



# **Bilcon Web Application Project**

## **Final Report**

**CS 319**

**Section 1**

**Team 02 - 01**

Hakan Muluk

Gün Taştan

Begüm Kunaç

Mustafa Gökalp Gökdoğan

Öykü Demir

Mehmet Onur Özdemir

Instructor: Eray Tüzün

December 17, 2023

<b>Build Instructions</b>	<b>2</b>
<b>User Manual</b>	<b>4</b>
<b>Work Allocation</b>	<b>14</b>
Hakan Muluk	14
Design Process	14
Implementation	14
Gün Taştan	14
Design Process	14
Implementation (I worked in backend in the project)	15
Begüm Kunaç	15
Design Process	15
Implementation (I worked as a backend developer in the project)	15
Öykü Demir	16
Design Process	16
Implementation	16
Mehmet Onur Özdemir	17
Design Process	17
Implementation	17
Mustafa Gökalp Gökdoğan	17
Design Process	17
Implementation	18

# Build Instructions

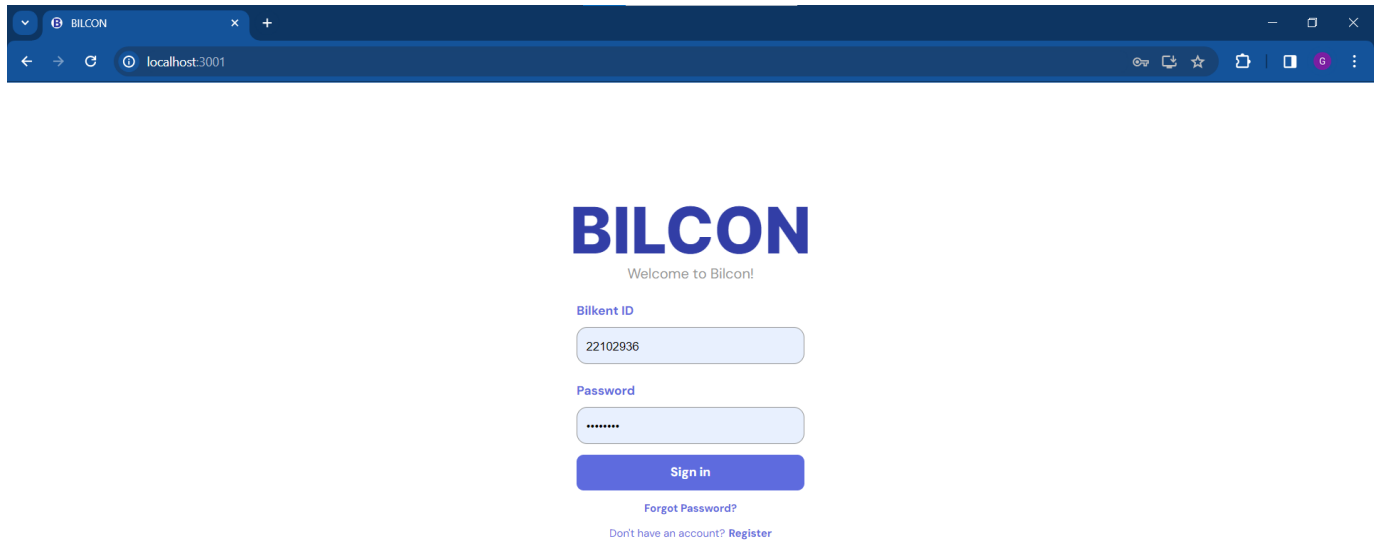
To build and run our application, one must install Node.js and MongoDB.

For building and executing Bilcon Web App:

1. Download the Windows installer from the Node.js® [web site](#) (Choose the LTS version that's shown on the left in the website).
2. Run the installer (the .msi file you downloaded in the previous step.)
3. Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings).
4. Restart your computer. You won't be able to run Node.js® until you restart your computer.
5. Confirm that Node has been installed successfully on your computer by opening a terminal and typing in the command `node --version`
6. To install MongoDB, install the msi package from their [official website](#). Here, note that version must be selected as 4.0.28 from the dropdown menu.
7. After hitting Download, click on the installed msi package.
8. Hit the NEXT button a few times until you see the download path which is written as Data Directory. Here you should note the path that is being downloaded.
9. After, hit the NEXT button until the installation finishes.
10. Now, go to your C Drive and create a file named data.
11. Then inside the newly created file, create a file named db.
12. Now, go to the path where MongoDB is installed (where you have noted in step 8) and from there go to the bin file.
13. Copy the directory of the bin and type "edit the system environment variables" to Windows search bar.
14. Then hit the Environment Variables button.
15. From the newly opened window, select Path for your current user and hit Edit.. button .
16. From the newly opened window, hit the New button and paste the path of the bin file that you have copied in step 13.
17. Now press OK buttons until all the windows have closed.
18. To check that MongoDB is installed correctly, go to CMD and type `mongo --version`
19. Having installed Node.js and MongoDB, clone the repository from GitHub to your local machine using the command `git clone` (or any other way that you are used to clone repositories).
20. Now open CMD, and type `mongod` in one terminal(window) and type `mongo` in another one. This will result in MongoDB listening on port 27017.

21. Now open Visual Studio Code and type `git checkout demo` to change your branch to demo.
22. Now open new terminal and navigate to backend directory by typing : `cd 'c:/Desktop/Bilcon/Bilcon/bilcon-backend-3/bilcon-backend'`  
Important Note: here “c:/Desktop/Bilcon” at the beginning of the command is where my local copy is located. Change and replace that part with the path where you have cloned the repository into.
23. After that type: `npm install --global nodemon`
24. Now to install other Node modules type: `npm i`
25. To run the backend of the project, type `nodemon main.js`
26. You should see the following message in terminal meaning that backend is running on <http://localhost:3000> : connected
27. Now open another terminal and navigate to frontend directory by typing: `cd 'c:/Desktop/Bilcon/Bilcon/bilcon-frontend'`  
Important Note: here “c:/Desktop/Bilcon” at the beginning of the command is where my local copy is located. Change and replace that part with the path where you have cloned the repository into.
28. To download required frontend modules, type: `npm i`
29. To run the frontend of the project, type: `npm start`
30. Type y to the question that is asked in this terminal.
31. Wait for your local machine to compile the code and open the browser for Bilcon Web App.
32. After waiting for a while, the website and application must be working.
33. To finish the execution of the application, type CTRL + C in both of the VSC terminals. Also type CTRL + C in the terminals that you have started running the MongoDB.

