CTIS411 Senior Project I

Software Requirements Specification

Left-Over!

(T)hank (G)od (E)ast (O)ver

Furkan Kılıç, 21601729

Mehmet Melikşah Özceyhan, 21703661

Nevin Didem Bilgihan, 21601723

Yağmur Cansu Gürkan, 21601486

Bilkent University

Department of Information Systems and Technologies

15.11.2021

Change History

File Name	Document Type	Deliverable	Version	Submission Date
Deliverable_2_v0.1	Google Docs	2	0.1	-
Deliverable_2_v0.2	Google Docs	2	0.2	-
Deliverable_2_v0.3	Google Docs	2	0.3	-
Deliverable_2_v0.4	Google Docs	2	0.4	-
411_Deliverable_2_TGEO	PDF	2	1	15.11.2021
411_Deliverable_2_v2_TGEO	PDF	2	2	07.12.2021

Project Team

Project Team Name		**(T)hank (G)od (E)ast (O)ver**	
Name, Surname	**Student Id Number**	**Department**	**E-mail**
Nevin Didem Bilgihan	21601723	CTIS	didem.bilgihan@bilkent.edu.tr
Mehmet Melikşah Özceyhan	21703661	CTIS	meliksah.ozceyhan@ug.bilkent.edu.tr
Furkan Kılıç	21601729	CTIS	furkan.kilic@ug.bilkent.edu.tr
Yağmur Cansu Gürkan	21601486	CTIS	yagmur91598@gmail.com

Project Details

Project Name	**Left-Over!**
Academic Advisor	**Neşe Şahin Özçelik**
Github URL	https://github.com/meliksahozceyhan/Senior-Project-LeftOver
WEB page	**-**

Executive Summary

This document provides a comprehensive description of the Left-Over! software system requirements. It determines what features Left-Over! must have and how its features must function. General ideas about the individuals' characteristics that are going to use this application will be given. Furthermore, it will provide details about the dependencies and assumptions of the end-product. Moreover, in this document, there are the functional requirements of the end-product that defines how the Left-Over project will be operated, and also use-case diagrams and descriptions are provided. In section six there is the modeling of the two most important use-case which are Add Item and List Item of the end-product. In the following section, the selected use-cases' prototype is designed. Section eight will define the non-functional requirements considering the usability, performance, software system, and other non-functional requirements of the end-product. The next section tells about the logical database requirements of the system which will bring clearance about how the system will store data and how the end product will interact with that stored data. The constraints part interprets the factors that limit the available options for developing the Left-Over!. In section 11 the verification approaches and methods are determined to qualify the software. The discussion part details the impact of the software requirement specification documents in many aspects.

Changelog

- Addition to Executive summary.
- The numbers which are less than 10, written in letters.
- The recycling company definition was deleted in section three.
- User stories added to the section of requirements prototype.
- In the functional requirements section, must usage changed as shall.
- Addition to scope section.
- More explanation to the system interface part.
- Addition to the communication interface.
- Change in memory constraints
- Functional requirements' bullet points changed.
- Change in multidisciplinary collaboration.
- Other non-functional deletions.
- Table names changed.
- Add item activity diagram, list item activity diagram, and use case diagram have changed.

Table of Contents

List of Tables

Pages

List of Figures

Abbreviations

GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IOS	iPhone/iPad Operating System
OS	Operating System
WS	Web Socket

1. Scope

The product scope explains the purpose and the influence circle. In this manner, since Left-Over! is a sharing application for the public, the purpose is to provide people an opportunity to meet their needs without paying, and the influence circle is public in general.

The end-product, the Left-Over! Application will enable user registration as sharer and/or recipient. Sharers will be able to add one or more products under two categories, which are wearable and consumable. Recipients shall search and list the available items and request it. When the request is accepted by the sharer, the application will start real-time messaging between them. In this way, the sharer and the recipient will get together with the help of Left-Over!. The transaction process will be face-to-face independent from any application related authority. Thus, the application will provide an age control at the registration process, in this way the security risk will be reduced. Moreover, the application will offer an evaluation which affects the ratings of user profiles after transaction to maintain the reliability of the items and the users. Furthermore, the application provides drop point locations for collectable items which are recyclable.

While preparing this document and collecting the requirements, team generally met and discussed to determine the requirements in this document. Since team has no company to validate the requirements, team decided the requirements by themselves so there are no conflicting requirements to re-evaluate. Team arranged a meeting with their team advisor to check the integrity of requirements specified on the document, also to decide which requirements to be implemented in the first milestone of the product.

2. Product Perspective

The product is supposed to be an open source, under the Massachusetts Institute of Technology License. It is a mobile based system implementing a client-server model. The Left-Over! provides a simple mechanism for users to donate or request an item. Also, it enables users to see drop points and recycling companies for their recyclable waste.

The following are the main features that are included in Left-Over!;

a) System interfaces;

The system will interact with Google and Huawei Maps for maps services and also it will interact with the Google and Huawei Locations kits for location services.

These map services will be used to display map views on devices which will help the application to show the location of predefined drop points for recyclable waste.

b) User interfaces;

The required user interfaces are listed below;

- Pages which include data entrance will be in a form format.
- When there is a new entry, such as signing up and adding an item, a toast message will be displayed shortly on the bottom center.
- Password fields will be in a hidden format.
- Users will be able to see their password by clicking the show button (eye icon).
- The application design will be relative to Android OS and IOS mobile systems.
- Page design will be optimized based on screen sizes on different devices.
- There will be a navigation bar which includes message, add item, home, GPS, account buttons on each page.
- There will be a back button on the top left corner on each page except the home page.
- The listed item components will be clickable which navigates to its own page for more information.
- Although pages will not have a zoom in and zoom out feature, only item images will have that feature.

- Category pages will have property to be scrolled.
- Date picker will be used in date information entrance.
- New items will be added in specified categories which are defined in the drop-down list.
- The profile picture will be circle format.
- When an upload item photo button is clicked, the pop-up window will be displayed with two options which are to take a photo and choose from the gallery
- Clicked events will use the dissolve animation during transition.

c) Hardware interfaces;

Following are the hardware interfaces that will be needed by the Left-Over! application to operate properly.

- The Back-End Server needs an ethernet card to process incoming packets.
- The system will need 2 ports on the hardware to work. One is for the back-end server and the other is for the Database Management System.
- System will need to access the physical camera located on the mobile phone to capture the photos of items that are going to be shared across users.
- The application will need to access a GPS chip to locate the users and items, also to use maps to see drop points for recyclable waste.
- Application will need to access the file system of the mobile devices to pick photos to create an item.
- Mobile applications will need a touch screen for interacting with the users.

d) Software interfaces;

Following are the software products used for the Left-Over! application;

- **PostgreSQL 14:** The Database Management system used by software to store data of the application.
- **NEST.JS:** To implement the project, nest.js framework has been chosen for its easy way to build API's and modular architecture will ease the development and maintainability after the project grows.

