CS319

Deliverable 4

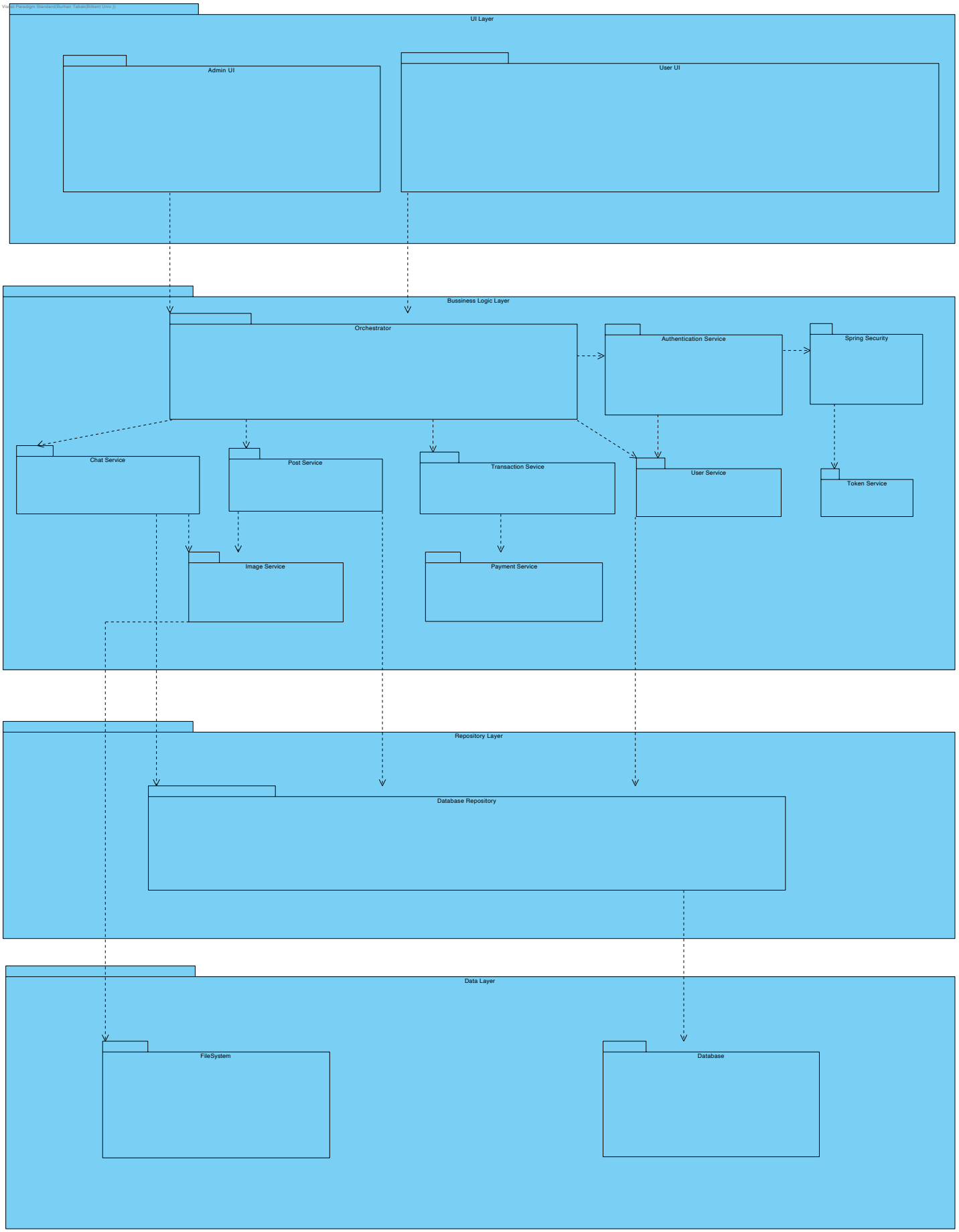
BILSYNC

Team 8

Section 1

Tuna Saygın - 22102566
Hüseyin Burhan Tabak - 22102516
Ahmet Tarık Uçur - 22102946
Işıl Özgü - 22102276
Kanan Zeynalov - 22101007

Table of Contents


Visual Paradigm Standard(Burhan Tabak(Bilkent Univ.))
UI Layer
Admin UI
User UI
Bussiness Logic Layer
Orchestrator
Authentication Service
Spring Security
Chat Service
Post Service
Transaction Sevice
User Service
Token Service
Image Service
Payment Service
Repository Layer
Database Repository
Data Layer
FileSystem
Database

1.0 High-Level Software Architecture

1.1 Subsystem Decomposition

We adopted a 4-layer subsystem decomposition design to enhance the overall structure and functionality of the system. The subsystems are organized into the UI, Business Logic, Repository, and Data Layer, each serving distinct purposes.

1.1.1 UI Layer

The UI Layer is responsible for user interaction and interface design, where people can communicate with the application. This layer increases the system's usability through a visually appealing and responsive interface. This layer communicates consistently with the Business Logic Layer.

Admin UI:

- This package comprises screens and interactions only the admin can access and use.
- This package consist of several screens such as User Management screen, Report Management screen. They will not be explained in detail since the scope of Subsystem Decompsiition diagrams not include these details.

User UI:

- This package contains screens that users interact with to navigate through our application and connect with other users via different mediums our platform supports, including but not limited to posts, chats and more.

