D4 Final

Design Goals

We are choosing understandability and user-friendliness as our top priority. In CampusConnect, we want to deliver ease of use for all users. No one should need to spend time learning an app in their busy Bilkent schedule. Thus, understandability and user-friendliness are our top design goals.

Understandability

Understanding the importance of clarity and comprehensibility, CampusConnect places a significant emphasis on the understandability of our platform. Recognising that a diverse range of users, each with unique perspectives and levels of familiarity with technology, will be engaging with our application, we have prioritised creating an easily understandable interface. To this end, we have designed a product to eliminate ambiguity and promote effective communication. Users interacting with CampusConnect can expect a clear and concise representation of information, allowing them to navigate the platform confidently. The straightforward and transparent design ensures that users can grasp the intricacies of new functionalities without undergoing a steep learning curve, fostering a sense of continuity in their experience with CampusConnect. By prioritising understandability, we aim to empower users with the knowledge and confidence to leverage our platform's capabilities fully. Whether users are tech-savvy or novices in digital environments, our design principles ensure that CampusConnect remains an accessible and transparent tool for Bilkenters.

User-Friendliness

By recognising the diverse array of people and personalities who will be engaging with our platform, whether they are contributing information or extracting valuable insights, the goal is to create a tool and an accessible and enjoyable experience for every user. Our user interface (UI) is intentionally designed to be straightforward, with simple event triggers that guide users seamlessly through the platform. This design philosophy draws inspiration from the familiarity of everyday experiences, ensuring that anyone who has ever navigated a market can intuitively grasp how to navigate and utilise CampusConnect. Moreover, our emphasis on user-friendliness extends beyond initial interactions to scalability. We recognise that users may need to scale their operations over time, and our intuitive design allows for seamless expansion. Whether users are inputting or outputting items, they can do so quickly and efficiently, adapting to the changing demands of their activities without encountering unnecessary complexity. One of our user-friendly approach's key advantages is eliminating boredom from the user experience. Complex and convoluted interfaces can lead to frustration and disengagement. In contrast, our design philosophy encourages users to interact with CampusConnect effortlessly, turning what might be perceived as a mundane task into a streamlined and enjoyable process.

Design Trade-offs

In accordance with our top priority design goals, understandability and user-friendliness, some design trade-offs have been made. The overall aim is to satisfy the needs of simple user behaviour.

Reliability vs. Modifiability

Prioritising reliability is foundational to establishing user trust and ensuring a consistent, stable user experience. In our application, where dependability is paramount for Bilkent user's trust, a reliable system fosters confidence among users. Users are more likely to perceive the system as user-friendly when it operates predictably, without frequent errors or disruptions. The emphasis on reliability aligns with the overarching goal of creating a system that users can confidently engage with and understand. As a trade-off, there is little room to modify CampusConnect since any fundamental change will require us to re-design and re-implement the entire software.