- **Flutter:** Flutter chosen to develop the mobile application because it gives us the ability to work with both android and iOS platforms. Thus, it will reduce the overall application development time.
- **Moment.JS:** Moment.js is a package that will ease the use and modify the date values.
- **i18n:** i18n will provide multi language support thus it will make the application available in different languages.
- **Socket.IO:** This packet will provide an easy way to develop a real time messaging platform. It will enable real time, bidirectional communication between web clients and servers. So, it will be used in the messaging part of the project.
- **Docker:** This Software will help the CI/CD (Continuous Integration/Continuous Delivery/Deployment) pipeline it enables quickly and easily packet the software product to be delivered and creates an isolated environment for software to run. Thus, this software will have a crucial part in the delivery of the application.

e) Communications interfaces;

Application will rely on the Hypertext Transfer Protocol (HTTP) and WebSocket (WS) protocols in order to work. With HTTP, the application will send all the required data from the backend server to frontend mobile applications. All data related to users, items, and drop points will be transferred using HTTP. Moreover, WS will be used to create a socket connection between devices and servers to enable person-to-person messaging, and item request notification will use the WS implementation.

f) Memory constraints;

The application will use 100-120Mb of internal storage and will have a memory usage of 90-95 MB.

3. User characteristics

The application is used by three main types of users which are recipient, and sharer. The sharer user type has sub-categories called individual and consumable service enterprises (restaurant, bakery, supermarket, hotel).

- The user of the application should have a smartphone which has either Android or IOS operating system.
- People who are using the application should be older than 18 and be literate.
- The application user needs to be familiar with the environment of the application stores and the interface of any application. This means that they should be application literate.

The users' main functionalities are listed below.

Table 1: Users' Main Functionality

Users	**Main Functionality**
Recipient	Books reusable, collectible, and consumable items
Sharer	Adds reusable, collectible, and consumable items

4. Assumptions and dependencies

This section will mention the assumptions and dependencies that will affect the main flow or operability status of the program. Thus, these assumptions are made for optimal usage and efficiency of the program. The following are the assumptions and dependencies of the Left-Over!;

- As stated in requirement R1 on the initial plan, the user's smartphone should have either IOS 13.2 (or greater) or Android OS 5.0 (or greater).
- It is assumed that the exact longitudes and latitudes will be found for recycle drop points.
- During the sign-in and log in process, the authenticated data will be used for security reasons.
- The required and efficient database which responds to the queries under one second will be used for storing user and item data.
- Since the application is a server-client based application there is a need for the internet connection. It will be assumed that the users will possess decent internet connectivity.

5. Functional requirements

In this section of the document there will be a list of main functionality and how the system will respond to different situations and different interactions of the user. Moreover, the use case diagram is shown and use cases are explained in detail at description tables.

The following are the Functional Requirements of the Left-Over!;

FR1. The system shall allow the user to sign up to the application by taking user information which includes the users' full name, e-mail, age, password, and address.

FR2. Application shall check user's birth date to ensure the user is greater than 18 years old in the registration process.

FR3. The application shall enable the user to log-in by email and password entrance after the sign-up process.

FR4. The application shall send a recovery email, in case a user forgets his/her password.

FR5. Users shall add an item by entering category, subcategory, photo, and expiration date or status information to be listed.

FR6. The application shall allow users to access their gallery or camera to add items' photos.

FR7. The application shall list all items in a selected category.

FR8. The application shall enable users to search for a specific item.

FR9. The users should be able to display the details of a selected item.

FR10. The application shall allow users to request items.

FR11. The application shall remove the requested items from the list.

FR12. The application shall re-list the requested items if the request has been withdrawn or when determined time is out without transaction.

FR13. The application shall send a notification to the sharer when his/her item is requested.

FR14. The application shall start messaging between the sharer and the recipient when the request is sent.

FR15. The sharer shall update the item transaction process as in-progress, completed or cancelled.

FR16. The application shall remove the items that have a transaction process as in-progress or are completed from the list.

FR17. The application shall re-list the items that have the transaction status as cancelled.

FR18. The application shall remove the consumable items that have an exceeded expiration date.

FR19. The application shall enable admins to add drop points for collectable item categories.

FR20. The application shall show collectable item drop points on the map considering city information.

FR21. The application shall enable the users to view the other users' profiles.

FR22. The application shall enable the users to edit their credentials.

FR23. Users may rate the other users after the transactions.

There is the Use-Case Model of the System;

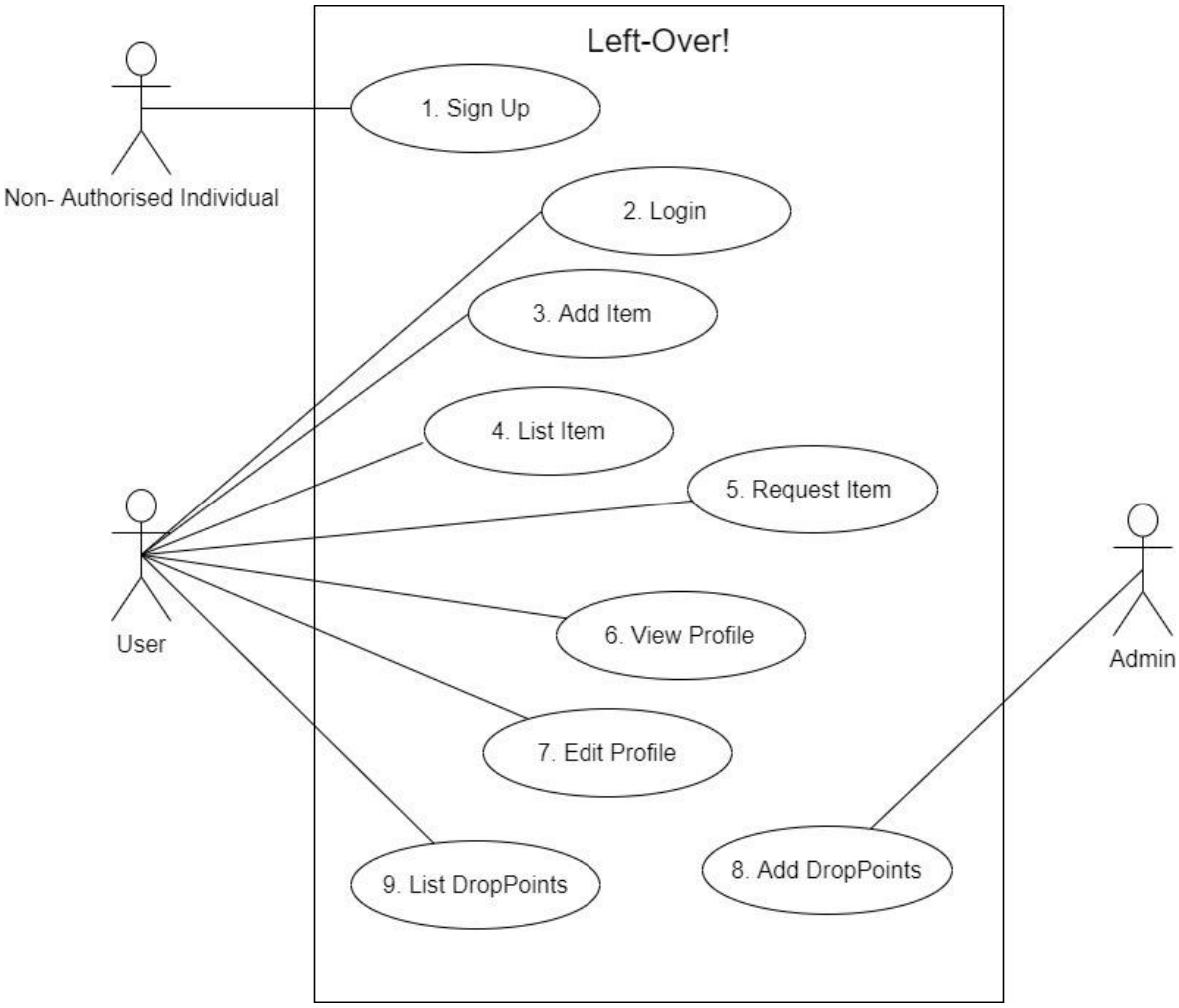

Figure 1: Use Case Diagram

And here are the detailed descriptions of each use case in the system;

