CS319 – Object Oriented Software Engineering

Section 3

Team 4 – FACID

Design Report

Çağatay **AKPINAR 22003508**
İlhami **ULUĞTÜRKKAN 22102546**
Alphan **TULUKCU 22003500**
Feza Emir **ÇELİK 22101910**
Deniz Can **ÖZDEMİR 22003854**

1.0 Introduction

CampusConnect is a website designed to make the lives of Bilkent University members easier. CampusConnect has attributes like second-hand sales for both textbooks for courses in Bilkent University and other belongings, It also provides communication services for Bilkent Members. This paper focuses on the design issues of the CampusConnect.

1.1 Purpose of the System

CampusConnect aims to create an inclusive and cohesive digital community within Bilkent University. It strives to facilitate the exchange of knowledge, resources, and experiences among students. The platform is dedicated to making the process of buying and selling second-hand books and furniture easy and cost-effective, reducing financial burdens on students. Additionally, we aspire to enhance transparency and communication between students and professors through the Questions and Suggestions section, ultimately improving the academic experience. Lastly, our Forum section fosters open dialogue on various university-related topics. Overall, our goals revolve around creating a thriving digital hub that addresses the diverse needs of our university community.

1.2 Design Goals

Campus Connect focuses on various goals, and the non-functional requirements show these goals:

1.2.1 Performance

Performance is the heartbeat of CampusConnect, with every aspect meticulously tuned to ensure lightning-fast interactions. Our login and logout processes are designed to take less than 2 seconds, allowing users to access their accounts swiftly. Navigating between pages is a seamless experience, taking less than 2 seconds to transition effortlessly. Whether you're uploading a post or entry with images, we prioritize speed – pictures from our database to your screen in under 5 seconds and post creation in less than 2 seconds, even with photographs. In the background, the entire system data is dynamically backed up to ensure data integrity and reliability. The importance of such performance measures cannot be overstated. A rapid, responsive system enhances user satisfaction, engagement, and the overall success of our platform. With CampusConnect, every second counts in making your online campus experience exceptional. These performance measures are vital to ensure that users have a smooth and efficient experience when using the CampusConnect platform. The speed of these actions directly impacts user engagement and the overall functionality of the website. Rapid performance is key to user satisfaction and success in providing an exceptional online campus experience.

1.2.2 Safety/Security

At CampusConnect, security and safety are paramount. We've implemented stringent measures to safeguard user data, ensuring that personal information is encrypted and stored securely within our database. In CampusConnect passwords of users are stored in the database as a hashed form, so even admins cannot access the passwords of users. Additionally, the app boasts robust authentication and authorization mechanisms, limiting access exclusively to individuals within the Bilkent University community. The account verification feature of our app provides the chance to create an on-campus community in the CampusConnect app. The importance of these security measures cannot be overstated. They are the foundation of user trust and confidence in our platform. As guardians of sensitive data, we prioritize the protection of personal information and access controls to ensure that CampusConnect remains a safe and secure digital space for our users, fostering trust, peace of mind, and a thriving online community.

