Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Engineering
Fall 2023

Deliverable 4

Design Goals,
High-Level Architecture.

Team 5

Cahit Ediz Civan (22003206)
Efe Kaan Fidancı (22102589)
Emir Tuğlu (22003165)
Görkem Kadir Solun (22003214)
Mete Enes Yılmaz (22102849)

Table of Contents

1. Introduction

This report discusses and explains the design and purpose of the software system. In the first part, the non-functional requirements of the software system are enacted and specified. In the second part, the design was expanded by focusing on the technical details of the software. This extension covers the architecture of the system. It includes separating different subsystems, relationships between hardware and software, data management, information access control and security, and boundary use cases.

1.1. Purpose of the System

The primary aim of the BEN (Bilkent Exchange Network) app is to establish a centralized and inclusive digital platform that facilitates communication and exchange-related activities among the students of Bilkent University. Another aim of BEN is to overcome the limitations imposed by fragmented communication channels, such as various Instagram accounts, and to foster a more interconnected environment within the university. The BEN app aims to revolutionize how the Bilkent University community interacts, communicates, and participates in exchange activities by offering a unified, user-centric, and adaptable digital platform.

1.2. Design Goals

Non-functional requirements specific to the application's needs have been carefully determined to provide a more livable experience for users and the software system. These goals were determined as a result of comprehensive background research, and a plan was created to implement them during the implementation process effectively.

1.2.1 Usability

The Bilkent Exchange Network is committed to delivering a smooth user experience through an intuitive and user-friendly interface, ensuring users and administrators can navigate the app effortlessly. Drawing inspiration from popular social media applications like Facebook, WhatsApp, and Instagram, our design approach aims to enhance user-friendliness. Accessibility is a key priority, and the application will adhere to accessibility standards to ensure inclusivity for all users. The responsive design of Bilkent Exchange Network will adapt to various devices and browsers, dynamically resizing elements to accommodate different screen sizes. Furthermore, the platform will provide concise and clear user documentation for UI elements, ensuring informative messages and guides to assist users in resolving queries.

We strive to continually improve and optimize the interface through rigorous user testing and feedback. By adopting user-centered design systems, we focus on deploying user units most effectively, with healthy packages and minimum download requirements. Iterative processes are designed based on user payments and preferences to create an application that meets and exceeds user expectations, ultimately providing a continuous and enjoyable experience for all users of the Bilkent Exchange Network.

1.2.2 Reliability

BEN is a social network; therefore, reliability is essential for Bilkent Exchange Network to make the application available to users with minimal downtime. The application should be available with at least a 95% uptime rate, ensuring users can access it whenever possible. Also, the application should handle errors gracefully. It should provide informative and precise error messages to prevent bad user experience.

1.2.3 Security/Safety

Bilkent Exchange Network should include robust user authentication and authorization mechanisms such as strong password policies and multi-factor authentication to ensure that only authorized users can access and modify data. Data stored in the database, such as user accounts and posts, must be reliably stored and protected by access controls and data encryption. Sensitive data, such as user passwords, must be encrypted with the SHA-256 algorithm before being stored in the database. Also, the application should respect and protect user information by sharing only the necessary information with other users.

1.2.4 Performance

The application's performance is paramount to ensuring a positive user experience. To respond to user requests in less than one second, swift responsiveness is a key focus. Users can expect to receive verification emails promptly, within one minute of their request, contributing to the efficiency of the onboarding process. Additionally, smooth navigation is prioritized, aiming for transitions between different pages to take less than three seconds. These time-sensitive objectives collectively contribute to the application's overall performance, enhancing user satisfaction throughout their interaction with the system.

1.2.5 Maintainability

The Bilkent Exchange Network should be designed and implemented flexibly, enabling further improvements and easy maintenance. The application’s code should be well-structured according to object-oriented design principles and documented to support future updates. Designing and implementing the application in this way makes adding new features easy and requires less effort to maintain the application.

