ChadList

NoBrainNoPain

Group Members:

Tomás Gres (315185),

Dragos-Daniel Bonaparte (315261),

Dan-Sebastian Ceapa (315162),

Chiril Luncasu (315171),

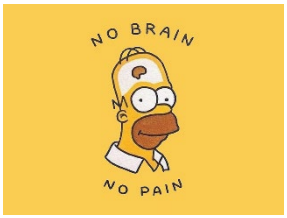
Matas Armonaitis (315263)

Supervisors:

Jakob Knop Rasmussen (JKNR)

Joseph Chukwudi Okika (JOOK)

VIA University College



54270 Characters

Software Engineering

3rd Semester

17 / 12 / 2022

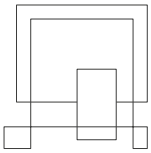


Table of content

Abstract 4

1. Introduction 6

2. Analysis 7

2.1 Functional Requirements 7

2.1.1 Critical Priority 7

2.1.2 High Priority 8

2.1.3 Medium Priority 8

2.1.4 Low Priority 9

2.1.5 Non-Functional Requirements 9

2.2 Use Case Diagram 9

2.3 Use Case – Fully Dressed Description (Manage Posts) 11

2.4 Activity Diagram (Manage Posts) 13

2.5 System Sequence Diagram 14

2.6 Domain Model 15

2.7 Security 16

2.7.1 Threat model Click or tap here to enter text.(Anon., 2022v) 16

2.7.2 Objectives 17

2.7.3 Threat prevention 17

2.7.4 Risks 19

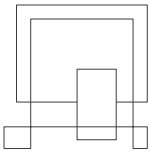
3. Design 20

3.1 System Architecture 20

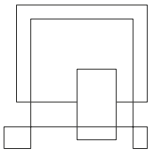
3.2 Technologies 20

3.2.1 Blazor Web Assembly 20

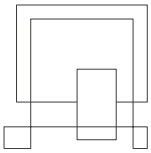
3.2.2 HTTP/REST 21



3.2.3 gRPC 21
3.2.4 Spring Boot Framework 22
3.2.5 JPA/Hibernate 22
3.2.6 PostgreSQL 23
3.3 Database 23
3.3.1 Entity Relations Diagram 23
3.4 Design Patterns 24
3.4.1 Adapter Pattern 24
3.4.2 Object Relational Mapping 24
3.4.3 Database Access Objects 25
3.4.4 Data Transfer Objects 26
3.5 Solid Principles 26
3.5.1 Single Responsibility Principle 27
3.5.2 Dependency Inversion Principle 28
3.6 Class Diagram 29
3.6.1 Presentation Tier: 29
3.6.2 Business Logic Tier: 30
3.6.3 Data Access Tier: 31
3.7 Sequence Diagram 32
3.8 Security Mechanisms 33
3.8.1 Authentication 33
3.8.2 Password Hashing 33
3.9 Graphical User Interface Design 34
3.9.1 Main View 34
3.9.2 Registration View 34

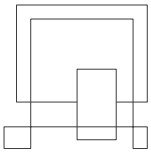


3.9.3 Marketplace View 35
3.9.4 Message view 36
4. Implementation 37
4.1 Data Access Tier 37
4.1.1 Implementation in Java 37
4.1.2 Code snippets 38
4.2 Business Logic Tier 42
4.2.1 Implementation in C# 42
4.2.2 Code Snippets 42
4.3 Presentation Tier 46
4.3.1 Implementation in C# Blazor 46
4.3.2 Code Snippets 46
5. Test 52
5.1 Test Cases 52
5.2 Business logic tier/Data Access tier tests 55
5.3 Unit tests 55
6. Results and Discussion 57
7. Conclusions 58
8. Project future 59
9. Sources of information 60
10. Appendices ..... **Error! Bookmark not defined.**



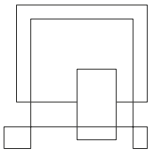
Abstract

A small startup business called ChadList entered the market with an ambitious vision towards the 12th United Nations Goals (Anon., 2022n), to be precise, Responsible consumption and production. The startup offered research in terms of the waste generated worldwide and got to an alarming conclusion that there is about 53.6Mt of e-waste in 2019 alone (Anon., 2022m), around 17Mt of textile waste only in 2018 (Anon., 2022e) and in 2017 the furniture waste generated by Americans was 12.2Mt (Anon., 2022l). Their approach to the problem was to offer people an easy and efficient way to sell or offer their unused items to other people that can make use of them. The startup's ideas and concerns were thereafter discussed with the Product Owner and the stakeholders to agree on a common and realistic vision of how the service will function. After which 22 functional requirements were selected carefully.
From the requirements given, the system will be composed of a simple set of actors, authenticated users, and unauthenticated users. Those requirements represent the main functionalities that can be performed by the aforementioned actors: creating a user which after will require the user to log in, once logged in, a user can create an item post and add images to it, and the marketplace can be accessed by either of the actors, an authenticated user is required to start a message with the seller, while the unauthorized user must rely on the phone number or email address, a user can edit his/her account, a user can edit an item and or change its status to sold or available.
The distributed system is composed of a 3-tier architecture which will offer faster development, improved scalability, improved reliability, and improved security making it clear, secure, and easy to maintain (Anon., 2022f). This layered architecture will allow the service to function everywhere regardless of location, making it more accessible to the end user. The system is presented as follows: The first tier, also known as the presentation tier is represented by a C# implementation of Blazor representing the user interface and it will be the tier that the end-user will interact directly with. The second tier will be the Business Logic Tier representing our middleware tier implemented in C# that will make sure that the third tier will be able to communicate back and forth with the first tier, but they must not communicate directly with each other creating a security flaw in the system. The third tier also known as the Data Access Tier is written in Java



and will serve as the persistence which is connected to a database created in PostgreSQL. The first and second tiers communicate through Representational State Transfer also known as REST, while the second and the third tier will communicate through Remote Procedure Call open sourced by Google also known as gRPC.

To assess the quality of the product, the team used both Black boxing and White boxing testing. The tests written conclude that the product proves to meet the requirements and the main ideas of the stakeholders and the Product Owner.

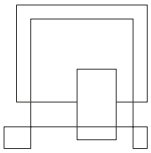


1. Introduction

Our customer is a non-profit organization that focuses on waste management. It has decided to enroll in the competition between recycling and wasting, hoping that its approach will motivate people to waste less and make use of items for longer. We were contacted by the company with the offer to develop a system that would help its users to sell and exchange their belongings that they no longer need.
The issue with most secondhand websites like eBay and Craigslist is that it is not available worldwide, and in consequence, the end users who would benefit from the service are excluded(Anon., 2022q). Another aspect that the startup wanted to address is the need for an authenticated user when buying an item. While Facebook Marketplace addresses the problem with worldwide coverage, it lacks anonymity for its users, requiring the end user to have an account before using the service which is something inconvenient for the user, especially if the user uses the service to only buy items.

The business owner decided to take on the opportunity and combine both worlds, where there is access to the service worldwide and anonymity for the end user that only uses the website for purchases. Having all those requirements laid down, the Product Owner, together with the stakeholders decided to include the following delimitations for the system:

- No map for the area where the product is located
- No shipping support whatsoever (Anon., 2022v)
- No monetary transactions through the website (Anon., 2022s)
- Users require an account only for selling items
- No complex recognition of illegal items (Anon., 2022t, Anon., 2022k)
- The website will limit its domain only to “.com” (The change of region will be done on the website)



-

To accommodate the requirements, the project decided to adhere to the Unified Process methodology. Having the Agile manifesto in mind, the project went sequentially through 4 phases:

I. Inception
II. Elaboration
III. Construction
IV. Transition

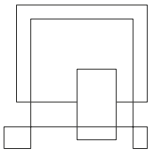
2. Analysis

The purpose of ChadList is to reduce wasting of reusable items throughout the world by giving everyone access to a platform where they can sell and buy those items quickly and reliably. The Product Owner put in a lot of effort in understanding the client as much as possible.
The system will have two types of users: unauthorized users and authorized users. Authorized users will be able to sell and buy items while unauthorized users will only be able to buy items.

2.1 Functional Requirements

2.1.1 Critical Priority

1. As a user I want to register, by specifying my first name, last name, phone number, email, date of birth and gender and log in specifying the email and password I used for the registration, in order to access my account.
2. As a user I want to be able to post an item for sale specifying its name, description of the item, price, currency and status of the item, in order to sell an item I don't need anymore.
3. As a user I want to message the user over an item they are selling in order to negotiate and ask details about the item.
4. As a user I want to be able to search for a specific item specifying the part of name, part of the description and min and max price in order to find an item that I am looking for.



5. As a user I want to be able to view details of a specific item in order to decide about the purchase.
6. As a user I want to be able to see all my active and previous chats with the sellers, to be able to verify information between what the seller said and what the item is about.
7. As a user I want to be able to report an item in case the item does not respect the rule about forbidden items.

2.1.2 High Priority

8. As a user I want to be able to edit my profile in order to modify any information that became outdated.
9. As a user I want to be able to delete or edit an item, in case I did any mistake during posting or I decided not to sell the item, or I sold the item.
10. As a user I want to be able to see all my notifications in order to keep track of messages and item updates.

2.1.3 Medium Priority

11. As a user I want to be able to have a list of items that might interest me in order for me to find better items.
12. As a user I want to be able to add an item to my wish list in order to save items that I am not sure yet that I want.
13. As a user I want to get a notification whenever there is an update on an item in my wish list in order to be up to date about my saved items.
14. As a user I want to be able to see the sellers rating in order to see if he is a trustable person.
15. As a user I want to be able to comment/rate a seller to help other users trust the seller or not.
16. As a user I want to be able to see a seller's profile in order to see any details that interest me such as address, rating or the amount of sold products.
17. As a user I want to be able to delete items from my wish list, in order to remove items that do not interest me anymore.
18. As a user I want to be able to change the password so that I don't get locked out of my account.
19. As a user I want to be able to delete my account in case I do not want to use the service anymore.

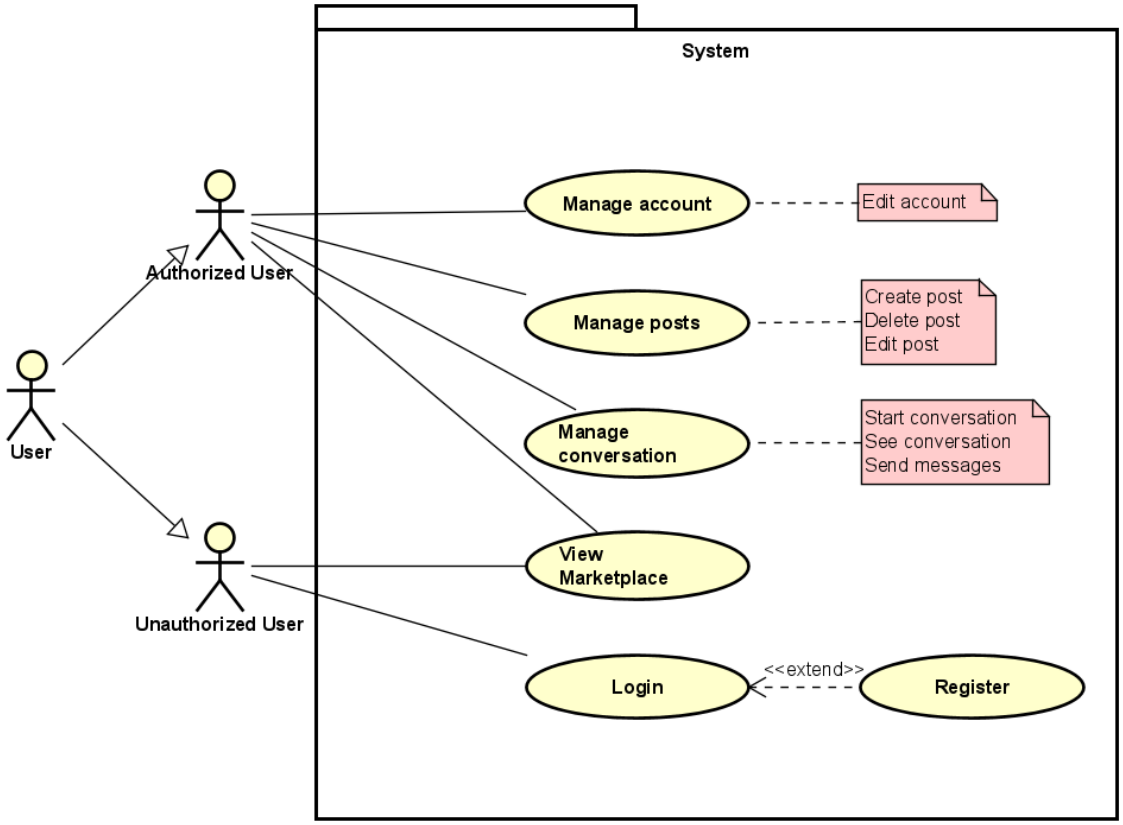
2.1.4 Low Priority

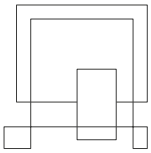
20. As a user I want to be able to sort the items by categories, in order to find what interests me.
21. As a user I want to be able to track how many people added a specific item to their wish list for me to take a faster decision about the item.
22. As a user I want to be able to see related items when I look at a specific post, in order to have a better chance of getting a better deal.

2.1.5 Non-Functional Requirements

1. The system must be able to work on Mozilla Firefox v106.0 and previous, Google Chrome v107.0 and previous and Microsoft Edge 107.0 and previous

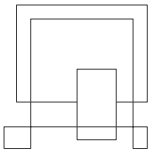
2.2 Use Case Diagram





Use case explanations:

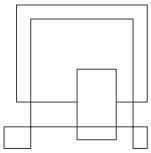
- Manage account
    - The user can choose to edit his/her profile at any time by pressing the person icon on the right top corner
- Manage posts
    - An authenticated user can create a post
    - An authenticated user can edit a post
    - An authenticated user can delete a post
- Manage conversation
    - An authenticated user can create a conversation with the seller
    - An authenticated user can see a conversation with a specific seller
    - An authenticated user can send messages in a conversation with the seller
- View Marketplace
    - Any user can view the items and press on them for a detailed view
    - An authenticated user can start a conversation with the seller from the detailed view
- Login
    - An unauthorized user can log into the system to gain access to more features
- Register
    - An unauthorized user can register into the system



2.3 Use Case – Fully Dressed Description (Manage Posts)

The following use case will represent the main page of the website whereas the user will spend most of the time. The use case shows the abilities of a user with an account.

Use case	Manage posts
Summary	The user can create/edit/delete a post.
Actor	User
Precondition	The user must be logged into his account.
Postcondition	The details and information for the post will be saved into a database.
Basic Scenario	Create Posts 1. The actor logins to his account. 2. The actor presses the marketplace button. 3. A window will be shown with all items that are for sale and their description. 4. If the actor wants to sell an item, he needs to press the create item post button. 5. A window will be shown, where the actor can add all the necessary information about the item. Edit Posts 1. The actor presses my item button. 2. The actor chooses the item, and presses the edit item button



	3. A window will be shown with the item where the actor can change the details about the post. 4. To save the changes, the actor should press the submit button below, and the changes will be saved. Delete Posts 1. Users can delete the post by pressing the delete item button. 2. The item will be deleted from the marketplace.
Alternative Scenario	
Exception	If one of the fields or more are empty, an error will be shown.

Table 1 – Manage posts – Fully dressed description

2.4 Activity Diagram (Manage Posts)

The following diagram represents the way the system manages posts. The diagram covers both main scenarios and alternate scenarios.

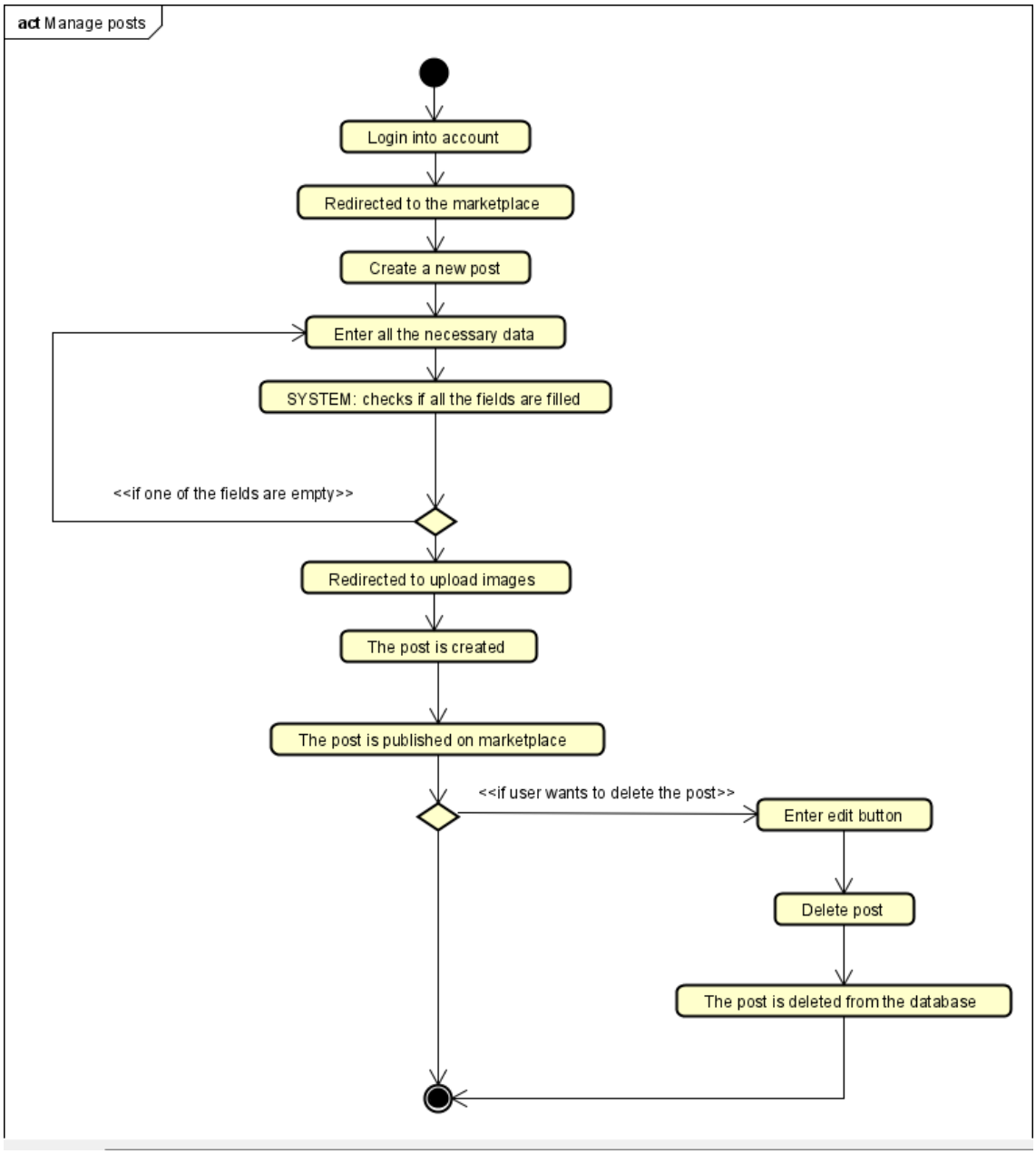


Figure 2 – Activity diagram – Manage posts

2.5 System Sequence Diagram

A system sequence diagram is used to represent how the system will respond to the user's input and every step that the system will perform towards completing its task. The following sequence diagram represents the steps that the user and the system must take in order to create a post.