# User Manual



The screenshot shows a web browser window with the address bar displaying 'localhost:3001'. The page features the BILCON logo at the top, followed by the text 'Welcome to Bilcon!'. Below this, there are two input fields: 'Bilkent ID' with the value '22102936' and 'Password' with masked characters. A blue 'Sign in' button is positioned below the password field. At the bottom, there are two links: 'Forgot Password?' and 'Don't have an account? Register'.

**BILCON**  
Welcome to Bilcon!

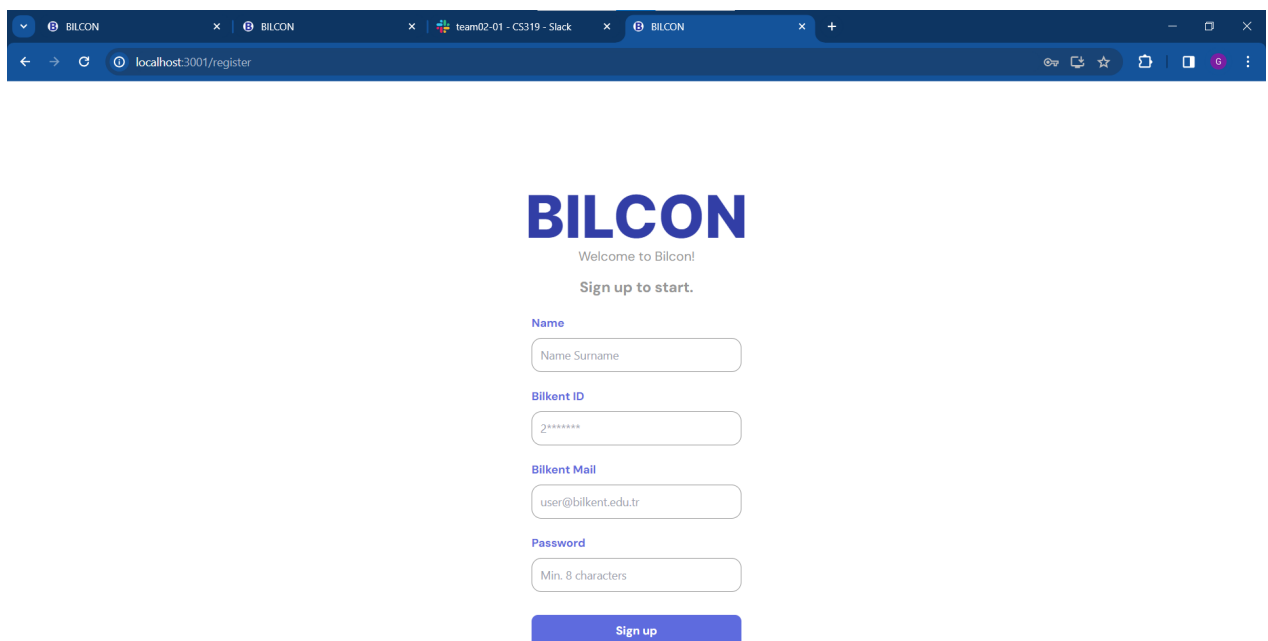
**Bilkent ID**  
22102936

**Password**  
\*\*\*\*\*

**Sign in**

[Forgot Password?](#)  
[Don't have an account? Register](#)

Users can log in to their account with the above log in screen. They should provide their credentials in order to log in. If they do not have an account, the user can create an account by clicking on Register. This takes the user to the registration page.



The screenshot shows a web browser window with the address bar displaying 'localhost:3001/register'. The page features the BILCON logo at the top, followed by the text 'Welcome to Bilcon!' and 'Sign up to start.'. Below this, there are four input fields: 'Name' with the placeholder 'Name Surname', 'Bilkent ID' with the value '2\*\*\*\*\*', 'Bilkent Mail' with the value 'user@bilkent.edu.tr', and 'Password' with the placeholder 'Min. 8 characters'. A blue 'Sign up' button is positioned below the password field.

**BILCON**  
Welcome to Bilcon!  
Sign up to start.

**Name**  
Name Surname

**Bilkent ID**  
2\*\*\*\*\*

**Bilkent Mail**  
user@bilkent.edu.tr

**Password**  
Min. 8 characters

**Sign up**

Users should provide their user information, then click on the link that is sent to their Bilkent mails in order to activate their accounts. The user cannot log in if their account is not yet verified.

If the user forgets their password, by clicking on Forgot Password? on the log in page, and providing their bilkent mails, they receive a password reset link which takes the user to a page which requires the user to provide a new password and then verify that password by typing it again. This process sets the password of the user to the new password.



