Bilkent University

Department of Computer Engineering

Campuswide Connection System

OnlyBilkent (team12 - 01)

Design Report

Anıl Altuncu - 21901880
Zehra İyigün - 22002913
İpek Sönmez - 22103939
Bartu Albayrak - 22101640
Gizem Gökçe Işık - 21803541

Instructor: Eray Tüzün
Teaching Assistant(s): Yahya Elnouby, Selen Uysal

Iteration 1

Contents

Design Report

OnlyBilkent

1. Introduction

The purpose of this report is to define the software system by explaining its purpose and modified non-functional requirements of the software system in the first report. The second part explains the software architecture, decomposition of different subsystems inside the system, relation between hardware and software, and data management. Lastly, the last part is reserved for the low-level design, which covers the object design, packages, class interfaces, and design patterns.

1.1.

1.2. The Purpose of the System

OnlyBilkent is designed to meet the diverse needs of the Bilkent University community under one web application. Exclusively designed for Bilkenters, this single application connects students, club boards, and graduates. OnlyBilkent empowers users to effortlessly buy and sell items, recover lost belongings, donate unused items, and simplify borrowing. Unlike other separate systems, OnlyBilkent stands out by streamlining and enhancing university life with accessibility, dedicated to fostering a more connected and harmonious campus experience for Bilkent University students, club boards, and graduates.

1.3. Design Goals

1.3.1. Usability

Our aim is to ease and fasten communication among the student network of Bilkent University. Therefore, the system should be easy to use and non-complicated. Consistent design elements, such as consistent symbols for post categories throughout every page of the application, will be maintained. A straightforward and simple left-side menu will display post categories, notifications, and announcements. Posting, messaging, and making an announcement will be done.

1.3.2. Maintainability

Given the possibility of future needs, such as incorporating new features, designing our application with maintainability in mind is significant. OnlyBilkent components will be organized into separate modules such as the post module, buyer-seller module, lender-borrower module, announcement module, etc.

1.3.3. Reliability

Users often seek to search for specific posts on OnlyBilkent, whether for educational purposes or to stay updated on campus events. Therefore, minimizing the data loss caused by system crashes is reasonable. Database backups will be performed every 12 hours. Using redundant database systems with replication to ensure data remains available even if one database server fails will minimize the data loss by 90%.

1.3.4. Safety

Authentication system where OnlyBilkent users need to log in with their own university e-mail addresses to ensure that they are from Bilkent University. This means that each user will have only one account associated with their university e-mail address. AES 128 Encryption will be used to encrypt the passwords of the users. After encryption, the password will be stored in the database. Admins will control the access authorization of the users.