1.2.3 Usability

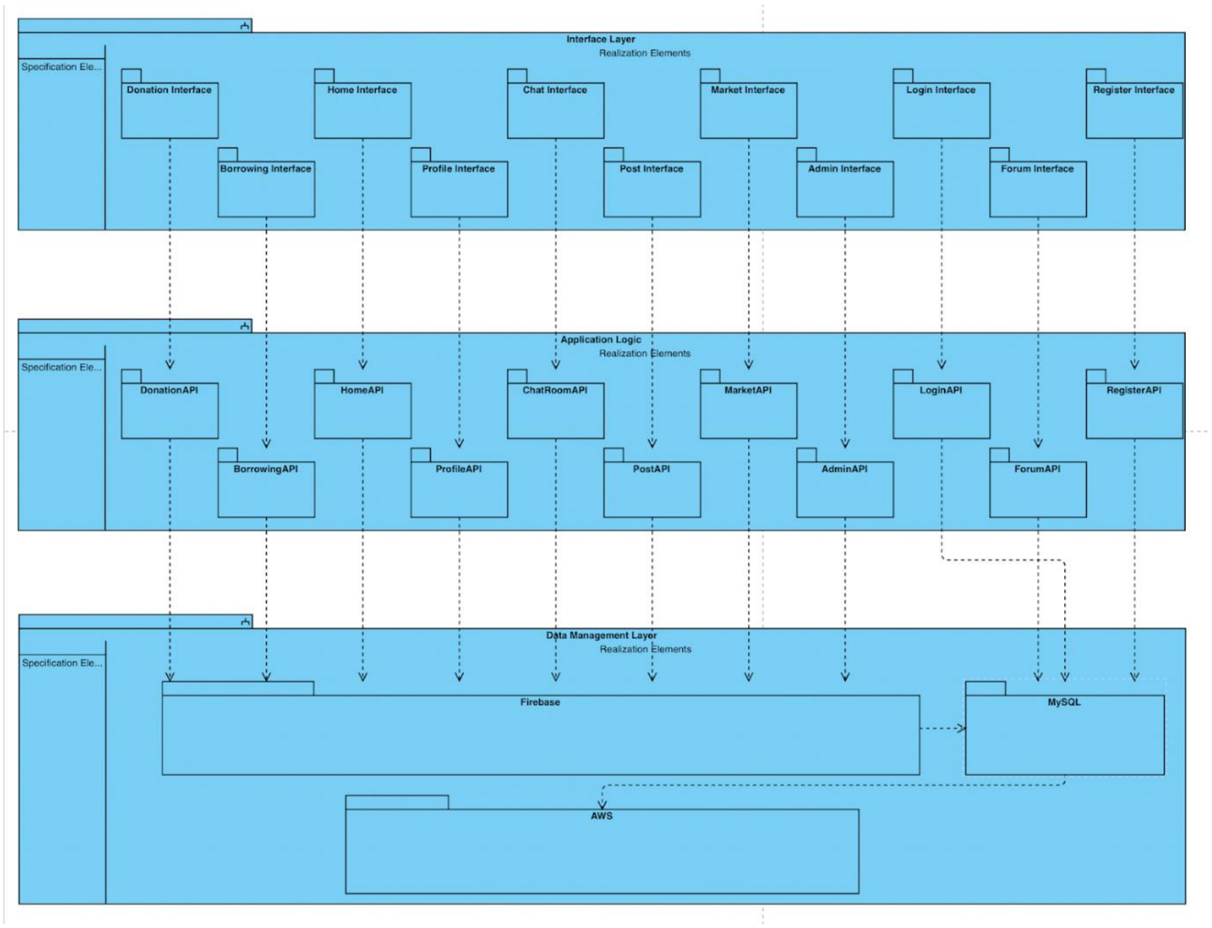
Usability is a cornerstone of CampusConnect's design philosophy, and it holds immense importance in delivering an exceptional user experience. Our user interface is meticulously crafted to be intuitive, accessible, and responsive, catering to users of different abilities. We have focused on creating an easy-to-use and simple user interface that ensures a seamless experience. Places of all buttons are on the top menu and easy to find and use for all users. Also, while adding a post, entry, borrowing, and donation, blanks that must be filled are located at the right of the page. The app's compatibility with popular web browsers, including Chrome, Safari, and Opera, guarantees that users can access CampusConnect with ease using any web browser. Additionally, we've chosen color tones (orange and white) that are easy on the human eye to prevent visual fatigue and ensure a pleasant reading experience. Our buttons are designed to be clear and user-friendly, streamlining interactions. By prioritizing usability, we aim to provide an accessible, enjoyable, and efficient platform for all users, promoting engagement and making CampusConnect a user's first choice.

1.2.4 Reliability

Reliability is at the core of CampusConnect, and its significance cannot be overstated. Our commitment to offering a dependable platform means that the CampusConnect app will be available 24/7, with minimal downtime reserved exclusively for essential maintenance. Furthermore, we've taken measures to ensure the system can recover gracefully from any unforeseen system failures, all without the loss of any critical data. We understand that reliability is the linchpin of user trust and satisfaction. It's the assurance of uninterrupted service, data integrity, and a seamless experience for our users. By prioritizing reliability, we aim to provide a platform that users can depend on day in and day out, fostering a strong sense of trust and loyalty within our digital community.

