CTIS411 Senior Project I

Software Design Description

Left-Over!

TGEO

Furkan Kılıç, 21601729

Mehmet Melikşah Özceyhan, 21703661

Nevin Didem Bilgihan, 21601723

Yağmur Cansu Gürkan, 21601486

Bilkent University

Department of Information Systems and Technologies

2.1.2022

Change History

File Name	Document Type	Deliverable	Version	Submission Date
Deliverable_4_v0.1	Google Docs	4	0.1	-
Deliverable_4_v0.2	Google Docs	4	0.2	-
TGEO_Deliverable_4_1	PDF	4	1	02.01.2022

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 gives detailed information about software design of the project. It covers analyzing models and plans which includes functional and non-functional requirements, software increments, and detailed software development environment explanation. The first increment consists of sign up, login, list and display item use cases, so the block diagram, interaction diagram, and deployment diagram drawn in the Software Design Description document are mostly related to these use cases. The third part, which is high-level design, mainly indicates architectures that are going to be used when developing the application. Decided architectures are Representational State Transfer and Model-View-View Model. The reason why they are chosen is that they are popular, extensible and the frameworks going to be used are naturally compatible with them. Furthermore, in the low-level design section, the first increment's use cases' pseudo-codes and their explanations are specified. Also, the discussion part details the impact of the Software Design Description document in many aspects.

Table of Contents

Page Number

List of Tables

Pages

List of Figures

Pages

Abbreviations

API	Application Programming Interface
FR	Functional Requirement
IDE	Integrated Development Environment
SDD	Software Design Description
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
URL	Uniform Resource Locator

1. Scope

This section of the document will briefly mention the planned activities that are going to be done by the team during the preparation of this document. The following parts cover the analysis model & planning, which includes functional and non-functional requirements, software increments, and frameworks, libraries, services, databases, and Application Programming Interfaces (APIs). Also, high-level (architecture) design is explained with logical, process, and physical view diagrams and design quality. Moreover, the low-level design of the use cases determined for the first increment, which are sign up, login, list_item, and display_item_detail, are shown with their algorithms. In this way, the productivity of the team while coding is aimed to be increased because of the clarity of what to do.

The Software Design Description (SDD) document is prepared based on the system required to develop Left-Over!. The requirements and use cases are mostly the same as the second version of the Software Requirements Specification (SRS) document. They are also used while drawing diagrams. Moreover, written frameworks, libraries, services, databases, and APIs, and planned incrementations are mentioned in the second version of the Software Project Management Plan (SPMP) document. In this document, high-level architecture and low-level design are determined newly. As a result, what to implement and how to implement the code is brought clear.

2. Analysis Model & Planning

1. Functional Requirements

In this section of the document, there will be a list of main functions 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 in version two of the SRS document.

