SENG_491/CMPE_491

Graduation Project

High-Level Design Report

For

KariyerLAB

TEDU Software/Computer Engineering Department

15.12.2024

Prepared by

Mustafa Karakuş	**Student ID : 48034466522**
Ahmet Hakan Aksoy	**Student ID : 30649805858**
Anıl Çelik	**Student ID : 29407012522**

Advisor:	**Elif Kurtaran Özbudak**
Jury:	**Tansel Dökeroğlu**
	Özlem Albayrak
	Venera Adanova

1. Introduction

Today's students face numerous challenges in building their careers, particularly in finding reliable internships, job opportunities, and scholarships. The increasing prevalence of misleading or fraudulent advertisements leads to wasted time, effort, and frustration. Many students struggle to access accurate information about working conditions, organizational culture, or feedback from previous employees, forcing them to make critical career decisions without adequate transparency. Additionally, the complexity of application processes and the lack of feedback from companies further discourage students, causing a decline in motivation and missed opportunities.

The digitalization surge has intensified competition, emphasizing the need for platforms like VizyonerGENÇ, which provides students with resources and opportunities to grow professionally. Inspired by such initiatives, KariyerLAB aims to go a step further by creating a centralized platform where students can confidently apply for internships, jobs, and scholarships. By enabling students to rate companies, the platform will discourage exploitative practices and promote accountability. Core features such as application tracking, document uploads, advertisement filtering, and feedback systems will empower students to make informed decisions and take transparent steps toward their career goals.

KariyerLAB aspires to foster a trustworthy ecosystem connecting university students, companies, and scholarship-granting institutions. This would ensure a more efficient and reliable pathway to success while addressing the gaps many young professionals face today.

1.1 Purpose of the System

KariyerLAB offers the opportunity to evaluate companies by going one step beyond the currently used application sites.

Students will be able to evaluate companies based on the results of their applications and indicate how the process went. Thus, companies are planned to manage the process transparently to protect their brand values.

According to the feedback, companies can see the necessary data to improve their process experiences.

1.2 Design Goals

1.2.1 Performance

The list of applications should be displayed to users in 2 seconds.
Query responses will take under 1 second.

1.2.2 Safety

Data Backup and Recovery: Implementing regular data backups and establishing procedures for data recovery in case of system failures or data loss incidents.

Disaster Recovery Plan: Develop a comprehensive plan to restore system functionality at the end of a natural disaster, cyberattack, or other unforeseen incidents.
Secure Transmission of Sensitive Information: Ensuring that sensitive information such as payment details or personal data is transmitted securely over the network using encryption and secure protocols (e.g., HTTPS).
Access Control and User Permissions: Implementing access control mechanisms to restrict unauthorized access to sensitive data and functionalities within the system.
Error Handling and Logging: Implementing robust error handling mechanisms to gracefully handle unexpected errors and log relevant information for troubleshooting purposes without exposing sensitive data.

1.2.3 Security

Internal Security Requirements: For internal security, the system uses TLS encryption for every data exchange. It performs routine security checks to prevent any risks and limits who can see what based on their job.
JWT Security Requirement: JWT is necessary to authenticate user sessions and securely transport identity information. Measures such as properly signing and encrypting JWTs, ensuring reliable key management, and setting appropriate token expiration times should be taken.
CORS Security Requirement: CORS must be configured correctly in the web application. This involves configuring CORS to allow the browser to load resources from other domains in a manner that does not pose security risks. Incorrect CORS settings may expose the application to risks where attackers could exploit security vulnerabilities.

1.2.4 Testing

The system should be tested regularly under different scenarios. Performance, security, and functionality tests should be performed to ensure that the system works correctly. In addition, improvement studies should be carried out based on user feedback.

1.2.5 Accessibility

It should be fully compatible with mobile devices and different screen sizes. The page's structure and content should be arranged to improve the user experience on different devices. The menu and page structure should be simple, understandable, and easy to navigate. Users should be able to easily access the information they are looking for.
Drop-down menus should be accessible with the keyboard or using assistive technology. In addition, menus and other navigation elements should work smoothly on mobile devices.

1.2.6 Exception Management

Users' errors should be handled appropriately by the system. If there is an issue, the user should be given instructions on what to do and be presented with informative error messages

1.2.7 Availability

Our initial target audience will be in Turkey; therefore, bug fixes and updates will be scheduled for midnight to minimize any potential disruption to user access.
Our system will be designed and published to be accessible 24/7, except for updates and developments.
A local network or cellular network connection is required to access the system.

1.2.8 Maintainability

Our subsystems will be developed using microservice architecture and the subsystems will be loosely coupled to each other so that maintenance can be done easily. Also, during updates, any new modules will be easier to integrate and will not affect other subsystems.

1.2.9 Usability

Users will not need to spend time learning how to use the application. The interface will be intuitive and user-friendly, ensuring seamless and straightforward interactions.

1.2.10 Legal and Regulatory Requirements

Personal Data Protection Law (KVKK): The provisions of KVKK must be applied for a system operating in Turkey.

There will be rules preventing unauthorized e-mail and SMS sending from the user.

Use of encryption to protect sensitive user information.

Preparation and approval of legal texts determining the user rights and limits of our site.

The images, software codes, or content used must comply with the licenses.