2. High-level Software Architecture

2.1 Subsystem Decomposition



We decided to use a 3 Layered Architecture, which contains an Interface Layer, Application Logic Layer, and Data Management Layer. Interface Layer contains the user interfaces used in the system as the boundary objects. Each interface has its own management package to handle the requests and communicate with the user. Application Logic Layer contains the backend APIs that correspond to the operations in the Interface Layer, executing the necessary logic and processing to fulfill user requests and application functions. This separation allows for a clean architecture where the user interface can evolve independently from the underlying business logic. It also facilitates scalability, as each layer can be scaled or modified without affecting the others.
The Data Management Layer is the core part of our system that takes care of safely storing, getting, and handling data. It uses Firebase, MySQL and AWS to store data. Firebase is utilized for storing images uploaded by users. It offers quick retrieval for the application and serves as an intermediary storage solution. MySQL acts as the primary database for storing links to the images in Firebase, user data, content, and other structured data required by the application. AWS is used for persistent storage, ensuring that data is available 24/7 with high reliability. This may be used in conjunction with other AWS services to enhance data management and scalability. Data Management Layer makes sure that all the information in the app is correct and consistent. By keeping data management separate, we can easily change how we store

data or make it handle more information if we need to. This setup also makes it easier to fix and improve the system since we can work on each layer by itself without making things too complicated or risky. In short, having this layer helps our app grow and change over time.

2.2 Hardware/Software Mapping

CampusConnect is a web-based project that will be accessed by search engines with both computers and smartphones. Therefore, it does not require any hardware to use. It will be deployed with Amazon Web Services EC2 so that anyone with the link can access and use the website. Django is a wide-support framework, and the whole back-end is developed with Django and Django-Rest Framework. It allows all modern search engines can use the webpage. On the other hand, CampusConnect is designed as a responsive website that uses the Bootstrap framework on its front-end part and consists of HTML, CSS, and JavaScript. Bootstrap helps the application be more relative and responsive to enhance usability. Therefore, Bootstrap and HTML5-supported devices can access the webpage.

2.3 Persistent Data Management

CampusConnect works in such a way that serves different types of users' data which they uploaded to CampusConnect to the other users of CampusConnect. These data are borrowings, donations, posts, and entries. Managing and categorizing this much data is vital for CampusConnect. For this reason, we preferred to use MySQL database because it has cross-platform compatibility, which means can run on various operating systems such as Windows, Linux, macOS, and more; it has scalability, which means it can efficiently handle growing datasets and number of users and it provides various security features including user authentication, access control, and encryption to ensure the safety of stored data. Also, we preferred Google Firebase for storing images of CampusConnects' data (borrowings, donations, posts, entries) because it provides simple API for file uploads and downloads, which makes it easy to integrate into the CampusConnect website, it automatically handles data traffic and delivers images efficiently even during periods of high demand, and it allows for real-time synchronization of data across clients. In addition, due to AWS RDS (Amazon Web Services Relational Database Service) provides real-time web-based remote database service to CampusConnect.

2.4 Access Control and Security

CampusConnect has an access control system in order to provide security for the app and prevent misuse of users. A user's access range is determined by his/her role. A user can either be a **user** or an **admin.** Users enter the app with an ordinary user interface and only execute certain actions that are restricted by the user interface. Our app determines the user's role according to the data that comes from the database during the login process. After determining that, the user can only see restricted interfaces and interface paths designed for the user role and cannot intervene with other users' items.
Admins enter the app just like common users. After the login process admin user sees a similar page with users, but there are small changes, such as the lack of an "add interface," which is used for sharing posts and entries. Since admins exist to provide good and safe service, they do not share anything. They just check shared items, profiles, and reported items. If there is any offensive or abusive usage of the app, they are allowed to intervene and edit or delete items and ban users. Therefore, admins can be considered as the police of the app environment.