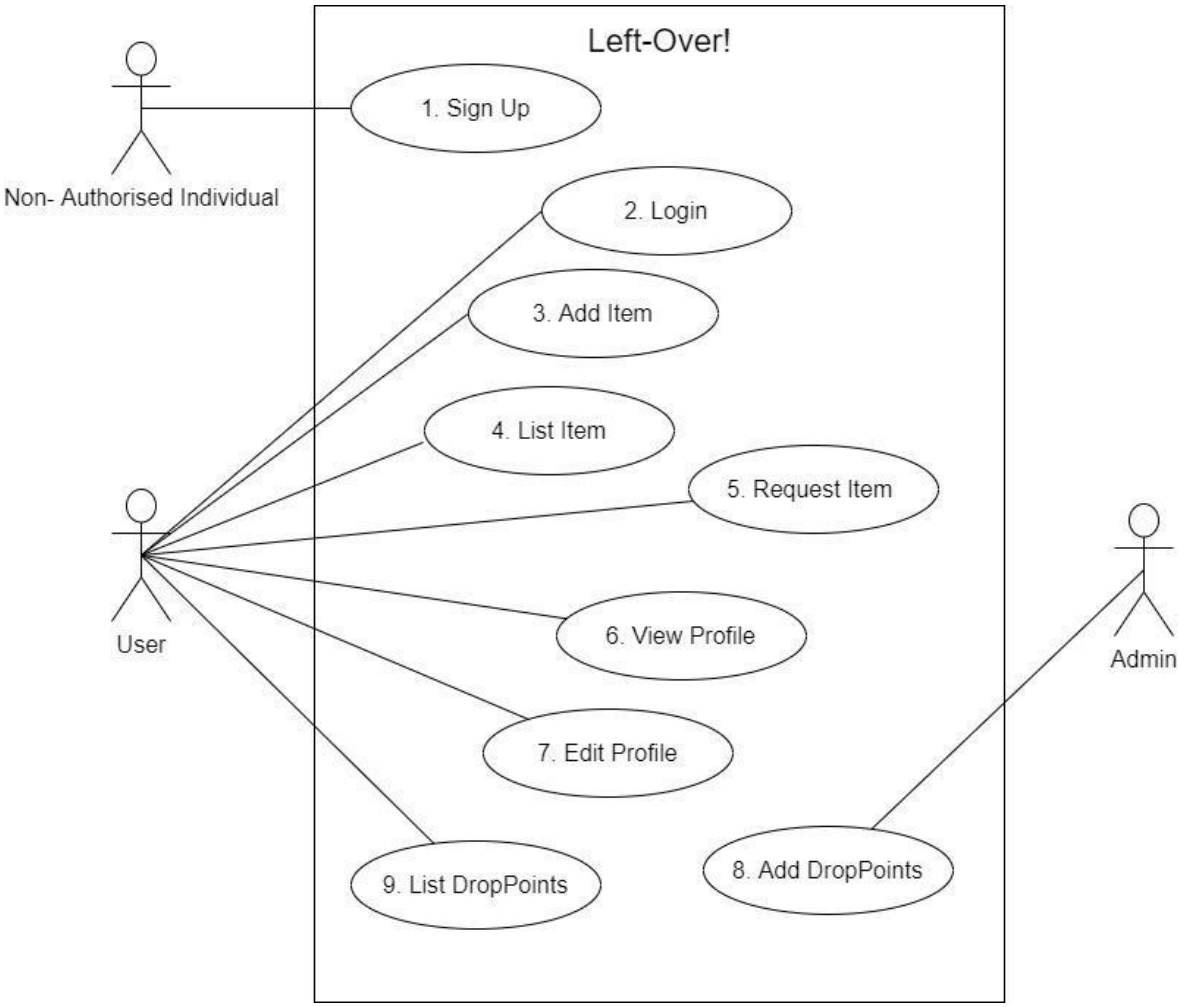

Figure 1: Use Case Diagram

The following are the Functional Requirements (FR) 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 canceled.
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 canceled.

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.

2. Non-Functional Requirements

This section of the document will specify the non-functional requirements of our software and will be inspected under 3 different subcategories which are usability, performance, and software system attributes.

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 in less than one second.

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

3. Software Increments

For the first increment which is 20% of the final product:

In the back-end part of the project, login, sign-up, list items and display item details endpoints will be provided. Database installation will be made, and related tables will be created to store user and item information. In the front-end part, login, sign-up, list items and display item details pages will be designed.

For the second increment which is 40% of the final product:

Add item and request item parts of the project will be completed. Back-end side will provide related endpoints and the front-end side will design the add item page and when a request is made, a notification will be displayed to the user.

For the third increment which is 60% of the final product:

In the back-end side, socket.io endpoints will be provided to enable messaging for the request item messaging part. Also, view user profile endpoints will be coded. On the front-end side, the messaging page and view user profile pages will be designed.

For the fourth increment which is 80% of the final product:

Add drop points and list drop points endpoints will be coded on the back-end side. Admin will use the add drop point endpoint to send a JSON post request to the database. Front-end side will design the list drop points page.

