System blocks account creation if the account already exists.
  - System blocks account creation if incorrect OTP is entered.
  - Implement secure password storage practices, such as hashing, to protect user passwords.
  - Provide a secure method for users to recover their account in case they forget their credentials.

- View Transaction History
  - Participating Actor: Bilkenteer
  - Entry Condition:
    - User is logged into the app.
  - Exit Condition:
    - User successfully views their transaction history.
  - Flow of Events:
    1. User navigates to their profile or account section in the app.
    2. User locates and selects the 'Transactions' tabs or a similar option.
    3. The system displays the user's transaction history, including purchases, sales, or any other relevant transactions.
    4. User will have the option to filter transactions by type i.e. ongoing transactions and completed transactions.
    5. User can also filter for transactions specific to purchases or rentals.
    6. User explores the relevant transactions history.
    7. User selects a specific transaction and explores the transaction details, including date, items, amounts, and any other relevant information.
    8. User can navigate back to the main transaction history or continue exploring individual transactions.
  - Special/Quality Requirements:
    - Allow flexibility in filtering transactions by type.
    - Ensure a clear and intuitive user interface for navigating and exploring transaction history.
    - Implement real-time updates to reflect any recent transactions or changes in transaction status.
- Report Bilkenteer
  - Participating Actor: Bilkenteer
  - Entry Condition:
    - User is logged into the app.
    - User encounters inappropriate behavior from another user.
  - Exit Condition:
    - User successfully submits a report, and appropriate actions are taken.
  - Flow of Events:
    1. User accesses the profile of the user to be reported.
    2. User selects the "Report User" option and provides details about the issue.

        3. Moderator reviews the report and takes necessary actions, such as warnings or account suspension.
    - Special/Quality Requirements:
        - The reporting user may receive feedback on the resolution of the report.
- Report Product Post
    - Participating Actor: Bilkenteer
    - Entry Condition:
        - User is logged into the app.
        - User is in the marketplace section of the app.
        - User encounters an inappropriate or suspicious product listing.
    - Exit Condition:
        - User successfully submits a report, and appropriate actions are taken.
    - Flow of Events:
        1. User accesses the product post to be reported.
        2. User selects the "Report Post" option and provides details about the issue.
        3. The moderator reviews the report and takes necessary actions, such as removing the post or contacting the seller.
    - Special/Quality Requirements:
        - The reporting user may receive feedback on the resolution of the report.
- Report Forum Post
    - Participating Actor: Bilkenteer
    - Entry Condition:
        - User is logged into the app.
        - User is in the forum section of the app.
        - User encounters an inappropriate or suspicious forum post.
    - Exit Condition:
        - User successfully submits a report, and appropriate actions are taken.
    - Flow of Events:
        1. User accesses the forum post to be reported.
        2. User selects the "Report Post" option and provides details about the issue.
        3. The moderator reviews the report and takes necessary actions, such as removing the post or contacting the author of the post.
    - Special/Quality Requirements:
        - The reporting user may receive feedback on the resolution of the report.
- Report Comment
    - Participating Actor: Bilkenteer
    - Entry Condition:
        - User is logged into the app.
        - User is in the forums section of the app.
        - User encounters an inappropriate or suspicious comment.
    - Exit Condition:
        - User successfully submits a report, and appropriate actions are taken.
    - Flow of Events:
        1. User accesses the relevant forum post and its comment.

2. User selects the "Report Comment" option and provides details about the issue.
3. The moderator reviews the report and takes necessary actions, such as removing the comment or contacting the author of the comment.

- Special/Quality Requirements:
  - The reporting user may receive feedback on the resolution of the report.

- Comment on Forum post
  - Participating Actor: Bilkenteer
  - Entry Condition:
    - User is logged into the buying and selling app.
    - User is viewing a forum post.
  - Exit Condition:
    - User successfully adds a comment to a forum post.
  - Flow of Events:
    1. User navigates to the forum section within the app and selects a post they want to comment on.
    2. User enters their comment in the provided text box at the bottom of post.
    3. User presses the post button and posts the comment to the forum.
    4. User has the option to respond to other comments on the forum post, creating a threaded discussion.
  - Special/Quality Requirements:
    - User may tag another user in the comment.
    - Implement moderation features like report comment.
    - If possible, implement real-time updates to show new comments on the forum post without needing a manual refresh.
- Tag User on a Post
  - Participating Actor: Bilkenteer
  - Entry Condition:
    - User is Commenting on a post
  - Exit Condition:
    - User successfully adds a comment with tagged user(s) to a forum post.
  - Flow of Events:
    1. User enters their comment in the provided text box at the bottom of post.
    2. Within the comment user adds @ symbol followed by the UserID of the user that commentator wants to tag.
    3. User presses the post button and posts the comment to the forum.
    4. System sends to notification to the tagged user using UserID
    5. Comment is added to the post.
  - Special/Quality Requirements:
    - If the tagged user’s ID is invalid or that user is suspended then the tagged notification is not sent.

