CS102	**Spring 2022/23**	Project Group	S3G3
Instructor:	**Uğur GüdükBay**		
Assistant:	**Sena Er**		

Criteria	TA/Grader	Instructor
Presentation		`
Overall		

~BAHÇEDEN~

SWIFTIES

Mehmet Akif Şahin, Mehmet Anıl Yeşil, Erfan FarhangKia,
Emir Ensar Sevil, Şükrü Eren Gökırmak

Detailed Design Report

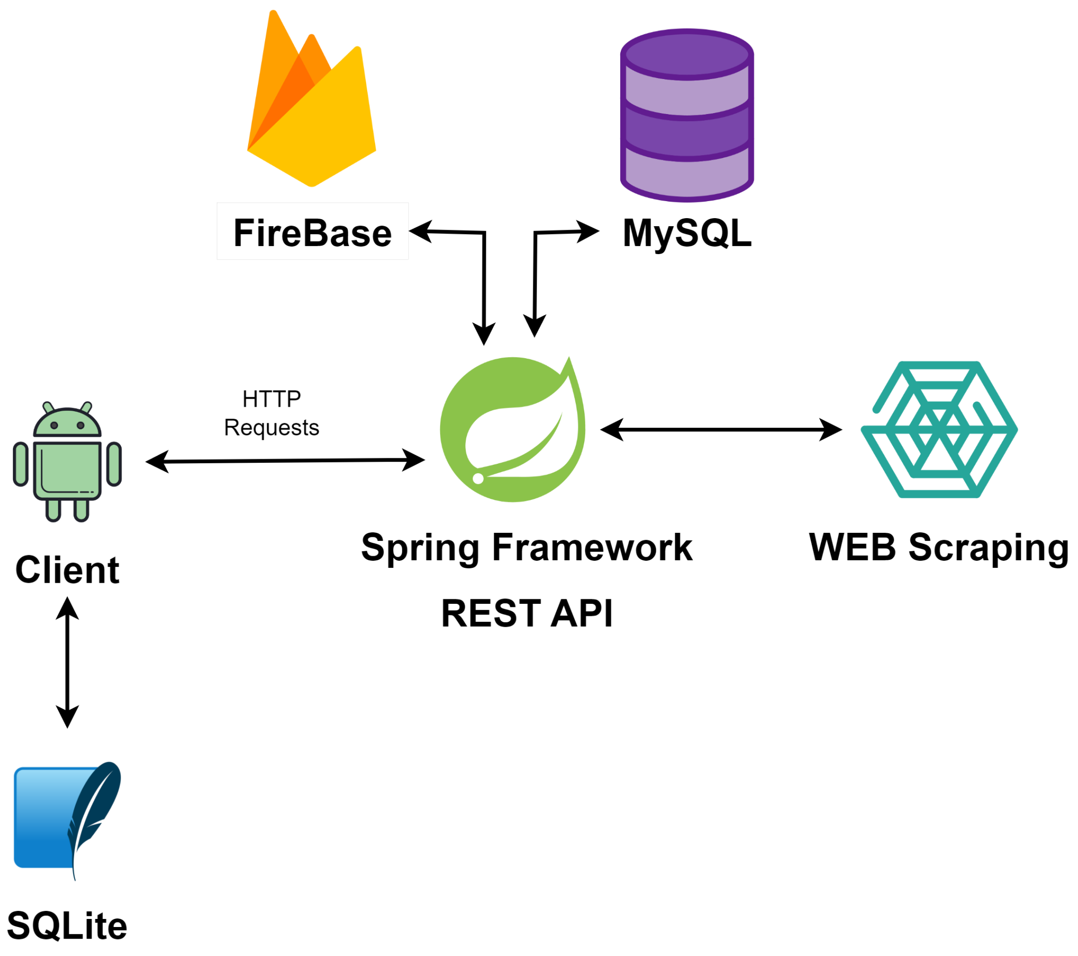
(version 2.0)

03 June 2023

1. Introduction

Bahceden is an innovative e-commerce Android application that eliminates intermediaries between local producers and customers. The traditional system often sees intermediaries buying products at low prices from local producers and then selling them to customers at inflated rates. Bahceden aims to prevent this exploitation by providing a platform for local producers to sell their goods directly to consumers, empowering customers to access healthier and more affordable products. To further assist both parties, Bahceden offers an intelligent price prediction system and up-to-date market data, enabling producers to make informed pricing decisions when listing their products on the app. Experience a transparent and fair marketplace with Bahceden, focusing on building sustainable relationships between local producers and customers.