Table 2:Use Case 1- Sign up

Use Case No:	Use Case 1
Use Case Name:	Sign up
Description:	Enables Users to create an account on the system
Actors:	Non-authorized Individual
Pre-conditions:	• The system is online • The application has an internet connection
Post-conditions:	User account has been created
Basic Flow:	1. System will ask users email, password, date of birth and other credentials needed for the system 2. Users will fill out the form accordingly 3. System will check if all required fields are entered 4. System will check if there is a registered user with the same email 5. System will check the password and retype password fields' equality 6. System will check if the user is above 18 years old 7. Connect to the database 8. Users Information will be saved to the database 9. The user's account will be created.
Alternative flows	A3.1 All fields are not filled by the user A3.2 System will display a message about empty fields A3.3 Continue with the Step 2 in the basic flow A4.1 Users enter an email registered on the system

	A4.2 System will inform the user saying that email is under use A4.3 Continue with Step 2 in the basic flow A5.1 Password and Re-Type password fields does not match A5.2 System will show an error message saying passwords does not match A5.3 Continue with Step 2 in the basic flow A6.1 User's age under 18 years old A6.2 System will give an error message that says you need to be at least 18 years old to register A6.3 Continue with Step 2 in the basic flow A7.1 Database System is not responding A7.2 Try to connect to database repeatedly for 1 minutes A7.2.1 If database responds, continue with the Step 6 in the basic flow A7.2.2 If the database does not respond, give an error message to the user, and close the application.
Frequency of Use:	Frequently
Priority:	Medium

Table 3: Use Case 2 - Login

Use Case No:	Use Case 2
Use Case Name:	Log-in
Description:	Enables users to log-in to the system
Actors:	User
Pre-conditions:	• The system is online • The user has an internet connection • User logged-in to the system
Post-conditions:	User has logged in to the system
Basic Flow:	1. System will ask for users email and password 2. System will check if there is a registered user with that email 3. System will check if the given password is true 4. The user will be logged in to the system.
Alternative flows	A2.1 Users enter an email that is not registered to the system. A2.2 System will inform the user saying that email is not registered to the system A2.3 Continue with Step 1 in the basic flow A3.1 Password does not match with the user's password A3.2 System will inform the user that is given password is not correct A3.3 Continue with Step 1 in the basic flow
Frequency of Use:	Frequently
Priority:	Medium

Table 4: Use Case 3 – Add Item

Use Case No:	Use Case 3
Use Case Name:	Add Item
Description:	Add Items to the Database
Actors:	User
Pre-conditions:	• The system is online • The user has an internet connection • User logged-in to the system
Post-conditions:	An Item from any category created.
Basic Flow:	1. User clicks the add item button 2. Application opens the add item page 3. System will ask for item details from the user. 4. System let user to take or upload a photo 5. System will check all required fields are filled 6. System will check if the expiration date of the item is past the current date. 7. System will insert the data of the item to the database. 8. System will prompt the user about item creation
Alternative flows	A4.1.1 User did not give camera access permission to application A4.1.2 App will prompt user to give camera access permission A4.1.3 Continue from basic flow 2 A4.2.1 User did not give gallery access to application A4.2.2 Application will prompt user to give gallery access permission

	A4.2.3 Continue from basic flow 2 A5.1 All fields are not filled by the user A5.2 System will display a message about empty fields A5.3 Continue with the Step 3 in the basic flow A6.1 User enter an item that has been expired A6.2 System will inform the user about that item has been expired A6.3 Continue with Step 2 in the basic flow A7.1.1 Database System is not responding A7.1.2 Try to connect to database repeatedly for 1 minutes A7.1.2.1 Database responds continue with the Step 5 in the basic flow A7.1.2.2 Database does not respond A7.1.2.3 Application gives error message and terminate use case A7.2.1 Database Insertion Failed
Frequency of Use:	Very Frequently
Priority:	High

Table 5: Use Case 4 - List Item

Use Case No:	Use Case 4
Use Case Name:	List Item
Description:	List item takes the items in the selected or searched category from the database and shows them on the screen.
Actors:	User
Pre-conditions:	• The system is online • The user has an internet connection • The user is logged in • There must be added items in the database to be listed
Post-conditions:	-
Basic Flow:	1. User opens Home Page 2. User clicks a category on the Home Page 3. Application opens the selected category's page 4. Application Connects to the database. 5. Items that are specified in the selected category are taken from the database 6. Taken items are listed on the screen
Alternative flows	A2.1. User clicks on the search bar A2.2. User is directed to the Search Page A2.3. User enters the category name A2.4. User clicks the search button A2.5. Continue with step 4 in the basic flow A2.3.1.1. User enters a non-existing category name A2.3.1.2. Application shows "There is no item in that category" message A2.3.1.3 Terminate activity

	A2.3.2.1. User clicks on the back button A2.3.2.2. Continue with step 1 in the basic flow A2.3.2.1.1. User clicks on the home button A2.3.3.1.2. Continue with step 1 in the basic flow A4.1 Database System is not responding A4.2 Try to connect to database repeatedly for 1 minutes A4.2.1 Database did not respond A4.2.2 Database problem try later message will be shown on the screen A4.2.3 Application direct user to the home page A5.1. There is no item specified under the selected category in the database A5.2. "There is no item in that category" message will be displayed on the screen A5.3. Terminate activity
Frequency of Use:	Frequently
Priority:	High

Table 6: Use Case 5 – Request Item

Use Case No:	Use Case 5
Use Case Name:	Request Item
Description:	It allows users to send a request for a selected item
Actors:	User
Pre-conditions:	A user must select an item to be requested
Post-conditions:	• The system is online • The user has an internet connection • The user is logged in • The requested item will be hidden from other users • The application will send a notification to the sharer • The application will create a chat between sharer and recipient
Basic Flow:	1. User selects an item. 2. User clicks on the request button.
Alternative flows	-
Frequency of Use:	Frequently
Priority:	High

Table 7: Use Case 6 – View Profile

Use Case No:	Use Case 6
Use Case Name:	View Profile
Description:	User Profile will be displayed.
Actors:	User
Pre-conditions:	• The system is online • The user has an internet connection • User logged-in to the system
Post-conditions:	-
Basic Flow:	1. The user will click a profile to display. 2. The system will display the clicked user's profile
Alternative flows	-
Frequency of Use:	Frequently
Priority:	Medium

Table 8: Use Case 7 – Edit Profile

Use Case No:	Use Case 7
Use Case Name:	Edit Profile
Description:	User Profile will be edited.
Actors:	User
Pre-conditions:	• The system is online • The user has an internet connection • User logged-in to the system
Post-conditions:	-
Basic Flow:	1. User clicks the edit button 2. Edit page will be opened 3. The user will edit his/her information 4. User click the save button 5. User will enter his/her password 6. Changed information will be saved
Alternative flows	A5.1 The password does not match with the user's password A5.2 System will display a warning message A5.3 Continue with Step 4 in the basic flow
Frequency of Use:	Less frequently
Priority:	Low