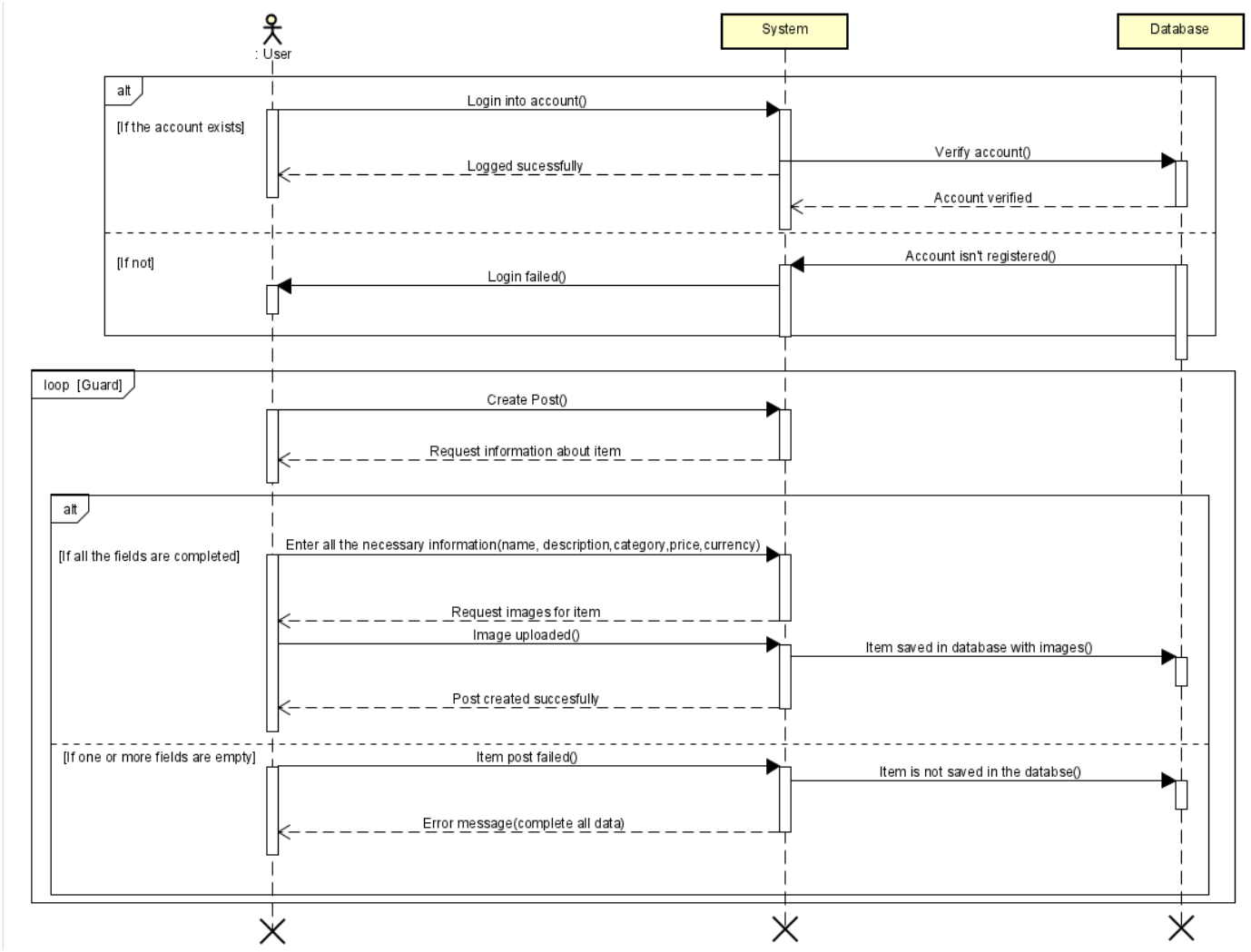


Figure 3 – System Sequence Diagram – Manage posts

2.6 Domain Model

The diagram shown below represents the relationships between the actor and the components. A **User**, after registering and creating an account, can decide to sell their **Item** (things they own and do not need anymore). If they wish to buy that item, they can start a **Conversation** with them to negotiate the price and delivery by sending a **Message**. After a message is sent, the seller will get a **Notification**.
Another **User**, decides to enter the marketplace to search for an item, the user not being authorized, can only see the item and will rely only on the **phone number** to purchase the item.

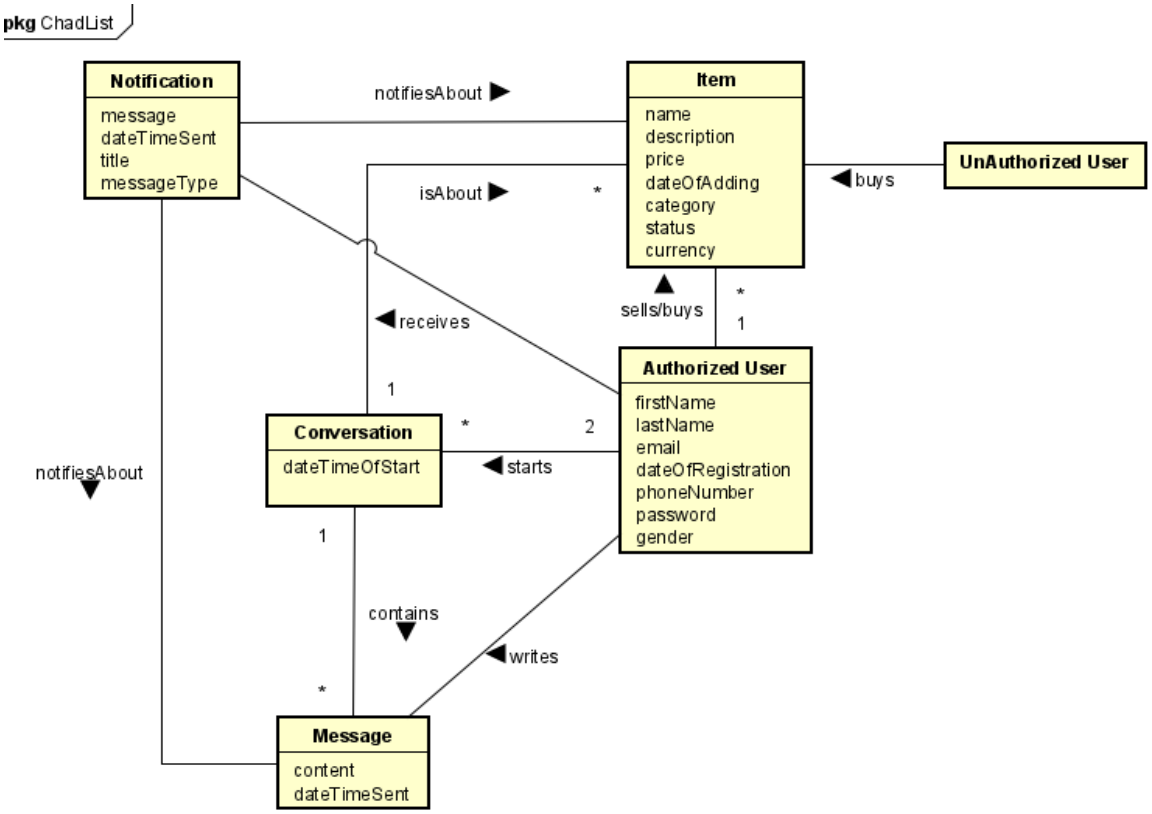
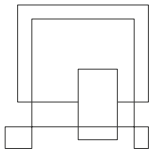


Figure 4 – Domain Model

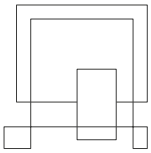


2.7 Security

2.7.1 **Threat model** Click or tap here to enter text.(Anon., 2022v)

Threat case	TPM	Objective violation	STRIDE	X.800	EINOO	Consequence
XSS (Cross-site scripting)(Wang and Zhang, n.d.)	Data not being validated	Confidentiality, Integrity, Authorization	Spoofing identity, Tampering with data, Information disclosure	Passive	External Network Online	Privacy is disclosed, User's data is stolen
MitM (Man in the middle)	Privacy and confidentiality breach	Integrity, Confidentiality, Authorization	Spoofing identity, tampering with data, information disclosure	Active and Passive	Insiders Network Online Offline	Privacy disclosed, The attacker can intercept information between them and the service.
Brute-force attack	User authentication, data authorization and validation is not handled	Authentication, Confidentiality	Repudiation, Spoofing identity, information disclosure	Active	External Network Online	Personal information disclosure, data modification, customers' identity stolen
DoS (Denial-of-service attack)	Host not having enough protection for such recurring attacks	Availability	Denial of service	Active	External Insiders Network Online	Server's unavailability to the users
The Heart Bleed Bug	The length of the message is not checked	Confidentiality, privacy	Tampering Information disclosure	Active: Modification	External Network Online	The system might return confidential data, such as the server's private decryption or signing key

Table 2 – Threat Model



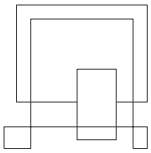
2.7.2 Objectives

The stakeholders requested that the website be secure. The Product owner communicated back to the team that confidentiality, and the integrity of the website should be of high standards.
First of all a website that does not provide confidentiality is not a trustable one, and in turn makes the website less probable to have any users.
Data confidentiality security mechanisms need to be implemented in such way such as unauthorized individuals will not be able to access it.
This will ensure that the system remains trustable, and the user will have its privacy when using the system.
As for integrity, the authorization was created in an ingenious way such as data integrity is assured to be changed only by authorized individual. Any sensitive information will not be edited without the user being authorized. The system integrity is maintained when even if a user accesses the URL of a page illegally the authorization will not allow access to anything on the page.
Lastly, the availability will be guaranteed by a strong and secure host. The system will have to be deployed on a host with good reviews and great security management for providing availability and prevent DoS attacks from occurring.

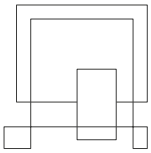
2.7.3 Threat prevention

The Product Owner presented the threats that the system will be prone to, to all the stakeholders following small changes in the requirements.
The prevention of the threats began with:

- **XSS prevention** (Anon., 2022u)**:**
    - Escape user input. Escaping means to convert the key characters in the data that a web page receives to prevent the data from being interpreted in any malicious way. It doesn't allow the special characters to be rendered.
    - Validate user input. Treat anything that originates data from outside the system as untrusted. Validate all the input data. Use an allowlist of known, acceptable, good input.



    - Sanitize data. Examine and remove unwanted data, such as HTML tags that are deemed to be unsafe. Keep the safe data and remove any unsafe characters from the data.
- **MITM prevention** (Anon., 2022r)**:**
    - Strong WEP/WAP Encryption on Access Points
    - Strong Router Login Credentials
    - Virtual Private Network
    - Force HTTPS
    - Public Key Pair Based Authentication
- **Brute-force prevention** (Anon., 2022ac)**:**
    - Longer passwords
    - Complex passwords
    - Limit login attempts
    - Using captcha
    - Two factor authentication
    - Cloudflare
- **DoS prevention** (Anon., 2022j)**:**
    - Good Cloud Mitigation Provider.
    - Strong Firewall
    - Internet Service Provider (ISP)
- **The Heart Bleed Bug prevention:**
    - Using a new and updated version of OpenSSL

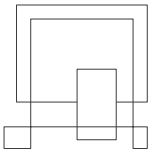


2.7.4 Risks

Threat mitigation is important, but the system must also take in consideration what is the risk of a threat and how much will it impact the service. This analysis will show which of those threats will have priority to be solved. By having a priority the system can be developed with the most important threat prevention, allowing the others to be addressed later in the lifetime of the system.

Threat case	Occurrence risk	Impact on the system	Priority
XSS (Cross-site scripting)(Wang and Zhang, n.d.)	Medium	Medium	Medium
MitM (Man in the middle)	High	High	High
Brute-force attack	Low	Low	Low
DoS (Denial-of-service attack)	Medium	High	Medium
The Heart Bleed Bug	Low	Medium	Low

Table 3 – Threat risk assessment



3. Design

3.1 System Architecture

In Figure 5, the system architecture and the communication protocols/middleware is shown visually. The presentation tier is implemented using Blazor Web Assembly (C#). From the presentation tier, the C# objects are converted to JSON objects and sent via HTTP protocol to the business tier. The business tier is implemented using .NET 6.0 (C#). The HTTP Controllers in the business tier handling the communication with the presentation tier follow Representational State Transfer (REST). The business logic then communicates with the persistence/database tier using Remote Procedure Calls (gRPC) and data sent through is serialized using Protocol Buffers. The database tier is implemented in Java and Spring Boot Framework. The database itself is implemented using PostgreSQL and Object Relational Mapping tool - Hibernate which is the implementation of the JPA specification.

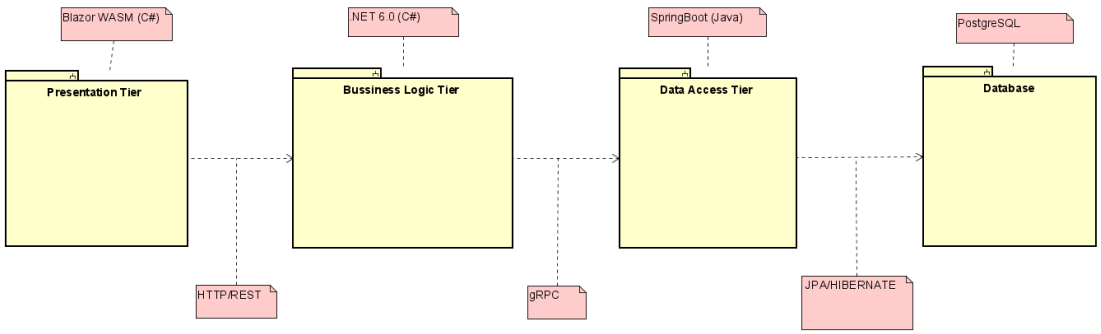
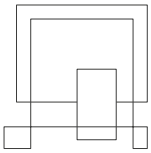


Figure 5 – Tier architecture

3.2 Technologies

3.2.1 Blazor Web Assembly

Blazor Web Assembly (Blazor WASM) was used for the design of the interactive user interface. The visual part was implemented using HTML, CSS, Bootstrap and the



functional part was written in C#. The graphical design was adjusted using Bootstrap library and CSS.

Authentication and authorization were implemented using this technology as well. If a user was successfully logged in, the system would grant him privileges to create new item posts, contact other users using our messaging system as well as editing their profile and the items they posted.

3.2.2 HTTP/REST

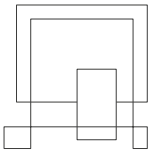
Hypertext Transfer Protocol and Representational State Transfer were used for communication between presentation and business tier. The HTTP protocol accesses resources identified by an endpoint and returns the relevant information about the requested resources.

On the presentation tier, the HTTP Requests: POST, GET, PUT, PATCH and DELETE were used. Each of the requests is sent to the business tier and there the http endpoint controllers (which were implemented following REST principles) handle the requests.

The POST requests are used to create new entities (for example new item post). GET requests are used to retrieve one or multiple entities from the endpoint. PUT and PATCH requests are used to update entities either by replacing the whole entity or just updating a part of the attributes of the entity respectively. The DELETE request is used to remove entities from the endpoint.

3.2.3 gRPC

Google Remote Procedure Call with Protocol Buffers are used between the business tier and the database tier. The gRPC uses Protocol Buffers by default as Interface Definition Language (IDL) for describing both the service interface and the structure of



the payload messages, which allows it to communicate between tiers implemented using different technologies.

GRPC provides four types of calls: Unary (where client sends single request and gets a single response), Server streaming RPCs (where client sends single request and gets a sequence of messages back), Client streaming RPCs (where client sends a stream of messages to the server, and after all messages are sent, the server sends a single response back to the client) and the Bidirectional streaming (where both sides send messages using a read-write stream, they can read and send messages independently and the message order in both streams is preserved)

In the system, the unary call was used for example for creating an item post, where a Create DTO was sent through, and a new Item message was sent back to the client. The Server streaming RPCs were for example used for retrieving all items the user posted. The last two methods were not used in our system. (Anon., 2022h)

3.2.4 Spring Boot Framework

Spring Boot was used in the database tier. Spring Boot is a Java Framework that is used to create micro services. It allows to create standalone Java applications that do not rely on an external server because it embeds a web server such as Tomcat or Netty. (Anon., 2022x)

3.2.5 JPA/Hibernate

Java Persistence API is Java specification focused on ORM (Object Relational Mapping). Hibernate is a Java Framework that implements JPA specifications. JPA represents how to define POJO (Plain Old Java Object) as an entity and manage it with relations using some meta configurations. They are defined either by annotations or by XML files. (Anon., 2022p)

3.2.6 PostgreSQL

PostgreSQL is a relational database system which supports both SQL (relational) and JSON (non-representational) object representations.
PostgreSQL database is created using JPA and Hibernate and used to store entities like users, items, messages etc. In the system. (Anon., 2022ad)

3.3 Database

3.3.1 Entity Relations Diagram

As a starting point of the database design, an entity relations model was created. The ER diagram follows the Domain model to a great extent, with some changes. For the user, a decision was made to use a special id attribute instead of using the email or phone number for simplicity.

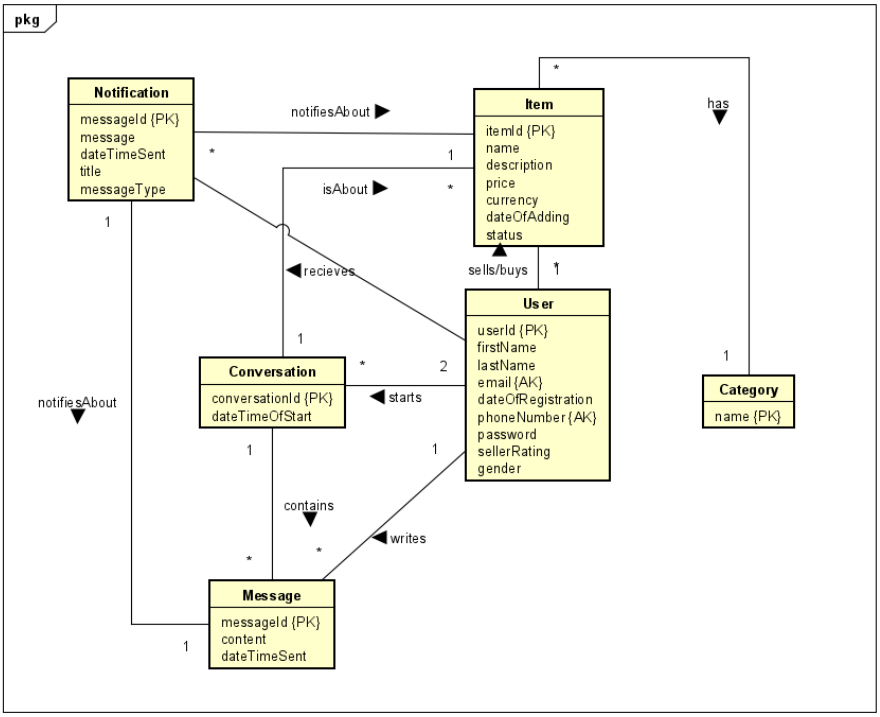
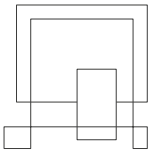


Figure 6 – Entity Relations Diagram



3.4 Design Patterns

3.4.1 Adapter Pattern

Adapter Patterns are used in the system for the JPA Repositories to be used by GRPC Services, in this example, the save method is mapped to createItem and updateItem methods, the findAll method is mapped to getItems method, findById is mapped to getItemById and deleteById is mapped to deleteItemById.