2. System Overview



2.1 Frontend

Android Studio

The user interface and frontend development of the Bahçeden Android e-commerce app will be implemented using Android Studio, a robust integrated development environment (IDE) designed explicitly for Android application development. Android Studio provides a comprehensive set of tools and resources that enable us to create an intuitive and visually appealing user interface for our e-commerce app. The main aim will be leveraging Android's native UI components, such as layouts and views, to design screens that align with the overall branding and user experience goals of Bahçeden. These components will be customized to showcase products, categories, user profiles, and shopping cart functionalities, providing an engaging experience for our app users. Android Studio's layout editor will design XML-based layout files visually, defining the structure and positioning of UI elements. Bahçeden will use various UI components, such as RecyclerViews, to enhance the app's usability and navigation flow. Additionally, it will incorporate appropriate animations and transitions to create a visually pleasing experience and provide user feedback. By utilizing Android Studio for the user interface and frontend development of the Bahçeden Android e-commerce app, we can leverage the powerful tools and resources provided by the IDE to create a visually appealing and user-friendly application. Integrating Android's native UI components will result in a high-quality app that provides a delightful shopping experience for our users.

Retrofit Library

Retrofit offers a high-level API that abstracts the complexities of network communication, allowing developers to focus on defining API endpoints and handling response data. With Retrofit, developers can create interfaces with annotated methods representing HTTP verbs and URL paths, letting Retrofit handle the rest. Bahçeden used Retrofit in its front end to easily build a bridge between the front end and the backend of the project by utilizing HTTP requests. These requests were used to transfer, manipulate, and transmit data from the application’s database to the user’s device, and vice versa.

Firebase

Firebase offers a comprehensive range of authentication services that can be seamlessly integrated into the Bahçeden e-commerce application. Bahçeden will incorporate Firebase for implementing sign-in and sign-up authentication features. Leveraging this technology will allow users to explore alternative authentication methods, such as Google Sign-In, allowing them to conveniently sign in using their existing Google accounts. This will enhance the user experience by eliminating the need for creating separate credentials. Firebase's analytics and monitoring capabilities will track authentication events and measure performance. This will enable Bahçeden to ensure a smooth and secure user experience within our e-commerce app.

2.2 Backend

Spring Framework

The Spring Framework is used to build a REST API for Bahçeden. The primary goal of this API is to provide a connection between the backend server and the frontend system. The API is designed to handle CRUD requests made by the user, allowing easy data manipulation within the databases. By pulling data from the front end and updating the corresponding records in the database, the API ensures that the data remains up-to-date and accurate at all times. The API is also responsible for retrieving data from relevant databases and passing it back to the front end for various tasks. These tasks include data visualization and statistical analysis. By enabling the front end to access and interact with data from the backend systems easily, the API helps to streamline workflows and improve productivity. By providing robust CRUD functionality and facilitating data transfer between systems, the REST API will ensure that the system operates smoothly and efficiently while providing flexibility for future enhancements and features.

Lombok Library

Lombok is a Java library that helps reduce the amount of repetitive code in Java projects. By adding annotations to Java classes, Lombok automatically generates common code constructs such as getters, setters, constructors, and toString methods. This eliminates the need for developers to write boilerplate code, making the codebase more concise and easier to maintain. Bahçeden used Lombok to make the development of the backend easier by eliminating boilerplate code.