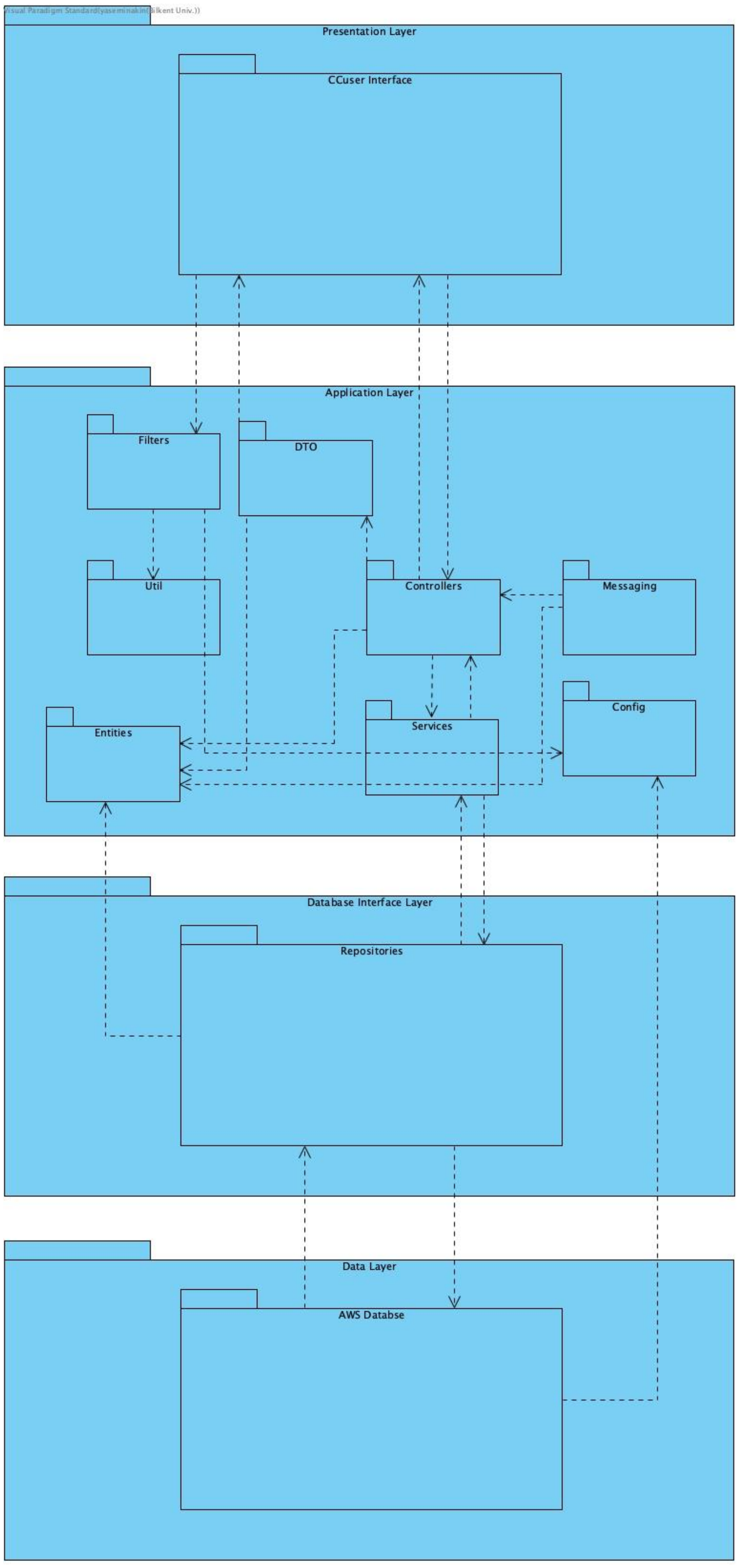
Efficiency vs. Flexibility

The prioritisation of efficiency contributes significantly to user satisfaction and resource optimisation. An efficient system enhances the user experience by providing quick responses and smooth interactions. Users find a responsive design more enjoyable to use, thus contributing to a positive perception of user-friendliness. However, increased efficiency implied decreased flexibility. Our operations support certain kinds of actions, and they are not supported. The users will need to fit their needs with the CampusConnect features.

Rapid Development vs. Backward-Compatibility

Prioritising rapid development aligns with the need for adaptability and a competitive edge in dynamic environments. Implementing minor changes quickly is a valuable attribute in maintaining a user-friendly system that remains relevant. Additionally, rapid development accelerates the time-to-market for new features and updates, facilitating continuous improvement. This iterative approach supports the overarching goal of meeting the needs of simple user behaviour while ensuring the system remains understandable and user-friendly through timely enhancements. As a trade-off, our updates may not work with the older versions. This is also the fact that prioritising efficiency and reliability required us to decrease backwards compatibility.

High-Level Architecture



1. Presentation Layer - CCUser Interface: This layer is responsible for handling all the user interface and user experience components. It interacts with the application layer to present data to the user and handle user inputs.

2. Application Layer:

- **Filters:** These are used to perform operations on requests and responses, such as authentication or logging, before they reach the core application logic.
- **Data Transfer Objects (DTO):** Objects that carry data between processes in order to reduce the number of method calls, particularly in a network environment.
- **Controllers:** They act as an intermediary between models and views to process incoming requests, perform operations, and return responses.
- **Services:** Contain the business logic of the application and orchestrate the application's response to user inputs or interactions.
- **Messaging:** Manages communication between different parts of the application or with other applications/services, possibly through message queues or service buses.
- **Config:** Holds configuration settings that can be used by different parts of the application, often externalised from the business logic.
- **Util:** Utility components that provide common functions across the application, such as helper functions or shared resources.
- **Entities:** These are domain objects that represent the data within the system and the relationships between that data.

3. Database Interface Layer - Repositories: These components abstract the data layer, providing a collection-like interface for accessing domain objects.

4. Data Layer - AWS Database: This is where the data is stored, managed, and retrieved. In this case, it indicates the use of a database service provided by AWS (Amazon Web Services).

Each subsystem is designed to handle specific concerns within its layer, following the separation of concerns principle to promote a more organised and maintainable codebase.