CS319 - Object Oriented Software Engineering

D4: Design Goals, High Level Architecture

Ahmet Kaan Sever - 22102278
Atakan Şerifoğlu - 22102313
Dila Tosun - 2212100
Konuralp Kılınç - 22101791
Perit Dinçer - 22103102
Selimhan Tokat - 22003145

1. DESIGN GOALS

1.1. Purpose of the System

The purpose of the system is to develop a reliable and easy to use online platform exclusively for the Bilkent University community for all the things related to second-hand sales, borrowing, donations, lost and found items. The system will be used by various community members such as students, academic staff, and administration personnel.

1.2. Design Goals

1.2.1. Reliability

Reliability is one the fundamental aspects of the platform. The goal is to ensure consistent and dependable performance. It is aimed to minimize downtime and errors, providing the users with a reliable and stable environment for their activities such as adding new items, looking for items, or checking up the status of their listed items. The technical part of the platform is planned to reach the coding standards to ensure the users trust the platform by delivering a seamless and error-free experience.

1.2.2. Maintainability

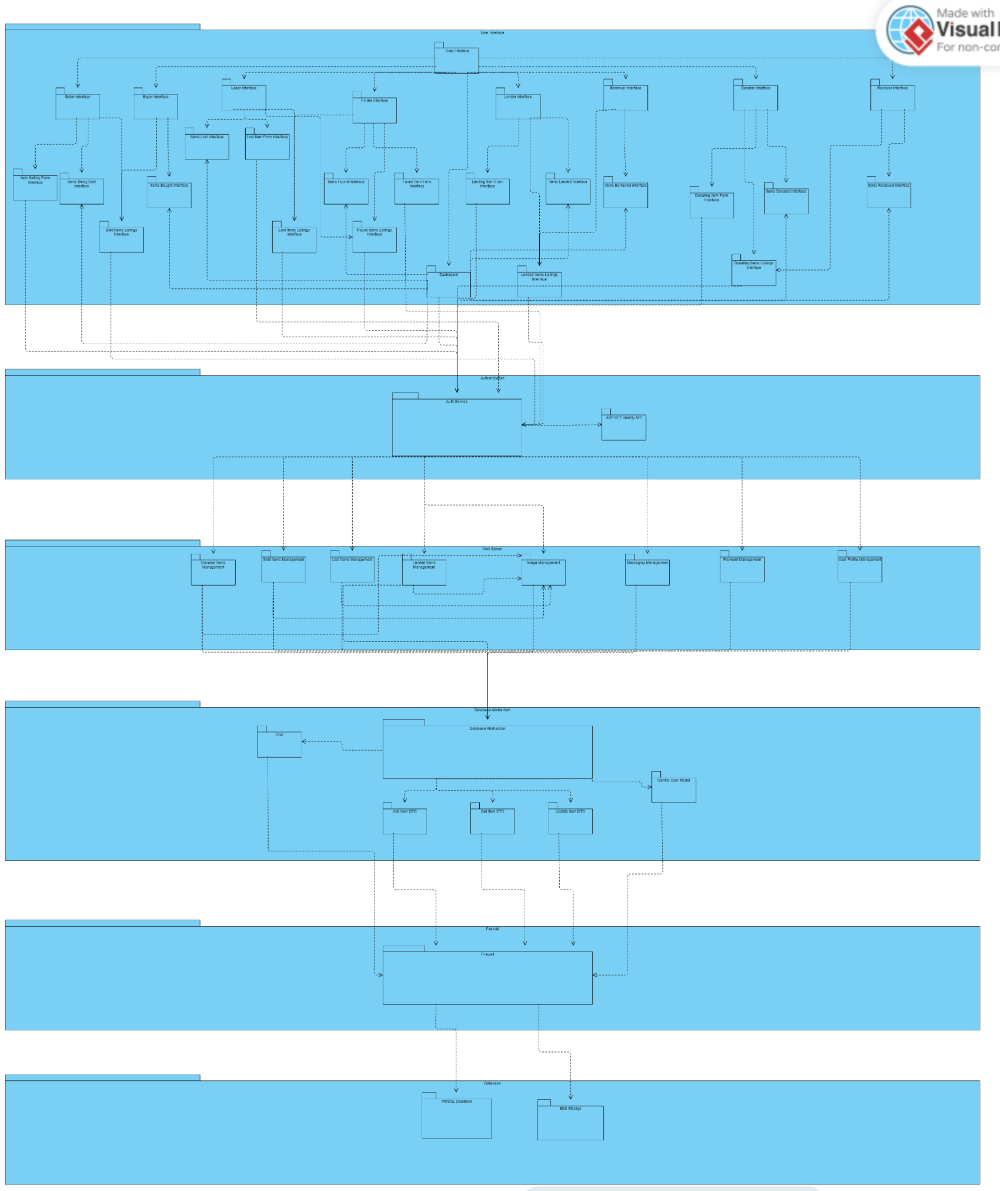
The maintainability of the platform is a key focus for long-term sustainability and ease of management. The system is dynamic with users adding new items to the webpage, and deleting the bought items from it. The item tracking statuses inside the system are made to be easily recognizable and modifiable for the future maintainability of the platform.