2.3 Data

MySQL

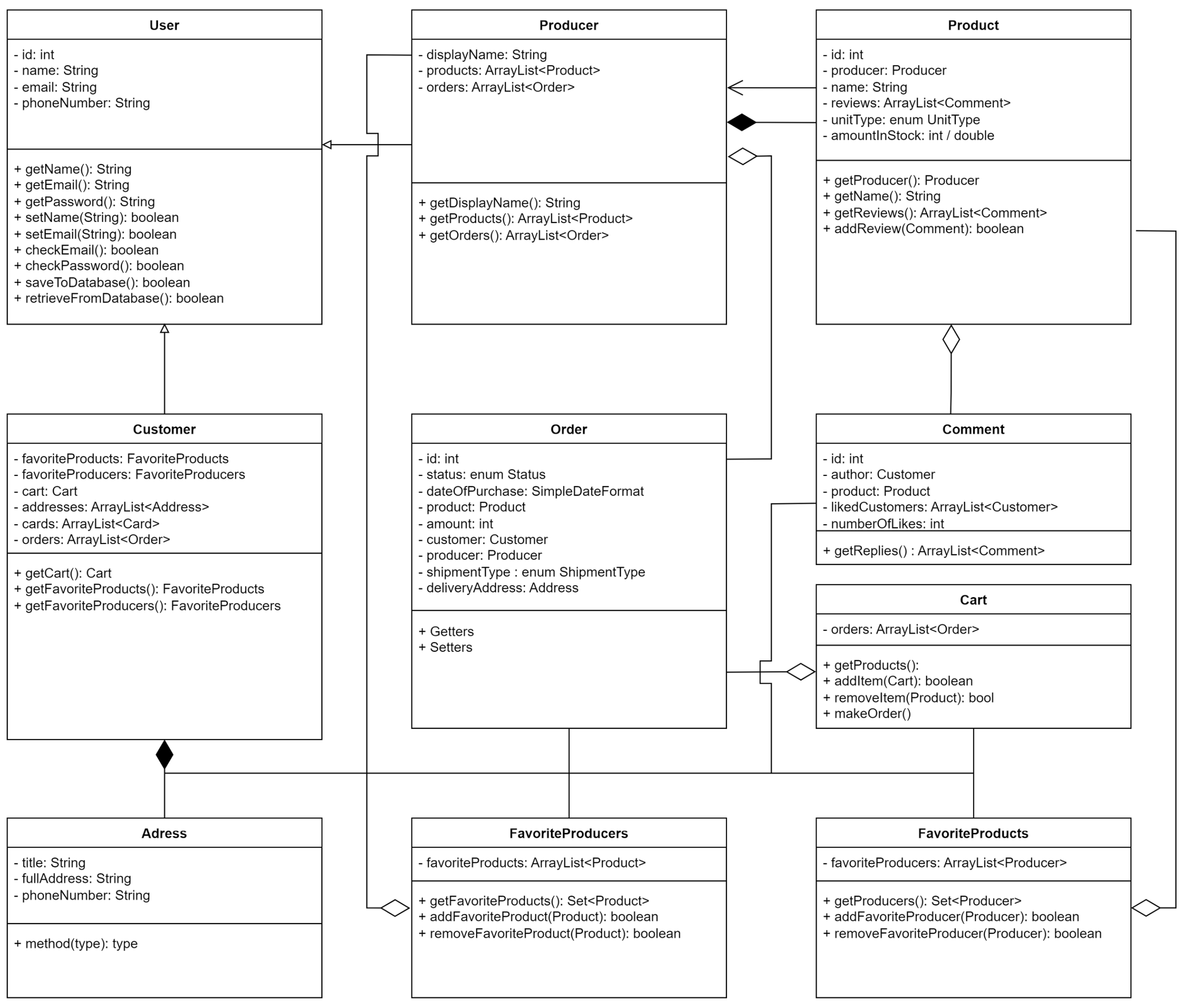
In our Android e-commerce app, Bahçeden, we chose MySQL as the database management system to store crucial data. We will employ a REST API architecture to facilitate app and database communication. The Spring Framework will be utilized to develop the REST API, ensuring efficient handling of HTTP requests and responses. This approach allows for a decoupled system architecture where the Android app communicates with the server-side through RESTful endpoints. The MySQL database will be designed to accommodate the specific requirements of the Bahçeden e-commerce app. We will define appropriate database tables to store product information, including their names, descriptions, prices, and availability. Similarly, tables will be created to store the producer’s details, such as shop names, locations, and contact information. Additionally, user-related data, like user profiles, order history, and authentication credentials, will be securely stored in the database. Leveraging Spring Data JPA will connect the MySQL database and the REST API seamlessly, enabling efficient CRUD (Create, Read, Update, Delete) operations.