For the fifth increment which is 100% of the final product:

As the final increment of the project, manual tests will be made. Also, with the help of the Amazon device farm, device compatibility tests will be made at this increment. After the test phase, the application will be packaged according to the Operating system specification. Moreover, the deployment of the application to the Play Store and App Store will be done. In the meantime, the back-end server application will be containerized and will be uploaded to Amazon Web Services. Thus, the application will be ready for market release.

4. Frameworks, Libraries, Services, Databases and APIs

The following table indicates information details about tools, libraries, programming languages, frameworks, database management system, API, document editor, and Integrated Development Environment (IDE) that are planned to be used including versions, brief descriptions, and Uniform Resource Locator (URL) of them.

All of the following frameworks, libraries, services, databases, and APIs but i18n and Google Maps for flutter will be started to use in the first increment. i18n will be implemented in the third increment and Google Maps for flutter is planned to use in the fourth increment.

Table 1: Software Development Environment

Name	Type	Version	Description	URL
Dart	Programming Language	2.12.4	Dart is a language optimized for fast apps on any platform which empowers flutter.	https://dart.dev
Typescript	Programming Language	4.5	TypeScript is a typed version of javascript which empowers NestJS.	https://www.typescriptlang.org
Flutter	Framework	2.5.3	Flutter will enable the creation of android and iOS apps from a single codebase.	https://flutter.dev
NestJS	Framework	8	NestJS framework will be used for building efficient, scalable server-side applications which will be used by mobile applications.	https://nestjs.com
PostgreSQL	Database Management System	12	The product will use PostgreSQL as a database management system to store data.	https://www.postgresql.org
Google Maps for Flutter	API	2.1.1	A Flutter plugin that provides a Google Maps widget.	https://pub.dev/packages/google_maps_flutter

Name	Type	Version	Description	URL
Google Docs	Document Editor	1.21.442.01.30	A document editor which enables collaborative works. It is used for creating project documents.	https://docs.google.com/
Microsoft Project	Project Management Tool	Version 2110	Microsoft Project is used to create a Gantt chart that lists the tasks in Leftover! project.	https://www.microsoft.com/en-us/microsoft-365/project/project-management-software
Draw.io	Chart and Diagram Tool	13.9.9	Draw.io is used to create class, activity, sequence, etc. diagrams of the system.	https://drawio-app.com
Figma	UI Design Tool	1.0	Figma is a UI/UX design tool for prototyping, design, and code generation.	https://www.figma.com
GitHub	Version Management Tool	2.34.0	Used to keep track of the versions of the project and to enable the project team to work collaboratively on code.	https://github.com
Moment.JS	Library	2.29.1	Moment.js is a package that will ease the use and modify the date data.	https://momentjs.com/
Socket.IO	Library	4.4.0	This library will provide an easy way to develop a real-time messaging platform.	https://socket.io/
Docker	Software	4.2.0	It enables the product to be packaged into a container which is a standardized executable component combining application source code with the Operating System libraries and dependencies required to run that code in any environment.	https://www.docker.com/

Name	Type	Version	Description	URL
i18n	Library	0.13.3	i18n will provide multi-language support thus it will make the application available in different languages.	https://www.i18next.com
Trello	Collaboration Tool	2.10.8	Trello is a tool to provide an environment to collaborate and manage projects.	https://trello.com

3. High-Level (Architecture) Design

1. Logical View

The responsibility of the User Management service is to maintain the state of user management and it uses the functions of services connected, which are Adding User, and Commenting and Rating Services. Item Management represents users engaged in item management. Item Management uses functions of Listing Item service to show items with a specified key or category, Adding Item service to add an item to the database, and Requesting Item service to allow item transaction between user pairs. Drop Point Management is responsible for drop point liability and maintainability. It uses the Listing Drop Point service to show current and available locations by using physical address mapping and Adding Drop Points service to allow location entries. The block diagram of the system is shown below.