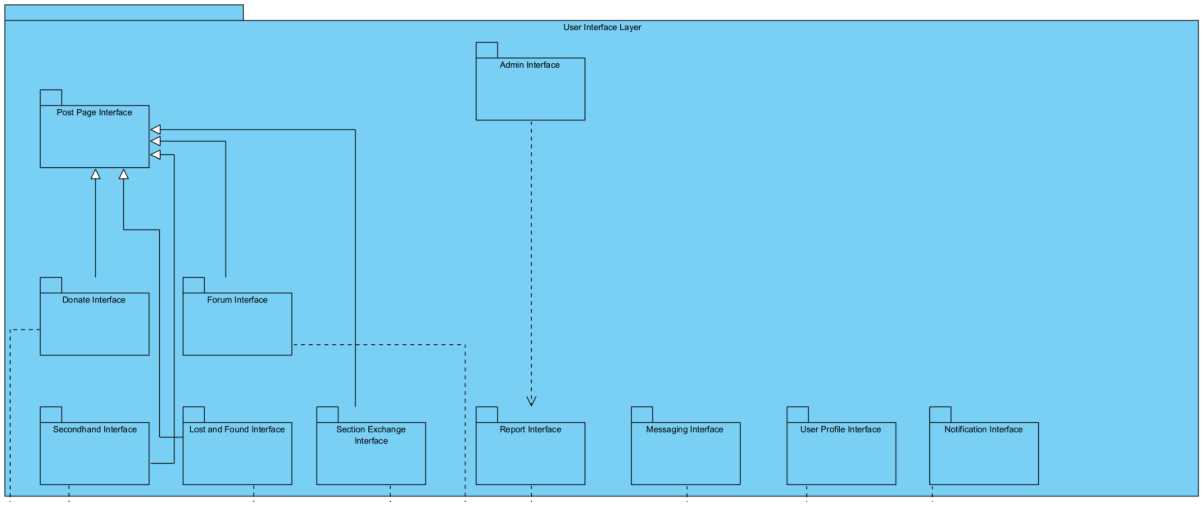
2. High-Level Software Architecture

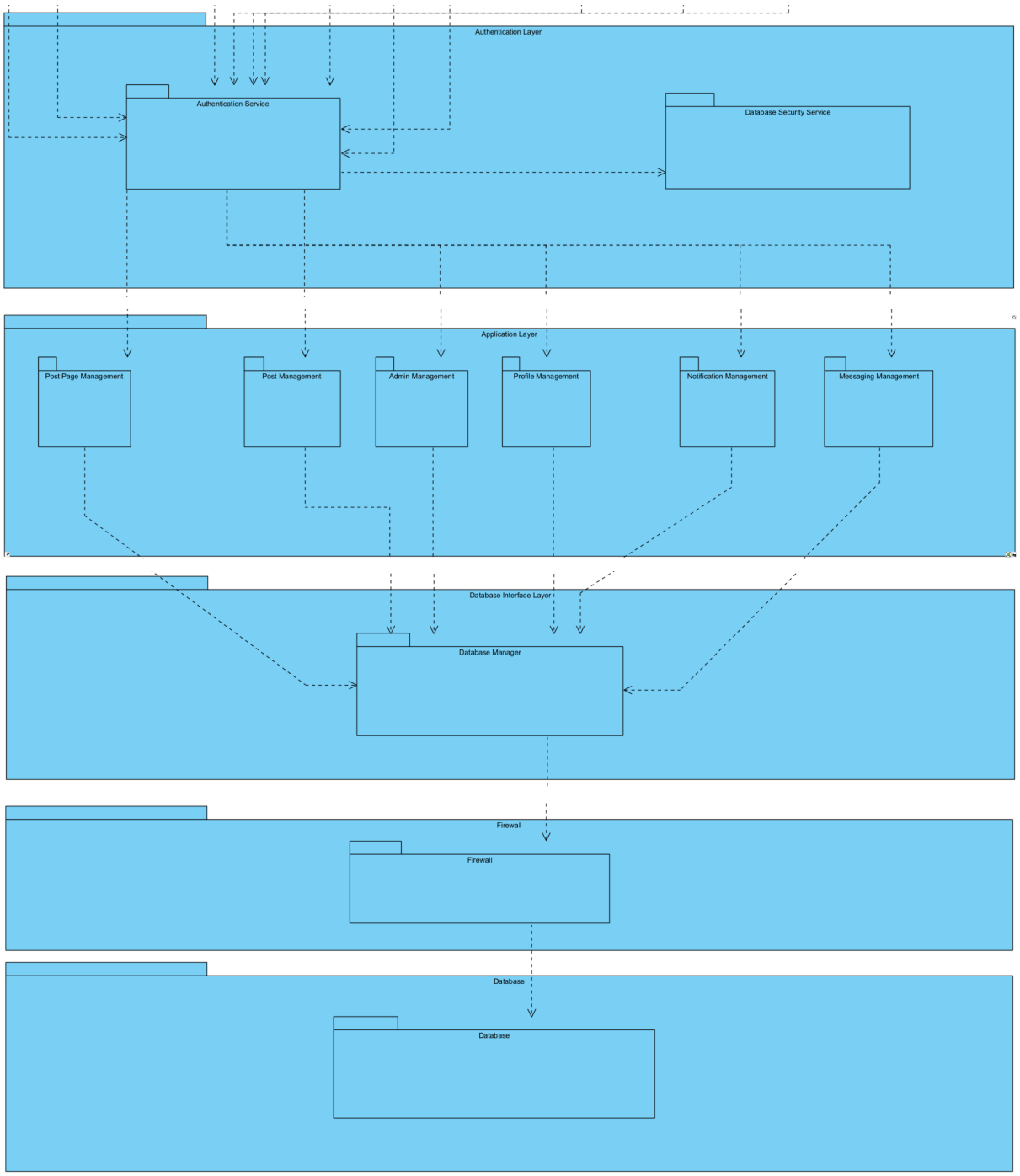
2.1 Overview

This section provides an in-depth look at the layers within the subsystem decomposition scope and explains hardware and software mapping in detail. Minimum requirements for the use of the program are indicated. Additionally, the use of the database regarding permanent data management is explained, and which data is stored is discussed in detail. In terms of access control and security, it provides a better understanding of program logic by providing control over the capabilities of different user types. Additionally, boundary