1.2.3. User Friendliness

The platform has a strong emphasis on user-friendliness to ensure a positive and intuitive experience for the Bilkent University community. The user interface has been designed to facilitate easy navigation across pages and items. The design is consistent among pages to increase similarities across features so that it is less complicated for the user. Users have the flexibility to personalize their experience, tailoring profiles and settings to their preferences. Visual clarity is optimized to eliminate unnecessary complexity, fostering a clean and straightforward design.

2. HIGH-LEVEL SOFTWARE ARCHITECTURE

2.1. Subsystem Decomposition



For a higher quality image: **Layer Diagram.png**

2.1.1. User Interface Layer

The user interface layer is responsible for dividing the possible interfaces and functions a user may use while engaged with the application. Listings for possible item types and forms for users to fill out are presented on this level, and the input data is passed down onto the layers below. The dashboard of the user is also presented here, which shows all the possible items the user has interacted with.

2.1.2. Authentication Layer

Authentication is done primarily through the ASP.NET Identity API. The information for login and register are passed through the API and proper authentication tokens are given for users. Only after proper authentication has been achieved will the user be allowed to interact with the functions of the below layers.

2.1.3. Web Server Layer

The Web Server Layer is responsible for the Web API operations. It consists of controllers which are responsible for handling HTTP requests and managing the communication between the server and the client. Controllers create, read, update and delete database items through database abstraction models. The Web Server Layer also executes the server-side code and executes the rendering of the proper HTML which will be sent to the client.

2.1.4. Database Abstraction Layer

Database abstraction layer consists of DTO’s, Chat Model and Identity User Model. Abstraction models are used to make the communication between the server and the database easier and modular. Each object represents a row in the database or a file in the Blob Storage. Also, DTO’s are used to control and limit the data flow between the server and the client. It prevents unnecessary data transfer by controlling the amount of data sent and received during add, update and read actions..

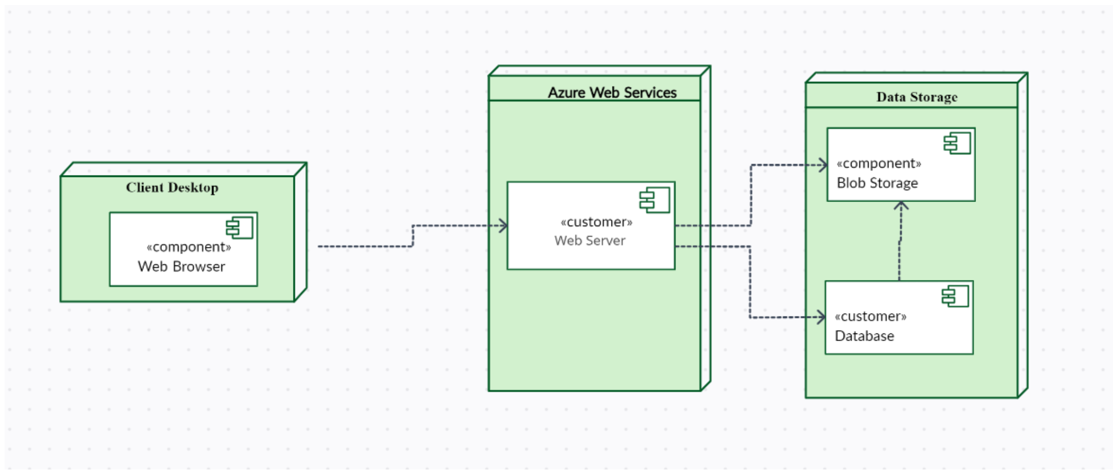
2.1.5. Firewall Layer

The Firewall layer is functionalized to check the IP’s of the computers trying to access the Database layer. Systems that have registered and allowed IP’s in Azure cloud system can read and write data according to its permissions. This allows the firewall layer to increase the security of the system by only allowing the authorized systems(in most cases it’s the server) to access the data.

2.1.6. Database Layer

Database layer is the data archive which comprises MYSQL Database and Azure Blob storage. Different database types allow the app to store several different items varying from chat description to listing images. Specifically, images and listings are stored at Blob storage while listing and user descriptions are stored at MYSQL Database. Files at the Blob Storage are linked to the users by file names at the database.

2.2. Deployment Diagram



2.3. Hardware/Software Mapping

The front-end of our web-based application is powered by the React.js framework (version 18.2.0). This choice was made due to React's versatility, allowing for extensive code reuse and delivering optimal performance compared to alternative frameworks.

On the backend, we have opted for the capabilities of .NET 6. This choice aligns with our goal of ensuring a scalable and well-maintained architecture, making use of the latest features and improvements offered by the .NET framework.