Compliance with copyright laws.

1.3 Overview

KariyerLAB is a centralized platform designed to address university students' challenges while building their careers. The platform aims to simplify the process of finding internships, job opportunities, and scholarships by providing a reliable and transparent environment. With features such as company ratings, applicant tracking, document uploads, and ad filtering, KariyerLAB empowers students to make informed decisions. The platform discourages exploitative practices by promoting accountability among companies and enables students to confidently take steps toward their career goals.
By connecting students, companies, and scholarship providers, KariyerLAB aims to create a reliable ecosystem that bridges the gap between young professionals and career opportunities, providing an effective and supportive path to success. KariyerLAB will be done in 3 parts, while creating this ecosystem, it stores the necessary information of students and companies in a database. It then sends this information to a Front-End interface for users to use the site via Back-End connections.

2. Proposed Software Architecture

2.1 Overview

This section will cover the decomposition of subsystems, hardware and software mapping, persistent data management, access control, software control mechanisms, and boundary conditions. Each topic will be explained in detail individually to ensure a clear understanding of the application's structure and logic.

2.2 Subsystem Decomposition

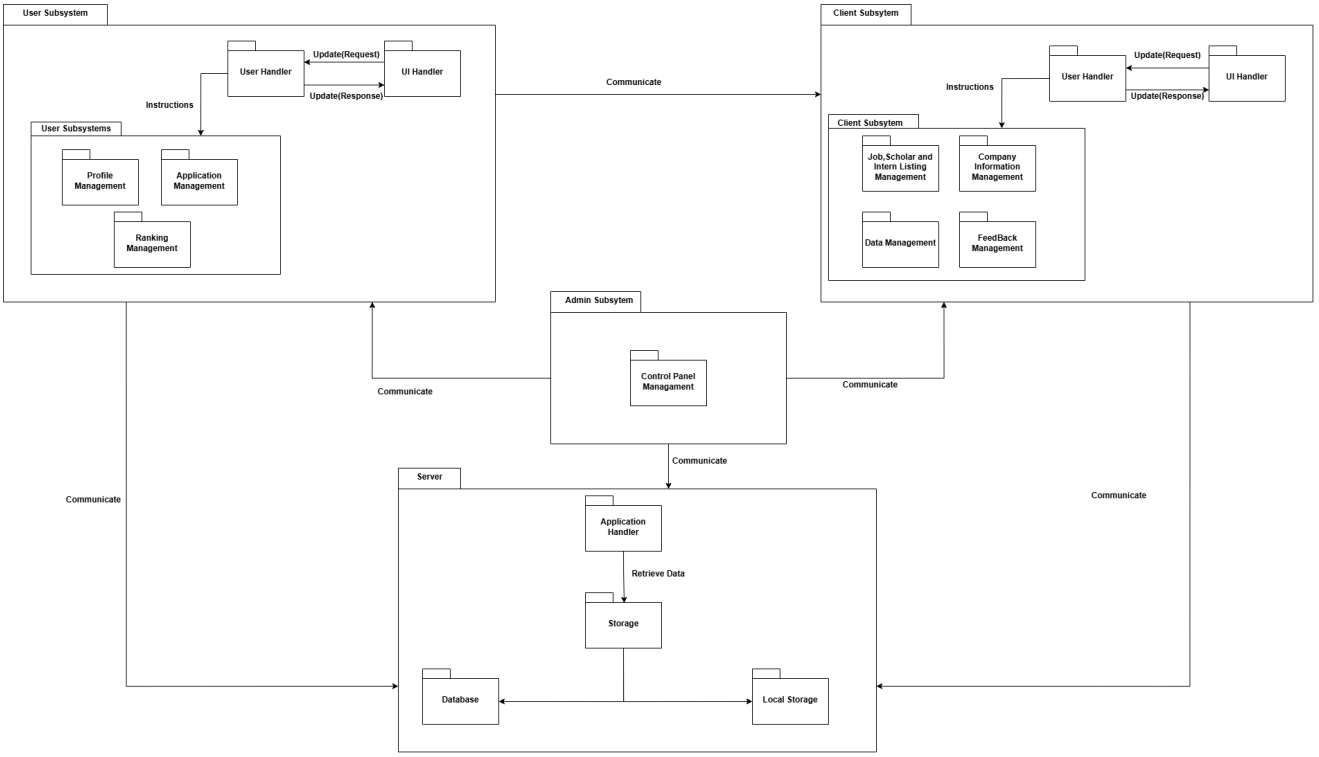
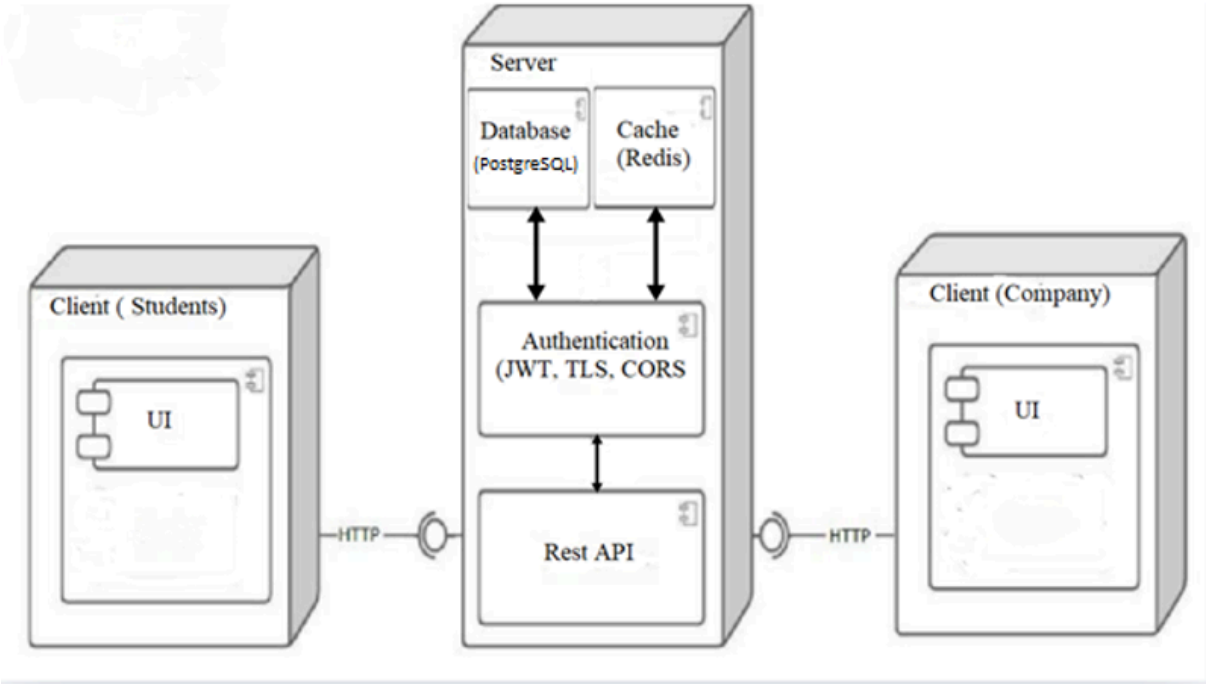