conditions that specify how the program will react to unexpected situations are explained in detail. This information increases the integrity and reliability of the program, contributing to a more robust experience for users.

2.2 Subsystem Decomposition

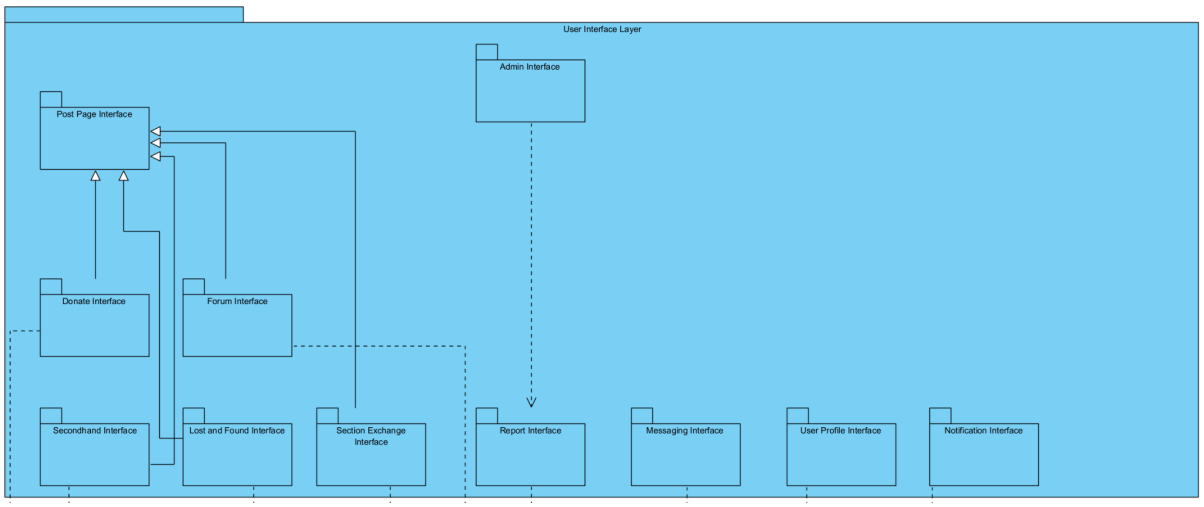
The system decomposes into five layers: User Interface Layer, Authentication Layer, Application Layer, Database Interface Layer, and Database Layer. The program structure is divided into five distinct sub-structures based on how they differ in related packages. This design would enhance our system's security, functionality, usability, and maintainability.