Access Control Matrix

	User	Admin
Login	X	X
Register	X	X
Renew Password	X	X
Share Post	X	
Share Entry	X	
Search for Post	X	X
Edit Own Post	X	
Edit Any Post		X
See Market	X	X
Filter Market	X	X
See Forum	X	X
See Detailed Post	X	X
See Detailed Entry	X	X
Comment to Entry	X	
Delete Own Post	X	
Delete Own Entry	X	
Delete Any Post		X
Delete Any Entry		X
View Own Profile	X	X
View Others Profile	X	X
Edit Own Profile	X	X
Edit Others Profile		X
Send Message	X	X
Report a Shared Item	X	
Report a Profile	X	

Ban a User		**X**

2.5 Boundary Conditions

2.5.1 Initialization

No installation is required to run the CampusConnect application; all necessary source files are fetched from the server when a user visits the CampusConnect domain. To use the application, the user should already have an account created in the database. The account creation phase is done through the registration process. After successful registration, users can log into the system through the login interface and access the features of CampusConnect.

2.5.2 Termination

CampusConnect operates on web servers and cloud-based services, ensuring continuous availability. The termination of the application would involve server maintenance or updates, which will typically be scheduled during low-usage periods. In case of any planned termination, users will be notified in advance through appropriate communication channels to minimize disruptions.

2.5.3 Failure

When a server failure or unexpected system issues occur, CampusConnect is designed to provide a seamless experience once the issues are resolved. Continuous monitoring and automated notifications are in place to alert administrators to any potential failures. Regular backups of the database are conducted to ensure data integrity and for being able to recover from unexpected events. In such scenarios, users of the application will be informed of the situation, and the development team will work punctually to restore normal operation.