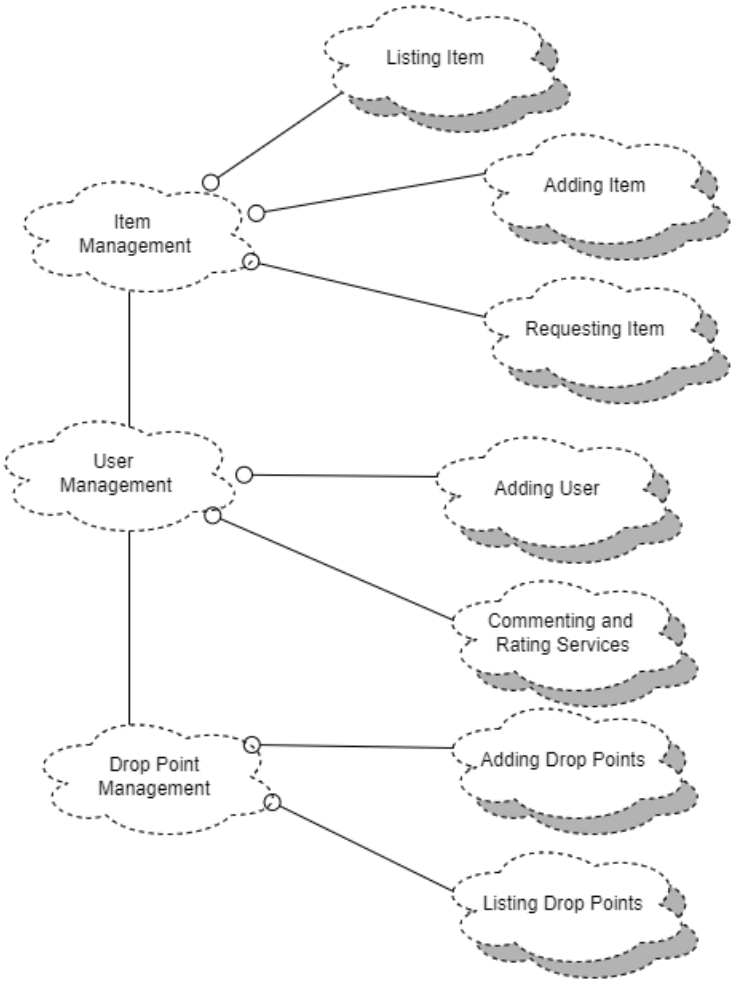

Figure 2: Logical View

2. Process View

In the following figure is the interaction diagram of the software system Left-Over! It gives a general idea about how the classes interact with each other to complete tasks issued by the users of the Application. Since the system makes use of the client-server architecture the mobile application is indicated as a block inside the interaction diagram.