2.2.1. User Interface Layer

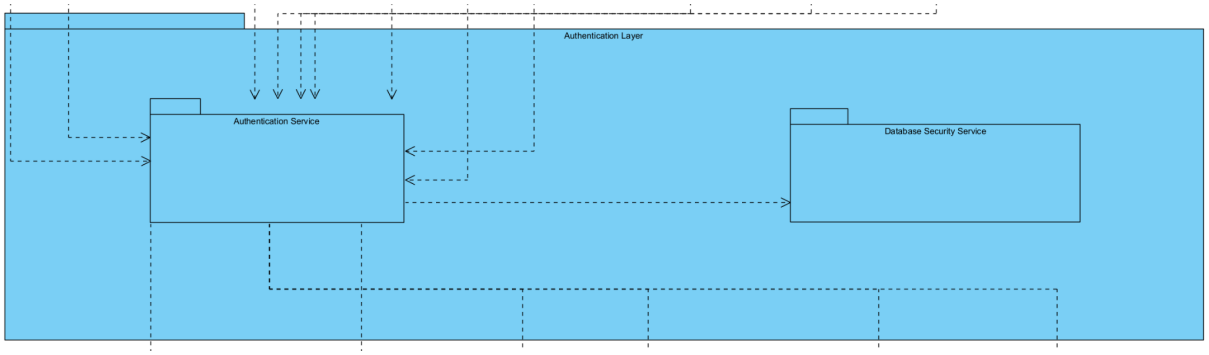
The term "User Interface Layer" describes how people interact with the displays, allowing them to connect with the software. Users can provide data in the requested fields, and the Application Layer will get this data.



- Admin Interface
    - This interface contains screens for admins to handle reports, remove posts, and ban users.
- Post Page Interface
    - This interface generalizes donate, forum, secondhand, lost and found, and section exchange interfaces.
    - It contains the common elements between those interfaces.
- Donate Interface
    - This interface contains screens for donate item posts.
    - This interface is generalized to the Post Page Interface.
- Forum Interface
    - This interface contains screens for forum elements.
    - This interface is generalized to the Post Page Interface.
- Secondhand Interface
    - This interface contains screens for secondhand item posts.
    - This interface is generalized to the Post Page Interface.
- Lost and Found Interface
    - This interface contains screens for lost and found item posts.
    - This interface is generalized to the Post Page Interface.
- Section Exchange Interface
    - This interface contains screens for section exchange item posts.
    - This interface is generalized to the Post Page Interface.
- Report Interface
    - This interface comprises screens for the admin to review and handle reports.
- Messaging Interface
    - This interface contains the required elements for messaging.
- User Profile Interface
    - This interface is composed of the elements of the user profile, such as name, description, posts, etc.
- Notification Interface
    - This interface contains notifications.