2.5.1 Buyer & Borrower

- Rent Product

    - Participating Actor: Borrower
    - Entry Condition:
        - User is logged into the app.
        - User is in the marketplace section of the app.
    - Exit Condition:
        - Rent request is accepted, and the user receives confirmation.
    - Flow of Events:
        1. User navigates to the marketplace section and performs a search for the desired item.
        2. User put a filter for rentables to show only rentable products related to their search.
        3. User selects the item they want to rent from the item listings.
        4. User clicks on the "Rent" button or a similar option associated with the selected item.
        5. User provides details such as rental duration, pickup/delivery preferences, etc.
        6. User submits the rent request to the system.
        7. The system notifies the owner of the item about the rent request.
        8. Owner reviews the rent request and approves it.
        9. User receives notification of the owner's decision.
        10. Product is added to both renter and borrower’s ongoing transactions.
        11. Product is removed from the marketplace temporarily until marked returned by the renter.
    - Special/Quality Requirements:
        - Borrower should be notified if the owner (renter) cancels the rental request and item stays in the marketplace.
        - Ensure timely notifications to both the borrower and owner throughout the process.
- Return Rented Item
    - Participating Actor: Borrower
    - Entry Condition:
        - User has borrowed an item.
    - Exit Condition:
        - Return request is submitted, and the user receives confirmation.
    - Flow of Events:
        1. User navigates to the his own profile page and clicks on the transactions tab from the menu.
        2. User clicks on the ongoing transactions tab and further filters it to rent transactions.
        3. User selects the order containing the product to be returned from the shown listing.
        4. User chooses the "Return Item" option.
        5. System notifies the renter of the return activity and asks for his confirmation.
        6. Renter approves and marks the item as returned.
        7. Borrower is notified of the approval.
        8. Transaction status of the item converts to completed and now shows in the completed transactions tab.
    - Special/Quality Requirements:
        - Allow other user who have added the same item in their wishlist to track the return status and receive updates on the return progress.

- Bid for Product
    - Participating Actor: Buyer and Borrower
    - Entry Condition:
        - User is logged into the app.
    - Exit Condition:
        - User successfully generates a bidding request.
    - Flow of Events:
        1. User identifies a product they are interested in buying/renting.
        2. User bids for the product, providing details like bidding price, and borrowing period if the product is rentable.
        3. System notifes the seller/renter about the bid.
        4. Product moves to the ongoing transactions tab of the buyer/borrower.
        5. Bidder is notified of the bidding results once the seller/renter manages the bids, i.e. accepts or rejects them.
    - Special/Quality Requirements:
        - Provide clear and real-time updates on the bidding status.
        - Implement secure and efficient payment processing for purchase bids.
        - Notify the buyer of the auction outcome promptly.
        - Ensure a user-friendly interface for bidding and viewing bidding details.
- Rate Renter
    - Participating Actor: Borrower
    - Entry Condition:
        - User is logged into the app.
        - User has rented an item from the renter.
    - Exit Condition:
        - Renter successfully provides a rating for a completed rental transaction.
    - Flow of Events:
        1. User navigates to their profile and selects the transactions tab.
        2. User navigates to the completed transactions tab.
        3. User filters the results to list the items they rented.
        4. User selects the product they want to rate.
        5. User clicks on the 'Rate Renter' or a similar option associated with the completed transaction.
        6. User enters a numerical rating or selects a rating from a predefined scale
        7. User has the option to add an optional comment or feedback to elaborate on their rating and submits it.
        8. User submits the rating and any accompanying comment.
        9. The renter's profile is updated to reflect the received rating.
    - Special/Quality Requirements:
        - Allow renters to rate completed rental transactions only.
        - Additionally allow user to have the option to navigate to rating window when notified about a completed transaction.
        - Implement a clear and user-friendly interface for the rating process.
        - Ensure that the rating system contributes to the overall profile reputation of the renter.