Figure 1: System Decomposition Diagram

2.3 Hardware/Software Mapping

This software and hardware map is a diagram that illustrates how the specified requirements are addressed. The software layer comprises the user interface (UI), database, REST API, and business logic layers. Users perform tasks such as creating CVs, applying for internships, and writing comments through the user interface, while the REST API layer securely communicates these actions with the database. Additionally, the business logic layer manages tasks like application submissions and comments. The database is optimized to meet security and performance requirements, JWT is used for user authentication, and all data transmission is encrypted via HTTPS. The hardware layer includes the server, data storage, and backup systems. The server is equipped with sufficient

processing power to meet high-performance demands and supports regular data backups and disaster recovery processes. This structure effectively addresses both functional requirements and technical needs such as security and performance.



2.4 Persistent Data Management

In the context of the provided requirements, persistent data management refers to the efficient storage, retrieval, and security of data throughout the system's lifecycle. The system will manage user data, job applications, CVs, comments, and evaluations using a relational database, ensuring that all critical information is securely stored and readily accessible. To meet the performance requirements, the database will be optimized with indexing and caching strategies to handle large volumes of data efficiently. Furthermore, data integrity and security will be ensured by regular backups, secure data transmission (using HTTPS), and proper access control mechanisms. The system will also provide mechanisms for disaster recovery, ensuring that data can be restored in case of system failures or unexpected incidents

2.5 Access Control and Security

Access control and security for the system will be implemented through strong authentication and authorization mechanisms. User access will be carefully managed to ensure that only authorized individuals can access sensitive data and functionalities. Secure authentication will be handled using JWT (JSON Web Tokens), which will ensure that user sessions are properly authenticated and that identity information is transmitted securely. All sensitive data, including personal information and job application details, will be encrypted during transmission using HTTPS to prevent unauthorized interception. Additionally, access to specific functionalities, such as managing user data or handling applications, will be restricted based on the user's permissions, ensuring that only authorized users can perform certain actions. Regular security audits and the application of security best practices will be conducted to identify and address potential vulnerabilities, ensuring the integrity and confidentiality of the system.

2.6 Global Software Control

2.6.1 Control Mechanisms

The KariyerLAB system includes general control mechanisms and processes of the software. These mechanisms are designed to ensure the safe and smooth operation of the system. Here are some important points:

- Error Management: The system detects and manages errors that may occur. Errors are reported to the user with understandable messages and reported to system administrators when necessary.
- Security Controls: Various security measures are taken to protect user data. These measures include data encryption and access controls.
- Access Controls: Users are given access to different parts of the system based on their roles, so only authorized individuals can access sensitive data.

2.6.2 Backup and Recovery

The KariyerLAB system makes regular backups to prevent data loss and stores these backups on secure servers. Thus, data can be easily restored in case of any problems.

2.6.3 Update and Maintenance

The KariyerLAB system is supported by continuous updates and maintenance work:

- Updates: The system is updated regularly to add new features and improve existing ones.
- Maintenance Work: The system is supported by regular maintenance work. These works are carried out to optimize system performance and detect potential problems in advance.

2.6.4 Performance Monitoring

System performance is continuously monitored and optimized:

- Monitoring Tools: Various tools are used to monitor system performance. These tools constantly monitor the overall health and performance of the system.
- Optimization: Performance data is analyzed and necessary optimizations are made for the system to operate more efficiently.