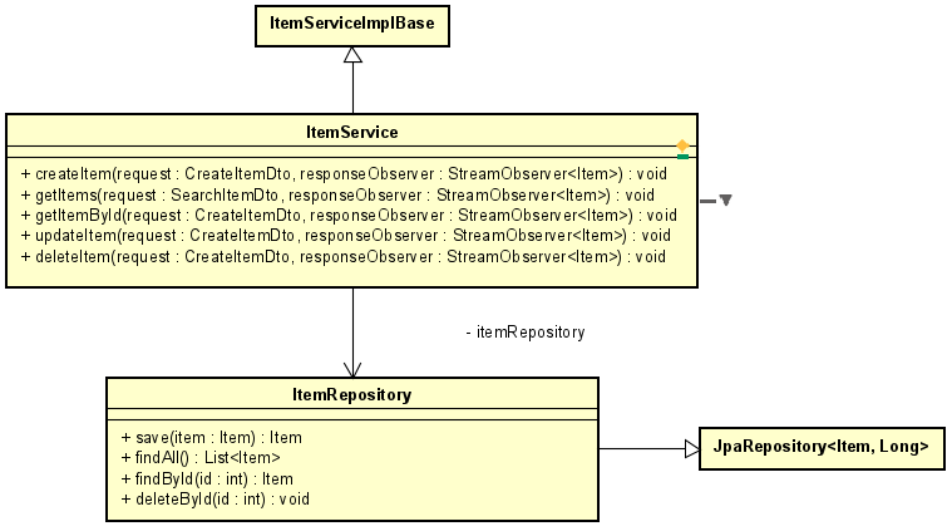
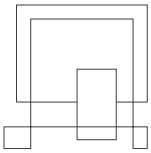


Figure 7 – Adapter Pattern

3.4.2 Object Relational Mapping

Object Relational Mapping is a design pattern that simplifies converting OOP objects to database entities and vice versa. In this system JPA and Hibernate are used and follow this design pattern.



3.4.3 Database Access Objects

DAO pattern is used to encapsulate the database tier access in a series of interfaces and thus allowing for the database tier to be replaced without having to extensively change methods and classes in the business tier.

Database Access Object contain all the basic CRUD methods (create, read/get, update and delete). Each database entity (user, item, notification, conversation etc.) has its own DAO.

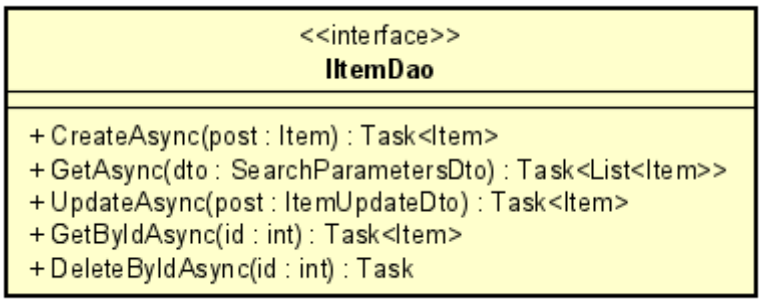


Figure 8 – Data Access Object

3.4.4 Data Transfer Objects

DTOs or Data Transfer Objects are objects that carry data between processes in order to reduce the number of methods calls.

In the system, the DTOs are used to specify necessary data for creating, updating or searching an entity, but leaving out information that is unnecessary.

For example, as the item's Id attribute is generated by the database, it is not necessary to specify it when creating the entity. Another unnecessary field is dateTimeCreated, as it will take the current time when it is saved to the database. The isSold attribute is also unnecessary, as in the beginning the item is not sold, so it is always false in the beginning. (Anon., 2022y)

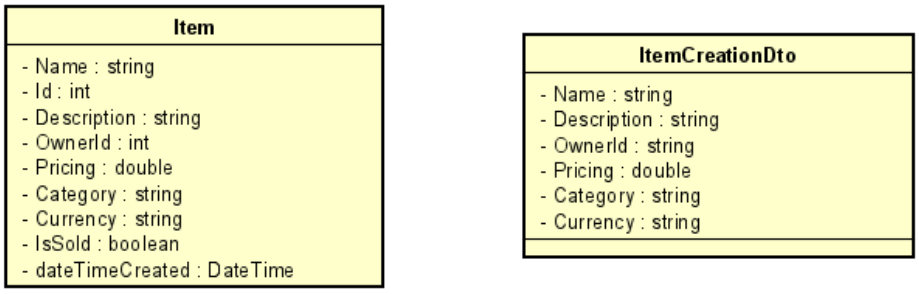


Figure 9 – Data Transfer Object

3.5 Solid Principles

The system was designed and implemented with great focus on the SOLID design principles, however, not all of the principles were useful for this specific project, so not all of them were used. (Anon., 2022w)

3.5.1 Single Responsibility Principle

In the business logic, there are controller classes, which are responsible for retrieving and posting data from and to the presentation tier. The logic classes are then responsible for validating the data sent from the presentation layer. Lastly there are Dao classes which are responsible for handling transfer of data to and from database tier. Each of those classes have only one responsibility.

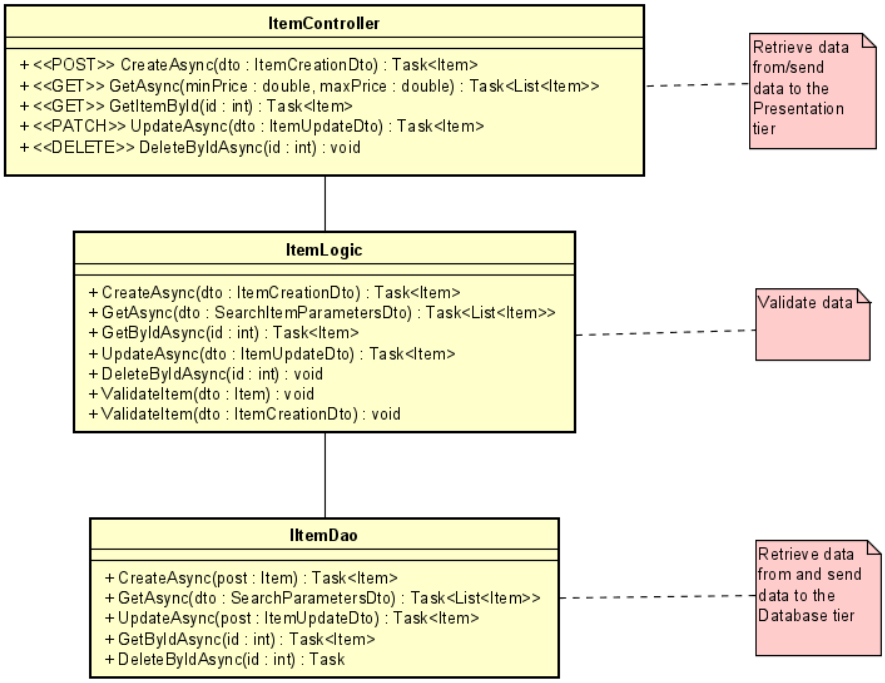


Figure 10 – Single responsibility principle

3.5.2 Dependency Inversion Principle

The Dependency inversion principle states that Entities must depend on abstractions, not on concretions. It states that the high-level module must not depend on the low-level module, but they should depend on abstractions.

In the system, this is used for example in the logic and DAO classes. The controllers rely on logic interfaces, instead of the implementation of the logic itself. Same for the logic classes, which rely on DAO interfaces instead of the DAO classes themselves.

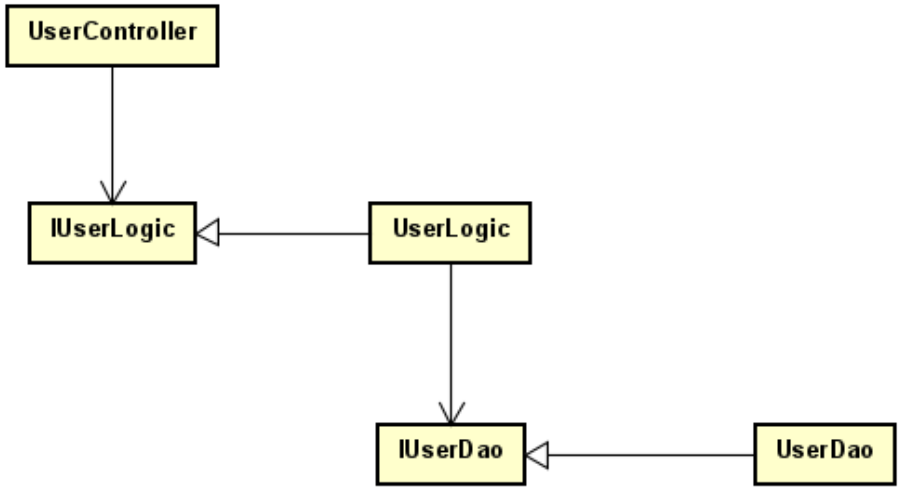


Figure 11 – Dependency Inversion Principle

3.6 Class Diagram

For a full resolution versions check in the appendixes

3.6.1 Presentation Tier:

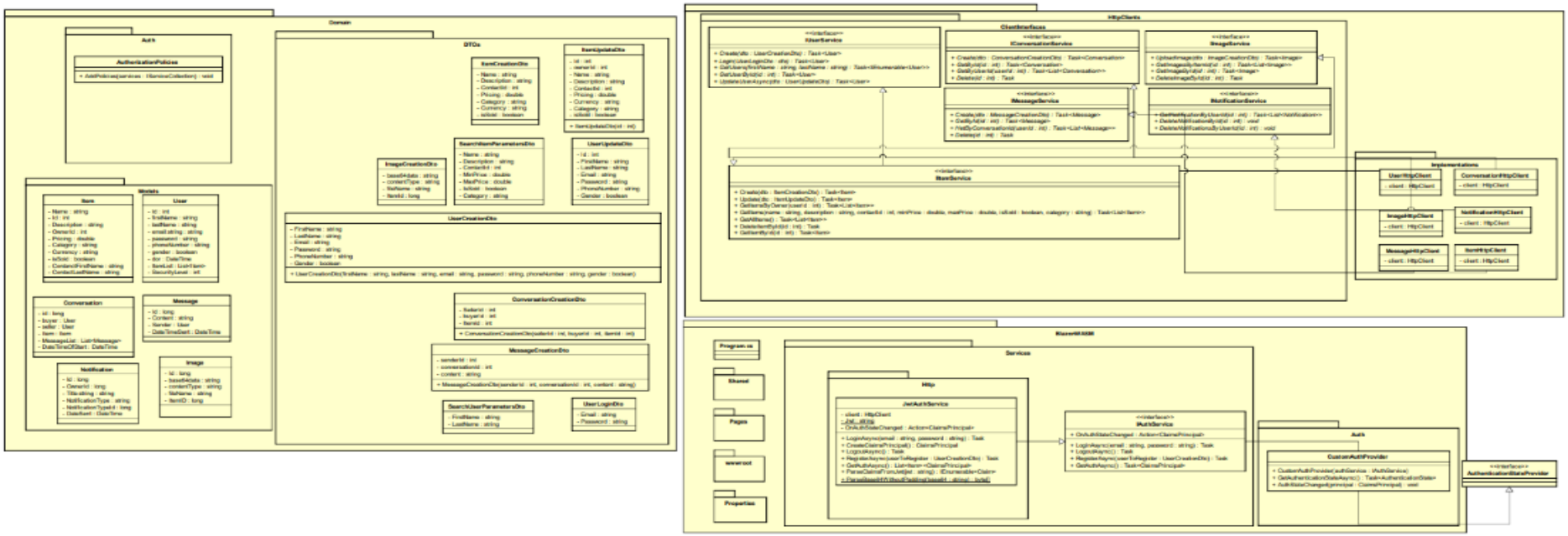


Figure 12 – Presentation Tier diagram

3.6.2 Business Logic Tier:

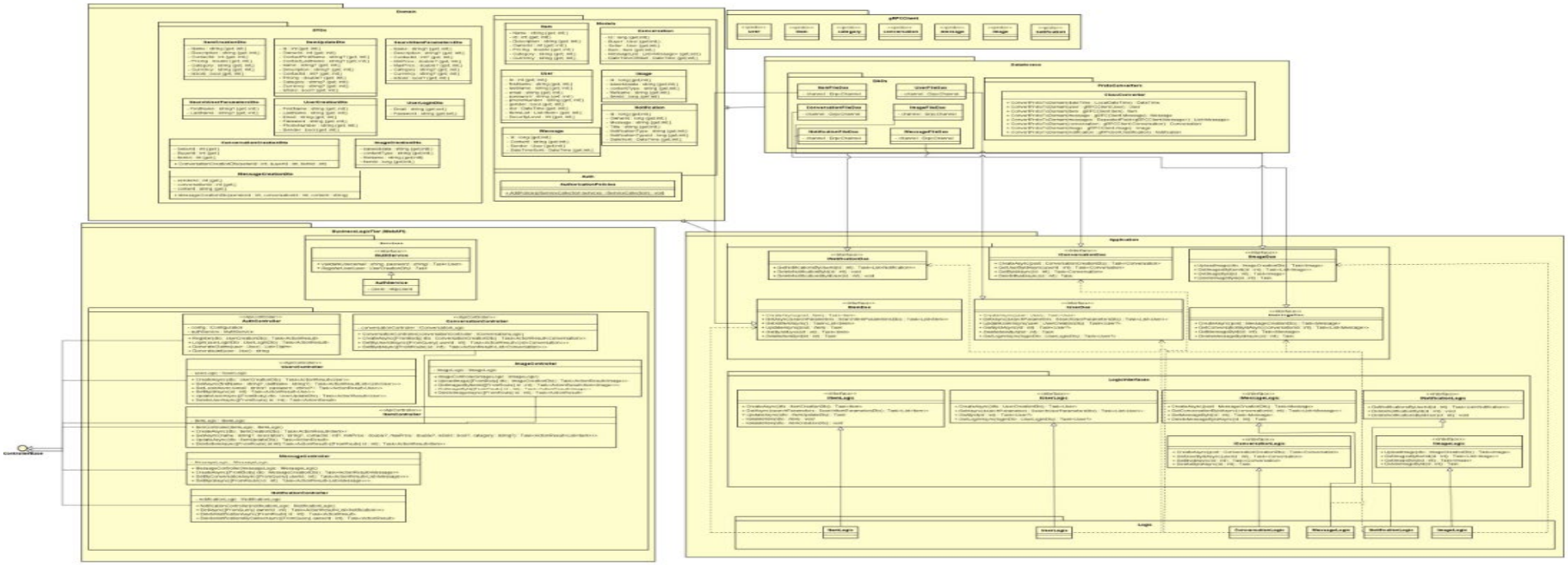
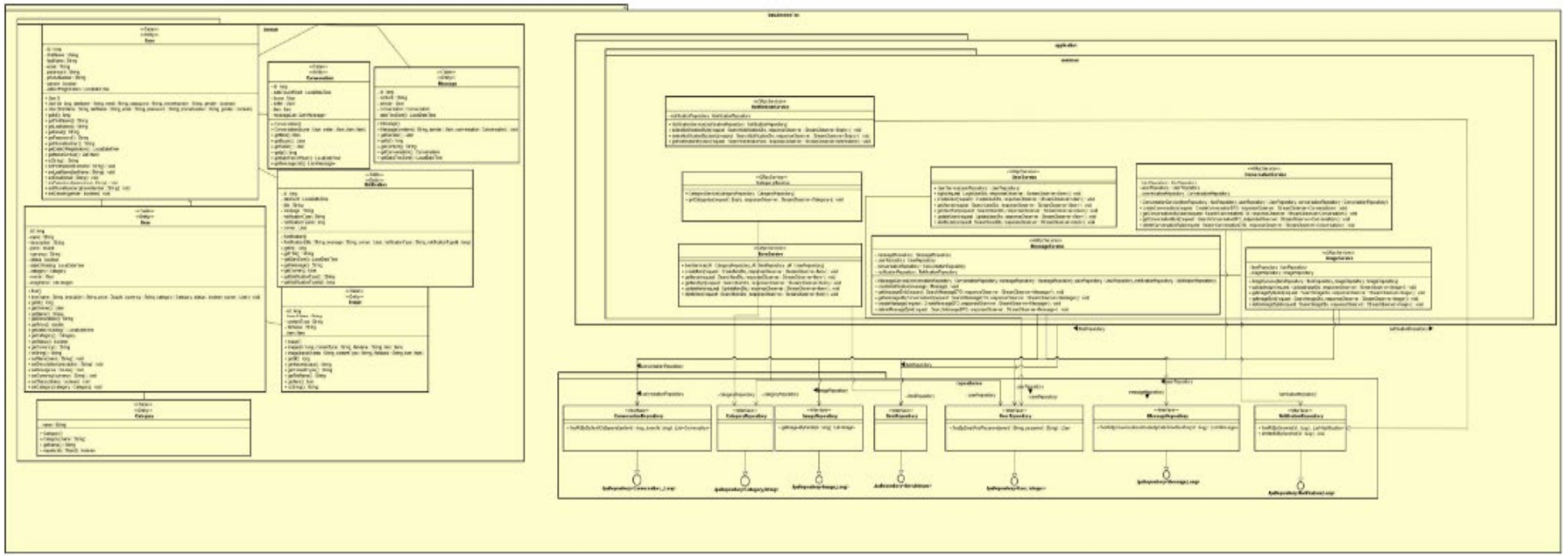


Figure 13 – Business Logic Tier diagram

3.6.3 Data Access Tier:



For the full Class Diagram in better quality, refer to Appendix L – Class Diagram

Figure 14 – Data Access Tier diagram

3.7 Sequence Diagram

The following sequence diagram shows the parts of the system that interact with each other in order to create a new item. When a user creates a new item by entering the information about it, that information is then sent through the view to the ItemHttpClient, where it is sent throguh HTTP protocol to the business tier. There the information is validated and afterwards sent to the database tier via gRPC. On the database tier it is sent to the ItemRepository and saved to the database.

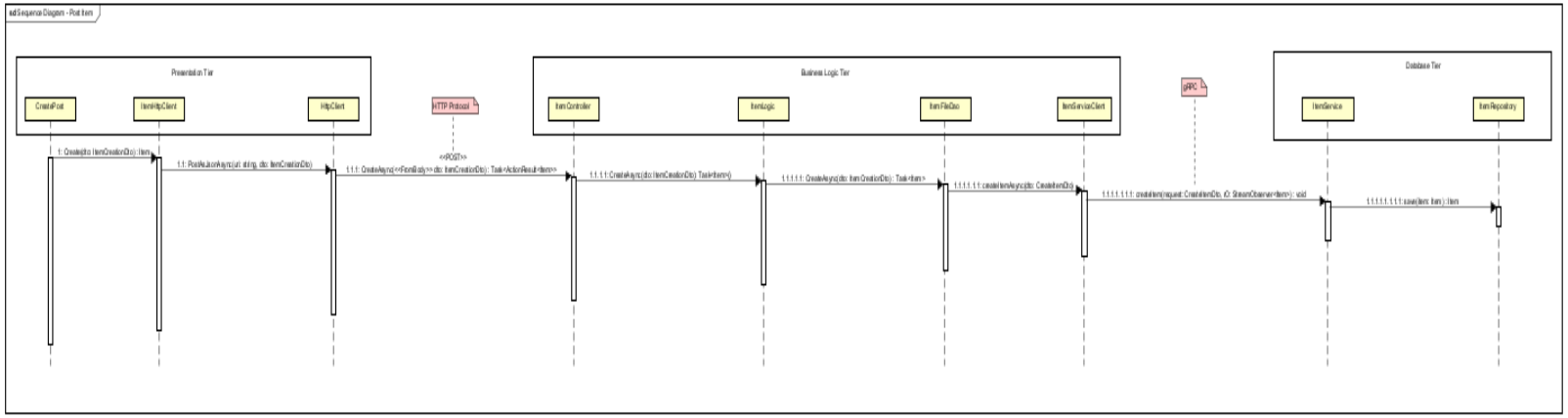
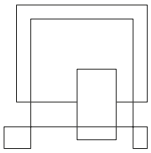


Figure 15 – Sequence Diagram

For a better-quality image see the Appendix J – Sequence Diagram



3.8 Security Mechanisms

3.8.1 Authentication

Authentications in systems is very important to differentiate between specific users as well as making sure that each user can only access their own account. There are three main types of authentication factors: something you know, something you have and something you are. Something you know is for example a password (it is something no one else except for you knows), Something you have is for example your smartphone and something you can be for example biometrics (face, fingerprints, voice). During Authentication process a user provides one of these factors but more usually a combination of those factors. For example, to access your bank account you need your smartphone (something you have), your face/fingerprint (to unlock your phone) and a service code or a PIN (to access the bank app itself).