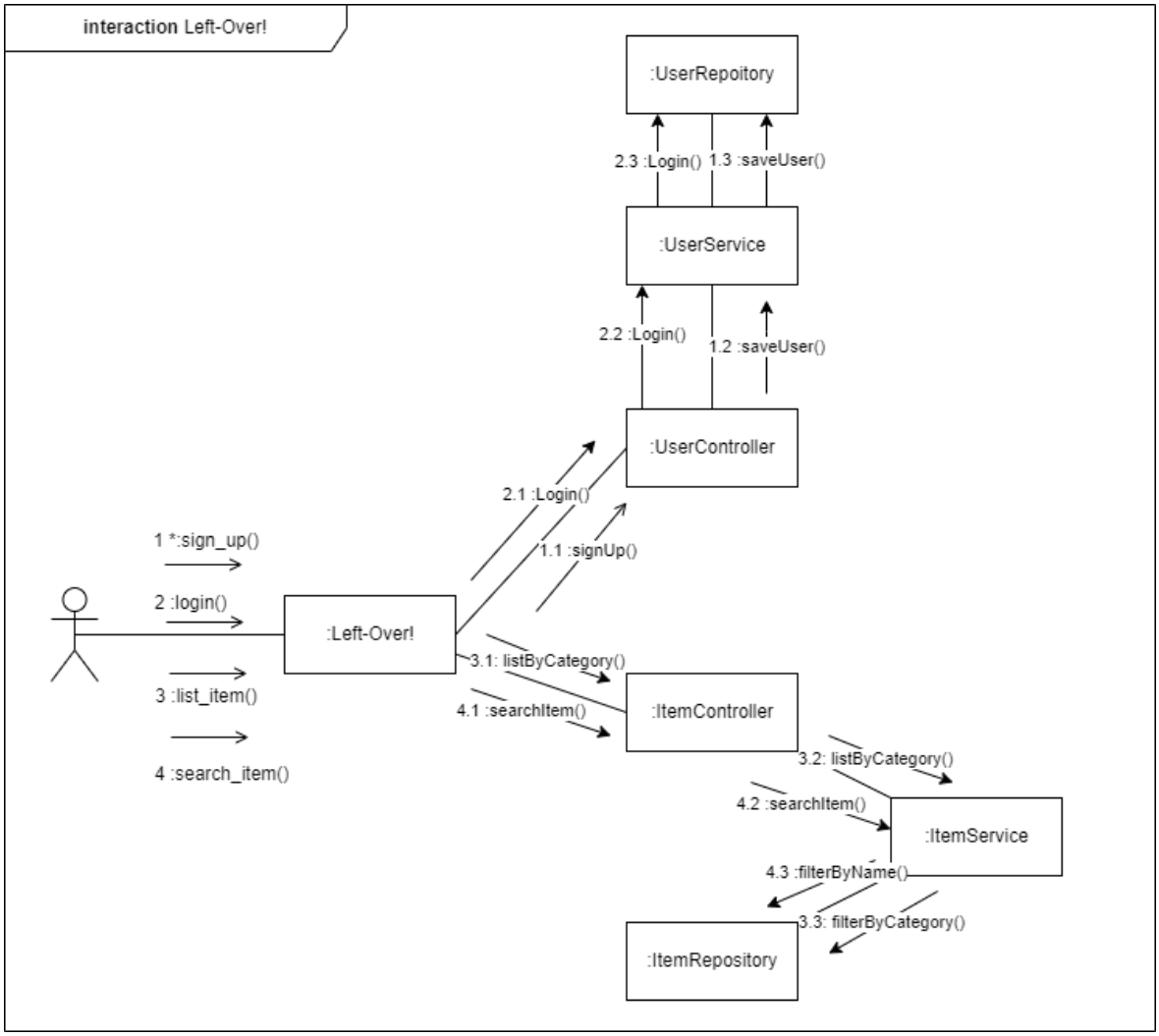

Figure 3: Process View

3. Physical View

Below is the deployment diagram of the project Left-Over!. Which clarifies the application's deployment details and which physical devices to use at the production stage. Also, this diagram clarifies the protocols that are going to be used between front-end and back-end and how the server-side application interacts with the database.