3.Object Design Trade-Offs

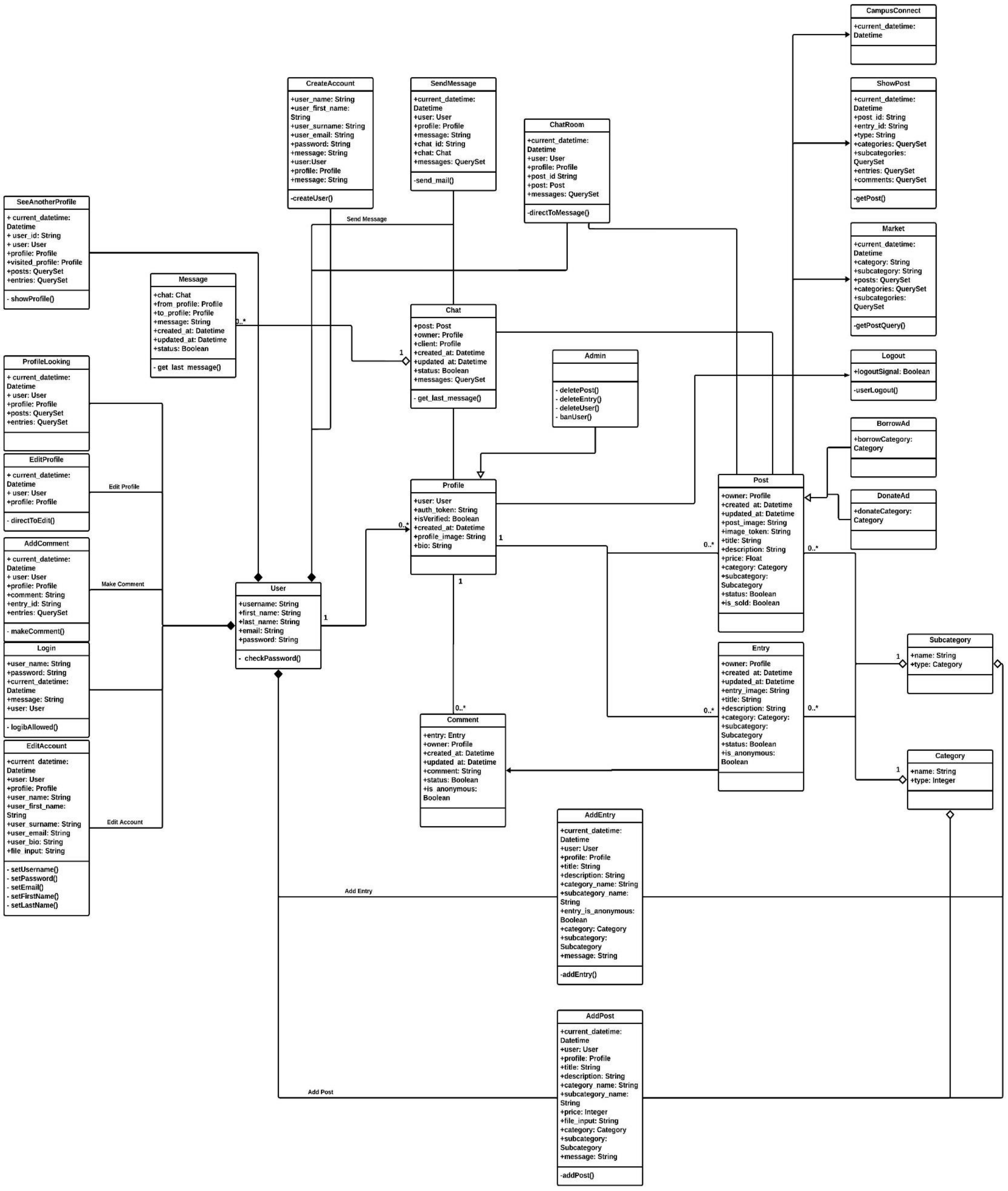
3.1 Readibility vs. Performance

In CampusConnect website, trade-off between readibility and performance is very significant. We used meaningful and explanatory variable and class names to improve code simplicity and also readibility from the perspective of all group members. This approach evolves collaboration between stakeholders. However, exclusive focus on readibility might lead to negatively impact on the CampusConnect's performance. Under these conditions, we decided to set the right balance between readiblity vs. performance. Even though we aware that code simplicity negatively affect code performance, at some parts of code we used extra lines or extra methods to make code more readible and understandable for all stakeholders. In conclusion, even though explanatory class and variable names do not affect the code performance, extra lines of code and extra methods degrades code performance. Although we were aware of this situation, we decided to sacrifice code performance in some places as we realized that code explainability is of great importance to us in the continuation of the project.

3.2 Security vs. Usability

In CampusConnect website, trade-off between security and usability has high importance. We used mail control system at registration stage in our code segments to improve website security. If mail of the user is not a Bilkent mail, registration application of a user is declined. In addition, CampusConnect has a authentication feature. After user register to the system, user must authenticate his/her account. If user register with the non-existent Bilkent mail, he/she fails at authenticate stage and after that account will be useless. These features of website reduces the usability of CampusConnect. Because user have to complete some processes to using the app with using his/her CampusConnect account and Bilkent Webmail. Although we were aware of this situation, we decided to put forward security rather than usability. In this way, CampusConnect provides its users with a reliable user environment consisting only of Bilkent University members.