2.6.3 Compliance and Standards

The KariyerLAB system is designed in accordance with industry standards and legal requirements:

- Industry Standards: The system has been developed in accordance with accepted industry standards in software development and data security.
- Legal Compliance: The system operates in full compliance with legal regulations and data protection laws (KVKK) in Turkey.

2.7 Boundary Conditions

2.7.1 Initialization

Users can start KariyerLab by logging in. If you are a new user, you will need to register. After logging in to the system, you will reach the home page. If you are logged in as a student, you can view the published advertisements. If you are logged in with a company account, you will be greeted by the advertisements you have previously published.

2.7.2 Termination

Users can safely close their sessions by logging out of the system from the relevant section (profile section) in the web interface.

2.7.3 Failure

If the user encounters any error while using the application, for example, if the user cannot comment due to the input entered in the wrong format or if the user cannot update the photo due to the wrong format selection, the user is notified with an error message with an appropriate toast.

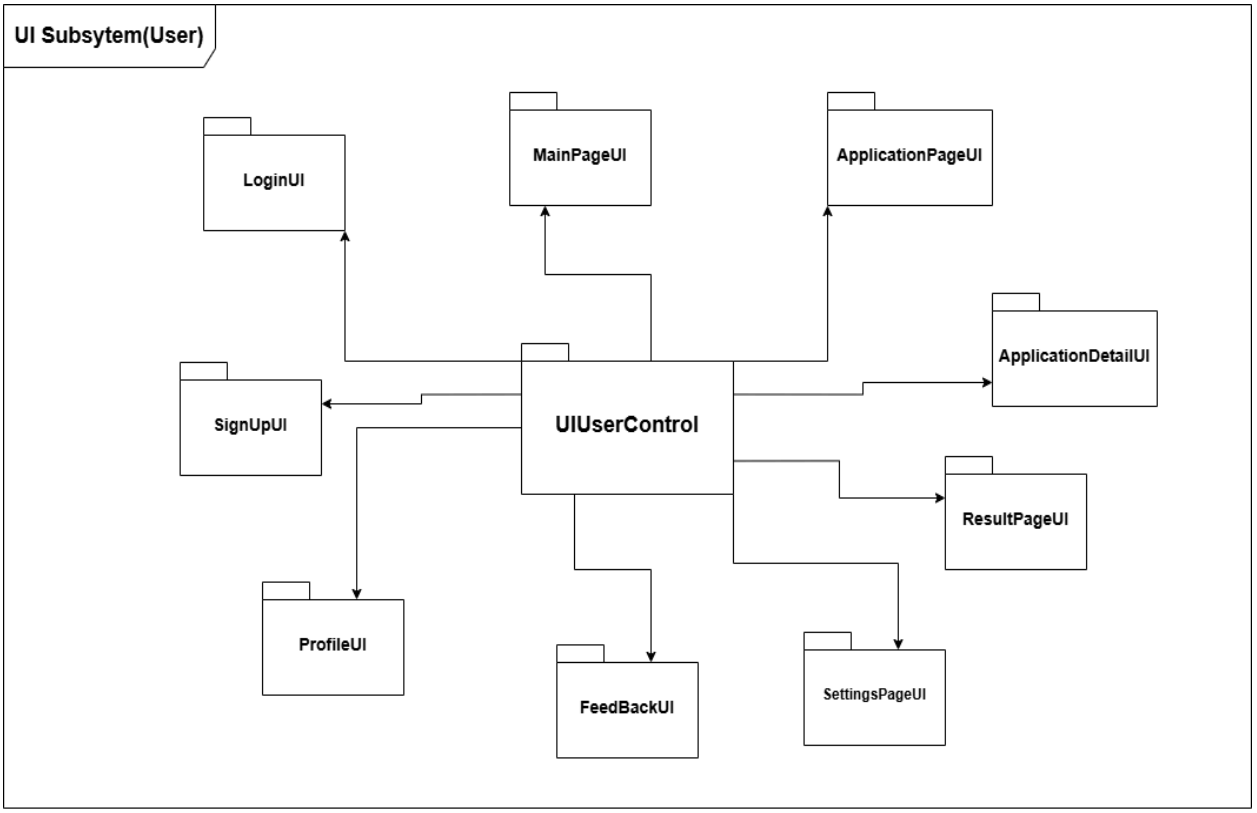
If the user's internet connection is interrupted while using KariyerLAB, the user will not be able to access KariyerLab anyway. When the connection is re-established, the user can continue to use KariyerLab without any problems.

When KariyerLAB works with third-party REST APIs, users may not be able to use certain functions when one of these REST APIs does not respond or an unexpected error occurs. For example, if an error occurs while retrieving company information via a REST API, the user is shown a message stating that there is a REST API error and the operation cannot be performed at this time.