        - Implement real-time updates to reflect any changes in the renter's rating on their profile.
- Subscribe to a Tag
    - Participating Actor: Buyer or borrower
    - Entry Condition:
        - User is logged into the app.
        - User is in the marketplace section of the app.
    - Exit Condition:
        - User is subscribed to a specific tag and receives relevant updates.
    - Flow of Events:
        1. User navigates through the marketplace and discovers a post with a tag that interests them.
        2. User clicks on the post to view its details.
        3. User identifies the relevant tag associated with the post, displayed prominently.
        4. User clicks on a tag associated with the post.
        5. The system displays on option to subscribe to the tag or view items relevant to the tag.
        6. User clicks on the 'Subscribe' or a similar option.
        7. Tptionhe system sends notifications with the subscribed tag to the user th ough his preferred choice for receiving notifications i.e. in-app or via email.
    - Special/Quality Requirements:
        - Allow users to manage their tag subscriptions and unsubscribe at any time.
        - Provide relevant and timely updates based on the subscribed tags.
- Add Product to Wishlist
    - Participating Actor: Buyer or Borrower
    - Entry Condition:
        - User is logged into the app.
        - User is in the marketplace section of the app.
    - Exit Condition:
        - User successfully adds a product to their wishlist.
    - Flow of Events:
        1. User identifies a product they are interested in adding to their wishlist.
        2. User clicks on the 'Add to Wishlist' button or a similar option associated with the selected product.
        3. Product is added to the user’s wishlist items’ listing.
    - Special/Quality Requirements:
        - Allow users to easily manage and modify their wishlist.
        - The user interface (UI) should dynamically update to reflect any changes made to the product post whenever a user views the post in the future.

        - Ensure a clear and intuitive user interface for adding products to the wishlist.
        - Implement real-time updates to reflect any changes in the user's wishlist.
- Remove Product from Wishlist
    - Participating Actor: Buyer or Borrower
    - Entry Condition:
        - User is logged into the app.
        - User is in the marketplace section of the app.
    - Exit Condition:
        - User successfully removes a product from their wishlist.
    - Flow of Events:
        1. User identifies the product they want to remove from their wishlist, either from the marketplace listings or from their personal wishlist listings.
        2. User clicks on the 'Remove from Wishlist' button or a similar option associated with the selected product.
        3. Product is removed from the user’s wishlist items’ listing.
    - Special/Quality Requirements:
        - Allow users to easily manage and modify their wishlist.
        - The user interface (UI) should dynamically update to reflect any changes made to the product post whenever a user views the post in the future.
        - Implement real-time updates to reflect any changes in the user's wishlist.
- View Wishlist Items
    - Participating Actor: Buyer or Borrower
    - Entry Condition:
        - User is logged into the app.
    - Exit Condition:
        - User successfully views their wishlist items.
    - Flow of Events:
        1. User navigates to their profile or account section in the app.
        2. User locates and selects the 'Wishlist' tab or a similar option.
        3. The system displays the user's wishlist listings.
        4. User will have the option to filter items by type e.g., rentables and sellables.
        5. User explores the relevant wishlist listings and interacts with them.
        6. User can navigate back to the user profile page or continue exploring individual posts.
    - Special/Quality Requirements:

        - Implement real time updates to reflect any recent transaction or change in product status i.e listed or available.
        - Allow flexibility in filtering listings by type.
        - Ensure a clear and intuitive user interface for navigating and exploring wishlist listings.
- Buy Product
    - Participating Actor: Buyer
    - Entry Condition:
        - User is logged into the app.
    - Exit Condition:
        - Buyer successfully purchases a product through bidding.
    - Flow of Events:
        1. Buyer identifies a product they are interested in buying.
        2. Buyer bids for the product.
        3. Seller manages the bids and accepts this bid.
        4. Buyer is notified of the approval and the transaction is logged into his completed transactions tabs.
        5. Buyer is asked by the system to rate the seller.
    - Special/Quality Requirements:
        - Buyer is notified by the system if his bid has been rejected by the Seller, in which case the buying process is unsuccessful.
        - Provide clear and real-time updates on the bidding status.
        - Ensure a user-friendly interface for bidding.