# BILCON

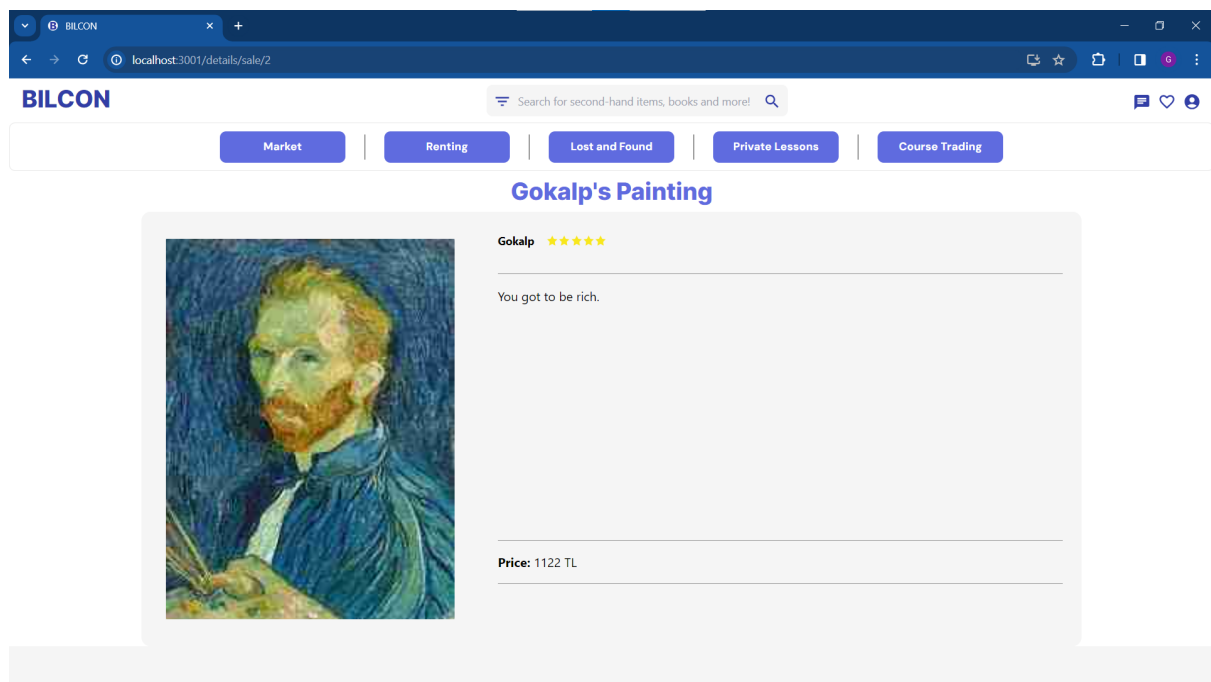
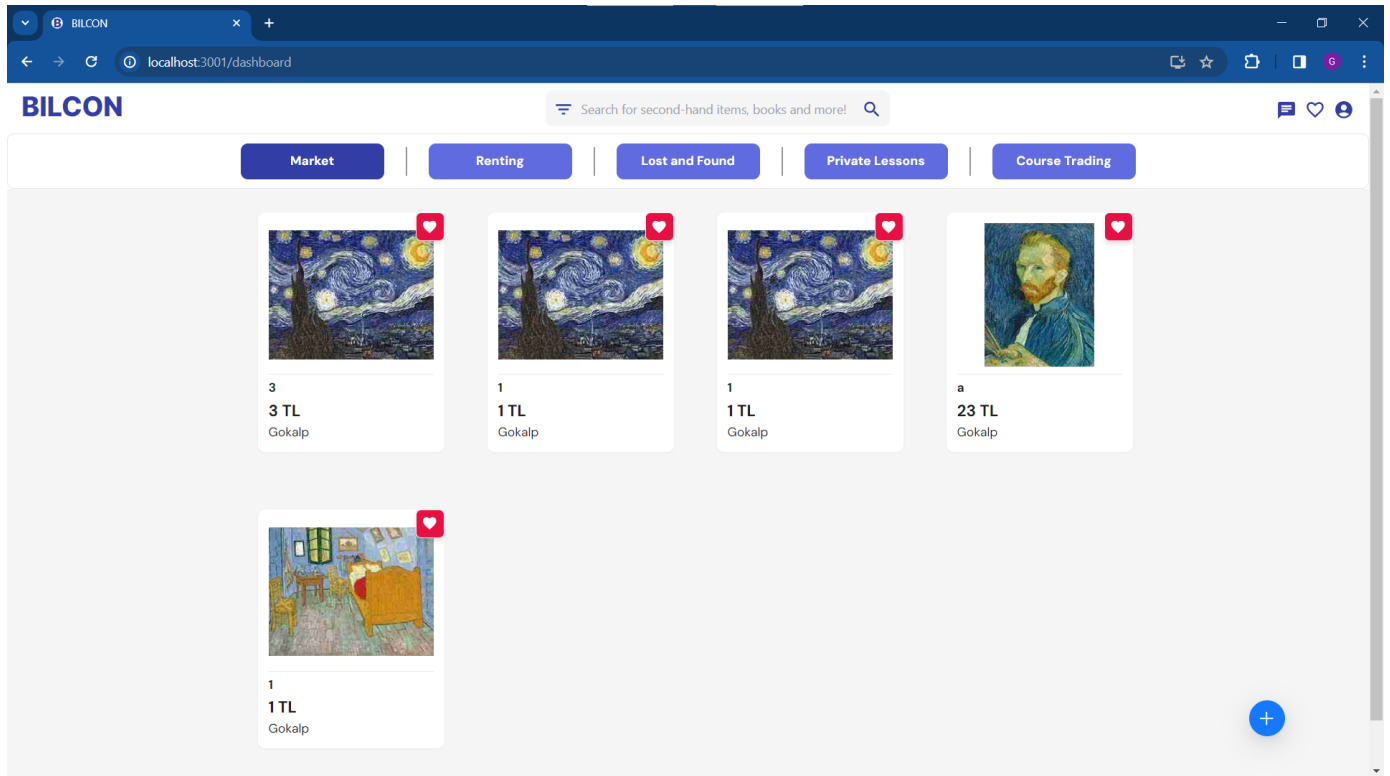
Welcome to Bilcon!

Please enter your school mail.