For data management, our project utilizes the Microsoft SQL Server (MSSQL), which offers a reliable and scalable database solution. To enhance our accessibility and create a seamless deployment process, we have chosen to host our MSSQL database on the Azure cloud platform. This decision stems from Azure's reputation for providing a secure and easily maintainable cloud environment.

In terms of hardware requirements, our project is designed to be highly accessible. Since it is a web-based application, users can access it through any device equipped with a modern browser. This inclusivity ensures that our application is compatible with various devices, including both computers and mobile phones. In addition it eliminates the need for any additional peripherals or accessories.

No specialized hardware components are necessary for our project, making it easily accessible to a broad user base. Whether users access the application on a desktop computer or a mobile device, the only requirement is a device capable of running a modern web browser. The application supports the latest versions of

Chrome, Firefox, Safari, and Edge, as well as older versions of these browsers back to Chrome 41, Firefox 21, Safari 6, and Edge 12.

2.4. Persistent Data Management

Our application will hold a large amount of data for a large number of users; therefore it was paramount to choose the proper database solutions to house such a project. We decided on using Azure MSSQL for storing data about the users and items, as its relational and table-based structure is extremely convenient for our data flow. It is also very easy to integrate with the Identity API for user authentication, which is another reason for it being chosen. The Azure Blob storage was chosen for storing images for items within the application because storing images in the database as byte form is inconvenient as it is slow to write and read. Blob Storage allows the server to store files in directories with their names. The stored files are mapped to the users in the database with their names. File names are stored in the DB and the file itself is in the Blob Storage.

2.5. Access Control and Security

Our platform is a web-based application, thus we provide an enhanced security and authorization to our users. The access control system is managed by attaining roles for users. Each of the users has certain roles that are assigned to them which are Buyer, Loser, Finder, Seller, Lender, Donator, Receiver, and Borrower. These roles are assigned to the user before certain actions such as buying an item.

On the client side, a user can only interact with the application after they have registered and logged in. Users are allowed to send and receive data from the backend through their authentication tokens, allowing us to ensure proper data safety within the app.

On the server side, a user's register information is stored onto the database; and their password is kept as a hash. When a user tries to log in, the appropriate info is cross-checked against the database, and if it is successful the user is assigned a temporary JWT authentication token and longer-lasting refresh token. This refresh token is used to repeatedly re-authenticate the user until its expiry date arrives.

Access Control Matrix

	Seller	Buyer	Loser	Finder	Lender	Borrower	Donator	Receiver
Register	X	X	X	X	X	X	X	X
Login	X	X	X	X	X	X	X	X
Change Password	X	X	X	X	X	X	X	X
Edit Personal Information	X	X	X	X	X	X	X	X
View Marketplace	X	X	X	X	X	X	X	X
View Item List	X	X	X	X	X	X	X	X
Add/Remove /Edit Found Items				X				
Add/Remove /Edit Lost Items			X					
Add/Remove /Edit Lended Items					X			
Add/Remove /Edit Marketplace Items	X							
Add/Remove /Edit Donated Items							X	
Send Message	X	X	X	X	X	X	X	X
Receive Message	X	X	X	X	X	X	X	X
See Item Details	X	X	X	X	X	X	X	X
Manage Contacts	X	X	X	X	X	X	X	X
View Contact Profile	X	X	X	X	X	X	X	X

2.6. Boundary Conditions

2.6.1. Initialization

Since CampusConnect is a web application, it is necessary for the user to log in to the website with an existing account in order to initialize the buying, selling, borrowing, donation and such processes. Once the user is logged in, the relevant data is fetched from the database to initialize the system. If the user is not logged into an account they won't be able to see any item list or manage the items. After log-in, the user will be directed to the dashboard, from where they can navigate to the rest of the application pages with the sidebar. When such a page is to be retrieved, pertinent data for the page will be pulled from the database.

2.6.2. Termination

There are multiple possible cases for CampusConnect to terminate. The first case occurs when the user logs out from their account. If a user logs out of the application, the last saved data of the user is stored in the database.

Secondly, an admin user may terminate the program for different purposes, such as maintenance or security concerns. In such a case, the users' data is saved to the database, and termination takes place.

Another possible termination case can be caused by an unexpected operation. In this case, the application might be forced to terminate. However, such kind of termination is mostly handled in the backend. In case of any such error, the user should be informed by necessary error messages and for the developer side, for quick identification there should be thorough logs that include error codes, timestamps, and contextual data.

Lastly, if a user's access token expires, the authentication service of the application will be informed. The authentication service will either refresh the access token of the user if it has not expired yet, which will not end with termination, or the user will be automatically signed off, causing the termination of the program.

2.6.3. Failure

If a problem occurs with the Azure hosting server, then the app will shut down automatically due to not having spare servers to relocate to. In such a case, the program and database revert to the last most recent state to ensure no data is lost or miswritten. A disruption in internet connectivity may also result in failure, which will require the user to log-in once more. In both cases, a pop-up will be presented to the user for debugging and problem diagnostics.