- Rate Seller
    - Participating Actor: Buyer
    - Entry Condition:
        - User is logged into the app.
        - User has purchased an item from the seller.
    - Exit Condition:
        - Buyer successfully provides a rating for a completed transaction.
    - Flow of Events:
        1. User navigates to their profile and selects the transactions tab.
        2. User navigates to the completed transactions tab.
        3. User filters the results to list the items they have purchased.
        4. User selects the product they want to rate.
        5. User clicks on the 'Rate seller’ or a similar option associated with the completed transaction.
        6. User enters a numerical rating or selects a rating from a predefined scale
        7. User has the option to add an optional comment or feedback to elaborate on their rating and submits it.
        8. User submits the rating and any accompanying comment.
        9. The seller’s profile is updated to reflect the received rating.
    - Special/Quality Requirements:
        - Allow renters to rate completed rental transactions only.
        - Additionally allow user to have the option to navigate to rating window when notified about a completed transaction.
        - Implement a clear and user-friendly interface for the rating process.

        - Ensure that the rating system contributes to the overall profile reputation of the renter.
        - Implement real-time updates to reflect any changes in the renter's rating on their profile.

2.5.2 Seller & Renter

- Create Product Posting
    - Participating Actor: Seller or Renter
    - Entry Condition:
        - The user is logged into the app.
        - User is in the marketplace section of the app.
    - Exit Condition:
        - The user successfully creates a new product post.
    - Flow of Events:
        1. User clicks on the “List an item" tab on the menu bar.
        2. User provides details about the product, including title, description, price, and relevant tags.
        3. User uploads upto 5 images of the product.
        4. User specifies if the product is for sale or rent.
        5. The system validates the entered information
        6. User confirms the creation of the product post.
        7. The system makes the post visible in the marketplace.
    - Special/Quality Requirements:
        - System informs the user if the uploaded images are of the incorrect format or size, and request a reupload.
        - System should not allow more than the prescribed number of image uploads.
        - Verify the accuracy and completeness of product descriptions to prevent misleading or incomplete information.
- Request New Tag
    - Participating Actor: Seller or Renter
    - Entry Condition:
        - The user is logged into the app.
        - The user is creating a new product post.
    - Exit Condition:
        - The user successfully submits a request for a new tag.
    - Flow of Events:
        1. User navigates to the ‘list an item’ tab and clicks on create new product post.
        2. System shows the create product post window.
        3. User selects an option to request a new tag.
        4. User provides details about the proposed tag and its relevance.
        5. The moderator reviews the request and add the tag to list of approved tags.
    - Special/Quality Requirements:
        - System should automatically reject the tag request if a same tag request has been rejected previously by the moderator and notify the user about it.

        - System should automatically give error message if the tag already exists.
        - Implement a clear and efficient process for tag requests and a review process for moderator approval.
- Delete Product Post
    - Participating actor: Seller or Renter
    - Entry Condition:
        - The user is logged into the app.
        - The user has authored the post that needs to be removed.
    - Exit Condition:
        - The user successfully deletes the product post.
    - Flow of Events:
        1. User navigates to the "Products" section within the app.
        2. User selects the product post to be deleted.
        3. User chooses the "Delete" option and confirms the action.
        4. The system removes the selected product post(s) from the marketplace.
    - Special/Quality Requirements:
        - If the app supports push notifications, the user may receive a notification confirming the successful deletion of the post.
        - Implement a clear and user-friendly confirmation mechanism to prevent accidental deletions.
        - Ensure that the deletion process respects user privacy, especially in cases where the post may have attracted interactions.
        - Implement moderation controls to prevent misuse of the deletion feature in case they were reported by some other user.
        - Maintain an audit trail or log of deleted posts for moderation or accountability purposes.
- View Product post metrics
    - Participating Actor: Seller or renter (Post author)
    - Entry Condition:
        - User is logged into the app.
    - Exit Condition:
        - User successfully views their product post metrics.
    - Flow of Events:
        1. User navigates to their profile or account section in the app.
        2. User locates and selects the 'Products' tabs or a similar option.
        3. User selects from sell and rent tabs to check their currently listed sellable or rentable products separately.
        4. User explores the relevant product listings.
        5. User selects a specific product and explores the post metrics, including views, bids, rental requests and other relevant information.
        6. User can navigate back to the main products listings or continue exploring individual post metrics.
    - Special/Quality Requirements:
        - Allow flexibility in filtering products by type.
        - Ensure a clear and intuitive user interface for navigating and exploring post metrics.