Bilkent Mail

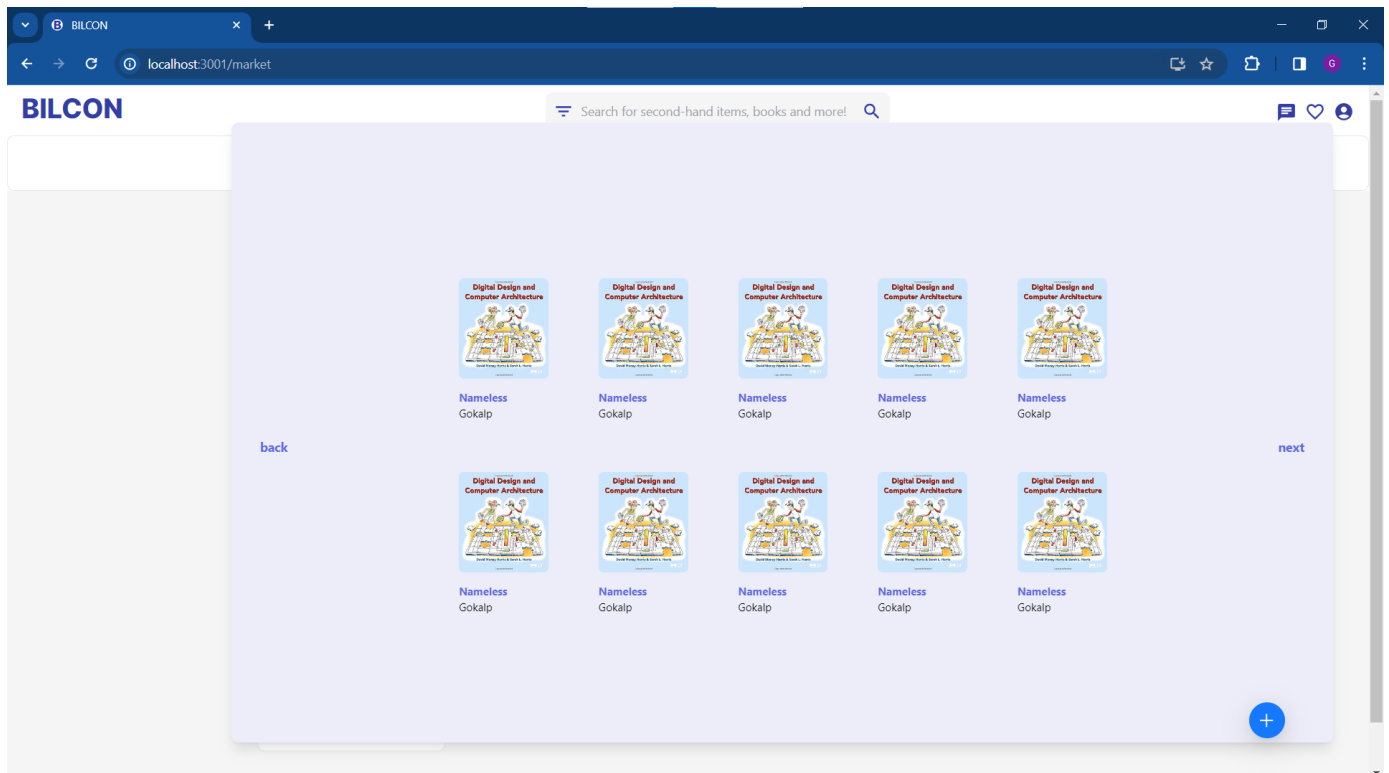
Send Mail

Users can view the items in different categories such as market, renting, lost and found, private lessons and course trading by clicking on different categories. If a user hits heart button that is on every items top right corner, that item will be added to users favorites list.

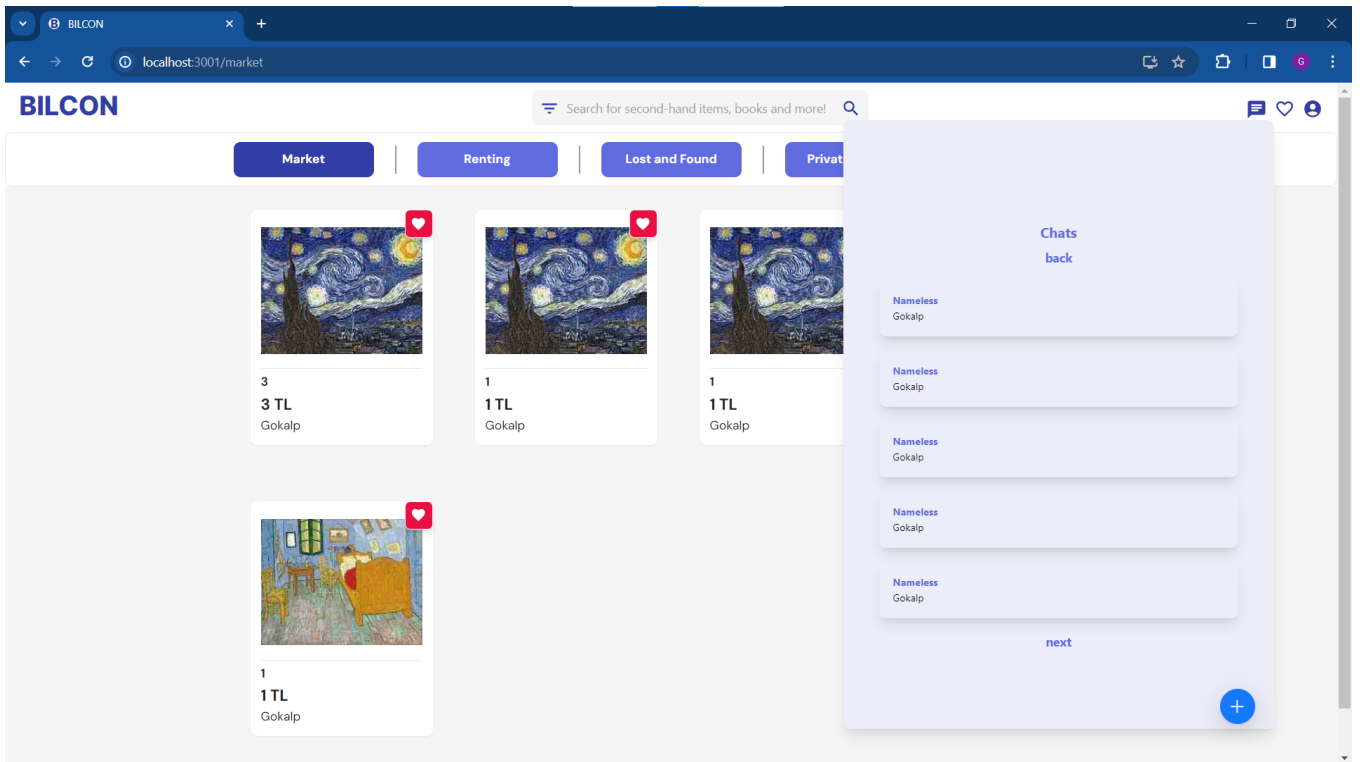


By clicking an item in the marketplace, users can view the details of the clicked item. In this product details page, users can see the owner of the item , alongside with his/her rating, description of the product and the price of the product.

Users have a favorites list, which consists of the items that users have marked as favorite by clicking on the heart. They can view their favorites list by clicking on the heart symbol which appears at the top right of the application.