3.8.2 Password Hashing

Password hashing is a way for the user to log into their account without their password ever being stored in the database. Instead of that, the password is changed using multiple mathematical functions which result in a string of a specific length of 64 characters (SHA256). The hashing algorithm is made in a way that whenever you enter a certain password, it will always result in the same hash. Therefore, when the user enters the password to log in, the password is hashed and then compared to the hash in the database. Another important ability of the hashing algorithm is that there is no way to get the password back from the hash. However, there are websites like crackstation.net which store hashes to many different passwords and are therefore able to crack the hash anyways. It is still important for the user to choose a strong password. (Anon., 2022i)

3.9 Graphical User Interface Design

The Graphical User Interface is in a form of web application made with Blazor Web Assembly which contains features for both authenticated and unauthenticated users. The view itself is made using HTML and designed using Bootstrap and CSS. The aim for the design was to be simple and easy to learn for new users.

3.9.1 Main View

The main page contains a navigation bar which enables the user to enter the marketplace and about us page and messages and a categories list as well as a search bar to look for items. After the user logs in, they can see their messages and their items listed for sale as well as their wish list.

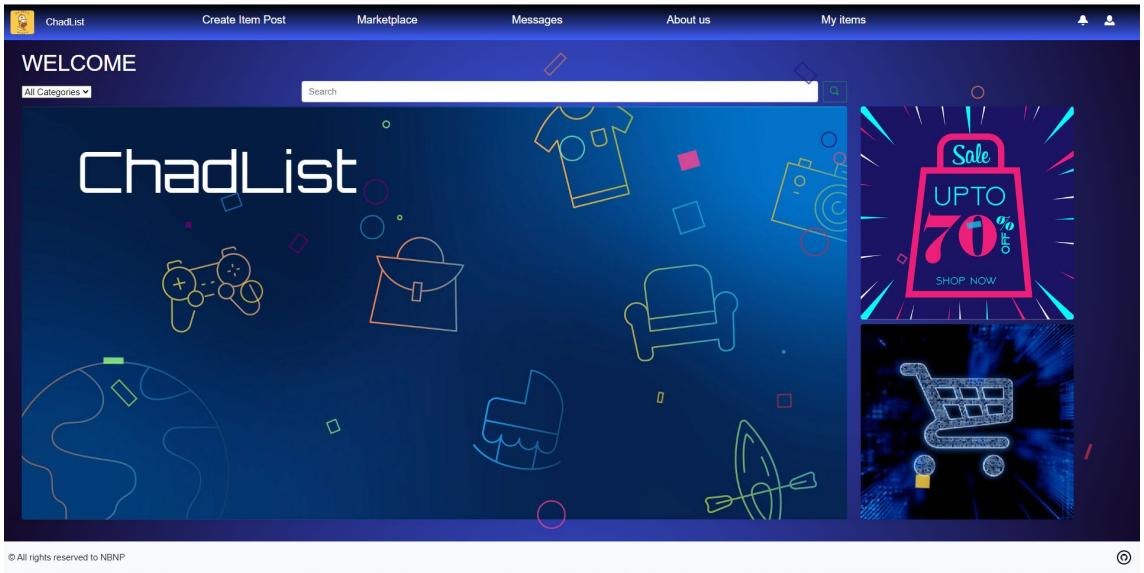


Figure 16 – Main page view (Anon., 2022a)

3.9.2 Registration View

Registration view is used for the users to register a new account. They enter their credentials and information about themselves for other users to recognize them.

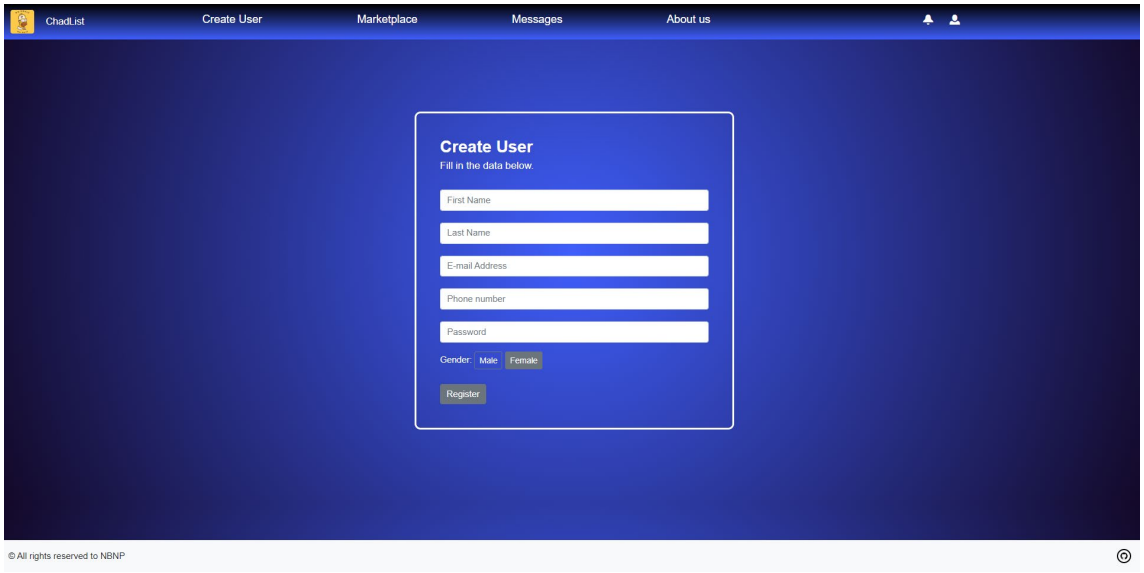


Figure 17 -Registration view

3.9.3 Marketplace View

In the Marketplace view the user can see all the items listed for sale, as well as to filter between them and selecting a specific category or a price range. Every item shows a main picture, a price, name and description.

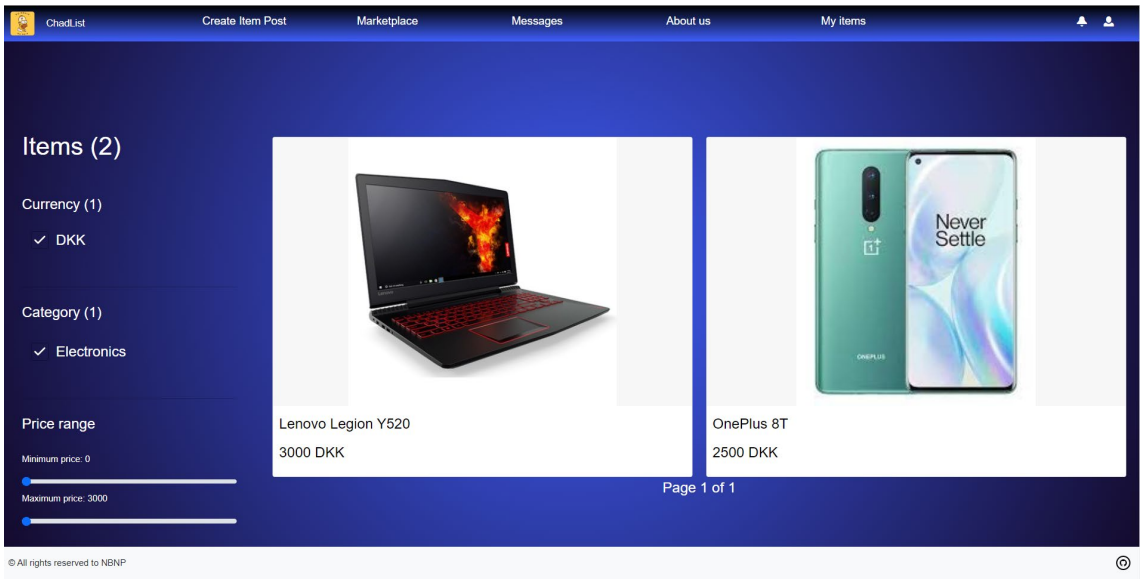


Figure 18 – Marketplace (Anon., 2022b)

3.9.4 Message view

In the messages view, the user can see all their conversations with other users. They are also able to read and send messages to other users.

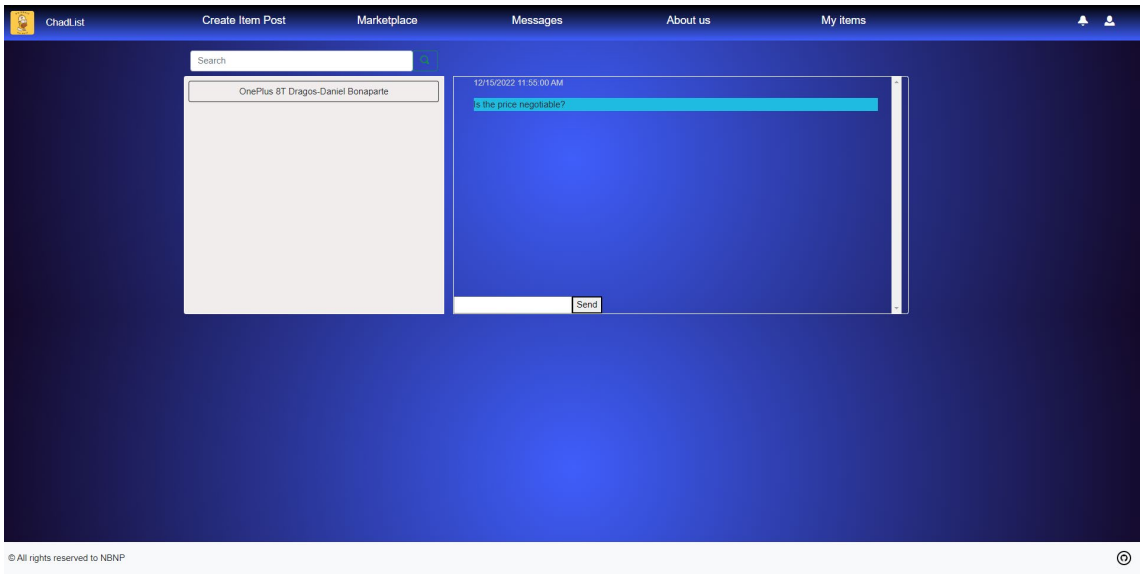


Figure 19 – Buyer messages view

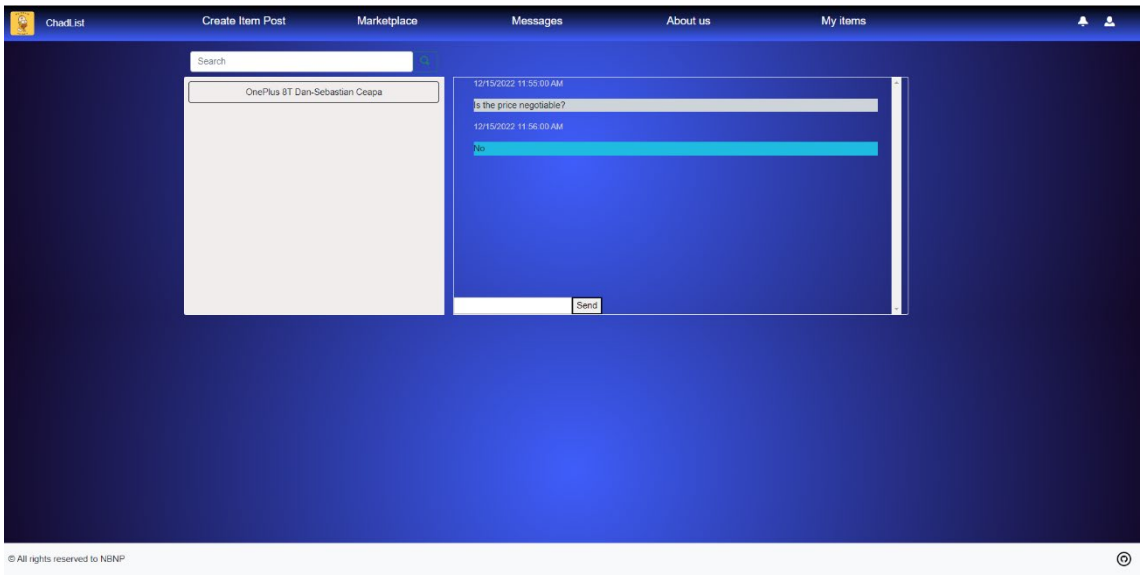
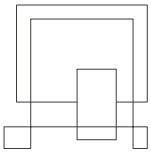


Figure 20 – Seller messages view

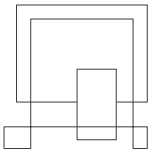


4. Implementation

4.1 Data Access Tier

4.1.1 Implementation in Java

After elaborating certain factors, the service's 3rd tier was chosen to be implemented in Java. It is responsible for handling and accessing the database and handle the gRPC requests.
The application is a Springboot implementation with dependencies to spring framework, Google's protobuf and Junit.
Spring framework provided the application with the necessary annotations, features and most importantly JPA/Hibernate, which is going to be the main way of communicating with the Database.
Google's protobuf is used to add in the project the .proto file type and provide the system with instructions on how to generate the protobuf.
Junit provide the necessary annotations for conducting tests on the code.
As the choice for PostgreSQL is only motivated by the fact that the developers worked with the technology previously and had a good understanding of it's language features.
More information about gRPC, Springboot and JPA/Hibernate can be found at chapter 3.2



4.1.2 Code snippets

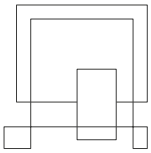
This chapter will contain explanations and images of actual code written in this tier.

Everything in the tier begins from declaring the domain classes as entities for the JPA's database generation feature

```
@Entity
@JsonPOJOBuilder
@Table(name="\"user\"")

public class User {
    2 usages
    @Id
    @SequenceGenerator(
            name="user_id _sequence",
            sequenceName = "user_id_sequence",
            allocationSize = 1
    )
    @GeneratedValue(
            strategy = GenerationType.SEQUENCE,
            generator = "user_id_sequence"
    )
    private Long id;

    4 usages
    private String firstName;
```

Figure 21 – Declaration of a domain class as entity

As seen above JPA/Hibernate requires the @Entity annotation for the class to be recognised as entity and be generated in the database. By using the annotation @Table, the programmer can choose to name the table, otherwise JPA/Hibernate will assign the name of the class as the name of the table. Even tough declaring the id is not necessary, by declaring the id manually using the @Id annotation and declaring the "id" as a sequenced value, the database will be able to ensure that every object inside the entity will have a unique id.



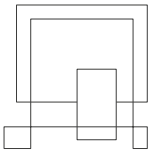
If the @Entity annotation is placed, JPA/Hibernate will take all the variables from the class and consider them as column in the future table. To exclude any variable, the @Transient annotation can be used.

After declaring all the domain classes as Entities, the system will require a way to communicate with the generated database. For this JPA/Hibernate comes with basic methods such as save(), delete(), get() and so on. The user also has the opportunity to declare his/her own methods.

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    2 usages  YoUnGi102
    Optional<User> findByEmailAndPassword(String email, String password);
}
```

Figure 22 – UserRepository interface

The annotation @Repository will indicate JPA/Hibernate that the following interface is going to extend the JpaRepository<User,Long> interface and use it to perform interactions with the database. The Repository is not required to contain anything else, but in some cases the user might declare other methods for ease of use.
The method “findByEmailAndPassword” is a user created method that will return the user that contains the matching pair of email and password.
In terms of JPA/Hibernate the process is to be repeated for all the domain classes.
As of now, the system can communicate with the database efficiently and easy, so now the implementation of gRPC begins.



Implementation of gRPC starts by creating the .proto files.

A proto file must specify the version of proto used and can include optional generation statements.

An important note is that the .proto file is going to be used in both server side (3rd tier) as well as in the client side (2nd tier).

In this file the programmer can define one or more services that will be used as methods later on in the code.

The “rpc” key word at the beginning of all the methods indicates that those methods are used for communication between the server and the client. The “returns” key word is used to define what will the method return after execution

All the parameters inside the methods and all the objects returned are declared inside the proto file as messages. In this example “LoginUserDTO” is defined as a message and contains two string variables that are going to serve as the email and password for the user.Important note about the defined messages is that the User used inside the roto file is not the same as the one inside the domain package despite the fact that they have the same variables.

```
syntax = "proto3";
option java_multiple_files = true;
option java_package = "dk.via.nbnp.databaseserver.protobuf";


service UserService {
  rpc login(LoginUserDTO) returns (User);
  rpc createUser(CreateUserDTO) returns (User);
  rpc getUsers(SearchUserDTO) returns (stream User);
  rpc getUserById(SearchUserDTO) returns (User);
  rpc updateUser(UpdateUserDTO) returns (User);
  rpc deleteUser(SearchUserDTO) returns (User);
}

message LoginUserDTO{
  string email = 1;
  string password = 2;
}
```

Figure 23 – User proto file – Java View

After the proto file has been written, the system will have to generate the protobuf. By using maven the protobuf can be generated easily and reliable, here all the messages and all the services declared in the proto file will be generated as java classes. Furthermore, the gRPC's server interface is now available for the developer to implement.

```
package dk.via.nbnp.databaseserver.application.services;

import ...

dragoselul +3 *
@GRpcService
public class UserService extends UserServiceGrpc.UserServiceImplBase {

    9 usages
    private final UserRepository userRepository;
```

Figure 24 – gRPC Service implementation

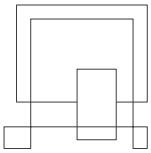
To start with, the implementation class of the gRPC server implementation is required to be annotated as a @GRpcService, otherwise the Springboot constructor will not initialize the class as a gRPC implementation, thus the server not being recognized. Furthermore the class must extend the service's implementation base such as all the "rpc" methods created in the proto file will be inherited and overridden by the developers necessity.

The instance variable UserRepository will be used to create queries and contact the database for any information requested through the server.

The following annotation @Autowired is used for constructor injection. Constructor injection is the method used by Springboot in order to initialize the class once and be available throughout the tier.

```
@Autowired
public UserService(UserRepository userRepository,
```

Figure 25 – Injection constructor



4.2 Business Logic Tier

4.2.1 Implementation in C#

The system is implemented in C# due to the easy of use with the presentation tier. The choice was also affected when choosing the way of communicating between the Data Access Tier and this tier, since the system was meant to be heterogenous, gRPC was chosen as it is available in both Java and C#.
As the stakeholders presented a wish for the system to be available for deployment on a tight schedule, the product owner chose Blazor on the Presentation Tier, thus in turn making the choice indirectly for the Business Logic Tier as well.

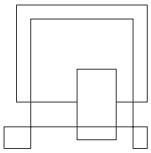
4.2.2 Code Snippets

For the sake of continuity, the following subchapter will continue with the gRPC implementation from the last chapter.
As of for the client side of gRPC, other than copying the proto files, the client must contain information about the IP address and port of the server implementation side. In turn, if the information is not provided or incorrect the client will display a “Not implemented” error.