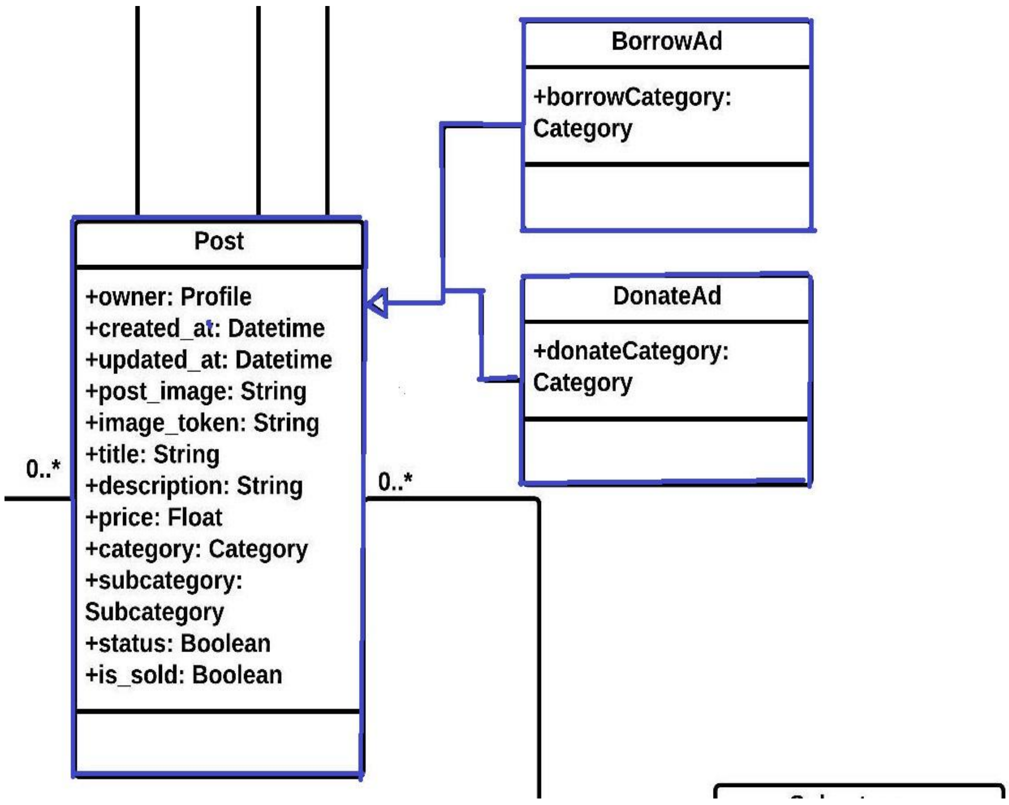
4.Final Object Design



5.Design Patterns

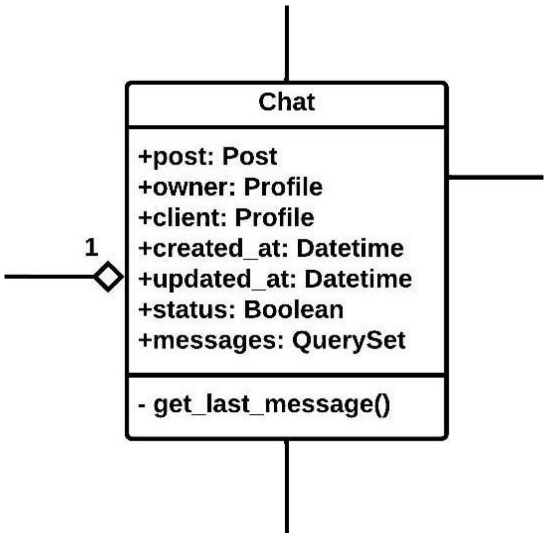
5.1 Strategy Pattern

In our project, we have used strategy pattern at Post, BorrowAd and DonateAd classes. In our project users can add adverts for borrowings and donations. These two process are very similar. For these reason users can choose and switch between these two classes at runtime. This flexibility not only simplifies the codebase but also ensures that users can seamlessly create different types of advertisements with distinct behaviors.



5.2 Façade Pattern

In our project, we have used Façade pattern at Chat class. It includes and controls message, profile and post classes. In this way, by encapsulating the complexities of these related classes behind unified Façade, the “Chat” class provides a simplified and cohesive entry point for users. This design choice enhances the maintability and readability of our code.



6.Packages

6.1 External Packages

- **Django:** This package is used for web applications in Python. It provides almost everything a web-project needs such as object modeling (ORM), admin panel and authentication support. It is suitable for building scalable and maintainable web applications. For these reasons we selected to use Django to build CampusConnect. This package interacts with Django Rest Framework, BootStrap and MySQL.
- **Django Rest Framework (DRF):** This package is used for building web APIs with Django. It extends Django’s capabilities. We selected to use DRF to integrate web APIs to CampusConnect. This package interacts with Django and ChatGPT API.
- **BootStrap:** BootStrap is a popular open-source front-end framework that streamlines web development. It incudes HTML and CSS design templates for various interface components, and with some JavaScript extensions. For these

reasons we selected to use BootStrap to build CampusConnect. This package interacts with Django.