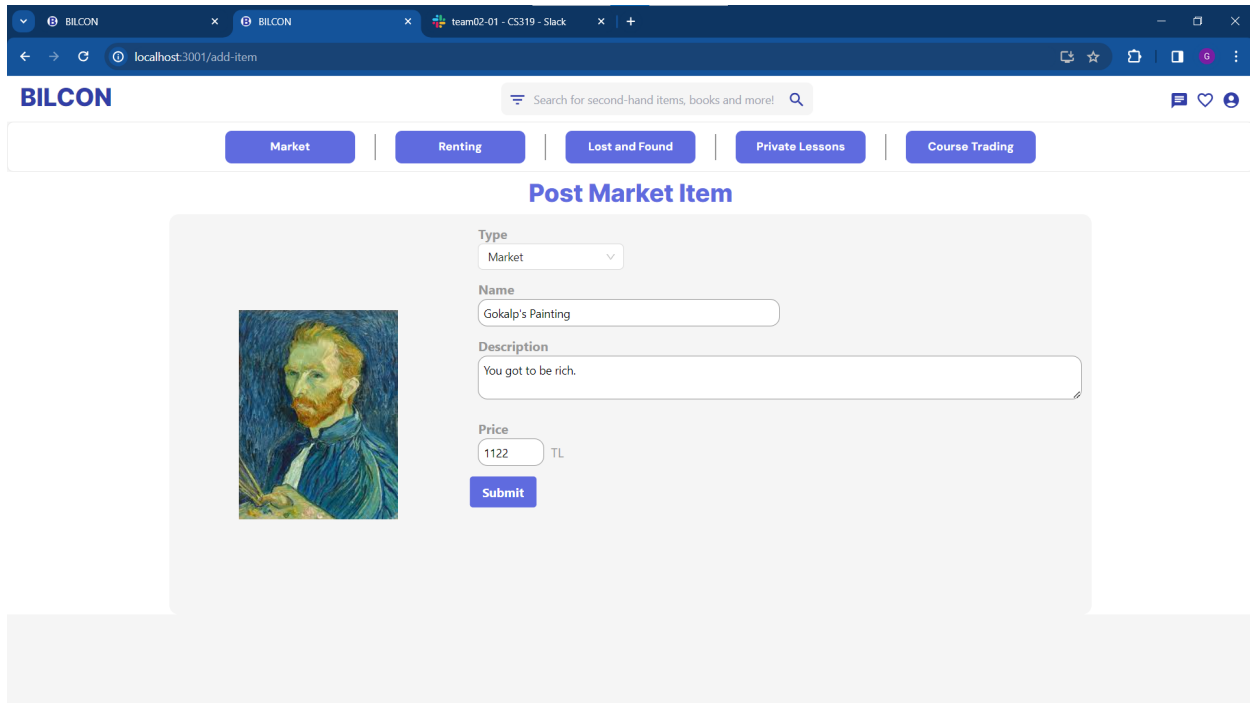






Messaged users and chats can be viewed as a “Chats” list by pressing the message box at the top right of the screen.

Users can post items by selecting its type and filling the necessary information about the product. For example for market items, users will fill name, description and price of the item as follows:



The screenshot shows a web browser window with the BILCON application. The browser tabs include 'BILCON', 'team02-01 - CS319 - Slack', and another 'BILCON' tab. The address bar shows 'localhost:3001/add-item'. The BILCON logo is in the top left, and a search bar with the text 'Search for second-hand items, books and more!' is in the top right. Below the search bar is a navigation bar with five buttons: 'Market', 'Renting', 'Lost and Found', 'Private Lessons', and 'Course Trading'. The 'Market' button is highlighted. Below the navigation bar is the title 'Post Market Item'. The form itself is a light gray box with a white background. It contains a 'Type' dropdown menu with 'Market' selected. To the left of the form is a placeholder image of a painting, which is a reproduction of Vincent van Gogh's 'Self-Portrait with Bandaged Ear'. Below the image is a 'Name' input field with the text 'Gokalp's Painting'. Below the name field is a 'Description' input field with the text 'You got to be rich.'. Below the description field is a 'Price' input field with the text '1122' and a 'TL' label. Below the price field is a blue 'Submit' button.

**BILCON** Search for second-hand items, books and more!

Market Renting Lost and Found Private Lessons Course Trading

### Post Market Item

Type  
Market

Name  
Gokalp's Painting

Description  
You got to be rich.

Price  
1122 TL

Submit

For posting renting items, users should determine price of the item alongside with its rent duration as follows:

The screenshot shows a web browser with the URL `localhost:3001/add-item`. The page header includes the BILCON logo, a search bar with the text "Search for second-hand items, books and more!", and navigation buttons for Market, Renting, Lost and Found, Private Lessons, and Course Trading. The main heading is "Post Renting Item". The form contains a "Type" dropdown set to "Renting", a "Name" field with "Gokalp's Painting", a "Description" field with "You got to be rich.", a "Price" field with "111" and a "TL" label, and a "Rent Duration" field with "12" and a "day" label. A "Submit" button is at the bottom.

**BILCON** Search for second-hand items, books and more!

Market Renting Lost and Found Private Lessons Course Trading

### Post Renting Item

Type: Renting

Name: Gokalp's Painting

Description: You got to be rich.

Price: 111 TL

Rent Duration: 12 day

Submit

For posting lost items, the user who lost the item should specify the location and time he/she lost the item.

The screenshot shows the same BILCON web application but with the "Post Lost Items Item" form. The "Type" dropdown is set to "Lost Items". The "Name" field is "Gokalp's Painting" and the "Description" field is "You got to be rich.". The "Info" section includes a "Location" field with "Bilkent" and a "Date" field with "14 / 03 / 2024". A "Submit" button is at the bottom.

**BILCON** Search for second-hand items, books and more!

Market Renting Lost and Found Private Lessons Course Trading

### Post Lost Items Item

Type: Lost Items

Name: Gokalp's Painting

Description: You got to be rich.

Info

Location: Bilkent

Date: 14 / 03 / 2024

Submit

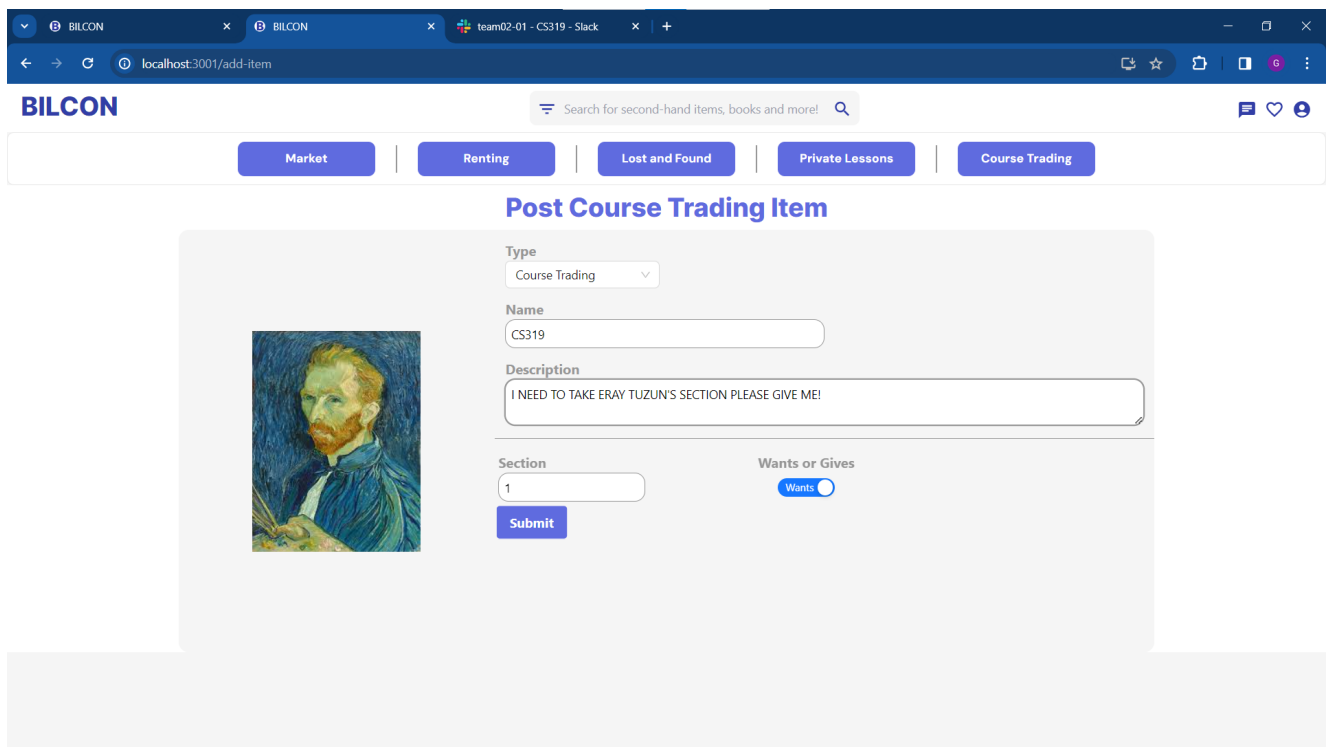
For posting found items, the user who found the item should specify the location and time he/she found the item.