```
1 usage
public class UserFileDao : IUserDao
{
    private readonly GrpcChannel channel = GrpcChannel.ForAddress("http://localhost:6565");
    private UserService.UserServiceClient ClientUser;

    public UserFileDao()
    {
        ClientUser = new UserServiceClient(channel);
    }
```

Figure 26 – gRPC client connection channel

The above image shows how the GrpcChannel is created, and how it is initialized.

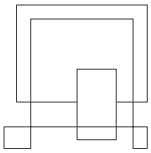


Here instead of extending the ImplBase of the gRPC and making the class a server as well, the programmer will create a variable of UserServiceClient and initialize it with the gRPC channel that has been created just above it.
Notice that the class implements the IUserDao which in fact contains the methods needed by the Presentation Tier.

```
public interface IUserDao
{
    1 usage  1 implementation
    Task<User> CreateAsync(User user);
    // Task<User?> GetByNameAsync(string firstName, string lastName);
    2 usages  1 implementation
    Task<List<User>> GetAsync(SearchUserParametersDto searchParameters);
    2 usages  1 implementation
    Task<User?> GetByIdAsync(int dtoContactId);
    1 usage  1 implementation
    Task<User?> GetLoginAsync(UserLoginDto loginDto);

    1 usage  1 implementation
    Task<User?> UpdateUserAsync(UserUpdateDto toUpdate);

    1 usage  1 implementation
    Task DeleteUserById(int id);
}
```

Figure 27 – IUserDao interface

The job of this class is to make sure that the information received from the Data Access Tier is valid, contains what is need and also make sure that the gRPC objects received are mapped back to the Domain classes.
After this part of the code, there comes the part where REST API must be implemented. It is a very crucial part of the code since this is the way of the Presentation Tier to communicate with the Business Logic Tier.



There are annotations also in C# which are inserted inside [ApiController].

```
[ApiController]
[Route( template: "[controller]")]
public class UsersController : ControllerBase
{
    private readonly IUserLogic userLogic;

    public UsersController(IUserLogic userLogic)
    {
        this.userLogic = userLogic;
    }
```

Figure 28 – REST controller

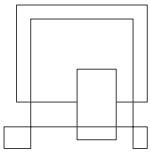
The [ApiController] annotation is used to instruct C#'s compiler that this class will perform CRUD operations inside it. The [Route("[controller]")] is used to create a URL such that everything is consistent. For example, the web URL https://localhost:7171 applied in the above image. For the system to differentiate which REST controller is accessing, the URL must change from the previous https://localhost:7171 to https://localhost:7171 + / + The name of the class without the "Controller" part which in this case will give https://localhost:7171/Users

The IUserLogic variable will make sure the information received arrives the gRPC client and will get back the response from the Data Access Tier.

C# uses injection on a similar level as Java for the Controller, as of IUserLogic, the interface will be initialized in the main class where the builder will take it as a service inside Program.cs.

```
builder.Services.AddScoped<IUserDao, UserFileDao>();
builder.Services.AddScoped<IUserLogic, UserLogic>();
```

Figure 29 – Injection in C#



Information gets through REST by providing either JSON strings, or querrys

```
[HttpPost]
public async Task<ActionResult<User>> CreateAsync(UserCreationDto dto)
{
    try
    {
        User user = await userLogic.CreateAsync(dto);
        return Created(uri: $"/users/{user.Id}", user);
    }
    catch (Exception e)
    {
        Console.WriteLine(e);
        return StatusCode(500, e.Message);
    }
}
```

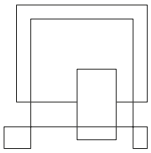
Figure 30 – Creating a user in REST

The above screenshot is used to create a User. REST uses CRUD as the template for the annotations, for example the [HttpPost] annotation above will be used to create always. As seen above the method also contains a parameter, a Data Transfer Object to be precise, which will be sent from the Presentation Tier to the Business Logic Tier as a JSON string. On the other hand, the bellow image shows that it is not the only way it can receive information, the use of queries is also very common among REST.

```
[HttpGet]
public async Task<ActionResult<List<User>>> GetAsync([FromQuery] string? firstName, [FromQuery] string? lastName)
```

Figure 31 – Having queries inside a REST method

Data Transfer Objects as name can imply, their main purpose is to make transferring data easier. For example, there may be cases like a Login when the system does not require all the information from a user just to perform an authorization. It is logical also that it is way easier to handle only the email and the password rather than filtering through the data to get everything needed for a simple login. Thereafter the system is cleaner and easier to change in case of any design flaw noticed or design rechoice.



4.3 Presentation Tier

4.3.1 Implementation in C# Blazor

As stated in the previous subchapter, the Product Owner together with the stakeholders decided that for a fast and reliable implementation Blazor should be used because of its high support from Microsoft, which also provides extremely important documentation for it. Even tough REST supports heterogenous systems, implementing everything in the same language was a very easy task and made the development go smoother.

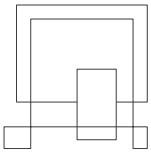
4.3.2 Code Snippets

To keep continuity furthermore the code snippets will start where the REST request are made.

```
public interface IUserService
{
    3 usages  1 implementation  More...
    Task<User> Create(UserCreationDto dto);

    1 implementation
    Task<User> Login(UserLoginDto dto);
    7 usages  1 implementation
    Task<IEnumerable<User>> GetUsers(string? firstName = null, string? lastName = null);

    3 usages  1 implementation
    Task<User> GetUserById(int id);

    1 usage  1 implementation
    Task<User> UpdateUserAsync(UserUpdateDto dto);
    1 implementation
    Task DeleteUserById(int userId);
}
```

Figure 32 – IUserService REST methods

This is more or less the same interface as the IUserDao (Figure TODO), those methods are used for all the tasks needed throughout the website.



Furthermore, the implementation will imply a HttpClient, which will need URL's to make requests for the REST controllers.

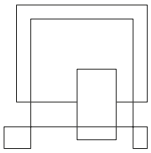
```
private readonly HttpClient client;

5 usages  Matas Armonaitis
public UserHttpClient(HttpClient client)
{
    this.client = client;
}
```

Figure 33 – HttpClient and Injection

In the above picture there is the HttpClient and the injection that takes place in the class implementing the IUserService interface.

```
public async Task<User> Login(UserLoginDto dto)
{
    using (SHA256 mySHA256 = SHA256.Create())
    {
        byte[] bytes = mySHA256.ComputeHash( buffer: Encoding.UTF8.GetBytes(dto.Password));

        // Convert byte array to a string
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString( format: "x2"));
        }

        dto.Password = builder.ToString();
    }
    HttpResponseMessage response = await client.PostAsJsonAsync( requestUri: "/Auth/login", dto);
    string result = await response.Content.ReadAsStringAsync();
    if (!response.IsSuccessStatusCode)
    {
        throw new Exception(result);
    }

    User user = JsonSerializer.Deserialize<User>(result, new JsonSerializerOptions
    {
        PropertyNameCaseInsensitive = true
    })!;
    return user;
}
```

Figure 34 – Login method



The above picture represents the Login method which hashes the password in SHA256(Anon., 2022g) after which the client creates a PostAsJsonAsync message, where the programmer provides the URL for the controller and the method that needs to be accessed, and the provided object that will be parsed to JSON, in the above case is the UserLoginDto. After a reply message is returned the JSON is then deserialized to the object declared by the programmer and then also returned to the website. If any error occurs or the authorization fails due to non-existing user, password, or email mismatch the method will throw an exception.

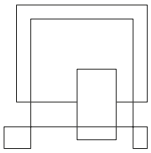
```
@page "/LoginSeeProfile"
@using ...
@inject IAuthService authService
@inject NavigationManager navMgr
@inject IUserService UserService
```

Figure 35 – Login page with injection for the services

The above screenshot contains the page URL and injects the IUserService and other services to be used for Loging in.

```
<CascadingAuthenticationState>
    <AuthorizeView Context="Auth:AuthenticationState ">
        <NotAuthorized>
            <div class="form-body">
                <div class="row">
                    <div class="form-holder">
                        <div class="form-content">
                            <div class="form-items">
                                <h3>Login</h3>
                                <p>Fill in the data below.</p>
                                <EditForm Model="@userLogin" OnValidSubmit="@HandleValidSubmit">
                                    <DataAnnotationsValidator/>
                                    <div class="col-md-12 ForInput">
                                        <InputText class="form-control" type="email" name="email" placeholder="E-mail Address" @bind-Value="userLogin.Email"/>
                                        <ValidationMessage For="() => userLogin.Email"/>
                                    </div>
                                    <div class="col-md-12" ForInput>
                                        <InputText class="form-control" type="password" name="password" placeholder="Password" @bind-Value="userLogin.Password"/>
                                        <ValidationMessage For="() => userLogin.Password" class="ForInvalidMessage"/>
                                    </div>
                                    <div class="form-button mt-3">
                                        <button id="submit" type="submit" class="btn btn-primary">Login</button>
                                    </div>
                                </EditForm>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </NotAuthorized>
```

Figure 36 – NotAuthorized view for Login



The above image represents the not authorized view of the page, this allows the programmer to create two pages in one, in this case if the user is not authorized, the page will display the login.

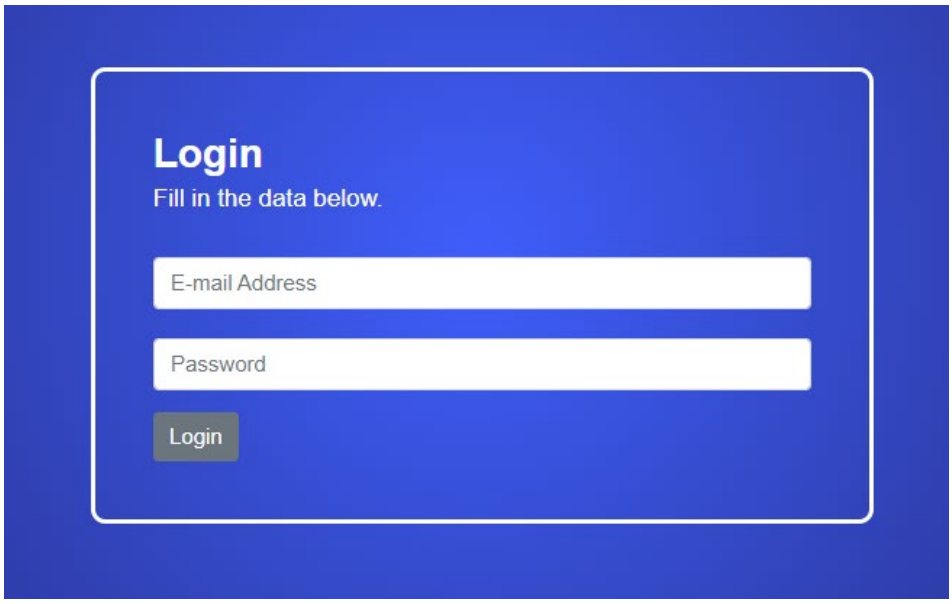


Figure 37 – Login view for not authorized

The page contains an EditForm which validates the data, and will disable the button to login if the input data is incorrect or empty.

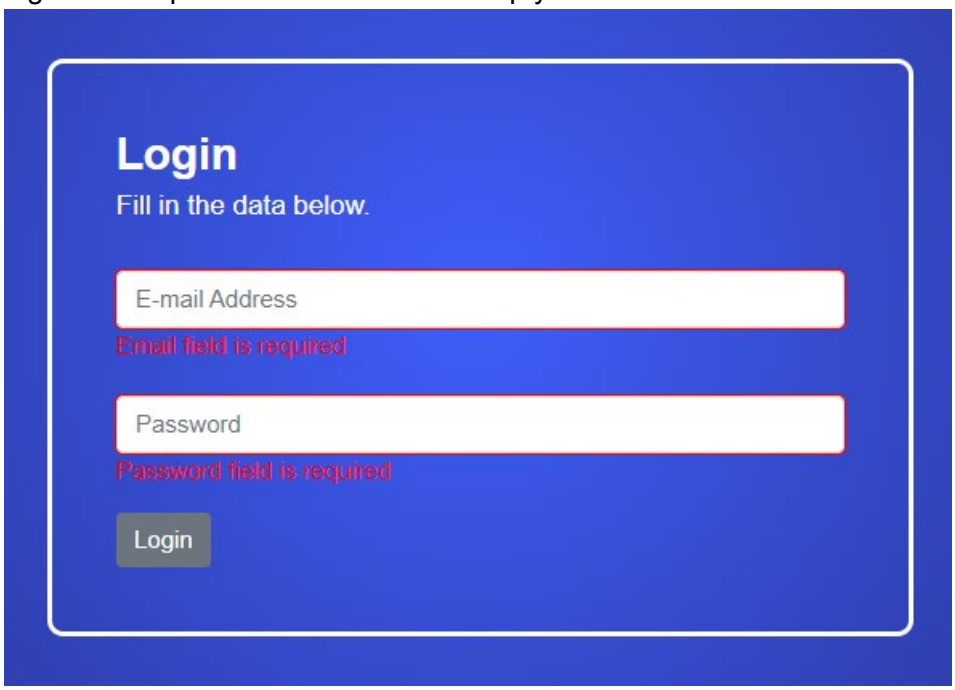
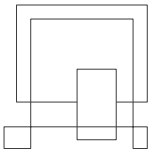


Figure 38 – Login form invalid



Following up is the Authorized view of the page

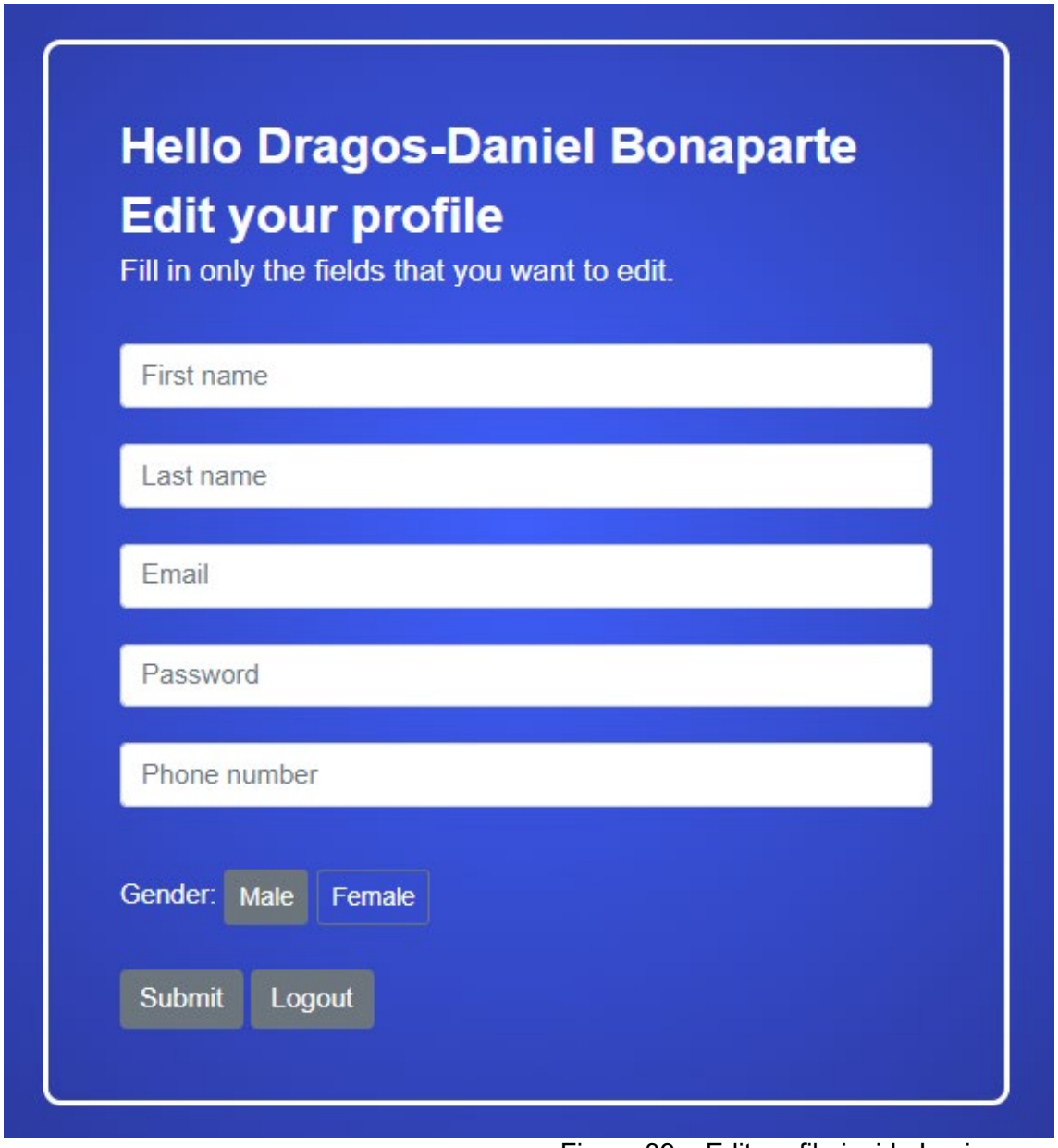


Figure 39 – Edit profile inside Login page

The above screenshot contains as well an EditForm and also has validation for incorrect information or empty.

```
<Authorized>
    <div class="form-body">
        <div class="row">
            <div class="form-holder">
                <div class="form-content">
                    <div class="form-items">
                        <h3>Hello @name.Split( separator: ' ')[0] @name.Split( separator: ' ')[1]</h3>
                        <h3>Edit your profile</h3>
                        <p>Fill in only the fields that you want to edit.</p>
                        <EditForm Model="@UserUpdateDto" OnValidSubmit="@UpdateUser">
                            <DataAnnotationsValidator/>
                            <div class="col-md-12 ForInput">
                                <InputText class="form-control ForInput" type="text" placeholder="First name" @bind-Value="UserUpdateDto.FirstName"/>
                                <ValidationMessage For="() => UserUpdateDto.FirstName"/>
                            </div>
```

Figure 38 – Authorized view of the login page

In the above snippet the Authorized tag shows the beginning of the authorized view, and then the EditForm (Anon., 2022f, Anon., 2022o, Anon., 2022d, Anon., 2022c) tag makes sure that the data is validated. If the data is invalid the ValidationMessage (Anon., 2022e) will appear displaying the error message defined in the model of the editForm, in this case the UserUpdateDto does not have any error message so the next screenshot is taken from UserLoginDto.

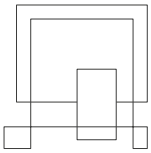
```
[Required (ErrorMessage = "Email field is required")]
```

Figure 39 – Error message

Because this is a Blazor implementation the page can contain C# code next to the HTML. The OnInitializedAsync() is a overridden method that is executed before the display of the page such as when the page requests the objects to display them, the page has already loaded them in the memory.