Jsoup and Selenium Libraries

Our project's primary objective of web scraping is to gather and organize data from various online grocery stores and wholesale market halls, enabling us to categorize specific products our app's producers will offer for sale. Our approach employs machine learning algorithms or information theory approaches to predict pricing for our producers. To achieve this, we utilized Jsoup for parsing static HTML and navigating through static web pages while relying on Selenium to obtain the source code of dynamically generated JavaScript pages and traverse them effectively.

3. Core Design Details

3.1 Model Classes and Interfaces


User
- id: int
- name: String
- email: String
- phoneNumber: String
+ getName(): String
+ getEmail(): String
+ getPassword(): String
+ setName(String): boolean
+ setEmail(String): boolean
+ checkEmail(): boolean
+ checkPassword(): boolean
+ saveToDatabase(): boolean
+ retrieveFromDatabase(): boolean
Producer
- displayName: String
- products: ArrayList<Product>
- orders: ArrayList<Order>
+ getDisplayName(): String
+ getProducts(): ArrayList<Product>
+ getOrders(): ArrayList<Order>
Product
- id: int
- producer: Producer
- name: String
- reviews: ArrayList<Comment>
- unitType: enum UnitType
- amountInStock: int / double
+ getProducer(): Producer
+ getName(): String
+ getReviews(): ArrayList<Comment>
+ addReview(Comment): boolean
Customer
- favoriteProducts: FavoriteProducts
- favoriteProducers: FavoriteProducers
- cart: Cart
- addresses: ArrayList<Address>
- cards: ArrayList<Card>
- orders: ArrayList<Order>
+ getCart(): Cart
+ getFavoriteProducts(): FavoriteProducts
+ getFavoriteProducers(): FavoriteProducers
Order
- id: int
- status: enum Status
- dateOfPurchase: SimpleDateFormat
- product: Product
- amount: int
- customer: Customer
- producer: Producer
- shipmentType : enum ShipmentType
- deliveryAddress: Address
+ Getters
+ Setters
Comment
- id: int
- author: Customer
- product: Product
- likedCustomers: ArrayList<Customer>
- numberOfLikes: int
+ getReplies() : ArrayList<Comment>
Cart
- orders: ArrayList<Order>
+ getProducts():
+ addItem(Cart): boolean
+ removeItem(Product): bool
+ makeOrder()
Adress
- title: String
- fullAddress: String
- phoneNumber: String
+ method(type): type
FavoriteProducers
- favoriteProducts: ArrayList<Product>
+ getFavoriteProducts(): Set<Product>
+ addFavoriteProduct(Product): boolean
+ removeFavoriteProduct(Product): boolean
FavoriteProducts
- favoriteProducers: ArrayList<Producer>
+ getProducers(): Set<Producer>
+ addFavoriteProducer(Producer): boolean
+ removeFavoriteProducer(Producer): boolean

User is an abstract class representing users using the app, storing their id, name, email, and phone number. Customer and Producer classes will extend the User class.

The Customer class will represent a customer using the app, storing their cart information, orders, addresses, favorite products, and producers. This class will have a composition relationship with the classes Order, Comment, Cart, Address, FavoriteProducts, and FavoriteProducts.

The Producer class will represent a producer in the app, storing their display name, orders, and listed products. This class will have a composition relationship with the class Product and will have an aggregation relationship with the class Order.

Product class will represent an item listed by a producer, storing the product’s id, name, the producer that owns the product, the product reviews, the unit used to measure the product, and the amount of product left in stock. This class will have an aggregation relationship with the Comment class and will have an association relationship with the Producer class.

The Comment class will represent a review of a product. It will store information about the customer or producer that wrote it, which product it was written for, the content of the review, and the number of likes it has, and will store the id of the parent comment if it is a reply.