The screenshot shows a web browser with the URL `localhost:3001/add-item`. The page has a header with the BILCON logo and a search bar. Below the header is a navigation bar with buttons for Market, Renting, Lost and Found, Private Lessons, and Course Trading. The main content area is titled "Post Found Items Item" and contains a form. The form has a "Type" dropdown menu set to "Found Items". There is a "Name" input field with the text "Gokalp's Painting" and a "Description" input field with the text "You got to be rich.". To the left of the form is a placeholder image of a painting. Below the form is an "Info" section with "Location" (Bilkent) and "Date" (14 / 03 / 2024) input fields. A "Submit" button is at the bottom of the form.

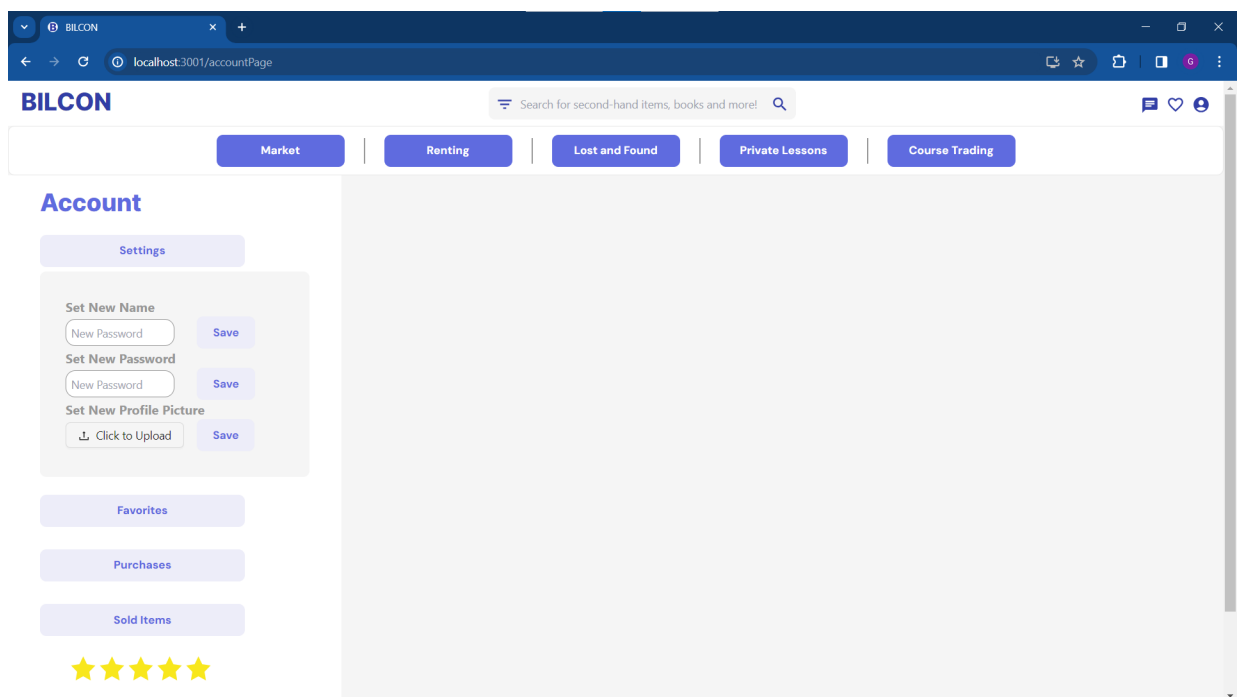
For posting private lesson items, users should specify the price per hour of the lesson they offer.