2.2.2 Authentication Layer

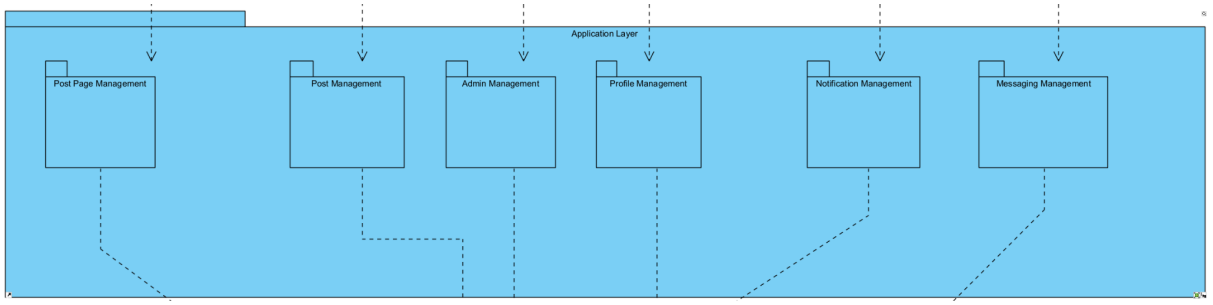
The authentication layer is located between the User Interface and Application layers. The main function of this layer is to validate and authorize user requests and safeguard the system against unauthorized access.



- Authentication Service
    - The Authentication Service is responsible for verifying the identity of users attempting to access the system.
- Database Security Service
    - The Database Security Service focuses on safeguarding the integrity and confidentiality of data stored in the database. It implements data encryption and access controls to prevent unauthorized access or tampering with sensitive information.

2.2.3. Application Layer

Control object packages that may modify the data by the obtained data make up the Application Layer. Under the User Interface Layer, data from the User Interface Layer is processed in the Application Layer to provide the referencing information. These control object packages interact with the Database Manager within the Database Interface Layer.

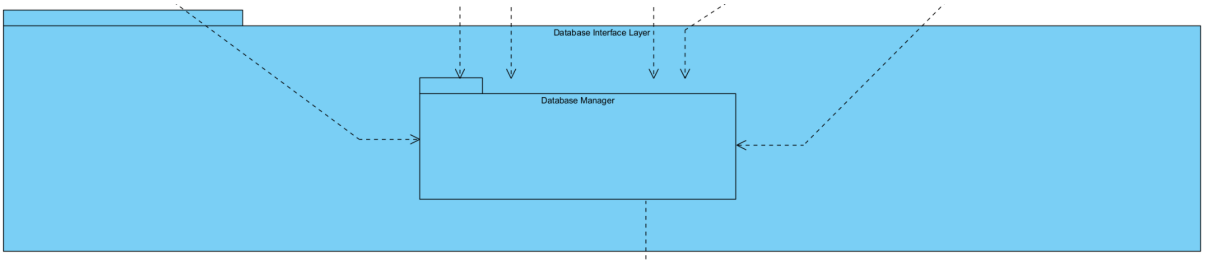


- Post Page Management
    - This package consists of classes related to post page management.
- Post Management
    - This package consists of classes related to post management.
- Admin Management
    - This package consists of classes related to admin functionality.
- Profile Management
    - This package consists of classes related to profile management.
- Notification Management

    - This package consists of classes related to notification management.
- Messaging Management
    - This package consists of classes related to messaging management.