Table 9: Use Case 8 – Add Drop Point

Use Case No:	Use Case 8
Use Case Name:	Add drop point
Description:	Adds a new drop point to the drop point list.
Actors:	Admin
Pre-conditions:	• The admin panel must be opened.
Post-conditions:	-
Basic Flow:	1. The admin opens the admin panel. 2. The admin clicks the add new drop point button. 3. The admin enters the drop point name. 4. The admin enters the drop point coordinates. 5. The admin presses the save drop point button.
Alternative flows	-
Frequency of Use:	Not frequently
Priority:	Low

Table 10: Use Case 9 – List Drop Point

Use Case No:	Use Case 9
Use Case Name:	List drop point
Description:	Lists the existing drop point list.
Actors:	User
Pre-conditions:	The user must be logged-in to the system.
Post-conditions:	• The Left-Over! system must show the new added drop point in the drop point list with name and map view.
Basic Flow:	1. The user presses the drop point list button. 2. The Left-Over! system gets the user's coordinates. 3. The Left-Over! system lists the existing drop points' names and coordinates considering the user's current location.
Alternative flows	A3.1 The user's coordinates could not be detected. A3.2 Show the drop point list located in Ankara.
Frequency of Use:	Frequently
Priority:	Medium

6. System Model

Considering the Use Case narratives mentioned in Section 5, activity diagrams, system sequence diagrams and analysis class diagrams of the add item and list item use cases will be displayed in this part.