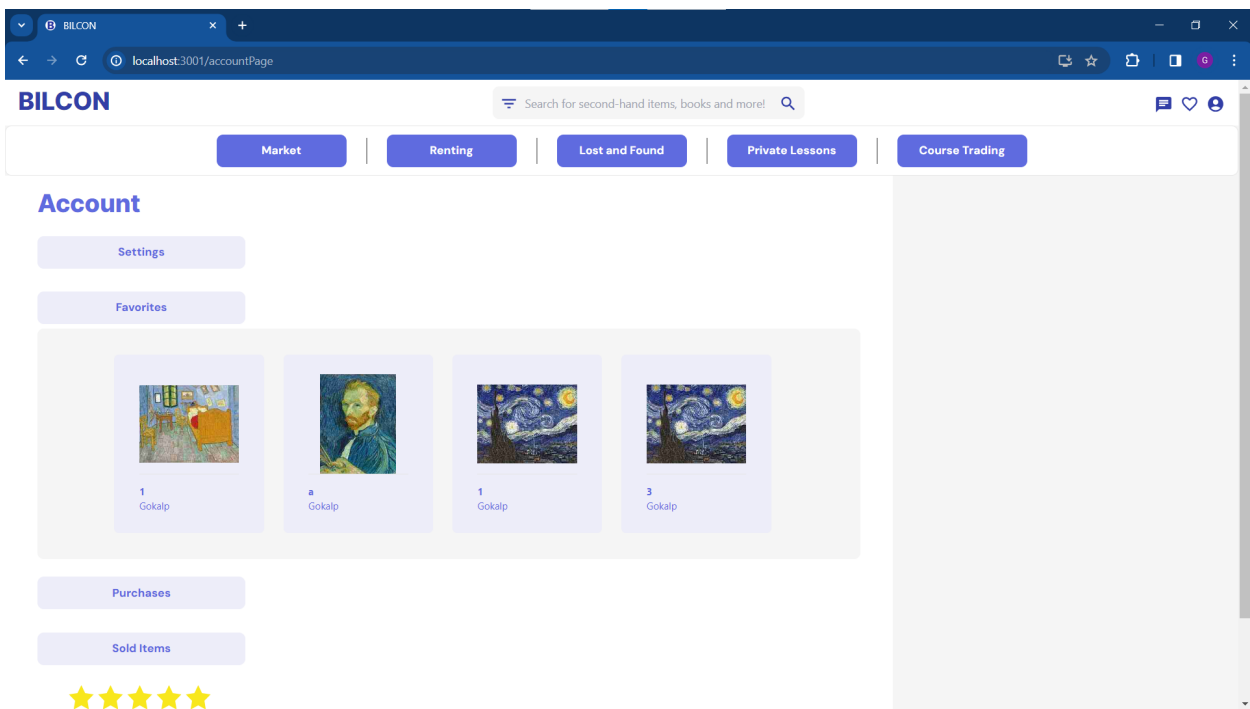
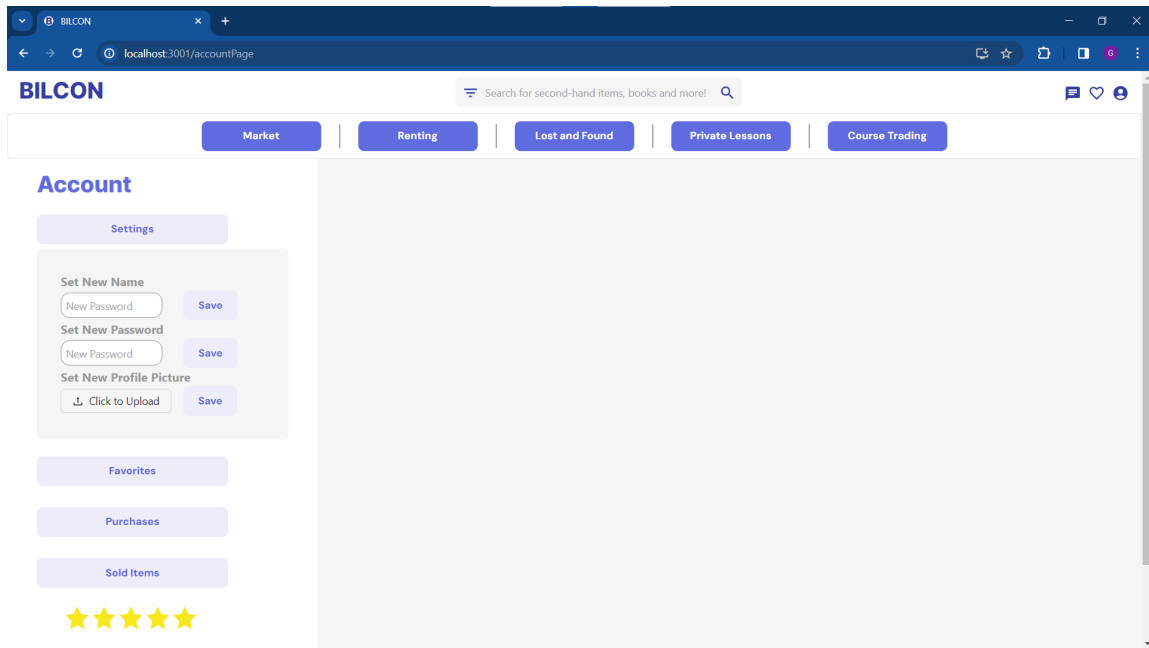
The screenshot shows the same web browser with the URL `localhost:3001/add-item`. The page has the same header and navigation bar. The main content area is titled "Post Private Lessons Item" and contains a form. The form has a "Type" dropdown menu set to "Private Lessons". There is a "Name" input field with the text "Gokalp's CS319 Lesson" and a "Description" input field with the text "wow". To the left of the form is a placeholder image of a painting. Below the form is a "Price" input field with the text "35" and a label "TL/hour". A "Submit" button is at the bottom of the form.

For course trading items, users should specify the name of the lesson they want, whether they want to get or give the lesson and the section that they want or give.



Users can see their own account page with details such as favorites list, purchases, sold items and current rating. From this screen, users can also change their password.





# Work Allocation

## Hakan Muluk

### Design Process

- Wrote some use cases' explanations, as different sections in D1.
- Created a sequence diagram and its explanation(including the scenario) in D2.
- Co-created the class diagram for both application and solution domains in D2 & D5.
- Wrote some of the Design Goals in D4.
- Co-wrote the Access Control and Security part. Also, co-created the Access Control Matrix in D4.
- Wrote some of the subsections of Boundary Conditions part in D4.
- Wrote one of the Design Patterns in D5.
- Actively contributed to every aspect of the project, including design, quality improvements, user experience enhancements, backend demos, system critical updates, and the merging and verification process.

### Implementation

- Implemented some parts of the User, UserDb and UserController classes.
- Implemented the Poster, Customer, PosterDb, CustomerDb, PosterController and CustomerController classes.
- Implemented the Item class and their subclasses which are SaleItem, RentItem, LostItem, FoundItem, PrivateLessonItem and CourseItem. Also the database module classes for these: SaleItemDb, RentItemDb, LostItemDb, FoundItemDb, PrivateLessonItemDb and CourseItemDb. Lastly, the ItemController interface and the children of the interface, which are SaleItemController, RentItemController, LostItemController, FoundItemController, PrivateLessonItemController and CourseItemController.
- Implemented different routes in main.js(the API) by using the classes that are mentioned above.
- Has done testing & debugging by using Postman.

## Gün Taştan

### Design Process

- Co-created and designed the Use Case Diagram for D1 in Visual Paradigm.
- Co-created and designed the Activity Diagram for D2 in VisualParadigm.
- Explained the Activity Diagram in report in D2.

- Co-created and designed the Sequence Diagram for D2 in VisualParadigm.
- Also took part in the Class Diagram for D2.
- Co-created the Subsystem Decomposition in High Level Architecture for D4.
- Co-explained what happens in each layer and explained their connections in D4.
- Co-created, designed and drew the detailed Class Diagram for D5.
- Co-created the build instructions and user manual for D6.
- Co-created the trailer of the project.
- Actively contributed to every aspect of the project, including design, quality improvements, user experience enhancements, backend demos, system critical updates, and the merging and verification process.