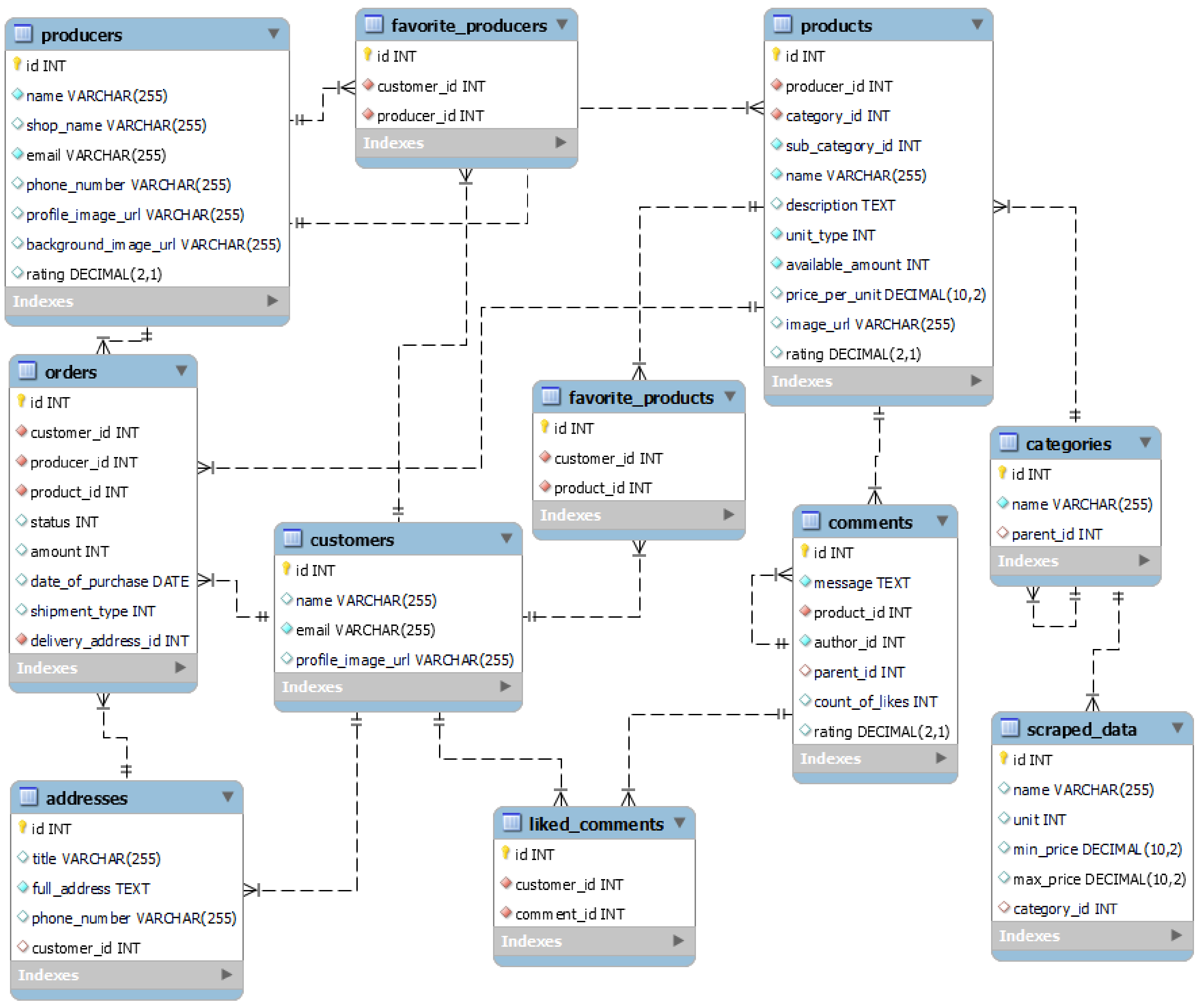
The Order class will represent an order that a user makes. It will contain information about the date it was made, which customer ordered it, which (unique) product was ordered, the amount of the order, the shipment type, and the delivery address. This class will have an implementation relationship with the interface IDGetter.

Cart class will represent a user's cart, containing the products and the desired amounts of said products in a user’s cart. It will have an aggregation relationship with the Order class.

The Address class will represent an address in the real world. It will store the title of the address, the full address, and the phone number associated with the address.

FavoriteProducers class will represent the list of favorite producers a customer has. It will have an aggregation relationship with the Producer class. Similarly, the FavoriteProducts class will represent the favorite products of a user and will have an aggregation relationship with the Product class.