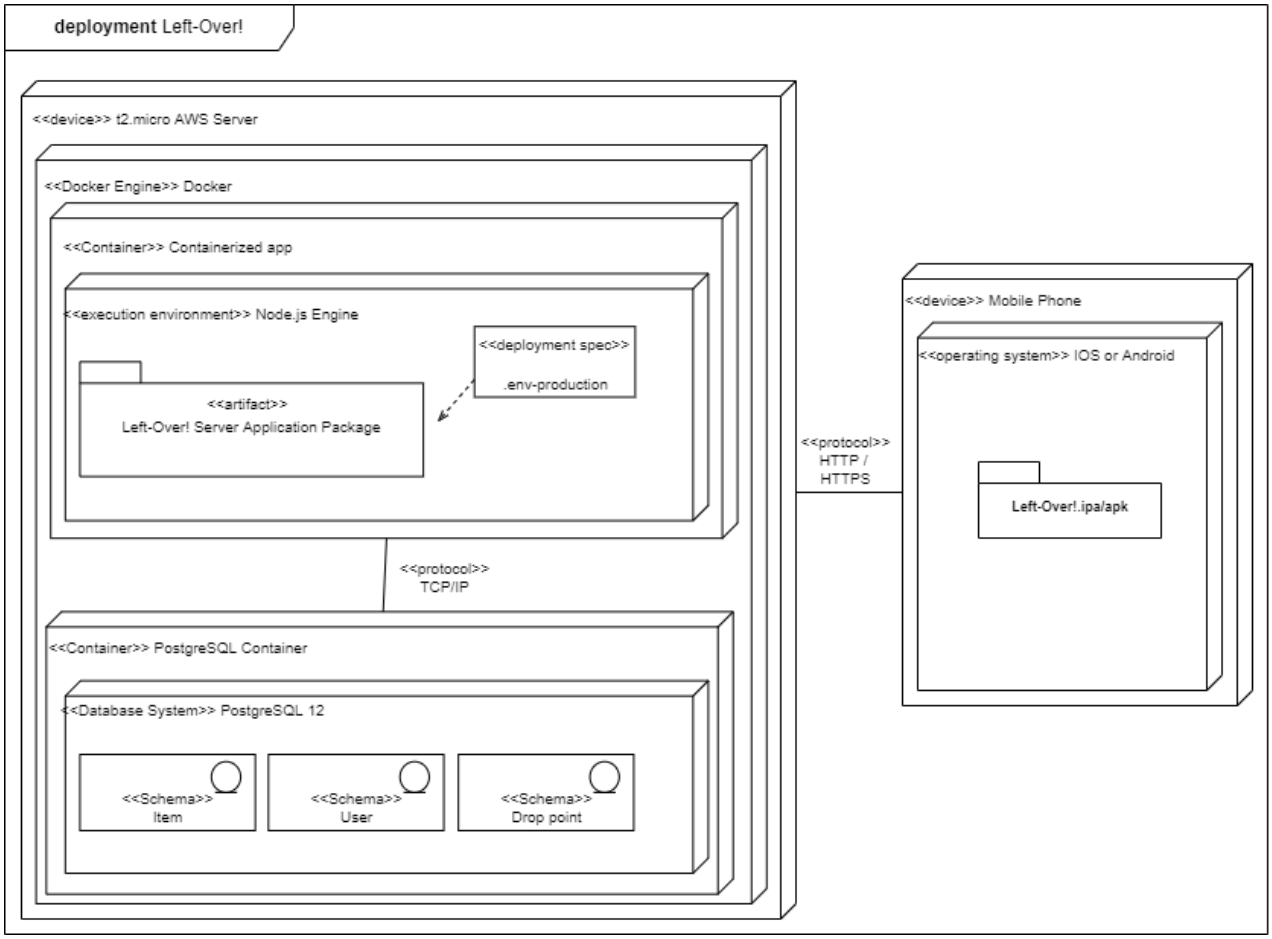

Figure 4: Physical View

4. Design Quality

The application's main architecture is based on the client-server architecture. The application will process all the valuable information on the server-side. And all the data which is crucial for the system to operate is held on the servers which host the NestJS application. The NestJS application will be written according to the Representational State Transfer principles. The Flutter Application, on the other hand, will follow the Model-View-View Model design principles which helps us to divide the functionality into different and meaningful modules, hence improving the code quality and allowing programmers to encounter fewer bugs and errors while developing the application. The system will be hosted on the Amazon Web Servers, as a result of this hosting the development team does not have to think about scalability and also reliability of the system. All the problems that might occur in those areas are handled by Amazon itself.

4. Low Level Design

- **Login Page:** User enters email address and password fields to authenticate their account. If the email address exists, then the system will check whether the password is correct or not. If the email address does not exist, the application will display a message as “Please register first!”.

Data: *Email_address and Password*

Result: *The authentication*

If Email_address is existed then

    If Password is correct then

        Goto home_page

        print “The account is authenticated.”

    Else

        print ”Invalid Password!”

    Endif

Else

    print “Please register first!”

Endif

End

- **Sign up:** The user will provide the required fields which are username, email address, password, confirmation password, date of birth, city, and address to register the system. The system will check the email address's existence and field entrance. If the request is satisfied, the application will control whether the required field entries. Afterward, the application will control the password and the confirmation password are equal to each other. According to if checks, related messages will be displayed.