- Add Item and List Item Activities Class Diagram

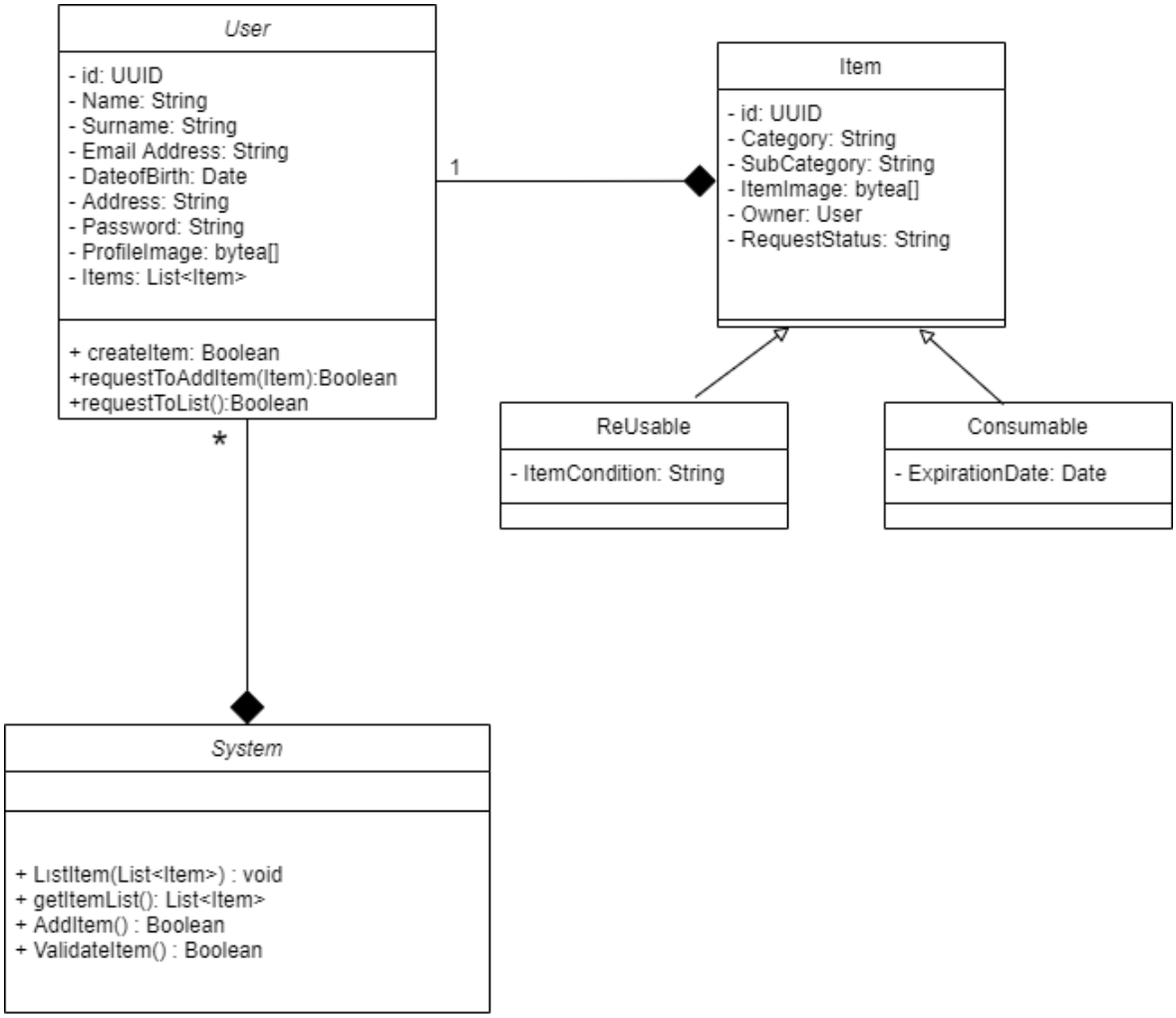

Figure 2: Class Diagram

- Add Item Activity Diagram

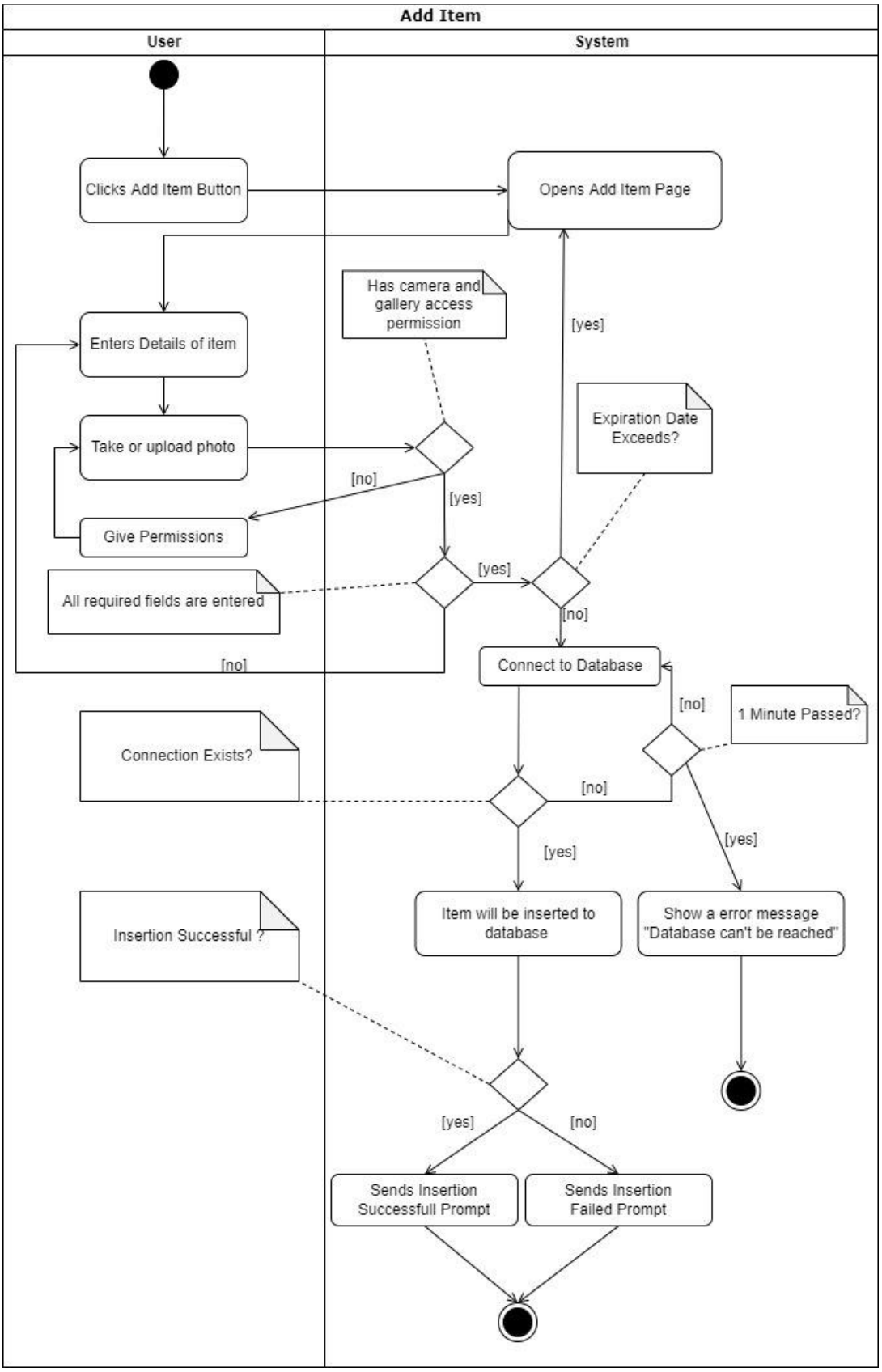

Figure 3: Add Item Activity Diagram

- List Item Activity Diagram

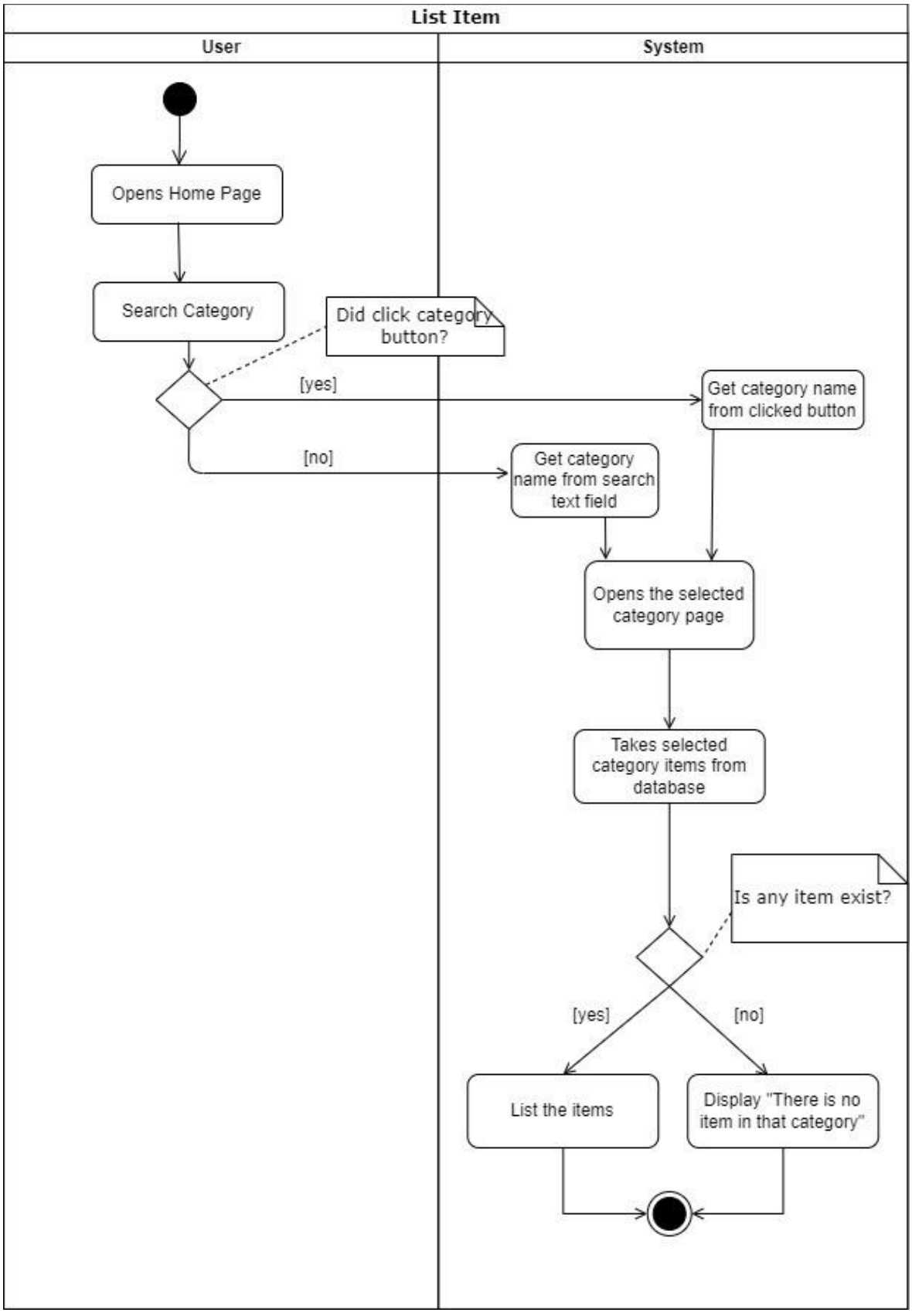

Figure 4: List Item Activity Diagram

- Add Item Sequence Diagram

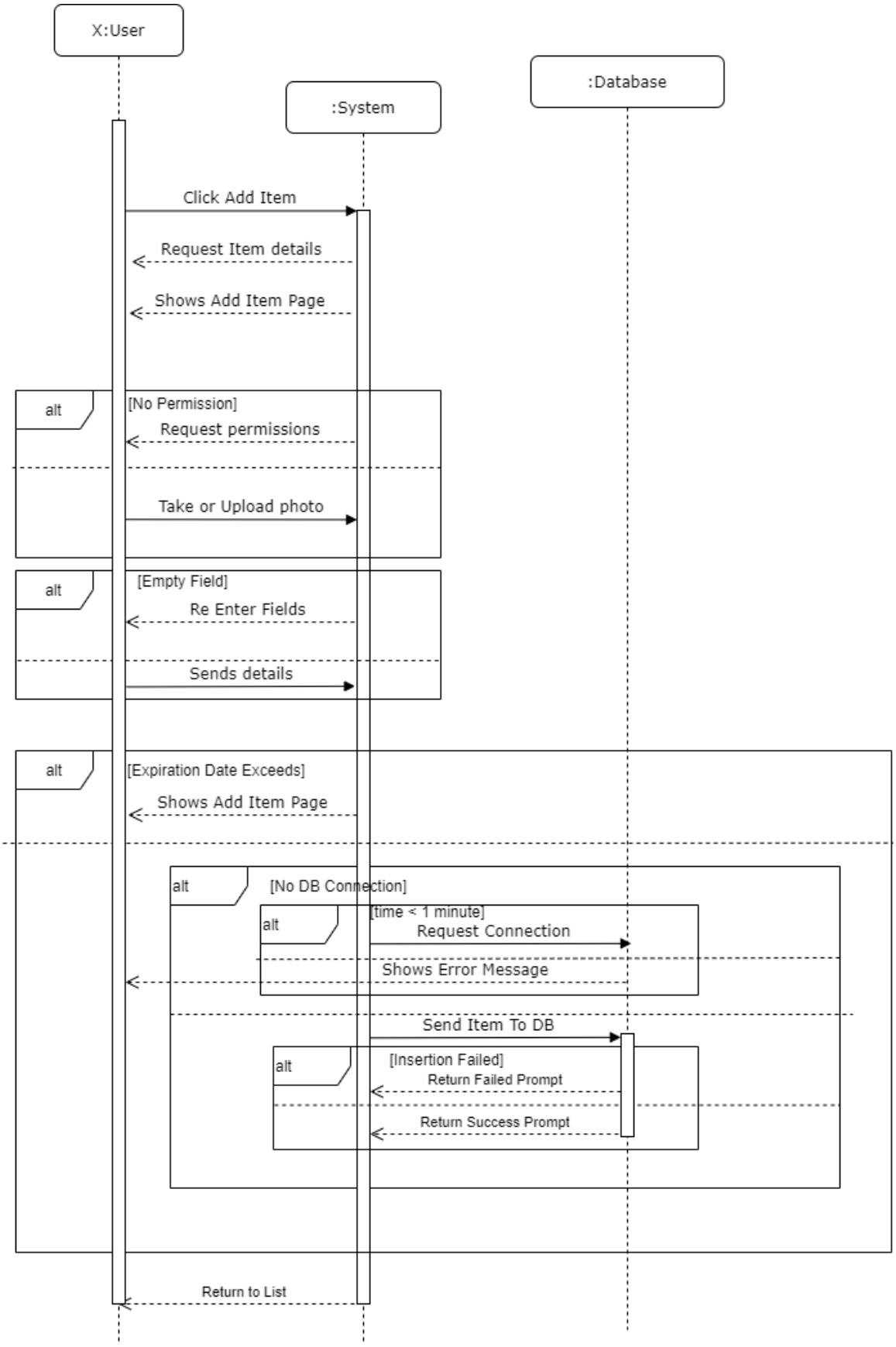

Figure 5: Add Item Sequence Diagram

- List Items Sequence Diagram

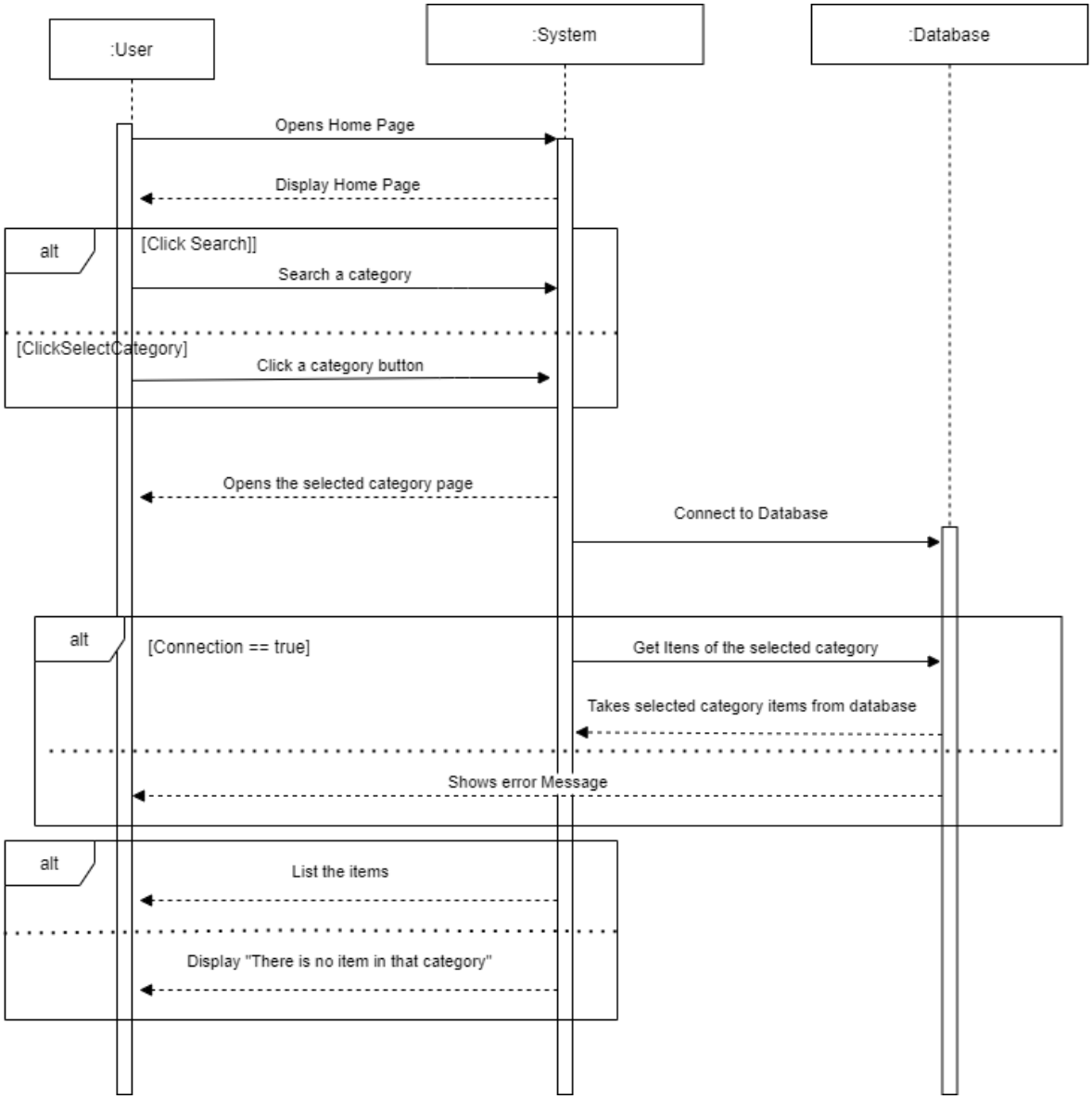

Figure 6: List Items Sequence Diagram

7. Requirements Prototypes

The following section mentions the prototypes of the project's two most important use-cases which are the add item and list item. This part will give a general idea about how the system will look and work considering modeled use cases above.

The following two figures illustrate the List Item of the system. When a user chooses a category which is consumable in this diagram, the subcategories under the related section will be displayed with an item image and their enterprise name. This screen provides a limited number of items for each category. Whenever the user clicks the view more button, several subcategory items will be displayed. By clicking the see more button in the second figure, a user can display more items.