2.2.4. Database Interface Layer

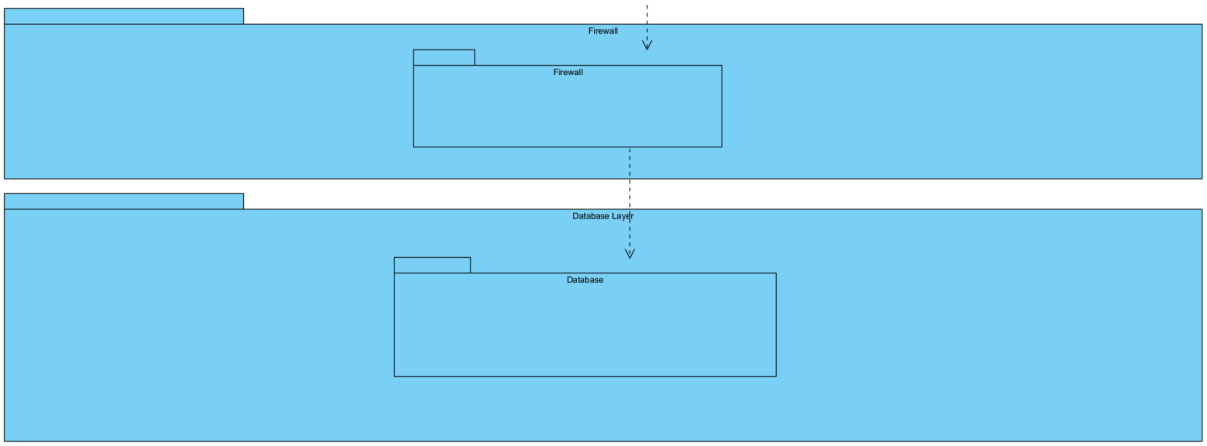
The single component in the Database Interface Layer is the Database Manager, whose job is to manage database instances. Between the Application Layer and the Database Layer sits this transactional management layer.



- Database Manager
    - This package consists of classes related to managing database instances.

2.2.5. Database Layer

Only the Database and Firewall are in the database layer. The database is responsible for keeping track of and storing the data that the system needs. On the database, the data is both read and written.

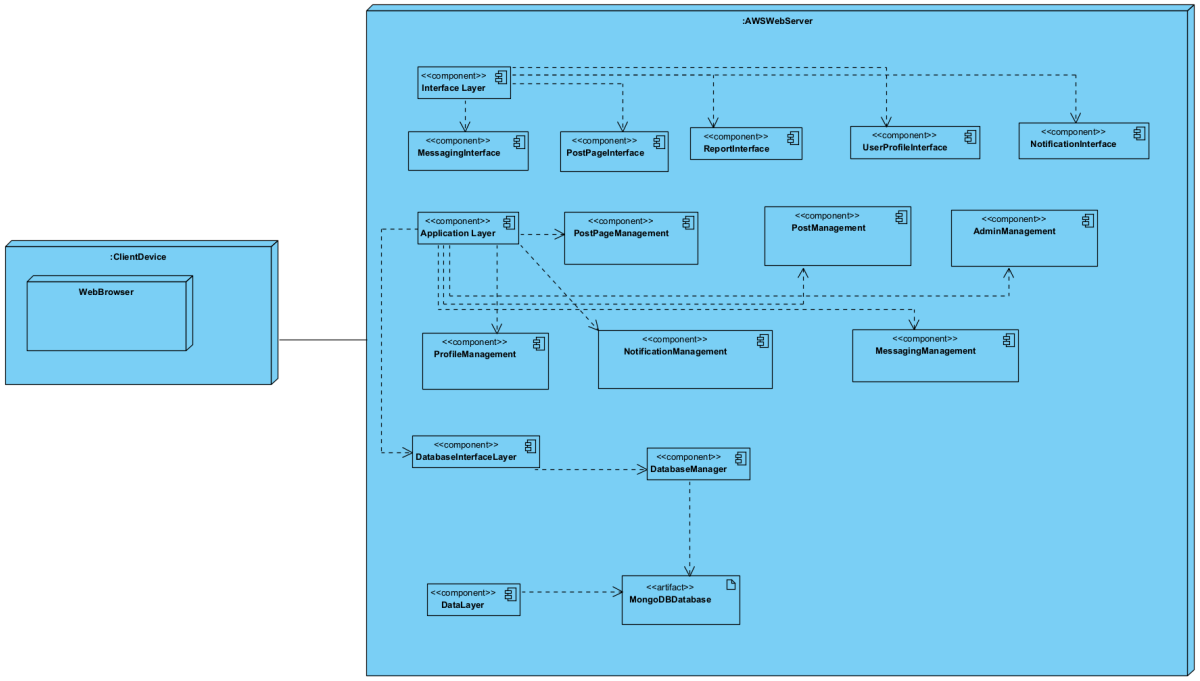


- Firewall
    - The Firewall is a critical security component within the Database Layer, acting as a protective barrier between the BEN app and external networks, safeguarding the system against unauthorized access, cyber threats, and potential vulnerabilities.
- Database
    - The Database is a fundamental component residing in the Database Layer, serving as the primary repository for storing and organizing the data essential

for the app's functioning. Its key responsibility is to provide a structured and efficient mechanism for storing and retrieving information.