```
@code {


    private UserLoginDto userLogin = new();
    private UserUpdateDto UserUpdateDto = new();
    private string? name;

    [CascadingParameter]
    private Task<AuthenticationState> AuthState { get; set; } = null!;

    protected override async Task OnInitializedAsync()
```

Figure 40 – Code inside login page



A very important note is that because the system is going to be new in the market, the Product Owner decided that there will be some processing involved on the client's side such as the filter for the marketplace.(Anon., 2022ab)

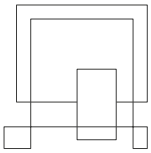
5. Test

To make sure that software the Product Owner expects is working, the software was tested every sprint. The test included some unit tests and some manual testing.

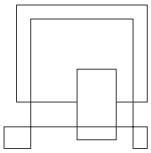
5.1 Test Cases

The manual testing was a crucial tool to make sure that everything in the system works as intended and the tiers cooperated with each other the way they were intended. The tests were created based on use case descriptions. For each implemented description a test was done to make sure it is implemented correctly. For these tests we used black box testing.

Scenario	Test scenario	Test steps	Test data	Expected results	Actual results	Pass /Fail
1	Login into the account	1. The user enters his email. 2. The user enters his password		The user is logged in	As expected	Pass
2	Creating a user	1. The user enters his credentials (First name, Last name, e-mail address, phone		The user is registered in the system	As expected	Pass

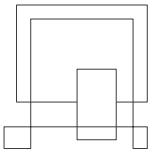


		number, password, and gender) 2. User presses the register button				
3	View Marketplace	1. The user presses the marketplace button. 2. The user will be redirected to the marketplace		The marketplace will show a list with all listened items	As expected	Pass
4	Filtering items	1. The user can filter items by choosing the category, currency, and price range		The marketplace will show the items with filter	As expected	Pass
5	See post details	1. The user presses the marketplace button 2. The user chooses the item he wants to buy		The program will show all the information about that item	As expected	Pass
6	Message seller	1. The user presses the marketplace button 2. The user chooses the item he wants to buy 3. The user presses the message seller button 4. The user will be redirected to messages		The user can see the conversation of the item he wants to buy and to message the seller	As expected	Pass
7	Create post	1. The user must login into his account 2. The user presses the create post button 3. The user enters all the necessary data 4. The user will be redirected to another window to insert a picture		The item will be placed on the marketplace	As expected	Pass



8	Edit post details	1. The user must login into his account 2. The user goes to his items 3. The user chooses the item he wants to edit 4. The user presses the edit button 5. The user edits the fields he wants 6. The user submits the changes		The item will be edited with the new changes both on users' profile and marketplace	As expected	Pass
9	Delete item	1. The user must login into his account 2. The user goes to his items 3. The user chooses the item he wants to edit 4. The user presses the delete button		The item will be deleted from users profile and marketplace	As expected	Pass
10	Edit user	1. The user must login into his account 2. The user presses the profile icon 3. The user chooses the information he wants to change 4. The user submits the changes		The new information about the user will be updated	As expected	Pass
11	Log out	1. The user must login into his account 2. The user presses the profile icon 3. The user presses the log out button		The user will be logged out from his account	As expected	Pass

Table 4 – Test cases



5.2 Business logic tier/Data Access tier tests

In order to test both tiers it was decided to make the tests in the business tier, but to also use the database server from data access tier for testing some of the use cases. Once again use case descriptions were used to create the test cases. This allowed to see when adding additional implementation if there were any bugs that influenced some of already completed use cases without the need for interacting with the presentation tier. Xunit library was used for these tests.

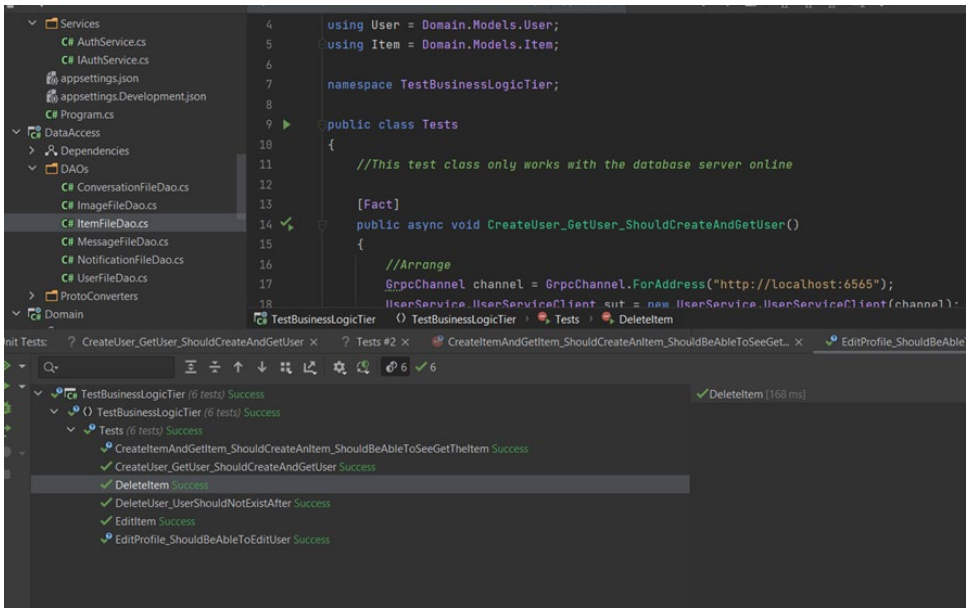


Figure 41 – Business Logic Tier/Data Access Tier tests

5.3 Unit tests

Because of the amounts of manual testing, testing in other tiers and the difficulty of automated tests, the development team decided to only implement necessary tests for the presentation tier. The tests are made to work without being dependant on other tiers, meaning it required mocking and quite a bit of research in order to make them work. Only necessary tests for the presentation tier to work were implemented in order for the development team to fit in to the time schedule.

Bunit and Nunit.autofixture frameworks and white box together with aaa(Arrange, Act, Assert) strategy was used to implement the unit tests. Bunit allowed to make the tests independent from the other tiers and autofixture allowed us to input random data without the need of manual input. In total 3 Tests for the presentation tier were implemented, which all passed in the end. In order to keep other more simple units like setters and getters working properly, manual testing was used throughout all of the sprints.

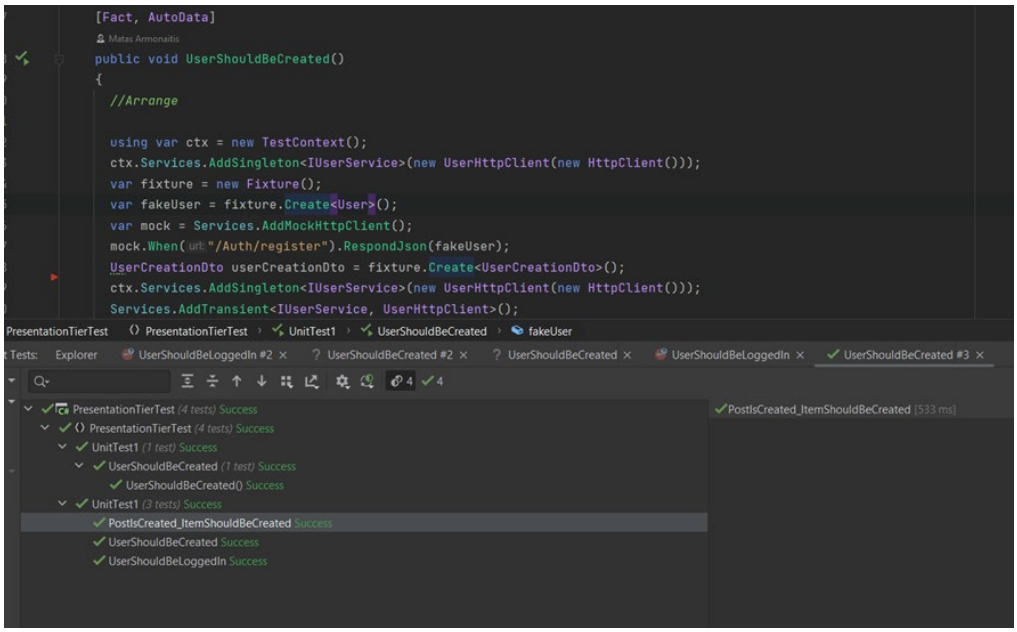
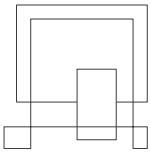


Figure 42 – Presentation Tier tests



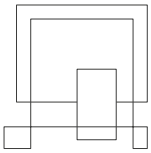
6. Results and Discussion

22 functional requirements total, including 7 critical, 3 high, 9 medium, and 3 low priority requirements, are listed in the Product Backlog. At first, the main objective of the development team was to satisfy the Product Owner by at least implementing the essential priority needs. The needs that were chosen for each sprint of the development process did not always begin with the most important and work their way down in priority; instead, if some of the lower priorities could be implemented in conjunction with the higher priorities, they were also added to the sprint.

Out of a total of 22, 11 conditions have been totally met. Due to a lack of time, the team opted to concentrate on the most crucial components of the program and make sure they function as intended and have through all necessary testing. The most crucial components of the software must be functional, and the other features can be added later, it was concluded after discussing this with the product owner.

The most challenging aspects of the project throughout development were encountered when implementing the critical requirements. Examples would include adding images to the Items, which needed extensive research to ensure it looked and worked as good as possible. The following example would be the time-consuming and labor-intensive process of integrating the data access and business logic tiers. Making the marketplace filtering function was the third challenging aspect because the team lacked much blazor and css experience. Fortunately, the team dedicated a lot of time to these implementations during the sprints, but it still took longer to implement than expected. The last issue was estimating how much work could be accomplished in each sprint. The team almost always fell behind schedule because they assigned too many tasks during each sprint, believing they could finish them on time.

At the conclusion of the project, the development team tested each developed feature to ensure that it is functional, and the product owner approved it after ensuring that it functions as intended. The program was black box and white box tested through manual testing and unit testing. In the end, the product owner approved it based on the requirements of the stakeholders.



7. Conclusions

The request of the non-profit organization was to help towards achievement of one of the United Nations goals towards sustainable future, specifically goal number 12 – responsible production and consumption. A decision was made to develop a system where people could easily give away their unused things or even sell them for some extra money instead of throwing them away in the garbage.

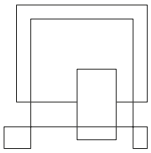
The solution was based on the need to have a simple system that can be used by anyone, so simplicity was a priority while developing the system. Scrum and Unified process were used as a development framework.

The system consists of 23 requirements (22 functional and 1 nonfunctional), which are covered in 6 use cases. The use cases were centered mainly around users, items, conversations and messages. In order to differentiate between users a login system was implemented, and each user is logged in using an email address and a password. If a user chooses not to register an account, they can still browse the items for sale. However, they will not be able sell items themselves or to contact the seller using the messaging system provided by the application. They will therefore have to contact the seller using a different means (for example phone number or email address).

Following these requirements, a decision was made to split the implementation into three parts: presentation (containing the user interface), business logic (responsible mainly for validation of the data) and database (responsible for storing data). The system was afterwards designed in a heterogeneous way. In order to achieve that HTTP protocol with RESTful API was used between presentation and business tier, and gRPC was used between business and database tier. The presentation tier was developed using Blazor Web Assembly, business logic was developed using .NET 6.0 and database tier using Java Spring Boot.

Implementation and testing were done hand in hand, meaning that after each finished requirement a thorough testing was concluded in order to make sure everything is working properly in both happy and unhappy scenarios. Black box testing was concluded using Swagger (for business tier), BloomRpc (for database tier). The user interface was tested manually by entering different values and trying different scenarios. The white box is done using BUnit, XUnit and NUnit.autowire.

The end of the project resulted in all critical and most of high priority requirements were met, which resulted in a working system that can be used by the end users.



8. Project future

A product's goal is to please its customers and give them high-caliber services. Because of this, a major priority during the entire development process was to admit the system's flaws and concentrate on fixing them in the future to give users a more positive experience.

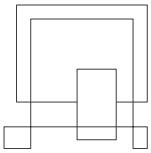
Implementing the security elements that were specified and covered in the design part would thus be the first step in a future upgrade plan. The system is currently susceptible to the vast majority of assaults that could compromise the system and its data, which is the cause of this. Therefore, it's crucial to make sure the system offers a safe environment for users to communicate in.

The delimitations mentioned in the Project Description were also among the initial features that the development team thought about including in a later version of the system. Implementing financial transactions within the system, for instance, would be a useful feature. This will enable system users to use their chosen means of making financial transactions, whether it be through the system's built-in features or a third-party API like MobilePay.

The management of shipping is a crucial feature that will need to be included in the future in order to provide system users with a positive user experience. The technology now only allows consumers to make purchases; however, shipping is taken care of off-camera.

The features that were not used throughout the development phase are additional features that might be considered during future developments. The "report" feature wasn't essential, but it would undoubtedly enhance the user experience. The same would apply to features that are specific to the products of the business, including having time-limited events, signing up for an item presale and automatically buying the item when it goes on sale.

When adding new features and upgrading the system, there are a few factors related to the code implementation that could also be considered. One of them is deeper class separation to concentrate on managing resources for just one functionality.



9. Sources of information

Banger, D., 2014. A Basic Non-Functional Requirements Checklist « Thoughts from the Systems front line.... Available at: https://dalbanger.wordpress.com/2014/01/08/a-basic-non-functional-requirements-checklist/ [Accessed January 31, 2017].

Business Analyst Learnings, 2013. MoSCoW : Requirements Prioritization Technique — Business Analyst Learnings. , pp.1–5. Available at: https://businessanalystlearnings.com/ba-techniques/2013/3/5/moscow-technique-requirements-prioritization [Accessed January 31, 2017].

Dawson, C.W., 2009. *Projects in Computing and Information Systems*, Available at: http://www.sentimentaltoday.net/National_Academy_Press/0321263553.Addison.Wesley.Publishing.Company.Projects.in.Computing.and.Information.Systems.A.Students.Guide.Jun.2005.pdf.

Gamma, E. et al., 2002. *Design Patterns – Elements of Reusable Object-Oriented Software*, Available at: http://books.google.com/books?id=JPOaP7cyk6wC&pg=PA78&dq=intitle:Design+Patterns+Elements+of+Reusable+Object+Oriented+Software&hl=&cd=3&source=gbs_api%5Cnpapers2://publication/uuid/944613AA-7124-44A4-B86F-C7B2123344F3.

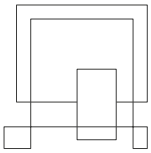
IEEE Computer Society, 2008. *IEEE Std 829-2008, IEEE Standard for Software and System Test Documentation*,

Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*,

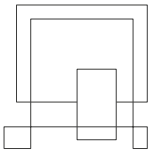
Mendeley.com, 2016. Homepage | Mendeley. Available at: https://www.mendeley.com/ [Accessed February 2, 2017].

YourCoach, S.M.A.R.T. goal setting | SMART | Coaching tools | YourCoach Gent. Available at: http://www.yourcoach.be/en/coaching-tools/smart-goal-setting.php [Accessed August 19, 2017].

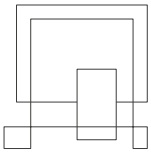
Anon. 2022a. *14 CSS Particle Backgrounds*. [online] Available at: <https://freefrontend.com/css-particle-backgrounds/> [Accessed 28 November 2022].



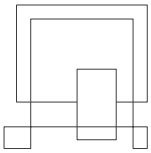
Anon. 2022b. *35 CSS Grid Examples*. [online] Available at: <https://freefrontend.com/css-grid-examples/> [Accessed 28 November 2022].
Anon. 2022c. *An Overview On InputRadioGroup Component In Blazor WebAssembly*. [online] Available at: <https://www.learmoreseekmore.com/2020/12/blazorwasm-inputradiogroup-overview.html> [Accessed 5 December 2022].
Anon. 2022d. *Blazor: How to use the onchange event in <select> when using @bind also? - Stack Overflow*. [online] Available at: <https://stackoverflow.com/questions/58125929/blazor-how-to-use-the-onchange-event-in-select-when-using-bind-also> [Accessed 5 December 2022].
Anon. 2022e. *Blazor University - Validation*. [online] Available at: <https://blazor-university.com/forms/validation/> [Accessed 5 December 2022].
Anon. 2022f. *c# - Binding Type double values in InputText of Blazor app Razor page - Stack Overflow*. [online] Available at: <https://stackoverflow.com/questions/68274549/binding-type-double-values-in-inputtext-of-blazor-app-razor-page> [Accessed 5 December 2022].
Anon. 2022g. *Compute SHA256 Hash In C#*. [online] Available at: <https://www.c-sharpcorner.com/article/compute-sha256-hash-in-c-sharp/> [Accessed 18 November 2022].
Anon. 2022h. *Core concepts, architecture and lifecycle | gRPC*. [online] Available at: <https://grpc.io/docs/what-is-grpc/core-concepts/> [Accessed 15 December 2022].
Anon. 2022i. *CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc.* [online] Available at: <https://crackstation.net/> [Accessed 15 December 2022].
Anon. 2022j. *Denial of Service and Prevention - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/deniel-service-prevention/> [Accessed 13 December 2022].
Anon. 2022k. *Facebook Marketplace Rules for you to know before putting up anything on sale | Apps*. [online] Available at: <https://www.republicworld.com/technology-news/apps/facebook-marketplace-rules-for-you-to-know-before-putting-up-anything.html> [Accessed 24 September 2022].



Anon. 2022l. *Furniture waste – The forgotten waste stream - Recycle Track Systems*. [online] Available at: <https://www.rts.com/blog/furniture-waste-a-growing-issue/> [Accessed 12 December 2022].
Anon. 2022m. *Global e-waste generation 2010-2019 | Statista*. [online] Available at: <https://www.statista.com/statistics/499891/projection-ewaste-generation-worldwide/> [Accessed 12 December 2022].
Anon. 2022n. *Home - United Nations Sustainable Development*. [online] Available at: <https://www.un.org/sustainabledevelopment/> [Accessed 12 December 2022].
Anon. 2022o. *How do I get the checkbox value if it is checked? | Blazor FAQ | Syncfusion*. [online] Available at: <https://www.syncfusion.com/faq/how-do-i-get-the-checkbox-value-if-it-is-checked> [Accessed 5 December 2022].
Anon. 2022p. *JPA vs Hibernate: Know The Difference - InterviewBit*. [online] Available at: <https://www.interviewbit.com/blog/jpa-vs-hibernate/> [Accessed 15 December 2022].
Anon. 2022q. *List on International eBay Sites*. [online] Available at: <https://galaxy.maropost.com/s/article/list-on-international-ebay-sites> [Accessed 24 September 2022].
Anon. 2022r. *Man in the Middle (MITM) Attacks | Types, Techniques, and Prevention*. [online] Available at: <https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/> [Accessed 13 December 2022].
Anon. 2022s. *Payment methods policy | eBay*. [online] Available at: <https://www.ebay.com/help/policies/payment-policies/payment-methods-policy?id=4269> [Accessed 24 September 2022].
Anon. 2022t. *Policies on restricted or prohibited items – overview | eBay*. [online] Available at: <https://www.ebay.com/help/policies/prohibited-restricted-items/prohibited-restricted-items?id=4207> [Accessed 24 September 2022].
Anon. 2022u. *Protect from cross-site scripting attacks - IBM Garage Practices*. [online] Available at: <https://www.ibm.com/garage/method/practices/code/protect-from-cross-site-scripting/> [Accessed 13 December 2022].
Anon. 2022v. *Shipping and delivery for buyers | eBay*. [online] Available at: <https://www.ebay.com/help/buying/shipping-delivery/shipping-delivery-buyers?id=4005> [Accessed 24 September 2022].



Anon. 2022w. *SOLID: The First 5 Principles of Object Oriented Design | DigitalOcean*. [online] Available at: <https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design> [Accessed 15 December 2022].
Anon. 2022x. *Spring Boot Tutorial | DigitalOcean*. [online] Available at: <https://www.digitalocean.com/community/tutorials/spring-boot-tutorial> [Accessed 15 December 2022].
Anon. 2022y. *The DTO Pattern (Data Transfer Object) | Baeldung*. [online] Available at: <https://www.baeldung.com/java-dto-pattern> [Accessed 15 December 2022].
Anon. 2022z. *The Environmental Crisis Caused by Textile Waste*. [online] Available at: <https://www.roadrunnerwm.com/blog/textile-waste-environmental-crisis> [Accessed 12 December 2022].
Anon. 2022aa. *Threats and Pitfalls.pdf*. [online] Available at: <https://via.itslearning.com/LearningToolElement/ViewLearningToolElement.aspx?LearningToolElementId=2613431> [Accessed 13 December 2022].
Anon. 2022ab. *Web application client vs server side processing*. [online] Available at: <https://www.pragimtech.com/blog/blazor-webAssembly/web-application-client-vs-server-side-processing/> [Accessed 5 December 2022].
Anon. 2022ac. *What Is A Brute Force Attack? How To Prevent It?* [online] Available at: <https://www.cloudways.com/blog/what-is-brute-force-attack/> [Accessed 13 December 2022].
Anon. 2022ad. *What is PostgreSQL*. [online] Available at: <https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/> [Accessed 15 December 2022].
Anon. 2022ae. *What is Three-Tier Architecture | IBM*. [online] Available at: <https://www.ibm.com/cloud/learn/three-tier-architecture> [Accessed 12 December 2022].
Wang, X. and Zhang, W., n.d. Cross-site scripting attacks procedure and Prevention Strategies. [online] https://doi.org/10.1051/03001.



10. Appendices

1. Appendix A – User Interface Design
2. Appendix B – Group Contract
3. Appendix C – Project Description
4. Appendix D – Test Cases
5. Appendix E – User Cases
6. Appendix F – Activity Diagrams
7. Appendix G – EER Diagram
8. Appendix H – System Sequence Diagram
9. Appendix I – Code
10. Appendix J – Sequence Diagram
11. Appendix L – Class Diagram
12. Appendix M – Sprint Burndown Charts
13. Appendix N – Domain Model
14. Appendix O – Astah Files

No Brain No Pain

ChadList

Chiril Luncașu (315171), Dan Sebastian Ceapă (315162),

Matas Armonaitis (315263), Dragoș-Daniel Bonaparte (315261)

Tomáš Greš (315185)

Software Technology Engineering

Semester 3

22/09/2022

Version: 0.1

Table of content

1. Background Description 1
2. Problem Statement 3
3. Definition of purpose 4
4. Delimitation 5
5. Methodology 6
6. Time schedule 7
7. Risk assessment 8
8. Sources of Information 9

Appendices (including Group Contract)

1. Background Description

Today everybody can agree that the amount of waste from all categories of products is just unbelievably huge. The project taken on by the team strives to lower the amount of waste that is being produced yearly.

There are out there a lot of web sites who are doing the same thing but, a lot of those web sites run only in specific countries, there is a chaos when it comes to what you are searching for and what you are getting, you cannot search the online shop without an account, very often it happens that the products that are being displayed are not well described and do not provide any useful information for the buyer.

Currently eBay is one of the biggest online second-hand shopping sites but, at the moment it is only available in 25 countries (Anon., 2022b), which is not ideal for the end-user since sometimes the cost of shipping is the same as the product you are purchasing. On the other hand, eBay supports payment by card, PayPal, PayPal Credit, Credit card or debit card, Apple Pay and Google Pay, and, it also has options for delivery, if the buyer is cannot pick-up the item in person (Anon., 2022c, Anon., 2022e).