        - Implement real-time updates to reflect any recent transactions or changes in post metrics.
- Mark Item as Returned
    - Participating Actor: Renter
    - Entry Condition:
        - The user is logged into the app.
        - The borrower of the item has started a return activity.
    - Exit Condition:
        - The user marks the item as returned, and the post is updated.
    - Flow of Events:
        1. User navigates to his own profile within the app.
        2. User navigates to the products section within his profile and selects the ongoing transactions button to show his ongoing transactions.
        3. User filters the rented products.
        4. User selects the product to be marked as returned from the listing.
        5. User chooses the "Mark as returned" option.
        6. Product is removed from the user's on going transactions.
        7. System asks the renter if he wishes to start renting the product again and if approved the product reappears in the marketplace.
    - Special/Quality Requirements:
        - The system updates the borrower that the return activity has ended, and the transaction moves to his completed transactions tab.
- Manage Bids
    - Participating Actor: Seller/Renter
    - Entry Condition:
        - User is logged into the app.
        - User has listed an item for selling/renting.
    - Exit Condition:
        - Seller/Renter successfully manages bids on their listed products.
    - Flow of Events:
        1. Seller/Renter navigates to their profile and then clicks on the 'Products' tab to show their listed products for selling/renting.
        2. Seller/Renter selects a product they want to manage bids for.
        3. The system displays a list of all received bids for the selected product, including bidder/renter information and bid amounts.
        4. Seller has the option to accept a specific bid, in which case all other bids are automatically rejected.
        5. Seller/Renter may choose to reject a single bid, signaling that they are not interested in the offered terms.
        6. The system sends notifications to the bidders/renters regarding the status of their bid.
    - Special/Quality Requirements:
        - Ensure a clear and user-friendly interface for managing bids.
        - Provide real-time updates on bid statuses.
        - Implement communication features to facilitate negotiation between the seller/renter and the bidder/renter.
        - Allow sellers/renters to easily track and manage bids for multiple products.
- Edit Product Post

    - Participating Actor: Seller/Renter.
    - Entry Condition:
        - User is logged into the app.
        - User has authored the post he wants to edit.
    - Exit Condition:
        - Seller successfully edits and updates a product post.
    - Flow of Events:
        1. Seller navigates to their profile and then to their Products tab.
        2. User finds the post he wants to edit, optionally using filters to navigate to the post.
        3. User clicks on the 'Edit' or a similar option associated with the selected product post.
        4. Seller updates information such as product title, description, images, price, or any other relevant details.
        5. The system may provide an option for the seller to preview how the edited product post will appear.
        6. Seller confirms the edits and clicks on the 'Save' or a similar option to update the product post.
        7. The system updates the product post with the edited information.
    - Special/Quality Requirements:
        - Allow sellers to easily perform edits on posts they have authored anywhere in the marketplace as well.
        - Ensure a clear and user-friendly interface for editing product posts.
        - Provide real-time updates on the status of the edited product post.
        - Implement validation checks to ensure that the edited information complies with any specified guidelines.
        - Log all edit history to facilitate tracking and verification in case the post is reported by other users.
- Rate Borrower
    - Participating Actor: Renter
    - Entry Condition:
        - User is logged into the app.
        - User has received an item back from the borrower.
    - Exit Condition:
        - Renter successfully provides a rating for a completed transaction.
    - Flow of Events:
        1. User navigates to their profile and selects the transactions tab.
        2. User navigates to the completed transactions tab.
        3. User filters the results to list the items they have rented.
        4. User selects the product they want to rate.
        5. User clicks on the 'Rate borrower’ or a similar option associated with the completed transaction.
        6. User enters a numerical rating or selects a rating from a predefined scale
        7. User has the option to add an optional comment or feedback to elaborate on their rating.
        8. User submits the rating and any accompanying comment.
        9. The borrower’s profile is updated to reflect the received rating.
    - Special/Quality Requirements:

        - Allow renters to rate completed rental transactions only.
        - Additionally allow user to have the option to navigate to rating window when notified about a return of item by the borrower.
        - Implement a clear and user-friendly interface for the rating process.
        - Ensure that the rating system contributes to the overall profile reputation of the borrower.

2.5.5 Item Founder & Item Loser

- Post on Lost forum
    - Participating Actor: Item Loser
    - Entry Condition:
        - User is logged into the app.
        - User has navigated to the Lost forum.
    - Exit Condition:
        - User successfully posts information on the forum.
    - Flow of Events:
        1. User clicks on the create post button in the lost forum.
        2. User provides detailed information about the item, including a title, description, images and relevant tags.
        3. The system provides confirmation message to the user before posting.
        4. User confirms and post is posted
    - Special/Quality Requirements:
        - Implement moderation features to review reported posts to prevent misuse or inappropriate content.
        - Ensure that sensitive information, such as contact details, is handled with privacy in mind.
- Post on Found forum
    - Participating Actor: Item founder
    - Entry Condition:
        - User is logged into the app.
        - User has navigated to the Found forum.
    - Exit Condition:
        - User successfully posts information on the forum.
    - Flow of Events:
        1. User clicks on the create post button in the found forum.
        2. User provides detailed information about the item, including a title, description, images and relevant tags.
        3. The system provides confirmation message to the user before posting.
        4. User confirms and post is posted
    - Special/Quality Requirements:
        - Implement moderation features to review reported posts to prevent misuse or inappropriate content.
        - Ensure that sensitive information, such as contact details, is handled with privacy in mind.
- Delete forum post
    - Participating Actor: Item founder or item loser (Forum Post Author)
    - Entry Condition:
        - User is logged into the buying and selling app.