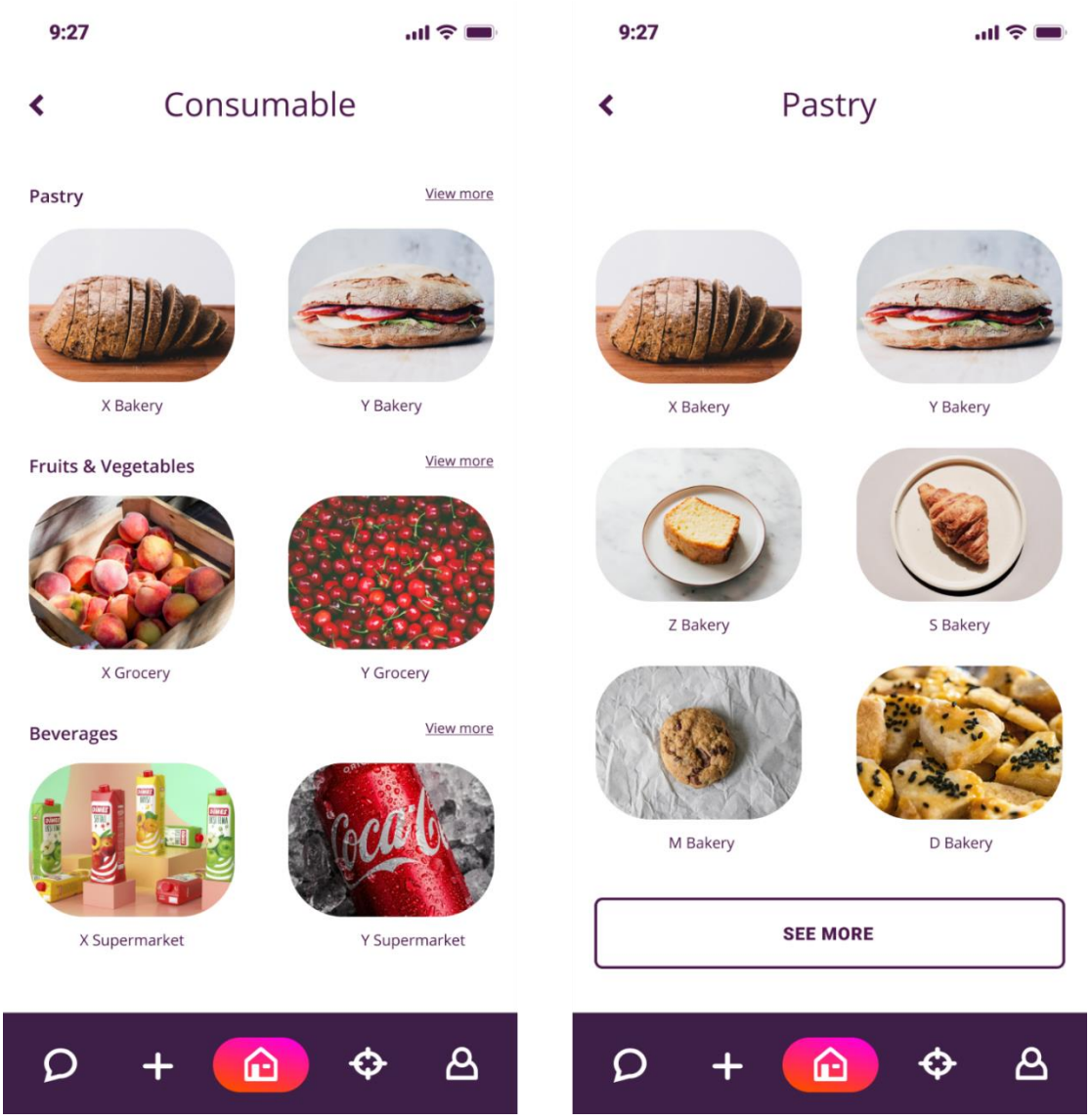

Figure 7: List Items Prototype

The consecutive diagrams indicate the add item prototype. A user will select the category and subcategory depending on the item which will be added. If the user selects the consumable category, the expiration date will be entered. However, if the user chooses the category of reusable, status will be entered. Furthermore, for both selections, the user will upload the item photo. By clicking the Add Item button, the item will be added to the system.

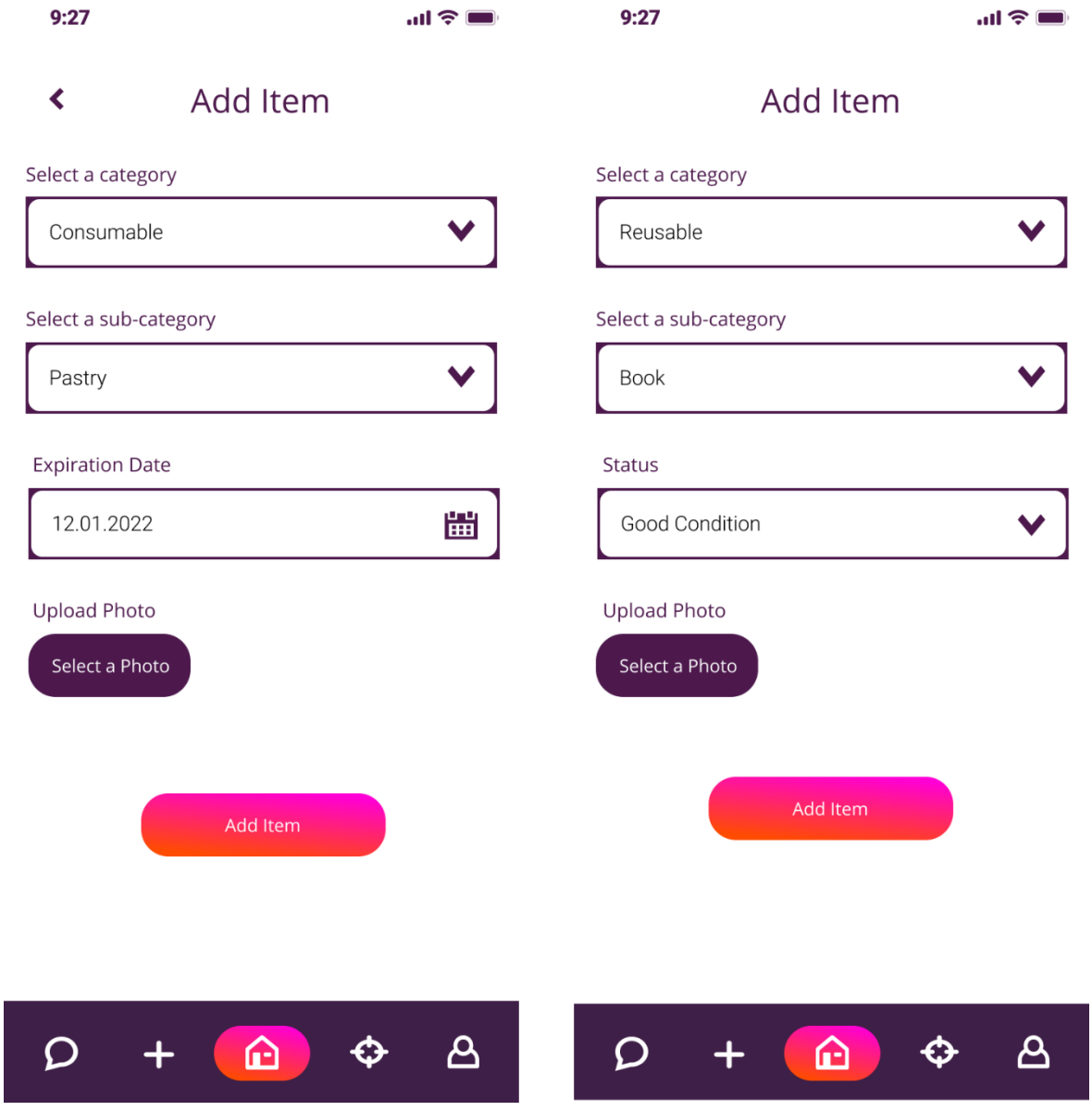

Figure 8: Add Item Prototype

8. Non-functional Requirements

This section of the document will specify the non-functional requirements of our software and will be inspected under 4 different subcategories which are usability, performance, software system attributes and other non-functional requirements.

1. Usability requirements

1.1- Application user interface design should follow material design concepts.