2. High-level Software Architecture

2.1. Subsystem Decomposition

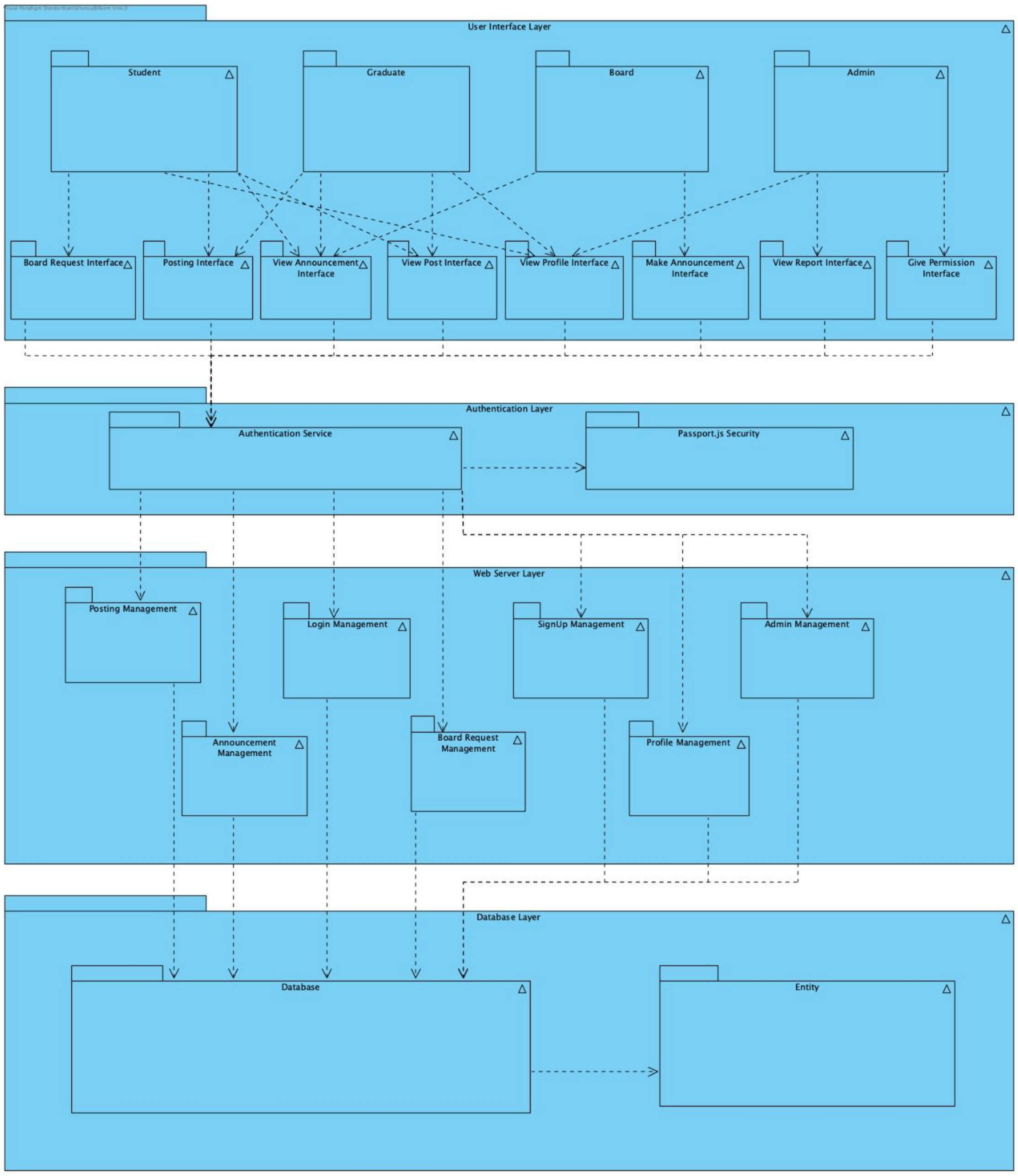

Fig. 1 Subsystem Decomposition Diagram

We utilized a four layered structure in our system, encompassing the User Interface Layer, Authentication Layer, Web Server Layer, and Database Layer. We aimed to decompose into subsystems regarding their distinct functions. Each subsystem holds the responsibility to call other systems ensuring the system remains maintainable while in action. This division also aims to enhance scalability by ensuring each layer operates independently. As a result, it complements our system to be easier to maintain, and functional.

2.1.1 User Interface Layer

The User Interface Layer is the interface between users, and the screens, providing interaction with our system. Through this layer, users can navigate between different screens by clicking on the designated areas or inputting information into designated places. The accessible screens are different for each user type being Student, Graduate, Board and Admin.

2.1.2 Authentication Layer

The Authentication Layer is responsible for ensuring secure access to the app, managing user logins, registrations, and verifying user identities. It implements Passport.js authentication protocols guaranteeing that user data remains accessible only to authorized individuals.

2.1.3 Web Server Layer

Web Server Layer acts as a management system of our system. This layer manages interactions between the User Interface, Authentication, and Database layers, ensuring functionality throughout the app.

2.1.4 Database Layer

The Database Layer is responsible for handling data storage, retrieval, and management. It stores user profiles, posts, direct messages, likes, and other relevant information securely. Implementing an optimized database structure ensures quick access to information, enabling swift responses to user queries such as filtering out posts regarding them being latest, oldest etc.

2.2. Hardware/software mapping

In our system, we've chosen Bootstrap for the frontend and Spring Boot for the backend. We're opting for React paired with JavaScript. Additionally our indirect use of HTML5 and CSS3 ensures support across all modern browsers. Our backend infrastructure relies on Java, Spring Boot, and MongoDB, with deployment planned on AWS servers. Considering the yearly student population of Bilkent University being around 12000 people, the number of student clubs in Bilkent University in the 2023-2024 school year being 111 and the total graduate number being around

55000 we anticipate a maximum of 15000 users. We are aware that the resources provided under the AWS's Free Tier servers might not fully support our system expecting up to 15,000 users due to limitations on instance types, capacities, and duration. But we plan on utilizing the free tier server offered by AWS's RDS service db.t3.micro with the specifications of 2 vCPU, 1 GB memory and a 3.1 GHz Intel Xeon processor.[1] We plan to go on a paid tier server from AWS considering the number of users.