2.3. Deployment Diagram

The communication between the client and server is depicted in this diagram. Its layout bears a striking resemblance to subsystem breakdown. Classifications that function closely together and may be categorized by kind are called components, whereas artifacts represent tangible aspects of the physical world.



2.4. Hardware/Software Mapping

For the front end, TypeScript and React will be used for the front end. TypeScript offers backward compatibility via transpiling to earlier JavaScript versions. Additionally, we utilize CSS3 and HTML5, which are supported by all current browsers. As a result, it merely requires a stable version of a current browser and does not require high system demands. MongoDB and C# will be used in our backend. The database and backend will be set up on MongoDB servers. Taking into account every member of Bilkent University, the number of users can easily surpass thousands. Given the volume of users, a sizable server cluster will be needed to gather all of the user data. The server's capacity should be promptly increased if these criteria are insufficient.

There are no specific hardware components needed for this project. The application is web-based. Therefore, its accessibility is not restricted to any specific device as long as the user has hardware that can run any contemporary browser. As a result, computers running contemporary browsers like Google Chrome (v41.0.2272 and up), Mozilla Firefox (v21 and up), Microsoft Edge (any version), Safari (v6 and up), and Opera (v11 and up) may use the

application. Standard computer accessories, including a keyboard and monitor, are needed to utilize the program.

2.5. Persistent Data Management

Since MongoDB offers simple access operations throughout the development process, it will be utilized as the database technology. Furthermore, Next.js is compatible with MongoDB, enabling even greater maintainability. As a result, using MongoDB to construct and manage our OOP model for the project will be simpler. The MongoDB database will hold information about users, posts, market items, and sent notifications. The server will be made available online via Amazon AWS. Because the itemThehange, the requests sent through the application must be dynamic. This will remove any potential discrepancy in the data that has been saved.

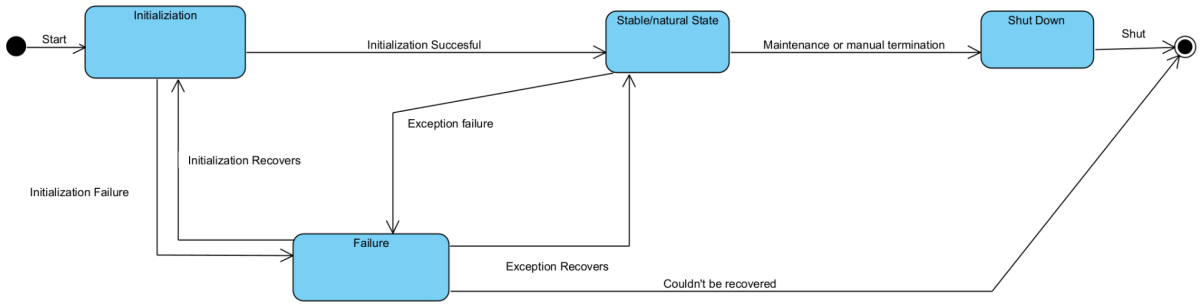
2.6. Access Control and Security

Because Bilkent Exchange Network (BEN) is an online program, it has to have more security and permission. The security of MongoDB will be supplied at the back-end logic. Built-in support for MongoDB's Security Authenticate will manage authentications. Moreover, MongoDB's Security may grant each user type the specified rights. Finally, built-in security features in MongoDB's Security help protect the application from frequent attack vectors. In addition to MongoDB's security features, Amazon AWS provides some of the finest defenses against potential attacks (such as denial of service). Actors in the application can directly contact the methods of control classes listed below by utilizing the buttons on the website interface.