On the other side Facebook create Facebook Marketplace which is available in a lot more countries compared to eBay but, its lack of organization, lack or false information about the product and the fact that the end-user must have an account in order to shop on the website makes it very annoying and frustrating to navigate. Facebook does not support any of the payment types that eBay has, but it has a useful map in case the seller does not want to show his address to the public, which helps with confidentiality. Unlike eBay, Facebook Marketplace does not support shipping, but since the website requires you to have an account, you can message each other instantly with a click of a button.

Finally, is Craigslist which is a very popular alternative to eBay or Facebook Marketplace. It is available in most of the countries, the end-user can navigate the website without an account but, the website in the team’s opinion looks very ruff, very complicated and a lot of the items that are posted do not contain precise information and that is without mentioning that the website does not have a search bar that can

search through all the categories at the same time. To be mentioned is that Craigslist does not have either payment support or shipping support.

All the websites mentioned above have policies regarding selling forbidden items (Anon., 2022a, Anon., 2022d). For example, eBay has restrictions when it comes to adult items, alcohol and many more, Facebook Marketplace has restrictions for selling Pets, drugs and many others.

2. Problem Statement

A User wants to either sell their item or buy an item they want through a website. The user needs to be able to chat with the seller or if the user is selling an item to be able to be contacted.

1. How to make the website to not be used for selling illegal items.
2. How to make users be able to communicate with each other when buying or selling
3. How to make the website available to everybody around the world.
4. How to enable users to buy items with or without an account.
5. How to make the website easy to navigate and organized.

3. Definition of purpose

The purpose of this project is to help the users of the system sell items they no longer use or need and buy items they would like to own but cannot afford or do not want to pay full price for.

4. Delimitation

- We will not make a map to see where the seller is selling their item.
- We will not make a function to ship items through our website.
- We will not make transactions through our website.
- We will not allow users to sell without and account.
- We will not use complex recognition for illegal items.
- The website will only have the “.com” domain.

5. Methodology

A decision was made that for the development process, we will use a combination of Scrum and Unified Process while following the Agile manifesto.

Scrum will be used to divide the time working on the project into a fixed period of time called sprints. In our case, each sprint will be 5 days long.
In a sprint the requirements chosen to work on will be split into smaller tasks, that the
development team work on independently, therefore each member will assign task to himself.

Apart from that a Daily Scrum meeting will be held every day for a maximum of 15 minutes (usually online), where every team member will give feedback about the task they have been working on, also what is going to be done next and what problems will be encountered.

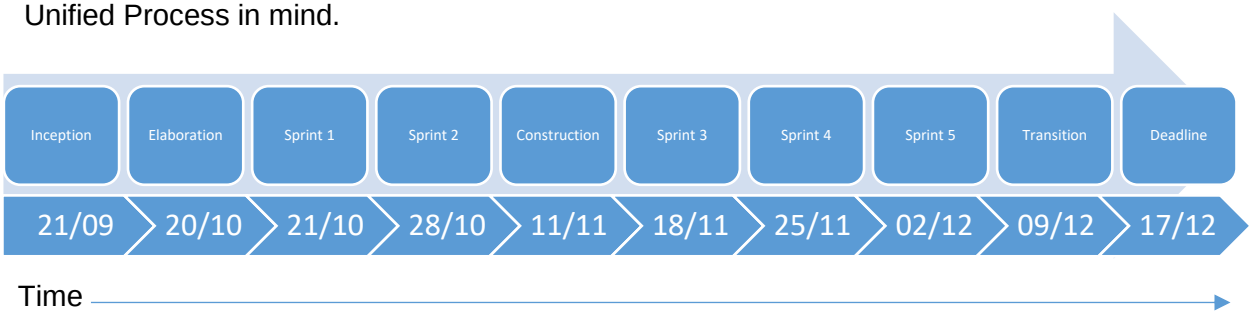
A Burndown chart will be created and updated every day, during the daily sprint meetings for each of the sprints, therefore the team will be able to keep track of how ahead or behind they are in the particular sprint.

There will also be assigned role of Product Owner and Scrum Master for two of the members of the team. The Product Owner will be responsible for the final product, and
he will test it after each sprint. The Scrum master will be responsible for the work going
as smoothly as possible. His responsibilities include: updating the Sprint backlog and the Burndown chart during each daily meeting and booking rooms for when the team meets.

The Unified Process (UP) phases Elaboration, Construction and Transition will be utilized in each of the sprints. For each requirement, the team will first create diagrams and design (Elaboration), then the database will be updated, classes necessary to implement requirement as well as testing if all of the implementation work as it is supposed (Construction) to make a working feature/part of a system (Transition).

6. Time schedule

Given that the expected workload is 27.5 hours per ECTS per student, we will be working a total of 275 hours per person in the group. This amounts 1375 of hours of work in the project. The workload will be spread out with the Scrum methodology and Unified Process in mind.



7. Risk assessment

Risks	Likelihood Scale: 1-5 5 = high risk	Severity Scale: 1-5 5 = high risk	Product of likelihood and severity	Risk mitigation e.g., Preventive- & Responsive actions	Identifiers	Responsible
Logical Flaws	3	5	15	Coding in pairs, and plenty of testing.	Logical flaws – the program does not work as intended.	Dragos
Database problem	4	4	16	Having a weekly backup from the database with essential information for the program.	Program cannot fetch information from the database	Chiril
Loss of data	5	5	25	There should be a copy of the code to use as backup in case the main website and application fails.	Website and application failure.	Dan
Server goes down	3	5	15	Checking the server host, check if there is not an unknown IP address, and switch to an alternate host until the main host is resolved.	If no server identifier is set, the DHCP server sets the server identifier based on the primary interface address used by the server to receive a client request.	Tomas, Matas

8. Sources of Information

Anon. 2022a. *Facebook Marketplace Rules for you to know before putting up anything on sale | Apps*. [online] Available at: <https://www.republicworld.com/technology-news/apps/facebook-marketplace-rules-for-you-to-know-before-putting-up-anything.html> [Accessed 24 September 2022].

Anon. 2022b. *List on International eBay Sites*. [online] Available at: <https://galaxy.maropost.com/s/article/list-on-international-ebay-sites> [Accessed 24 September 2022].

Anon. 2022c. *Payment methods policy | eBay*. [online] Available at: <https://www.ebay.com/help/policies/payment-policies/payment-methods-policy?id=4269> [Accessed 24 September 2022].

Anon. 2022d. *Policies on restricted or prohibited items – overview | eBay*. [online] Available at: <https://www.ebay.com/help/policies/prohibited-restricted-items/prohibited-restricted-items?id=4207> [Accessed 24 September 2022].

Anon. 2022e. *Shipping and delivery for buyers | eBay*. [online] Available at: <https://www.ebay.com/help/buying/shipping-delivery/shipping-delivery-buyers?id=4005> [Accessed 24 September 2022].

Appendices

Group Contract

Group Name: No Brain No Pain (NBNP) Date: **10/09/2022**

These are the terms of group conduct and cooperation that we agree on as a team.

Participation:

We agree to participate in all the tasks, work hard, socialize with each other and feel good while working.

Communication:

We agree to communicate mainly on physical meetings and on Discord. Alternatively, we will use messenger and ultimately phone calls

Meetings:

We agree to meet on every Wednesday with some exceptions and with consent from every member we shall meet at least one more time per week.

Conduct:

We agree to be active in the group, willing to work and finish tasks before the deadline.

We shall vote every idea of any members to have the satisfaction of being heard.

Conflict:

If we get into argues, we shall end the conflict at once and stop it from escalating.

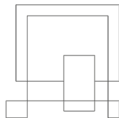
Deadlines:

Every member agreed to be active and respond before each deadline with their work.

Other Issues:

We agree to try to prioritize distributing tasks more, and, in as equal manner as possible.

Group member's name	Student number	Signature
Chiril Luncașu	**315171**	
Matas Armonaitis	**315263**	
Dan Sebastian Ceapă	**315162**	
Dragoș-Daniel Bonaparte	**315261**	
Tomáš Greš	**315185**	



Process Report - SEP

Group 2

Group Members:

Dragos Daniel Bonaparte 315261

Dan Sebastian Ceapa 315162

Matas Armonaitis 315263

Chiril Luncasu 315171

Tomas Gres 315185

Supervisors:

Jakob Knop Rasmussen(JKNR)

Joseph Chukwudi Okika(JOOK)

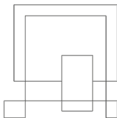
Characters 30204

Software Engineering

3rd Semester

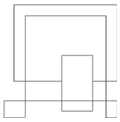
17 / 12 / 2022

VIA University College



Contents

1. Introduction 3
2. Group Description 5
Personal Profiles 5
Cultural Background 8
3. Project Initiation 10
4. Project Description 11
5. Project Execution 12
6. Personal Reflections 14
6.1. Tomas Gres 14
6.2. Dan Sebastian Ceapa 15
6.3. Matas Armonaitis 16
6.4. Chiril Luncasu 17
6.5. Dragos Daniel Bonaparte 18
7. Supervision 19
8. Conclusion 20

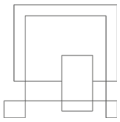


1. Introduction

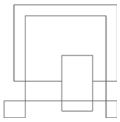
When we found out the classes will be separated this semester, we knew we're going to stay together as a group. Although last semester we had some roadblocks while developing our app, we still found similar goals we strived to achieve. Our team at the time could be improved, even though we had an amazing setup where everybody had their own role and brought their own contribution to our project, we still thought we could use a breath of fresh air. While scouting for people that we could add to our roaster we were carefully analysing our new colleagues. One stood out from the start, which helped us make a choice quite fast. Tomas was a perfect addition to our group. His knowledge, determination, will for success and sense of humour made us understand that he is quite a catch. So, after carefully discussing it with the prior group members we embarked on our journey to success together.

We would obviously lie to you if we said that the project requirements were easy to grasp at the start. Terminologies like heterogeneous distributed system, security threat models or 3 tier architecture were foreign to us, and quite frightening. Despite our morale being down, we had a lot to learn during the semester while attending our courses and doing our homework regularly as a group, so we make sure everybody understands every detail, so we don't lose anyone on the road. Sometimes we had to take a deep dive into the subjects ourselves and find a way to make everyone understand the concepts better by teaching them to each other which boosted our teamwork. We did ask for a lot of help from the supervisors as well, which led to us spending our time on important things rather than cluelessly debugging our code with no use. The methodologies that we used this semester played a major part as well. Scrum was a hit with amazing results, which probably meant we must use it again with a new tool this time. We decided to use Jira to assist us in keeping track of the roadmaps, burndown charts and sprints. Also, we tried a new technique called Scrum poker, which made us understand more about time management and how we can make our work more efficient. Unified Process helped us with our system and the teamwork we were exerting, and how can we forget about Agile. Last semester one of our teachers made us fall in love with the Manifesto for Agile Software Development, so we made sure to use it again! We really think that without the procedures enumerated above we wouldn't be able to get a final project we are proud to present.

The countless meetings, the sleepless night while brainstorming together and all the mentor meetings we had at the start made us take the correct path to our goal which was ultimately finishing the semester with amazing results. We infused our code, proposals, and assumptions with the theory we've learned to get a promising result.



Despite our different characters and overwhelming personalities, we still managed to find each other's strengths quickly. Everybody contributed with something new, and we went through with our grandiose ideas. In the next chapters we'll be talking more about how our differences made us the unit that we are in Group Description, we'll talk about how we started the project and how ChadList became our child in Project Initiation. Of course, Project Description will contain the process of making the project description, and surely mentioning Unified Process, Scrum meetings, Agile development, and artifacts in Project Execution. You will get everyone's individual opinion about the project in Personal Reflections and we'll highlight our supervisor cooperation in the Supervision Section. Conclusion will be considered as the icing on the cake where we will give a general overview of our opinions about the whole process of this semester project.



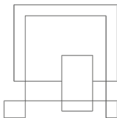
2. Group Description

So, after the last semester we decided to stay in the same group. Sometimes it's hard to find individuals who have the same goals in mind as you, but in this case everybody agreed on one thing, everyone wanted to have a great project, get a desirable grade and have fun while doing it. We consider our group very diverse, mostly because we come from different backgrounds, countries, and everybody has a different personal profiles, which makes everybody responsible of some of the characteristics in the group. We are proud to announce that even though we were so different last semester, at the start we felt that we were lacking a creative mind that would bring new ideas and implement them masterfully. That's where Tomas came into the picture. He is very different from us. He may look like he isn't very fun to be around, but if you get to know him, he is an ingenious mind that loves teamwork and excels in the tasks he dedicates himself to. So now with 5 members we chose to jump headfirst into our tasks.

Personal Profiles

As we mentioned last semester we are very different when it comes to character types, SCRUM roles were quite easy to choose. First, we saw the amazing job our Green Matas did last semester as a SCRUM master because of drive to nourish relationships with people, his capability to make harsh decisions when needed, and friendliness. It was a clear pick and we're happy of our decision. He was protecting the team from the Product Owner while fostering communication, maintaining the atmosphere as friendly as possible, and meeting facilitations. When talking about the Product Owner we had a harsher time. We were happy with what our Blue Dan did, but we challenged ourselves to find a new type of Product Owner that would spark some new ideas or points of view. That's why we chose our Yellow Dragos. He is enthusiastic which would put much higher goals for us, something that made us work harder, he is persuasive which meant that our SCRUM master had to work a lot more on protecting us from his wraith of new assignments we wanted to give us, but mostly he is sociable, which made us think of a real life situation where your Product Owner is very sociable and knows what he wants to do with his idea, but he needs people to achieve his goals, someone who would be very happy to start working, and very fast with his unpredictable decisions.

As always, we will back up our claims by showing our personal profiles a little lower. We learned a lot from last semester. After having some problems with issuing authority, lack of motivation, influence, and fear of appearing weak, but we are proud to finally say that we worked on our weaknesses, and we consider our team to be perfect. When it came to the issuing authority, last semester we learned that if we showed some boldness and some group members tried to seek status which solved our problems we would succeed. When it came to our fear of

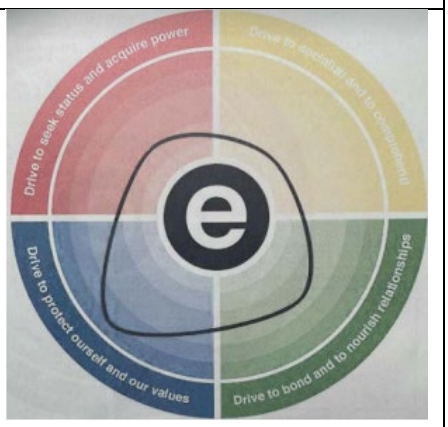


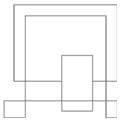
appearing weak, lack of motivation, and lack of influence we usually solved it by talking things out and trying to find a solution all together, because that's why we were assigned as a team, to go through hard times together, not to fight on alone. This meant enhancing our productivity and teamwork every time we went through these things, and that led us to describe ourselves as a total unit.

We consider that we can describe our team with one word, which is harmony, because we complement each over. We have Dragos who is a very Yellow person, always enthusiastic and sociable, happy to help and dynamic, we also have Matas, who is encouraging with everything, he cares about everyone because he values our sanity, always patient and relaxed with every little thing he does. We have Dan and Tomas that are precise, formal all the way through, always questioning and interested in the process, and lastly we have Chiril, who even though represents a lot of the Blue characteristics, he has enough Red to be flexible into that role. Always competitive, strong-willed, purposeful, and demanding from his peers, which leads to the authority we needed so much.

We learned from last semester and put some strict boundaries when it comes to laziness and time frames, which made our group prosper. The personal profiles were a very big turning point in our relationships in the group, because they showed us who should do what and who we can rely on for tasks like leading, creativity, uniting people, and enthusiasm filling.

In a nutshell, we really think we did a great job with SCRUM roles, assigning group tasks and matching everyone based on their personality type by looking at our results from last semester. We consider that our hard work finally paid off, and we can openly say we are all living in harmony.

Matas Armonaitis	Dan-Sebastian Ceapa
	
Dragos Daniel Bonaparte	Chiril Luncasu
Tomas Gres	



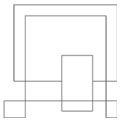
Cultural Background

We talked a lot about this one last semester, and we think our opinions didn't change this semester. A big part of our group is the fact that we have the same cultural background, all sharing a common history, the same community back home that shaped us into what we are today, and of course, the fact that 3 of us have the same nationality, which lead to a better understanding of our needs.

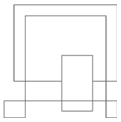
First, because our cultural background is extremely similar, and we've spent more than a semester together, we managed to understand everyone pretty well. No new challenges have been met, which we pride ourselves with. Another thing that is worth noting is that we didn't have any conflicts whatsoever. That boosted our confidence a lot, hence we know what everybody looks forward to when starting the project. Despite needing someone last semester to act as a human translator sometimes, we went over all the harsh topics last semester and we caught our new member up to speed, which saved us a lot of time and effort when it came to people being mad and not understanding themselves and their feelings related to the task put in front of us.

We tried to make sure our similar cultural background doesn't get in the way of our understanding the other cultures, or the supervisors that might not have the same values as all of us. Tomas really helped us with this, he comes from a slightly less harsh/strict country, and he was the fresh look we were looking for last semester. On one hand he was coming up with ideas of how to do things differently and we were psyched to implement them and try them out hoping for a different result. On the other hand, a lot of the ideologies we have already proven to work in the group were met with a warm welcome by him, because he as well thought those are great skill expressions and implementations.

As we mentioned last semester our groups most important dimensions were Disagreement, Communication, and Leading. We can't stress enough how much we love being direct while communicating. When we disagree, we are very confrontational, which leads to fast results and accurate expectations. When disagreeing in our group we base everything on facts which are represented in an open manner. When communicating to everyone we mostly use a low-content approach, which enhances productivity because we don't waste time on small details we can figure out later. Being direct and having a low-content way o speaking is very desirable way of explaining yourself in our group, so we cherish it. Lastly, but not least, leading is the centre piece of our communication. We love the egalitarian approach where everyone has a say, where everyone accepts new ideas and where everyone can speak up when needed.



To sum up, we are a team that comes from similar cultural backgrounds, and we all cherish some similar rules when talking to each over. We define ourselves as egalitarians that can easily do a task when met with a low-content direct type of speaking.



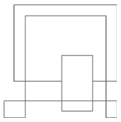
3. Project Initiation

Our group was formed out of the friendship and comradery towards each over. Dragos, Matas, Dan, and Chiril already knew each over and were very happy with how they worked together last semester they decided to work together again. Chiril saw Tomases potential and tried to add him into the group even before the semester started. Because Tomas had a bad experience with his group from the last semesters, he decided to join Chiril and his group on their journey, becoming a part of NBNP.

The name of our humble group is NBNP. It stands for “No Brain No Pain” – which is an inside joke that we designated as the final name of the group. The friendly, fun, happy atmosphere always allowed for jokes between our group members. We a lot of meetings outside of working hours as well, making sure we are bonding as friends as well, not only as colleagues. As we said last year we used a lot of our brain power for this project, so the name should be taken as a joke!

Our customer requested a system where users can buy and sell things they do not use anymore. Users that want to buy items can go onto our website and pick something that they like, talk directly to the seller and maybe bargaining the price, making it a lot cheaper than buying a new one. Also, we should mention that we tried to follow one of the UN goals called Responsible Consumption and Production which is evidently implemented by us trying to reduce waste because most of the time “one man’s trash another man's treasure”.

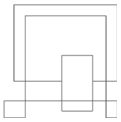