3.2 Database Schema



The SQL schema presented here is designed to manage Bahçeden. The schema consists of seven tables: Producers, Addresses, Products, Customers, Comments, Orders, and two additional tables, FavoriteProducts and FavoriteProducers, to allow customers to keep track of their preferred products and producers.

producers: This table stores information about the producers, which includes their id, name, shop name, email, phone number, profile image URL, background image URL, and rating.

customers: This table stores information about the customers including their id, name, email, and profile image URL.

categories: This table stores information about product categories. Each category has an id, name, and parent_id which is a foreign key referring to the id field within the same categories table, meaning categories can be nested within each other.

products: This table stores information about products, including id, producer_id (referring to producers), category_id (referring to categories), sub_category_id, name, description, unit type, available amount, price per unit, image URL, and rating.

favorite_products: This table is used for tracking customers' favorite products. It includes a customer_id (referring to customers) and a product_id (referring to products).

favorite_producers: Similarly, this table is used for tracking customers' favorite producers, and includes customer_id and producer_id (referring to producers).

addresses: This table stores the shipping address information for customers, which includes id, title, full address, phone number, and customer_id (referring to customers).

orders: This table stores the order information, which includes id, customer_id, producer_id, product_id (all referring to their respective tables), order status, order amount, date of purchase, shipment type, and delivery address_id (referring to addresses).

comments: This table is used to store comments made by customers. It contains fields for id, message content, product_id (referring to products), author_id, parent_id (referring to the id of another comment, in case of nested comments), count of likes, and a rating.

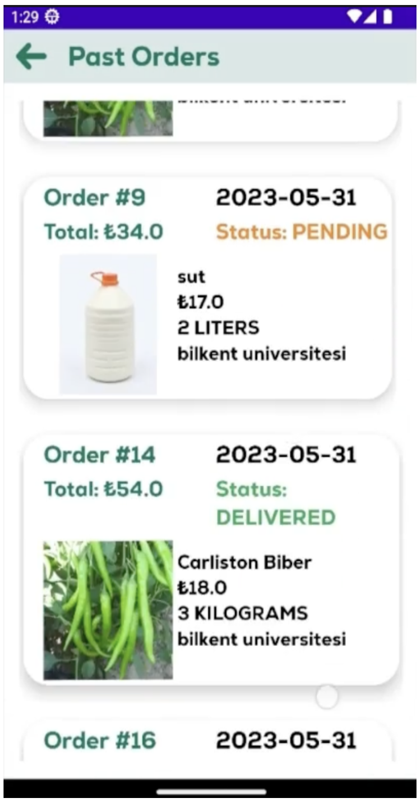
scraped_data: This table is used to store data that has been scraped from other sources, which includes id, name, unit, minimum price, maximum price, and category_id (referring to categories).

liked_comments: This table is for tracking which comments have been liked by which customers. It includes a customer_id (referring to customers) and a comment_id (referring to comments).

4. Task assignment

Customer UI Design	Erfan FarhangKia, Mehmet Akif Şahin, Mehmet Anıl Yeşil
Producer UI Design	Şükrü Eren Gökırmak, Mehmet Anıl Yeşil, Emir Ensar Sevil
Frontend	Mehmet Akif Şahin, Şükrü Eren Gökırmak, Emir Ensar Sevil
Backend	Mehmet Anıl Yeşil, Erfan FarhangKia
Web Scraping	Mehmet Akif Şahin, Erfan FarhangKia

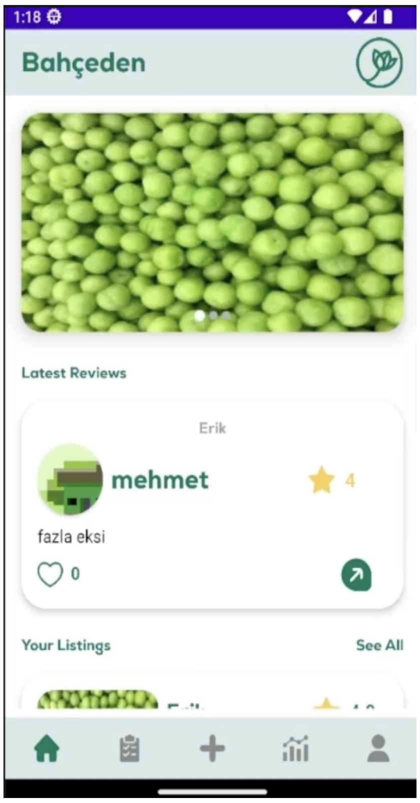
5. In-App Screenshots


Past Orders

Order #9 2023-05-31
Total: ₺34.0 Status: PENDING
sut
₺17.0
2 LITERS
bilkent universitesi