        - User authored a forum post.
    - Exit Condition:
        - User successfully deletes their forum post.
    - Flow of Events:
        1. User navigates to the his personal profile and clicks on the forum posts tab.
        2. User navigates to the required post optionally using filters to narrow down the results.
        3. User clicks on the ‘delete post’ button or equivalent option on the post.
        4. The system presents a confirmation prompt to ensure the user intends to delete the post.
        5. User confirms the deletion.
        6. The forum post is immediately removed from public view.
    - Special/Quality Requirements:
        - Implement a clear and user-friendly confirmation mechanism to prevent accidental deletions.
        - User should be able to deletes a post he authored directly inside forums as well.
        - Ensure that the deletion process respects user privacy, especially in cases where the post may have attracted comments or interactions.
        - System keeps a log of the deleted posts for cases where the post may have been subjects to report(s) by other users.
- Edit forum post
    - Participating Actor: Item founder or item loser (Forum Post Author)
    - Entry Condition:
        - User is logged into the buying and selling app.
        - User authored a forum post.
    - Exit Condition:
        - User successfully edits their forum post.
    - Flow of Events:
        1. User navigates to the his personal profile and clicks on the forum posts tab.
        2. User navigates to the required post optionally using filters to narrow down the results.
        3. User clicks on the ‘edit post’ button or equivalent option on the post.
        4. System allows user to edit post components like title, description, images, tags etc and clicks on post.
        5. The system presents a confirmation prompt.
        6. User confirms the prompt.
        7. The new edits are dynamically applied to the post.
    - Special/Quality Requirements:
        - Implement a clear and user-friendly confirmation mechanism for post edits.
        - User should be able to edit a post he authored directly inside forums as well.
        - Ensure that the edit process respects user privacy, especially in cases where the post may have attracted comments or interactions.
        - System keeps a log of the deleted posts for cases where the post may have been subjects to report(s) by other users.

3. NFR

- Response Time:
    - All actions related to user inputs should be completed within 2 seconds.
- Data Integrity:
    - There should be no data loss in the event of a system crash during any upsert action.
- Notification latency:
    - Notification to User: Must be sent within 2 minutes of the request.
    - Account Activation: Send the activation notification immediately (within 5 seconds) after the user completes the account creation process.
    - Report Submission Confirmation: Must be sent within 3 minutes of report submission.
- Email Activation Link Validity:
    - The activation link or code sent for account activation should remain valid for 48 hours.
- Session Inactivity Timeout:
    - App should automatically log out user after 90 minutes of user inactivity.
- Profile Picture Requirements:
    - If applicable, allow users to update their profile picture with a JPEG or PNG file format, and a size restriction of a maximum 5MB.
- Image Upload Processing:
    - Ensure that image uploads are processed within 5 seconds, providing a seamless experience for the seller.
- Search Query Performance:
    - Search queries in the system should return results within 3 seconds, regardless of the size of the dataset.

4. Tech Stack

4.1 Frontend:

- Framework:
    - Next.js on Page router will be used as it offers server-side rendering and static site generation, which can lead to improved performance and better SEO. Page-based routing helps organize the codebase around a file-system-based router.
- UI Components:
    - Ant Design will be used as it provides a comprehensive suite of high-quality React UI components that are well-documented. It accelerates development with a standardized UI, ensuring consistency and saving development time.

4.2 Backend:

- Framework:
    - Spring Boot will be used as it is known for simplifying the bootstrapping and development of new Spring applications. It minimizes boilerplate code, offers easy configuration, and has a vast ecosystem, which helps build large-scale applications.
- Database Management System:
    - PostgreSQL will be used as it is an open-source relational database. Jakarta ORM allows for a clear abstraction of the database layer, simplifying the interaction with PostgreSQL and ensuring that complex transactions and queries are managed effectively.