	Buyer	Seller	User	Admin
AuthenticationManager	authenticate() signUp(...) signIn(...)	authenticate() signUp(...) signIn(...)	authenticate() signUp(...) signIn(...)	authenticate() signIn(...)
ProfileManager	setPicture(...) setTitle(...) seeOwnPosts()	setPicture(...) setTitle(...) seeOwnPosts()	setPicture(...) setTitle(...) seeOwnPosts()	setPicture(...) setTitle(...) seeOwnPosts()
PostManager		createSecondhandPost(...) editSecondhandPost(...) removeOwnPost(...) createContent(...) setCategory(...)	createSecondhandPost(...) createExchangePost(...) createBorrowingPost(...) createLostFoundPost(...) createDonationPost(...) createForumPost(...) createEntry(...)	createSecondhandPost(...) createExchangePost(...) createBorrowingPost(...) createLostFoundPost(...) createDonationPost(...) createForumPost(...) createEntry(...) setCategory(...) editPost(...) removeOwnPost(...)

	Buyer	Seller	User	Admin
			editPost(...) removeOwnPost(...) setCategory(...)	
Page	sortByDate(...) sortByValue(...) filterByDate(...) filterByCategory(...) search(...) sendMessageToOwner(...)		sortByDate(...) filterByDate(...) filterByCategory(...) search(...) sendMessageToOwner(...) sortByVote(...) sortByValue(...) filterByValue(...)	sortByDate(...) filterByDate(...) filterByCategory(...) search(...) sendMessageToOwner(...) sortByVote(...) sortByValue(...) filterByValue(...)
DirectMessage	composeMessage(...) sendMessageTo(...)	composeMessage(...) sendMessageTo(...)	composeMessage(...) sendMessageTo(...)	composeMessage(...) sendMessageTo(...)
ReportManager	reportPost(...) reportUser(...)	reportUser(...)	reportPost(...) reportUser(...)	getReport(...) resolveReport(...)
NotificationManager	getNotifications(...)	getNotifications(...)	getNotifications(...)	getNotifications(...)
AdminManager				forceRemovePost(...) banUser(...)
DatabaseManager				

2.7. Boundary Conditions



2.7.1. Initialization

Since BEN is a web-based application, an existing username is required to log in to the site. When the application is started, when user login occurs, the relevant data is pulled from the database to initialize the system. Users without an account can also access the application just to create one. When the user logs into the application, the relevant page options appear in the sidebar. When a page is clicked, the information on the page, such as the items on the secondhand page or posts in the forum section, is pulled from the database. Also, since it is one of the important parts that perform various application operations, AWS S3 credentials

should be provided for different users' information and uploaded files, and AWS S3 setup should be checked regularly. Finally, the database must be initialized to start the application, and BEN-specific credentials must be passed. If necessary, the server's IP address to access the database must be provided by the database for it to respond. In addition to powering the backend operation of the application, this information is used to check referential integrity by internally using access and refresh tokens generated when users log in.

2.7.2. Termination

Termination of BEN can be accomplished in three different ways. First, when a user logs out of the application, the application is closed with the last saved data stored in the user's database.

Second, due to a failure in the system, an unhandled exception may arise that forces the application to terminate. However, such cases are usually handled on the backend side, allowing the system to exit without problems or display meaningful errors to the user.

When the user's access token expires, a request is sent to the authentication unit. The authentication service refreshes the user's access token if the refresh token has not yet expired. Otherwise, the user will be automatically logged out when the access token expires. This ensures the system's healthy functioning in terms of security and user session management.

2.7.3. Failure

BEN will run on AWS servers and operate without having a backup web server. In this case, if a problem occurs in AWS, the application may fail, in which case the program will be restarted, returning to the database with the last saved data.

Furthermore, since BEN is a web application that works over the internet, losing the user's internet connection may also create a potential problem. In this case, the user can access the application by simply logging in again, and the application will be launched with the last saved data so that even if an outage has occurred, the user's experience continues smoothly.