3. Subsystem Services

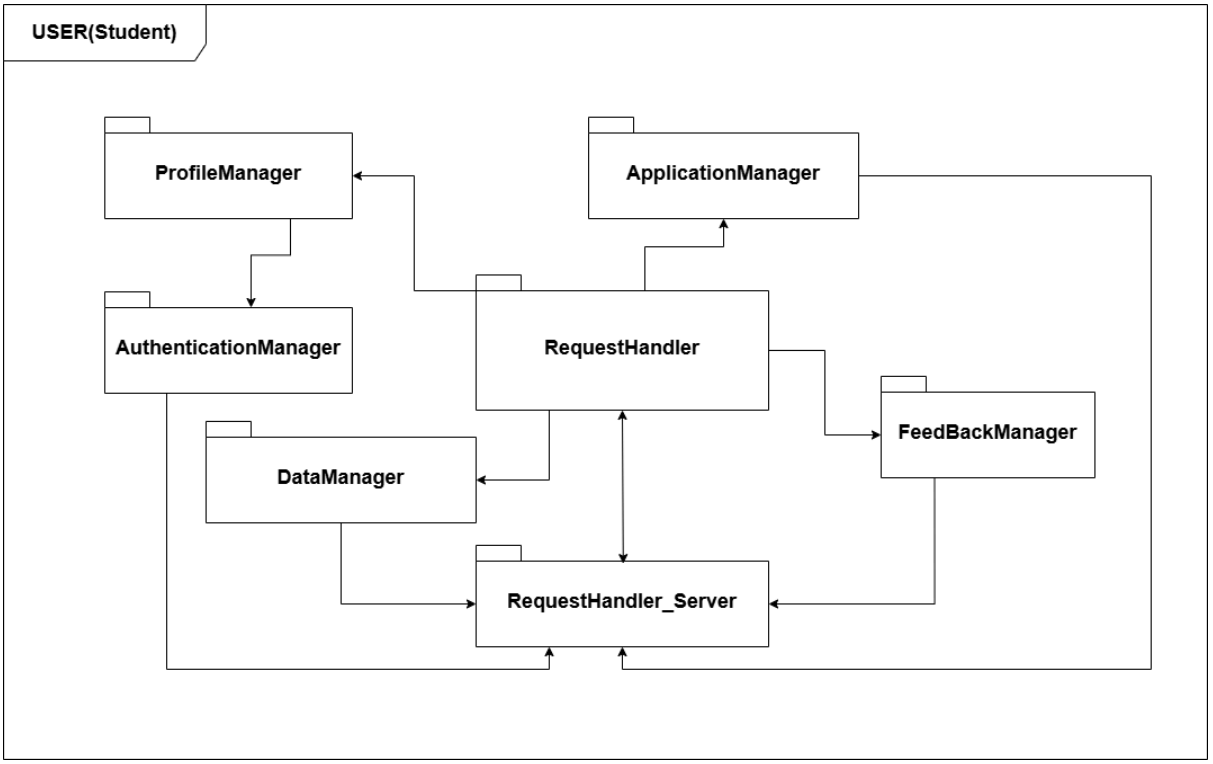
3.1 User Subsystem (Students)

3.1.1 UI Subsystem



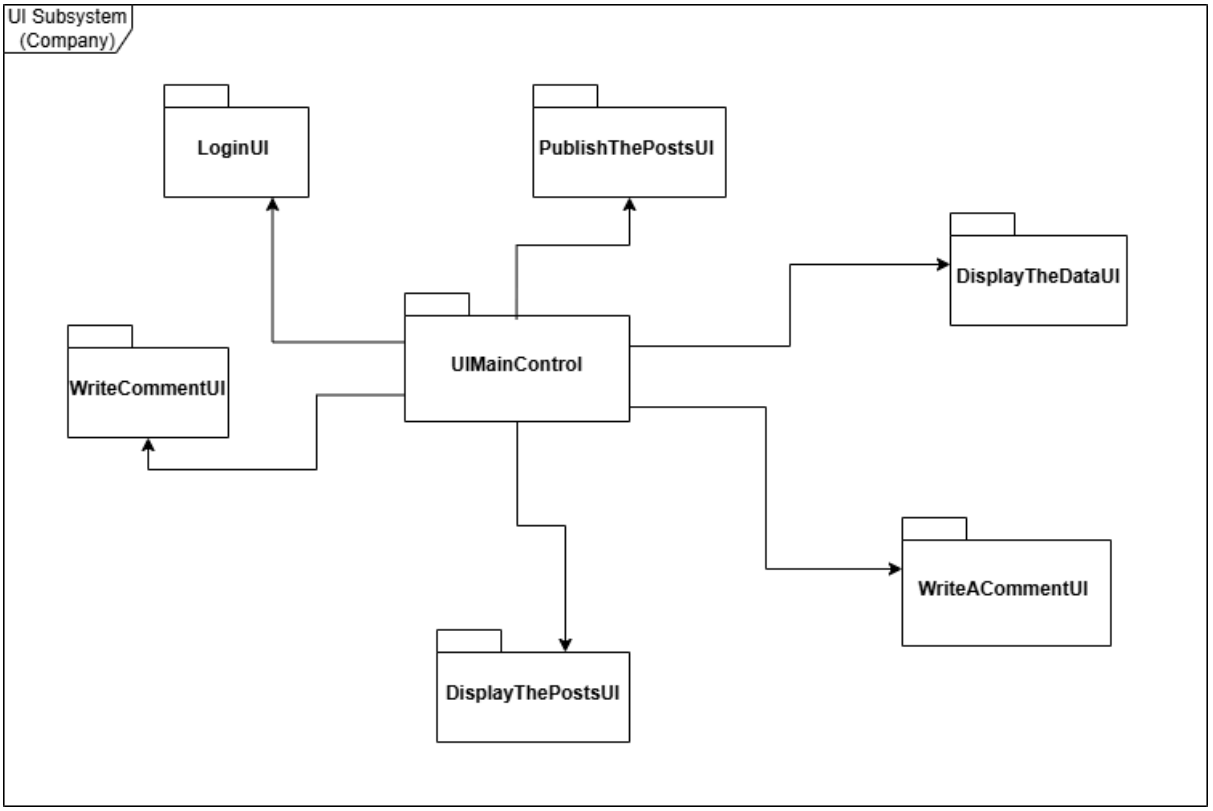
LoginUI: User interface for the login.
SignupUI: User interface for the signup.
ProfileUI: User interface for the arrange their profile settings.
FeedbackUI: User interface for sending feedback.
SettingsPageUI: An interface where the user can make personal settings within the platform.
ResultPageUI: The interface where users will see the application results.
ApplicationDetailUI: An interface where users can get information about the job postings they will apply for and see the conditions.
ApplicationPageUI: The interface where users will apply.
MainPageUI: The main screen interface that users will see after logging into the site.