Data: *Username, Email_address, Password, Password_confirmation, Date_of_birth, City, Address*
Result: *Registering the application*
If Email_address is not empty **and** Email_address does not exist **then**
If Username is not empty **and** Password is not empty **and** Password_confirmation is not empty **and** Date_of_birth is not empty **and** City is not empty and Address is not empty **then**
If Password is equal to Password_confirmation then
print "Successfully registered."
Else
print "Please make sure that passwords are the same!"
Endif
Else
print "Please fill required fields!"
Endif
Else
print "Entered Email address is already registered!"
Endif
End

- **List Item:** System will display all items in the given category name.

  Data: *Category_name*
  Result: *Listing items depending on the selected category*
  for each item **in** Category_name **do**
      display items

- **Display Item Detail:** User will provide an item id by clicking on an item on the list item page. By using this id, the system gets the item details and displays them on the detail page.

  Data: *Item_id*
  Result: *Displaying the selected item's information in detail*
  If Category **in** Consumable **then**
      display item_name, category, subcategory, photo, expiration_date
  Else if category **in** reusable **then**
      display item_name, category, subcategory, photo, status
  Endif

5. Discussions

The discussions mention the validity and applicability of the SDD document, its details, and its characteristics. Moreover, SDD'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 the building of the system. The limitations can be categorized into several groups. The awareness of the categories minimizes their effect of them on the project. Scheduling is one of the most significant limitations that the team must follow. The strict deadlines in the software process model and documentation process should be considered. The constraint can be explained with the calculation of time and budget information on the SDD document according to the deadlines of the project. Therefore, the team may need to work overtime to accomplish the system design specified in the SDD document.

2. Applicability of the project under real life situations

Under real-life situations, the SDD document will need to describe the design of the system fully enough to allow software development considering what is to be built and how it is expected to be built. Since this is a student project, both system design expectations and its application are under the control of the team. Thus, this document will become less effective for this project compared to the real-life documents.

3. Health and Safety Issues

While preparing this document team members can be stressed due to the strict schedule and heavy workload. This excessive stress might cause several effects on team members' psychology. 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. Moreover, since the development is conducted during the pandemic and team members gather outside, there is a high risk of infection for team members.

4. Legal Issues

There are no legal issues considering the SDD. Since this is a student project, team members do not need to follow the Turkish Social Security Administration's working regulations in terms of time and budget.

5. Economic Issues and Constraints

During writing the SDD document, the team members must stay late for many days, so some of them had to turn home by taxi. Also, since the document is detailed and takes a long time to complete, team members gather more frequently, and it affects the gasoline expenses of those who have cars. Moreover, 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 the school is determined as a meeting point to make SDD document meetings regularly. Also, since it is a school project and there is no decision-maker other than team members, design descriptions are determined according to the team itself.

7. Producibility-Manufacturability

Left-Over! does not serve a tangible product directly, it offers software. Hence, there will not be continuous production. Therefore, software design is determined in a way that once the product is released according to system requirements specified before, it will not need any maintenance and improvements. Hence, the SDD document will not be touched. Furthermore, the software design was set at the last part of the documentation process, so what to do and how to do will not be changed further.

8. Social, Political and Ethical Issues

When preparing this document and the process of deciding software design description, the team distributed the work equally among them to prevent a feeling of injustice about the work done by them. Moreover, during the SDD documentation process, the right to freedom of thought and expression, which is one of the main issues of fundamental human rights, is reserved. All members expressed their opinions freely without exposing any discrimination.

9. Multidisciplinary Collaboration

There is no multidisciplinary collaboration in software design since all members of the team are from the Information Systems and Technologies department including the developer team and team advisor. Thus, no advice or consultancy is taken from other disciplines.

10. Environmental Issues

Google Documents is used as an online documentation tool and draw.io is used for online diagraming to prevent paper waste during the software design description process. However, all team members use their personal computers separately for documentation, which causes more electricity consumption. Thus, the process planned to finish the project with minimum latency to keep electricity consumption at the minimum.