If we talk about our research we can say that we learned from last semester and tried to do a lot of it. All of us went on their favourite platforms that resemble our project and then documented everything while looking for common features. We started using UP at it’s fullest by doing Inception early on with all the data that we accumulated. We tried to make sure it fulfilled the requirements and our expectations. We didn’t forget about the SCRUM methodology that was given to us. We really liked it last semester, it made things so much easier and accessible so we decided to keep as close to it this semester as well. First, our Product Owner and SCRUM master were always talking to each over and over. When they would find compromises the SCRUM master would tell us what exactly we have to do which made the tasks that much easier. Our burndown charts show that we progressed confidently through our tasks. We tried our best to maintain a good communication with everyone, find common expectations and finish Inception without any conflicts.



4. Project Description

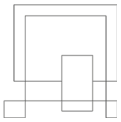
This time again, we are proud to say that our Project Description was a smooth sail. We are already experienced with working with a big project and a lot of tasks. Report Writing is already a stable for the 3rd semester already so we are happy that at least something remained the same during all the semesters, where we didn't find anything new. Learning from the mistakes from last semester was easier due to the harsh mistakes we made that blocked our way to success, we really took those lessons to heart and tried reimagining the report based on hard evidence from last semesters where our documentation was actually right.

We were happy with what we've written down as a first sketch of our Project Description, and we finished it quite early, at least we expected a project like this to take much more time than we took, but there is a saying that states "your eyes are afraid, but your hands do" with which we can entitle our first journey into the Project Description sphere. Next, we had a mentor meeting where we were told our Project Description had some small flaws but overall it was a success. This boosted our morale, because ultimately, we thought that we'll have the same headache with it as we did last semester. Last semester taught us that having vague expectations yet somewhat defined goals will get you far. Of course as we mentioned many times before SCRUM roles played a huge part here as well, while the Product Owner was very excited about the project, our SCRUM master tried to keep him in touch with reality and the actual things we can implement in such a limited time. Our supervisors made sure we're going down the right path which enhanced the developing experience. Before we knew it the Project Description was a success, we developed a UN related, Product Owner appreciated, SCRUM master approved, and supervisor liked piece of information that set our goals clearer and our expectations that were quite vague to something more prosperous.

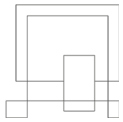


5. Project Execution

Project Execution this semester was more stressful than we expected. Even though we knew what we were doing and have done this multiple times before we still had problems understanding where to start and how to go through with our plans. Despite last years loss of morale, this year we didn't lose it throughout the whole project which is a win in our books. We were surely going into Elaboration phase, and nothing could stop us.
Elaboration wasn't a problem for us. We figured out fast what we want to do by finishing the Project Description, so modelling didn't take as long as we figured it will. Analysis and Design were thrown at us with a high velocity, but we managed it. Implementing something directly after designing it wasn't a foreign concept to us, so it was manageable. Another amazing SCRUM use is the fact that you can revisit your past progress and re-do some things that don't feel right. We had enough time for Testing some things in Implementation, but not a lot of them, because our project was built in a way where without a core component, item viewing, which was a critical requirement, we couldn't continue with our journey. It was quite a bummer that we couldn't see heaps of progress further on in Construction but we all agree that we did a lot of important work and the base we built was enough.
After finally nailing Elaboration, we continued to an unexpected Construction. Even though we tried to closely monitor every task at our meetings, even though we did some mentor meetings, and even though we tried to surgically develop our application things just didn't go our way. Some of the tasks we were given were very broad and we couldn't understand a lot of things. After a moment of weakness we manned up and started looking for responses from different resources we could find online, which helped us a ton. Our app started gaining colour, we finally had items and a working website that we could use to show them on. Later the same thing that we finally achieved will break and we will struggle with understanding why doesn't it work, but that's just a usual day of a Software Engineer. While still modelling the database and changing our type of database last moment we made a lot of progress by finding finally finding a way to authorize users, which surprisingly worked very well, we almost didn't have any problems with it. After creating the users and the items, and the ability to view items we though that we should start working on the messaging system inside our humble website. It was easier than we thought, which was unexpected as well. We first thought that it would be much harder to implement messages in the Data Access and Logic Tier, but we managed because we had bright minds in the team that thought of a full system that didn't have to be redone. It was easy to implement for the Presentation tier as well. We are proud of how much we've done in such a short span of time.
Testing was a big part of every sprint as well. We used White box and Black box testing to make sure our expectations would be met but at the same time try to find bugs for the new problems that might've appeared. We used bunit, x unit and Nunit.autowire for testing the presentation tier



which was quite harsh, and we used x unit for the Logic tier, which wasn't that bad. W tried our best to make Tests every sprint and check every new implementation
We were nearing getting into Transition, where we felt proud. A major part of our program was working. The Presentation tier was beautiful and working. The Logic tier didn't have any major bugs that would stop us from advancing with our endeavours and Data Access Tier was implemented well as well, so we didn't have a lot to fear besides the Tests that had to be redone because of the change in functionality we always implemented. Even though we did it in the Construction we still felt like we were changing things fast so we couldn't keep up the tests with the new versions, so we just waited a little until we knew that this was the last version. It is very important to mention that we tried hard to follow the UP Iterative Development, SCRUM roles, and approach to requirements so we had the best result by following exactly what we were told to do.



6. Personal Reflections

6.1. Tomas Gres

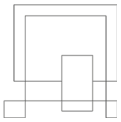
This semester has brought a lot of new information and knowledge which we the implemented in the semester project. We learned a lot about separating a system into multiple tiers, where each has its own purpose. We have also learnt about security and authorizations of users, to secure our applications even better. This semester was the hardest for me in terms of subjects and all the new information, that we had to implement immediately into assignments or projects. The thing I enjoyed learning about the most was Computer Architecture. It was interesting to learn how a computer actually works and how much effort had to be put in for us to have today's computers and smartphones.

For the semester project group, as I was a new member in an already well-oiled machine, I had to adjust to the new group. But I felt welcomed by my team members which has made it a lot easier. The group consisted of people from 4 different countries, but because we are all from Eastern European countries, we had a lot in common and a lot to talk about, so we had a very good chemistry. We had quite a few team-building events in form of trips or just meeting together and having fun. I believe we worked pretty well together and we managed to do considerate job on the project and all of our assignments.

We decided to repeat last year's development methods Scrum and Unified Process, as we all agreed that it is something we are very familiar with already and it was much simpler and easier to follow than first semester's Waterfall. We split requirements into smaller tasks and worked on that specific requirement until it was marked as done and then we moved on to the next one. That way it was much easier to see what was implemented and what was not and it brough us a little bit of satisfaction seeing a feature working.

As for the implementation itself, it was very challenging in the beginning to get all three tiers running. We had to use .NET and Blazor Web Assembly which we did not previously used, but there were very detailed tutorials which showed us how it is done and it was not as hard. However I myself had a hard time working on the front-end of the application, as I am more suited for back-end work, so anytime I had a task which included working on the presentation tier, I struggled, but I managed to finish my tasks every time. Another new technology we used was Spring Boot Framework. Although we were already familiar with Java, the framework itself was little harder to grasp, but we managed to do that as well.

We also decided to use a technologies which was just briefly explained in the class – JPA & Hibernate which was used for mapping java classes to database tables. For the middle ware technologies we used RESTful web API with HTTP protocol and Google's Remote Procedure Calls. Implementing REST API was not as challenging as we thought and we managed to make that work pretty easily. The hardest part of the project for me was implementing gRPC



communication. It took us multiple hours before we managed to make the service work how we intended.

6.2.Dan Sebastian Ceapa

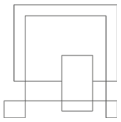
Because it's good to not know everything, I came to the realization during this project that even if we know some things and have some knowledge, we will always need to learn something new. How to connect the three tiers was, in my opinion, the most difficult aspect of this project (Presentation Tier, Business Logic Tier, and Data Access). I initially found the Adapter and DTO design patterns to be somewhat challenging, but as I gained knowledge and experience through the project and in class, it started to become less difficult.

We created a new group contract, checked to see that everyone was eager to work, and, of course, considered the fact that four of our group members had jobs. Initially, it was rather difficult to get into the work mindset, but after a few days, we grew accustomed to it. Every day, excluding weekends, we typically began at 10:00 AM.

It was best to follow UP and SCRUM in order to operate more productively. We were a little puzzled at first because we had the option to choose what we wanted to do this semester, but after some group discussion, we decided that we would go with the project, and the supervisors also assisted us in understanding exactly what we had to do. As we implemented SCRUM, our group was separated into several roles. I was given the role of Developer for this period, which presented a significant challenge for me. Throughout the assignment, I was able to better understand how SCRUM can benefit teams.

Even while I'm happy with the project's accomplishments, I realize there is still more work to be done to make it better.

On the one hand, we could communicate clearly and work effectively together. In order to handle some challenges effectively, we loved debating the issues at hand. Everyone was smiling and sharing some memes to brighten someone's day in a very welcoming environment. I can tell that there was a sense of harmony among us, and this makes it much easier to perform to the best of your ability. We also faced difficulties this semester because we had to work in five-person groups. It took us a while to find out how to work together, but eventually we were able to divide the work and support one another when we were having trouble. In the end, we had a good experience together, and I hope we will work together again in the upcoming semesters.



On the other hand, some coworkers share the same working hours, making it challenging in some cases to split projects so that everyone may manage to do them on time.

We tried to demonstrate this in our app since we truly liked it and created it with passion, and every team member contributed their all to ensure they knew they could handle it.

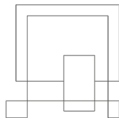
6.3. Matas Armonaitis

The project for this semester was both unique and similar to the one from the previous semester. The first semester was unsettling, thrilling, and fresh; by the second semester, I was more self-assured and somewhat aware of what was ahead. The only similarity is that I am still in the same group as I was during the previous semester, with the addition of a new member to our group as one of the differences. I thought it would be roughly the same as it was in the other project when I started this one, but it turned out to be much harder.

Starting with the similarities, this semester we continued to use SCRUM as our methodology, which made it simpler as I was already familiar with it. I functioned as the SCRUM master in this group once again, but I felt that I used this role considerably less than I did last semester because things like ensuring that the tasks are completed and that the team is effective in working on the project required much less interaction from me because everyone was motivated and completed their tasks without my help. This was also affected by the fact that we completed the majority of the project online while working from home, and it appeared to be quite successful for everyone as we could always contact one another if we needed assistance. This was also practical because it would be quite challenging to get everyone to meet at the same time because we all have jobs and work at various times.

Now about the things that were different. The fact that we are in the same group as last semester means that we already know each other and how we each work, which is a difference. It also helped that we were friends as well as co-workers because we would hang out after work and during breaks from university, which allowed us to work in a relaxed atmosphere without getting into arguments. Another difference was that the coursework was much harder than it was previous semester; instead of just applying fundamental concepts, we had to use complicated software.

The connecting the different tiers of our project was one of the most challenging aspects for me to comprehend. I myself found Java to be difficult to understand, so I delegated that task to my co-workers to ensure proper implementation. C# was the portion that was easiest for me to comprehend because I was eager to work with it.

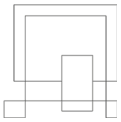


I primarily tested during the project and did some work in the presentation and business logic tiers. I initially believed that testing would be similar to last semester, but that was not the case. Since the system's implementation was much more complicated this semester, testing took significantly more study and time to complete. I used B Unit testing for presentation tier testing, which necessitated much research. I had to learn by doing when testing the business logic tier because I could not find any instructions on how to do it. This also required a lot of time, so I could not help my time much in other tasks unfortunately.

Even though we worked online the most of the semester, this year's teamwork was far better than it was. Each activity was assigned a rough time estimate, and we then completed the tasks ourselves. Even though we did not complete all of the user stories, most people were doing and finishing their tasks on schedule.
Morality of the team was like last semester awesome. Don't get me wrong, it was still quite difficult, especially at the beginning of the project, but we still managed to make fun of the challenging circumstances and maintain our motivation. Since we always talked English, including in our free time, there was no language barrier.

6.4. Chiril Luncasu

This semester was a bumpy ride. A lot of work had to be done and a lot of new things had to be learnt. We started off as usual with the sentence “We’ll do everything different from last semester”. When we started learning about the 3 tier architecture, I didn’t really understand it and everything looked really weird to me. There wasn’t anything I could base the new knowledge on, something I learned on my own before, which scared ma a little. When we started project description I understood that I will have to work a lot this semester, just because of the overwhelming tasks, and everything that looked really foggy. I would lie if I said the SEP period was a slow and annoying ride. It went by quite fast, which amazed me. A lot of things were done quick, and a lot of models were pushed quick. I myself did a lot of the models and Core concepts of our product, which I am very proud of, besides of course I completed several other tasks like messaging, notifications, main presentation tier layout, logic tier for items, diagram creations and report writing. I think that everybody gave their best when working on this project and everybody had a say when it came to coming up with new ideas. Also, I would add that I bonded even more with my teammates. Before we were already close, but now we saw each over thrive in our habitats. I like the fact that everybody took the tasks they thought they could complete and they sent them out as soon as possible for an amazing result. If I could go back I would surely do some things different, for example I would try to work more on the project because I feel like I could have implemented more things have I not tried to fix errors that weren’t as critical as I thought. But otherwise, I think I did well. There were a lot of things we had problems with, for example the connection between the Data Access tier and the Logic tier, some design issues with



Presentation tier, and a lot more small issues we needed to solve. Happily, we didn't have any major crashes that could make our project experience feel bad. So overall I would consider this semester as one with a big learning curve but very rewarding in the end. I would gladly go onwards with everything that I learned and I will try to manage my tasks better next time to get an even better result.

6.5. Dragos Daniel Bonaparte

I was wondering what new and exciting technologies we will learn throughout the course of this semester, and I got to say that my expectations were met. It was a real adventure to surf around so many technologies, get to create code that could be deployed and used as an actual system. A very cool way of motivating students to try harder and harder to exceed their limits. Everything was so confusing at the beginning, but as I got to leap deep and deeper in the technologies and in the documentation, I started to realize more and more that the programmers that created those technologies were geniuses.

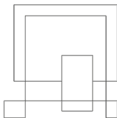
First, after trying to understand what that specific technology is supposed to do, I can say that it made me gain a deeper understanding and a logical view for the future when researching any other technology. Really helped me understand every step in our code execution and created a very logical and natural flow of execution throughout the whole system.

Our team followed UP and SCRUM again since it became so natural and also indispensable to us. Would like to mention that this semester our sprints were longer, which was a great advantage since all of us have a workplace and cannot always work at the same schedules. For example, for me it happened sometimes that when the sprints were starting in the middle of the week it was very difficult for me to finish my tasks since I had other required things to do on top of that such as an important meeting, work or so on, great improvement in our team's overall efficiency.

Also, a great improvement was that the team was focused more on dividing tasks, which made it possible for us to work a lot more independently and just compare parts of features together.

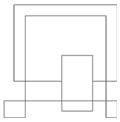
It was a great semester overall; everything was going well and smoothly. Every time I finish a project period, I have such a better understanding of all the topics, and after hand in I can already think about a better way on how to code the system such that we are more efficient and more engaged in the process.

Again, after such a project period I can say that I gave my all, everybody gave their all and despite anything that is what matters, I had fun coding this system, but more importantly had the fun together with my group.



7. Supervision

An important role in our group work was supervision. We really love the way our supervisor responded to our questions and the way he communicated how we should go through with some tasks. For example, in their emails they would always use a low-content approach and would be very direct which directly corelates with our cultural background. When we did meetings we got a very direct response that we really like, and a lot the suggestions felt really easy to implement just by the way they explained them, even though sometimes we would struggle last semester to even understand the teacher. We are very thankful for the aid they gave us, and for the inspiring attitude towards how we should approach our project.



8. Conclusion

In a nutshell, we would like to say that we liked this semester. The project we chose was manageable and entertaining at the same time. We tried our best to fulfil all the requirements, and we tried to bring a perfect product forward. Despite leaving out some requirements and not implementing everything perfectly, we made sure that the other parts are perfect, and we are proud of the final result. Also, a good thing we did is asking for supervision, and following the SCRUM roles with UP and the Agile manifesto in mind. If we could describe our group with one word we would say harmony, and if we could describe this project with one word, we would say initiative.