3.1.2 User Subsystem



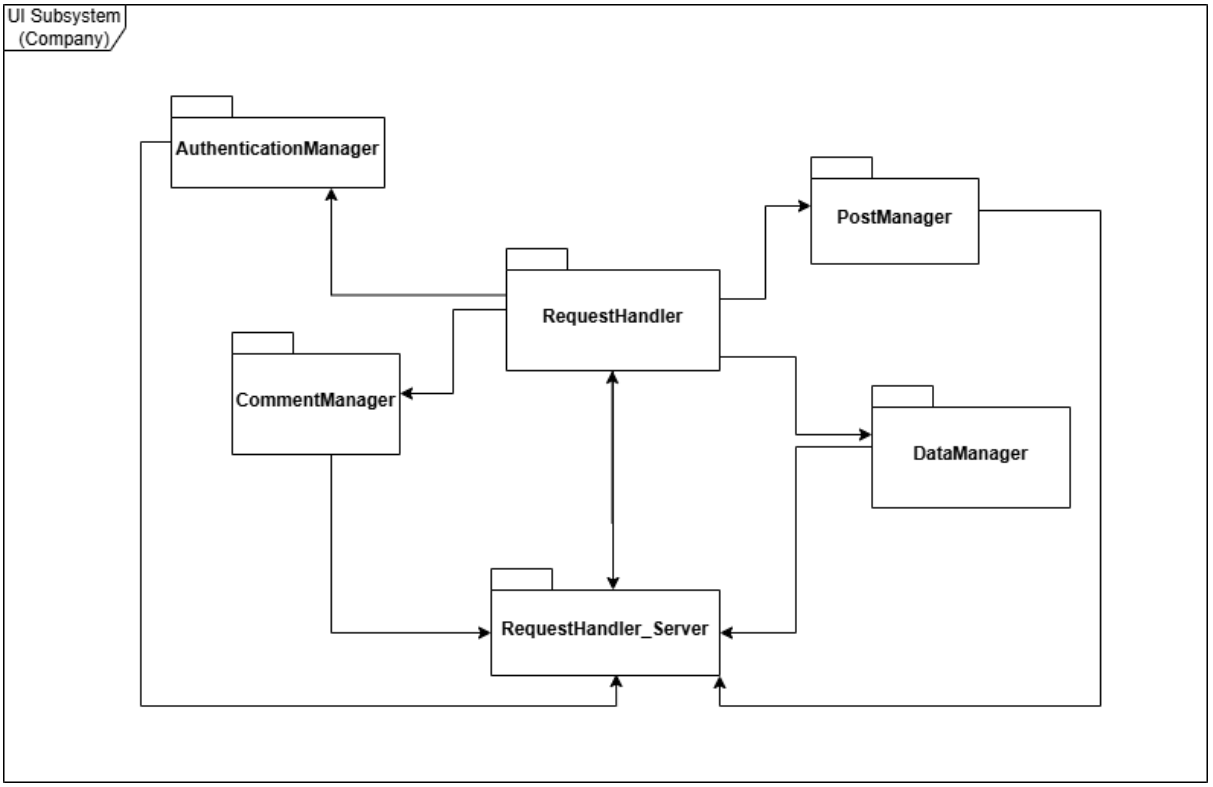
RequestHandler: Processes responses from the user interface and transmits the relevant data to the server or other system components.
RequestHandler_SERVER: Responsible for processing responses from the server or controlling system components, sending appropriate responses to the user interface.
AuthenticationManager: Manages the authentication process to ensure a secure connection with the system server.
ProfileManager: Controls the management and modification of profile details when necessary.
ApplicationManager: Manages users' applications for job postings opened by companies.
FeedBackManager: Manages evaluations made by users.
DataManager: Allows users to manage evaluations made by users in the database.