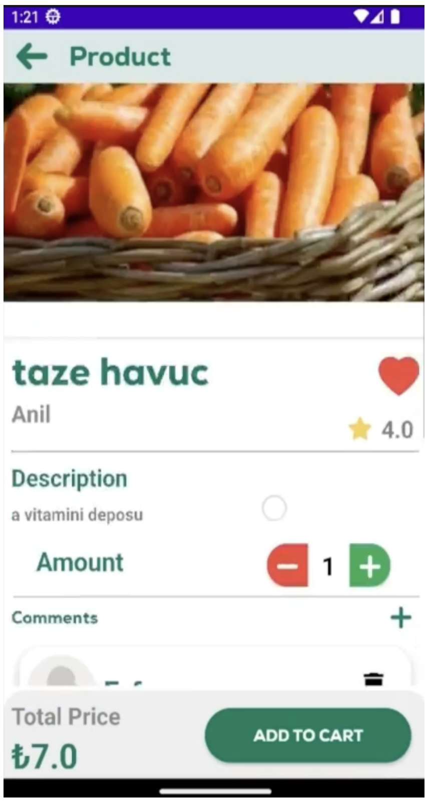
Order #14 2023-05-31
Total: ₺54.0 Status: DELIVERED
Carliston Biber
₺18.0
3 KILOGRAMS
bilkent universitesi

Order #16 2023-05-31

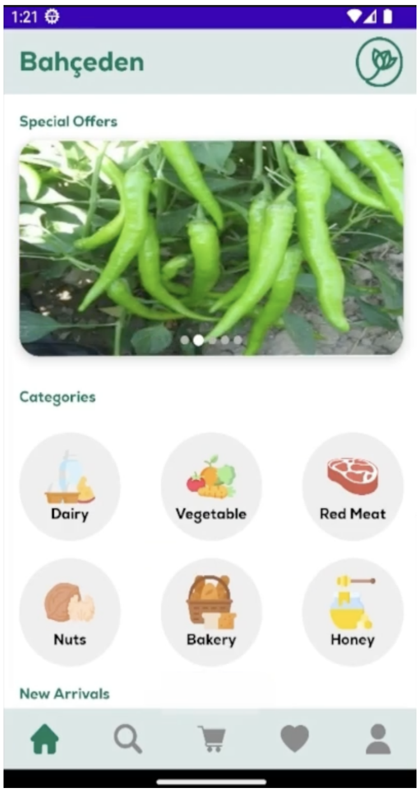

Bahçeden

Latest Reviews
Erik
mehmet 4
fazla eksi
0

Your Listings See All


Product

taze havuc
Anil
4.0
Description
a vitamini deposu
Amount 1
Comments
Total Price
₺7.0
ADD TO CART


Bahçeden

Special Offers
Categories
Dairy
Vegetable
Red Meat
Nuts
Bakery
Honey
New Arrivals

6. Reflections

What did you dislike about project work?

While the project was overall a great learning experience, certain aspects were less enjoyable. For instance, the iterative process of debugging was at times frustrating. Additionally, the lack of clear initial specifications led to some confusion and required extra time for adjustments later in the project.

What was the most difficult aspect of it?

The most difficult aspect of the project was effectively coordinating within the team, especially during remote collaboration due to differing schedules and time zones. Ensuring that everyone was on the same page and moving forward cohesively was a challenge.

What would you do differently if you had to start over again?

If we were to start over again, I would advocate for a more structured planning phase before diving into the coding. Spending more time initially to create a comprehensive roadmap and clarify roles and responsibilities could save time and reduce confusion in the long run. Regular check-ins and progress updates could also be implemented to enhance team communication.

How much time did you spend on it?

Due to the complexities and unexpected hurdles faced during the project, a substantial amount of time was spent. If we had to quantify, it would be roughly 15-20 hours a week over the course of the semester, totaling approximately 300 hours.

Are you proud of what you have achieved?

Despite the challenges faced, the final product was a testament to the team's resilience and hard work. Seeing the application function as intended and reflecting on the progress made from the beginning, we can confidently say that we are proud of what was achieved.

What weren’t you able to achieve?

We couldn’t achieve the implementation of the payment part and also we couldn’t achieve implementing a graph on our analytics part.