Implementation (I worked in backend in the project)

- Implemented some parts of User, UserDb and UserController classes (written classes, methods and database schemas).
- Implemented various routes and requests in main.js. These are then handled by the UserController class. These includes the following
- Implemented register functionality where it included sending verification mail, and verifying the user. Here, we used JWT, bcrypt and salting/hashing techniques and modules. Responsible for safely securing data in the database.
- Implemented login and session based authentication. Used cookies for creating sessions.
- Implemented logout route, which was responsible for maintenance and destroying the session.
- Implemented getting account page which can be seen in main.js
- Implemented change and forgot password routes and handled them.
- Implemented submitting transactions with TransactionDb and TransactionController classes (handling bought and sold items)
- Implemented rating system which can be seen in main.js
- Implemented requestContact route which can be seen in main.js
- Done testing and debugging by using Postman.

## **Begüm Kunaç**

Design Process

- Co-created the Use Case Diagram using VisualParadigm for D1.
- Co-created and designed the Class Diagram using VisualParadigm for D2.
- Co-created the Activity Diagram using VisualParadigm and gave the explanation of the diagram for D2.
- Took a part in the Sequence Diagram using VisualParadigm for D2.
- Co-created the Subsystem Decomposition for D4.



- Co-wrote the explanations for each layer of the Subsystems for D4. Explained what happens in each layer and explained their connections.
- Co-created and designed the detailed Class Diagram using VisualParadigm for D5.
- Co-created the build instructions and user manual for D6.
- Co-created the trailer of the project.
- Actively contributed to every aspect of the project, including design, quality improvements, user experience enhancements, backend demos, system critical updates, and the merging and verification process.

Implementation (I worked as a backend developer in the project)

- Implemented some parts of User, UserDb and UserController classes (written classes, methods and database schemas).
- Implemented various routes and requests in main.js. These are then handled by the UserController class. These includes the following
- Implemented register functionality which includes sending verification mail, and verifying the user. Here, we used JWT, bcrypt and salting/hashing techniques and modules. Responsible for safely securing data in the database.
- Implemented login and session based authentication. Used cookies for creating sessions.
- Implemented logout route, which was responsible for maintenance and destroying the session.
- Implemented getting account page which can be seen in main.js
- Implemented change and forgot password routes and handled them in the UserController class.
- Implemented submitting transactions (handling bought and sold items) with TransactionController class and TransactionDb.
- Implemented rating system of the user which can be seen in main.js, and the UserController class.
- Implemented requestContact route which is handled in the UserController class, by sending mails to users when there is a contact request. This can be seen in main.js
- Utilized from Postman for testing and debugging.

## Öykü Demir

Design Process

- Took active part on deciding on the tech stack of the project.
- Co-created the Non-Functional Requirements in D1.
- Wrote the explanations for each Use Case in the Use Case Diagram of D2.
- Created Mockups for D2 as well as the User Interface Design of the application.

- Co-created Design Tradeoffs in D4.
- Wrote Hardware/Software Mapping, Persistent Data Management and Access Control and Security parts of D4.
- Actively contributed to every aspect of the project, including design, quality improvements, user experience enhancements, backend demos, system critical updates, and the merging and verification process.

## Implementation

- Created Requests.js class that performs axios requests for the frontend.
- Mainly responsible for frontend, worked actively in both client and server sides of the application.
- Created all product components in ./screens/products that are compatible with the input parameters.
- Co-created Feed, and connected filtering and search methods of backend to it. Passed arguments to each page and header according to the filter and search values.
- Connected the backend of the application with the frontend.
- Co-implemented each page of the application in line with the User Interface Design
- Implemented Chat connection with the backend. Called sendMessage and getConversation methods.
- Implemented database and backend connection of login and register.
- Implemented navigation between screens in register and login according to sessions.
- Wrote redirections in case of session ending in backend and implemented them on frontend according to the response.
- Connected addItem to the backend. Created the required request body according to it with parameters.
- Wrote ProductList and BlogList classes for better pagination in feed.
- Wrote custom Upload component and set post requests according to it.

## Mehmet Onur Özdemir

### Design Process

- Wrote some use cases and their logical explanations as partitioned within the group in D1.
- Created Sequence Diagrams along with their explanations in D2, the sequence diagrams were custom made upon a given scenario but were functionally generalizable.
- Co-created Class Diagram in D2 & D5.
- Co-wrote Design Goals in D4, specifically the goal of maintainability.
- Co-wrote Access Control & Security in D4.

- Wrote Boundary Conditions and its State Diagram in D4.
- Co-wrote Design Patterns in D5.
- Actively contributed to every aspect of the project, including design, quality improvements, user experience enhancements, backend demos, system critical updates, and the merging and verification process.

#### Implementation

- Implemented and designed the structure of User, UserDb, UserController Classes.
- Implemented the MessageController, MessageDb, and Main classes for purposes of chatting within the application.
- Implemented different routes for the application, for the frontend partition of the project.
- Implemented a chatContext class for possible use of the frontend, using WebSockets and frontend logic such as dependency arrays within different use effects.
- The return values of the chatContext class involved attributes like messages, currentChat, creating chats, sending text messages, returning user's chats for the possible use of react components.
- Implemented the backend structure of the user and possible conversation attributes between 2 different users.
- Did testing & debugging & confirming the functionality using Postman.

### **Mustafa Gökalp Gökdoğan**

#### Design Process

- Co-created the Class Diagram for D2
- Explained the classes and their functionality further for D2
- Wrote the Introduction for D2.
- Coded our first prototype for D3 with Öykü.
- Explained Design Goals for D4.
- Designed, drew Subsystem Decomposition for D4.
- Drew and explained the Deployment Diagram for D4.
- Wrote Design Trade-offs for D4.
- Actively contributed to every aspect of the project, including design, quality improvements, user experience enhancements, frontend demos, system critical updates, and the merging and verification process.

#### Implementation

- I have created Account, Details, Forgot Password, Verification and Feed(Market, Second Hand, Renting...) Pages.

- I have developed pop-up error messages component.
- I have created a grid system to show products in feed.
- I have created Product, Header, Feed and Navigation Menu components.
- Enhancements to the header, including features like Chats and Fav updates, FavPop component, and improvements in the FilterView component.
- Several commits dedicated to enhancing the overall quality and user experience, addressing issues such as input control for login and registration screens and implementing stability improvements for the login process.
- Implementation of frontend features, including the addition of FloatButton and Item Page components.
- Commits associated with merging pull requests, verifying changes, and ensuring the smooth integration of features and fixes.