3.2.1 User Subsystem (Company)



LogInUI: User interface for the signup.
DisplayTheDataUI: Interface used to display data to the user.
WriteACommentUI: Interface for users to write comments.
DisplayTheDataPostsUI: Interface used to display posts to the user.
PublishThePostsUI: Interface for users to publish their posts.
WriteCommentUI: Interface allowing users to write comments on posts.

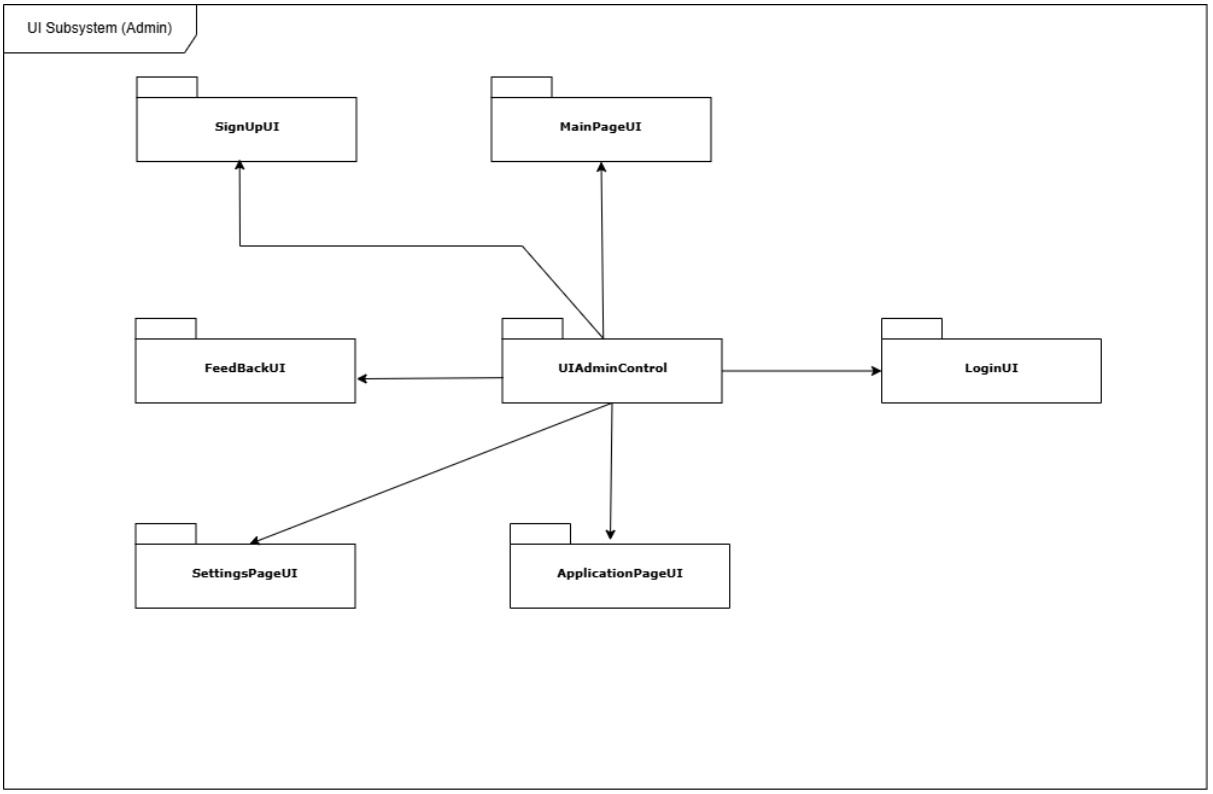
3.2.2 User Subsystem (Company)



AuthenticationManager:Handleslogin and authentication process.
PostManager: Manages publishing and displaying the posts.
CommentManager: Manages user comments.
DataManager:Manages data storage and retrieval.
RequestHandler:Process responses from the user interface and transmits the relevant data to the server or the system components.
RequestHandler_Server: Receives and provess incoming requests from the client side.

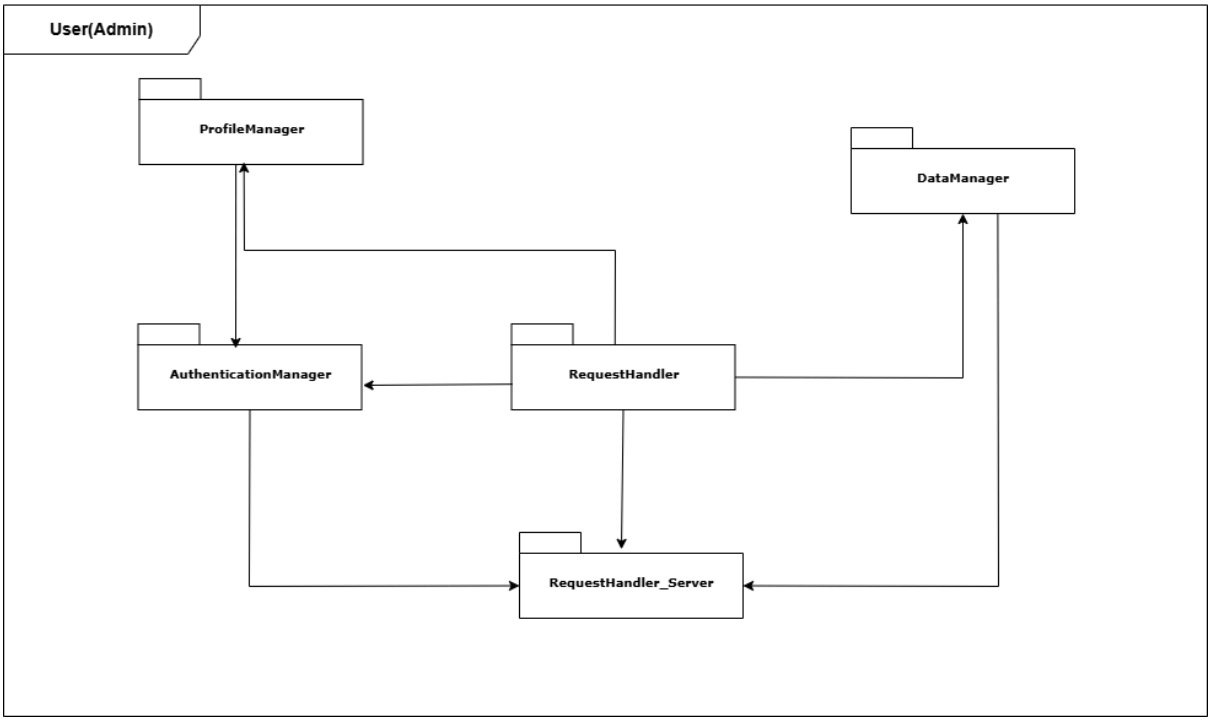
3.3 User Subsystem (Students)