1.1.2 Business Logic Layer

The Business Logic Layer focuses on the core functionalities of the application with control objects from the data received from the UI Layer. It can modify the data and ensure the integrity of the system and information throughout the scope of the application. These control object packages communicate with the Database Repository in the Repository Layer.

Orchestrator

- This package consists of control objects that pass information to other packages received from the UI Layer packages.
- This package is associated with Chat Service, Post Service, Transaction Service, User Service, Authentication Service, User Management UI, Report Management UI, Login UI, Group Management UI and Account UI packages.

Authentication Service

- This package consists of control classes that regulate the authentication of the users.
- This package is associated with Spring Security and User Service packages.

Spring Security

- This package consists of control classes that get data from the Authentication service and send it to the Tokenization process.
- This package is associated with Token Service and Authentication Service packages.

Token Service

- This package consists of control classes related to Spring Security for information exchange.
- This package is associated with the Spring Security package.

User Service

- This package consists of control classes that receive and process the Users' data and exchange information with the Database repository.
- This package is associated with Orchestrator, Authentication Service, and Database Repository, which is in the Repository Layer.

Chat Service

- This package consists of control classes related to managing the chat system according to the given data by the users.
- This package is associated with Orchestrator, Image Service, and Database Repository in the Repository layer.

Post Service

- This package consists of control classes that manage the Posting system in the application.
- This package is associated with Orchestrator, Image Service, and Database Repository, which is in the Repository Layer.

Image Service

- This package consists of control classes related to image processing.
- This package is associated with Chat Service and Post Service packages.

Transaction Service

- This package consists of control classes related to payment system management.
- This package is associated with Orchestrator and Payment Service packages.

Payment Service

- This package consists of control classes that will receive information from the User and process the transaction system.
- This package is associated with the Transaction Service package.

1.1.3 Repository Layer

Repository Layer acts as an intermediary between the Business Logic Layer and the underlying data storage. Handles data retrieval and storage operations, providing a clean and standardized interface for the Business Logic Layer to interact with the Repository. It contains only the Database Repository package.

Database Repository

- This package consists of the classes related to database management.
- This package is associated with the Chat Service, Post Service, User Service, and Database package in the Data Layer.

1.1.4 Data Layer

The Data layer manages the physical storage and retrieval of data from databases or other data sources from the application. It contains FileSystem and Database packages, which ensure data security, integrity, and persistence.

FileSystem

- This package refers to the external storage of the File system.
- This package is associated with the Image Service package in the Business Logic Layer.

Database

- This package refers to the external database.
- This package is associated with the Database Repository package.

2.0 Design Goals

Since our application combines a social media platform and a marketplace, our design goals are determined by what our possible end users expect from our platform. Because the project is in the design and development phase, the design goals are shaped according to the project's current version.

2.1 Functionality

According to our field observation, our users were interested in a platform that increases their chance to interact with fellow Bilkenters in an anonymous form. This need arose from the general shyness in the community. Moreover, our possible end users depended on a single platform, Instagram, to borrow, sell, or announce general complaints about the community. However, these types of interactions are unsuitable for a platform on Instagram as every post needs to contain an image, and there is not much content separation as people choose to connect over a single account, bilkent_itiraf_ediyor. Our main goal was to separate these contents into forum posts and trading posts; moreover, this separation allows the users only to see what they are interested in, increasing user experience. Our other functionalities allow users to complete sell/buy and borrow/donate operations inside our platform, keeping the users engaged with our platform. Similarly, creating different chat types in our platform increases user engagement as the conversations are not limited to the posts' comment section. These different functionalities are our highest priority, as our platform aims to fill a space in the app market that contains the most useful functionalities of popular social media platforms.

2.2 Usability

Our intuitive user interface provides a smooth user experience. Moreover, it is possible to switch between posts and chats, and even some trading posts, such as second-hand posts, automatically create a private chat for buyers and sellers to communicate their delivery method after the system initiates the transaction. Additionally, the system functionality does not clash with usability. Each interaction type, such as post interaction, chat interaction, and profile operations, has its page or section on the main page to create an understandable structure within the application.

2.3 Rapid Development

Since our project is a term project, one of our most important goals is to rapidly develop our project so that we can deliver it to our client. Moreover, our agile methodology allows us to iterate through the project often and catch mistakes early on. Moreover, this goal pushes our developing team to utilize reusable interfaces and libraries.

3.0 Design Trade-offs

3.1 Functionality vs. Usability

Higher functionality generally clashes with usability. However, we tried to prevent this by introducing separate pages or menus for main functionalities. Additionally, we can maintain usability by using pop-up pages while introducing different functionalities.
To set our project apart from our competitors, we, as a team, decided that functionality should come before usability. That is why we prioritize implementing different functionalities such as being able to publish different types of posts such as forum posts, second-hand, borrowing, donation and implementing chat in trading posts to allow communication during negotiation and delivery process as well as implementing a “deliver-first-then-obtain-money” approach during payment process to reduce possible scammers.

3.2 Rapid Development vs. Functionality

The rapid development process may lead our team to prioritize the main functionalities and eliminate the rest. Our team tried to balance these trade-offs by determining milestones for each one or two-week period.
However, even though it is important to develop a software in a fast-paced manner, especially in a term project, to set our project apart from our competitors, we aimed to deliver extra functionalities users may like. Thus, we prioritize functionality over development speed.