2.3. Deployment Diagram

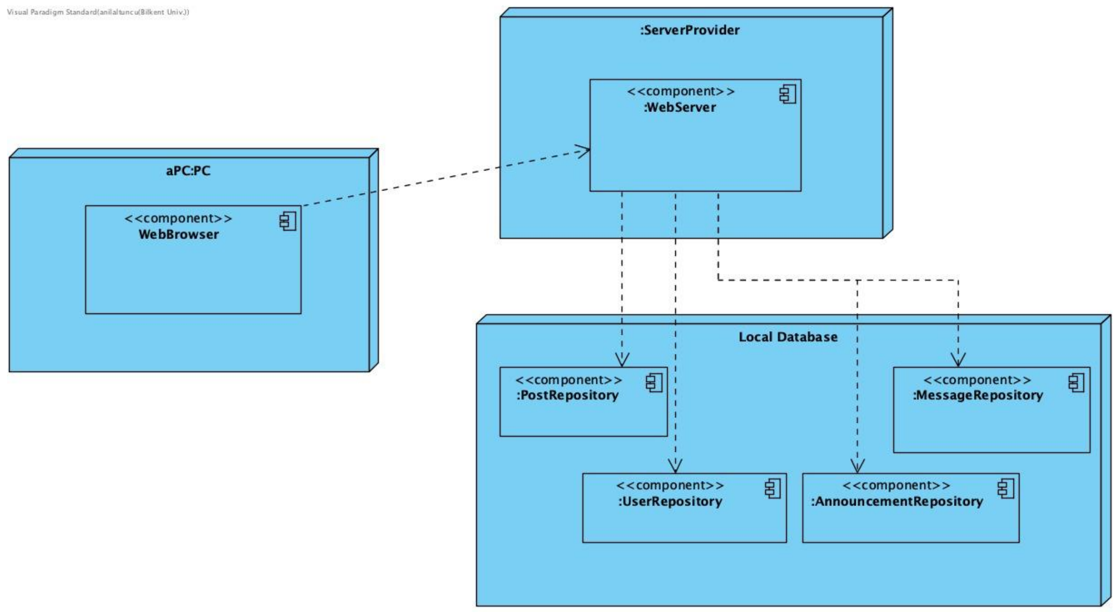
Visual Paradigm Standard(anilaltuncu(Bilkent Univ.))
:ServerProvider
<<component>> :WebServer
aPC:PC
<<component>> WebBrowser
Local Database
<<component>> :PostRepository
<<component>> :MessageRepository
<<component>> :UserRepository
<<component>> :AnnouncementRepository

Fig. 2 Deployment Diagram

Fig. 2 displays the interaction between user and server. Mainly the objects that inherit the Sendable class such as Post, Message, Announcement are stored in the database.

2.4. Persistent data management

OnlyBilkent system significantly relies on complex data, with different types of sendable objects like posts, announcements, and messages and different types of users like students, board accounts, and admin. Therefore, we chose a NoSQL database, MongoDB, to manage this data. MongoDB is a popular open-source NoSQL database management system that falls under the category of document-oriented databases. Document-oriented database MongoDB offers an intuitive data model that is fast and easy to work with and a flexible schema that allows the data model to evolve as application needs change.[2] Unlike relational databases, document databases' schema is dynamic and self-describing, so we don't need to pre-define it in the database first. Fields can vary from document to document. Structures can be modified at any time, avoiding disruptive schema migrations.[3] We will use MongoDB for user and sendable object data. Moreover, other data, such as images and documents, will be stored in the remote file system and will be accessed via file paths stored in the database.

2.5. Access control and security

OnlyBilkent is a web-based application, and it is exclusively for Bilkent University students and graduates. Therefore, our app requires enhanced security and authorization. We use Node.js for our back-end, and Passport.js is a widely used authentication middleware of Node.js that makes it easy to implement authentication and authorization. Passport.js cleanly encapsulates this functionality while delegating unrelated details, such as data access to the application.[4] The access control is managed by attaining roles for users. Each of the users has certain roles assigned to them. These roles are Student, Graduate, Board Account, and Admin. We use protected routes to restrict the access of different user types to certain user interfaces. This way, users can only see their own user interfaces.

2.5.1 Access Control Matrix

	Student	Board Member	Graduate	Admin
Login	X	X	X	X
Edit Profile	X	X	X	X
Renew Password	X	X	X	X
Create Post	X		X	
Edit Post	X		X	
Ban Profile				X
Add Announcement		X		
View Announcement	X	X	X	X
Remove Announcement		X		X
Edit Announcement		X		X

Report Profile	X		X	
View Reports				X
Search Post	X		X	
Direct Message	X		X	
Give Board Member Permission				X
View Board Member Requests				X
Request Board Member Account	X			

3. Glossary and References

[1] “Amazon RDS Instance Types | Cloud Relational Database | Amazon Web Services,” [Online]: https://aws.amazon.com/rds/instance-types/.
[2]”MongoDB Documents” [Online]:https://www.mongodb.com/document-databases, [Accessed- 14 Nov, 2023]
[3]”MongoDB Documents” [Online]:https://www.mongodb.com/document-databases#what-makes-document-databases-different-from-relational-databases [Accessed- 14 Nov, 2023]
[4]”Passport.js Documents”:https://www.passportjs.org/concepts/authentication/password/ [Accessed- 14 Nov, 2023]