1.2- Labels on the buttons should be understandable.

1.3- Icons used in the application should be compatible with the provided service.

1.4- All user interface elements must resize according to the screen size of the device.

2. Performance requirements

2.1- 99% of the request should be processed less than 1 second.

2.2- 95% of the time response time of the system should be under 5 seconds.

3. Software system attributes

3.1- System should be available to users 95% of the time.

3.2- System should include a daily backup plan to prevent data loss of the users.

3.3- System should use hashing and encrypting techniques to store the critical information of the user.

3.4- Encryption method to use should be SHA256.

3.5- System should create logs at certain events such as creating a user and registering an item.

3.6- Logs that are created by the system should be trackable.

3.7- Critical parts and interfaces of the program must be used only by authorized users.

3.8- User Application should be written in Flutter.

3.9- Server Application should be written in NEST.js.

3.10- A User should only change data that is registered by themself.

4. Other Non-Functional Requirements

There are no other non-functional requirements. All non-functional requirements are determined above.

9. Logical Database Requirements

This part of the document will bring clearance to ways how the system stores data which is going to be used by the product to function correctly. Thus, this part will consist of the entities in the system, how the product stores that information, frequency of use for the entities, how products access the data store, how entities relate to each other in terms of relationships, integrity, and retention requirements.

Here is a list of important entities in the system and what they represent in the system.

- **User:** Representation of a user of the system, a user can have a specific type that typifies the different roles a user has.
- **UserType:** Represents the classification of the user a user must have a single UserType on the other hand a UserType could be an element of multiple users. If there are any Users which are using a record of UserType system will not allow that entry to be removed.
- **ConsumableItem:** Represents an item that is going to be shared between users under consumable category and that is registered to the system.
- **ReusableItem:** Represents an item that is going to be shared between users and under the reusable category that is registered to the system.
- **DropPoint:** Represents a DropPoint on the map owned by Recycling Companies and Municipality that collects the recyclable waste like glass, paper, and metal. A DropPoint could be related to a User which has a specific UserType.
- **Comments:** Represents a comment made to a user by a user to rate and give opinions of the user. It has a strong relationship with the entity user. A comment must have a user and a User could have multiple comments. When a User is removed from the data store all related comments are deleted accordingly.
- **BookingRequest:** This entity represents the transaction between two users. It has a relationship with the User and ReUsableItem Entity. This entity is a frequently used entity since it tracks the core activity of the activity.

10. Constraints

The constraints in the software project are the most limiting factors for the software production of the project team. Several external, regulatory or project limitations may occur in the Left-Over! project. However, these constraints will be minimized by the team by understanding their definition clearly and finding solutions to reduce them.

Schedule is one of the constraints in the Left-Over! project. Project has a strict deadline for the software tasks and the project team must stick to this schedule. This situation can put pressure on the team for very time-consuming tasks and reduce software productivity.

Furthermore, Left-Over! will be developed by a software development kit called Flutter which released in 2017. It is a relatively new software development kit, and this may result in an insufficiency to find related resources or information while conducting research about the software.

11. Verification

Software verification process is an essential part of the project because it provides the evidence that the system or system element performs its intended functions and meets all the expected requirements [1]. In order to find the errors, gaps, or missing requirements in comparison to the actual requirements, various testing tools and techniques will be used in the project.

The four main stages of testing which are unit testing, integration testing, system testing, and acceptance testing will be considered as testing techniques to be used while testing the software. Unit testing will focus on the specific units of the software to determine whether each one is fully functional. Integration testing will allow the team members to combine all the units within a program and test them as a group. System testing will be a complete application test to evaluate whether the system fully satisfies all the expected requirements. Acceptance testing will determine whether the system is ready for release to customers or jury [2].

Although the project software development team will manually test whether the software meets the expected requirements by looking at the product from the end users' perspective, some test automation tools will be used to get more accurate and faster results. Amazon web services device farm will be used to test and qualify the software.

12. Discussions

The discussions mention the validity and applicability of the requirements document. Furthermore, SRS's effects are interpreted in social and universal aspects considering their social, environmental, and legal implications.

1. Limitations and Constraints

Limitations and constraints may influence how the team will manage the projects. The limitations can fall into several categories. By recognizing these categories, the effect of them can be minimized. Scheduling is the crucial limitation that the team must follow. The determined time periods of the software process model and deadlines should be considered. The constraint is determining the software requirements since there is no customer and a specific end user, the team members must identify the requirements by themselves.

2. Applicability of the project under real life situations

Under real life situations the requirements document will need to have more generic scope and the team has a chief engineer to check the work done in the requirement phase. However, this team does not have such a chief to check the work done before submitting it. Also, companies may design and focus their requirements for a specific user group but in this case, there is no specified target group.

3. Health and Safety Issues

While writing this document team members can be stressed because of the tight schedule and heavy workload. This excessive stress might cause different effects on team members' psychology. Also, while developing the project team members will use computers and will have to look at screens for long hours so this might cause some sighting issues.

4. Legal Issues

According to the Turkish Government, processing of the personal data such as date of birth, address, and contact information should be agreed upon by the users. Thus, when the team is writing the requirements of the project this law should be considered.

5. Economic Issues and Constraints

During writing the SRS document, team members must stay late for many days, so some of them had to turn home by taxi. Also, since it was a long document, team members gathered more frequently, and it affected gasoline expenses of team members who have cars. Furthermore, since the team does not have an office to work in, meetings were made outside, which led to an increase in food and coffee expenses.

6. Sustainability

A common point around school is determined as a meeting point to make SRS document meetings regularly. Moreover, software requirements are designed in such a way that further software and requirement improvements could be easily adopted.

7. Producibility-Manufacturability

Left-Over! does not serve a tangible product directly, it offers software. Hence, there will not be continuous production. Therefore, the requirements are determined in a way that once it is released, it will not be touched except for occurred problems, new feature development, and compulsory upgrades.

8. Social, Political and Ethical Issues

As mentioned in the social, political, and ethical Issues part of the initial plan, many consumable service enterprises do not participate in recycling efforts. Left-over! product requirements will be planned to attract these enterprises to contribute recycling. Thus, in a world where many people suffer from hunger and misery, it will be more ethical and humane not to waste and contribute to recycling. In addition, product requirements are designed regarding the privacy of users. Confidential information of the people who want to donate or request an item will not be publicly disclosed or shared with others.

9. Multidisciplinary Collaboration

Multidisciplinary collaboration is an indispensable part of the project to broaden the horizon of the team, gain a new and correct perspective. Considering the project requirements, opinions of people from the relevant disciplines are taken to determine the requirements correctly. Project team consulted the Bilkent University Faculty of Law students in order to observe the legal compliance of the project requirements. In

addition, potential end users are contacted, and product requirements is changed considering their opinions.

10. Environmental Issues

Left-Over! product requirements aim to influence the environment directly because as mentioned in environmental issues part of the initial plan, it is designed to contribute to sustainability by increasing recycling and preventing waste. In this sense, product requirements are planned to make a platform for the people and consumable service enterprises to bring them together and create a sharing mechanism.

13. References

[1] "Verification", *Dau.edu*, 2021. [Online]. Available: https://www.dau.edu/tools/se-brainbook/Pages/Technical%20Processes/verification.aspx. [Accessed: 08- Nov- 2021].

[2] "The Four Levels of Software Testing", *Segue Technologies*, 2021. [Online]. Available: https://www.seguetech.com/the-four-levels-of-software-testing/. [Accessed: 08- Nov- 2021].