3.3.1 UI Subsystem



LoginUI: User interface for the login.
SignupUI: User interface for the signup.
FeedbackUI: User interface where feedback is reviewed and deleted if necessary.
SettingsPageUI: An interface where the user can make personal settings within the platform.
ApplicationPageUI: Interface where necessary arrangements can be made for opened advertisements and applications.
MainPageUI: The main screen interface that users will see after logging into the site.

3.3.2 User Subsystem (Admin)



RequestHandler: Processes responses from the user interface and transmits the relevant data to the server or other system components.
RequestHandler_SERVER: Responsible for processing responses from the server or controlling system components, sending appropriate responses to the user interface.
AuthenticationManager: Manages the authentication process to ensure a secure connection with the system server.
ProfileManager: Controls the management and modification of profile details when necessary. Keeps the details of profiles opened as Admin.
DataManager: Allows the admin to monitor interactions made by users, such as deleting a comment or removing an inappropriate post.

3.4 Server Subsystem

The server subsystem is comprised of three primary components: the remote server layer, the mapping layer, and the data storage layer.

3.4.1 Remote Server Layer

This layer manages incoming HTTP requests via a REST API while maintaining security protocols. To process and handle these requests effectively, we utilized Express.js. This framework simplifies integrating functionality into the request lifecycle, enabling efficient handling of HTTP interactions

3.4.2 Application Subsystem

KariyerLAB consists of two main components: application creation and application submission. Companies are responsible for creating internship, job, and scholarship listings, while students can browse and apply for these opportunities. Once an application is submitted, it is routed to the system's data storage for further processing. This allows potential matches to be identified for students or companies, and the matched opportunities are made available for review in a streamlined manner. Students can input their preferences, such as skills, interests, or location, while companies define the requirements and details of their postings. When an application or new listing is created, the system utilizes a matching algorithm that evaluates compatibility between the candidates and the opportunities available. This algorithm runs server-side, ensuring students are not burdened with running complex calculations on their devices. To ensure accountability and transparency, the subsystem allows students to rate and review their experiences with companies. This feedback mechanism provides valuable insights for both students and organizations. Companies can analyze these reviews to improve their processes, while students gain clarity about working conditions, organizational culture, and other essential factors.

3.4.3 Data Storage Subsystem

Data is stored for later use.
PostgreSQL is used to store users' information, companies' information, application information, comments, and results.
Local storage is used to store users' relevant critical informatio,n such as profile photos, and critical information (ID number).

4. Glossary

REST API: Representational State Transfer (REST) is a stateless architectural style based on a client-server design model.

HTTP Request: HTTP, as an application layer protocol, facilitates the transfer of hypermedia documents.

JWT: JSON Web Token is a compact, self-contained token used for securely transmitting information between parties as a JSON object, often for authentication and authorization.

KVKK: The law that was created to protect users' data. (Kişisel Verilerin Koruma Kanunu)

Admin: The people who have the system owner.

Users (Students): Students who registered in the system

Client (Companies): Companies posting ads on the website.

5. References

[1] GeeksforGeeks, "REST API Introduction," GeeksforGeeks, https://www.geeksforgeeks.org/rest-api-introduction/, accessed December 27, 2024.

[2] Cloudflare, "What is HTTPS?" Cloudflare, https://www.cloudflare.com/learning/ssl/what-is-https/, accessed December 27, 2024.

[3] PostgreSQL Documentation, "PostgreSQL: The World's Most Advanced Open Source Relational Database," PostgreSQL, https://www.postgresql.org/, accessed December 27, 2024.

[4] Node.js Documentation, "About Node.js," Node.js, https://nodejs.org/en/about, accessed December 27, 2024.

[5] JWT.io, "Introduction to JSON Web Tokens," JWT.io, https://jwt.io/introduction, accessed December 27, 2024.