- **MySQL (Amazon Web Services):** This package is an open-source relational database management system that uses SQL for accessing and managing data stored in realtional databases. This package interacts with Google Firebase, EC2 and Django.
- **Google Firebase (Pyrebase4 Framework):** This package is used for storing images in CampusConnect at backend. This package interacts with MySQL.
- **ChatGPT API:** This package is used for detecting use of immoral words in chats and comments by users. This package interacts with Django Rest Framework.

6.2 Internal Packages

- **Views Package:** Views package is super package of all boundary classes.
    - **Donation Package:** This package involves all related classes to donation process.
    - **Borrowing Package:** This package involves all related classes to borrowing process.
    - **Home Package:** This package involves all related classes to homepage which users see when they login to their accounts.
    - **Profile Package:** This package involves all related classes to profile and features relevant to profile.
    - **Chat Package:** This package involves all related classes to chatting process.
    - **Post Package:** This package involves all related classes to adding post process.
    - **Market Package:** This package involves all related classes to market and features relevant to market.
    - **Admin Package:** This package involves all related classes to admins’ features.
    - **Login Package:** This package involves all related classes to logging in to account process.
    - **Forum Package:** This package involves all related classes to forum and features relevant to forum.

    - **Register Package:** This package involves all related classes to registering to CampusConnect process.

7.Class Interfaces

- **ProfileLooking:** This class works as a manager class. It is used for looking other users' profiles.
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
        - posts: Stores the posts of profile looked.
        - entries: Stores the entries of profile looked.
- **EditProfile:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
    - Methods:
        - directToEdit():
- **AddComment:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
    - Methods:

        - makeComment(): Uploads the comment to the CampusConnect

- **Login:**
    - Attributes:
        - user_name: Stores the username of user.
        - password: Stores the password of user.
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - message:
        - user:
    - Methods:
        - loginAllowed(): Controls the login process according to correctness of username and password from database.
- **EditAccount:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
        - user_name: Stores the username.
        - user_first_name: Stores the firstname.
        - user_surname: Stores the surname.
        - user_email: Stores the email.
        - user_bio: Stores the bio.
        - file_input: Stores the Google Firebase link of profile image.
    - Methods:
        - setUsername(): Updates old username with the new username.
        - setPassword(): Updates old password with the new password.

        - setEmail(): Updated old email with the new email.
        - setFirstName(): Update old firstname with the new firstname.
        - setLastName(): Update old lastname with new lastname.
- **SeeAnotherProfile:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user_id:
        - user:
        - profile:
        - visited_profile:
        - posts: Stores posts of another profile.
        - entries: Stores entries of another profile.
    - Methods:
        - showProfile(): Shows the informations of another profile to current user
- **CreateAccount:**
    - Attributes:
        - user_name: Stores newly created username.
        - user_first_name: Stores newly created firstname.
        - user_surname: Stores newly created surname.
        - user_email: Stores newly created email.
        - password: Stores newly created password.
        - message: Stores newly created message.
        - user:
        - profile:
        - message:
    - Methods:
        - createUser(): Creates user if conditions are provided at CampusConnect.
- **SendMessage:**
    - Attributes:

        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
        - message: Stores the message which will be sent.
        - chat_id: Stores the id number of chat.
        - chat:
        - messages: Stores the messages of chat.
    - Methods:
        - send_mail(): Sends message to another user.

- **ChatRoom:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
        - post_id: Stores the id number of post which is two users chatting about.
        - post: Stores the post which is two users chatting about.
        - messages: Stores the messages of chatroom.
    - Methods:
        - directToMessage():
- **Campus_Connect:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
- **ShowPost:**
    - Attributes:

        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - post_id: Stores the id number of post will be showed.
        - entry_id: Stores the id number of entry will be showed.
        - type:
        - categories: Stores the category of post or entry.
        - subcategories: Stores the subcategory of post or entry.
        - entries: Stores the entries about entry.
        - comments: Stores the comments of post.
    - Methods:
        - getPost(): Shows post or entry to user.
- **Market:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - category: Stores the category of filter if user filtered in market.
        - subcategory: Stores the subcategory of filter if user filtered in market.
        - posts: Stores all posts at market.
        - categories: Stores the all categories to show user when user wants to filter at market.
        - subcategories: Stores the all subcategories to show user when user wants to filter at market.
    - Methods:
        - getPostQuery(): Show the results of filters of user to user.
- **Logout:**
    - Attributes:
        - logoutSignal:
    - Methods:

        - userLogout(): Allows the user to log out of the application

- **BorrowAd:**
    - Attributes:
        - borrowCategory: Stores the category of borrow.
- **DonateAd:**
    - Attributes:
        - donateCategory: Stores the category of donation.
- **Subcategory:**
    - Attributes:
        - name: Stores the name of subcategory.
        - type:

- **Category:**
    - Attributes:
        - name: Stores the name of category.
        - type:
- **AddEntry:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
        - title: Stores the title of new entry.
        - description: Stores the description of new entry.
        - category_name: Stores the category name of new entry.
        - subcategory_name: Stores the subcategory name of new entry.
        - entry_is_anonymous: Stores whether the new entry is anonymous or not
        - category: Stores the category of new entry.
        - subcategory: Stores the subcategory of new entry.

        - message:
    - Methods:
        - addEntry(): Adds the new entry to the CampusConnect.
- **AddPost:**
    - Attributes:
        - current_datetime: Stores the current time to show user the time activities (adding post, adding entry, etc.) was made or how long ago activities (adding post, adding entry, etc.) was made.
        - user:
        - profile:
        - title: Stores the title of new post.
        - description: Stores the description of new post.
        - category_name: Stores the category name of new post.
        - subcategory_name: Stores the subcategory name of new post.
        - price: Stores the price of new post.
        - file_input: Stores the Google Firebase image link of new post.
        - category: Stores the category of new post.
        - subcategory: Stores the subcategory of new post.
        - message:
    - Methods:
        - addPost(): Adds the new post to the CampusConnect.
- **Message:**
    - Attributes:
        - chat:
        - from_profile: Stores the profile which sent message.
        - to_profile: Stores the profile which receive message.
        - message: Stores the message.
        - created_at: Stores the time which message is created.
        - updated_at:
        - status:
    - Methods:

        - get_last_message():
- **Chat:**
    - Attributes:
        - post: Stores the post which chat is started for.
        - owner: Stores the owner of post which chat is started for.
        - client: Stores the client of post which chat is started for.
        - created_at: Stores the creation time of chat.
        - updated_at:
        - status:
        - messages:
    - Methods:
        - get_last_message(): Shows last message of chat.
- **User:**
    - Attributes:
        - username: Stores the username of user.
        - first_name: Stores the first name of user.
        - last_name: Stores the last name of user.
        - email: Stores the email of user.
        - password: Stores the password of user.
    - Methods:
        - checkPassword():
- **Profile:**
    - Attributes:
        - user:
        - auth_token:
        - isVerified: Stores whether profile is verified or not.
        - created_at: Stores the creation time of profile.
        - profile_image: Stores the Google Firebase image link of profile photo.
        - bio: Stores bio of profile.
- **Comment:**
    - Attributes:
        - entry: Stores the entry which comment is made for.

        - owner: Stores the owner of entry which comment is made for.
        - created_at: Stores the creation time of comment.
        - updated_at:
        - comment: Stores the content of comment.
        - status:
        - is_anonymous: Stores whether comment is anonymous or not.
- **Admin:**
    - Methods:
        - deletePost(): Deletes the post of any user.
        - deleteEntry(): Deletes the entry of any user.
        - deleteUser(): Deletes the user from CampusConnect.
        - banUser(): Bans the user from CampusConnect.
- **Post:**
    - Attributes:
        - owner: Stores the owner profile of post.
        - created_at: Stores the creation time of post.
        - updated_at:
        - post_image: Stores the Google Firebase image link of post images.
        - image_token: Stores the tokens of image.
        - title: Stores the title of post.
        - description: Stores the description of post.
        - price: Stores the price of post.
        - category: Stores the category of post.
        - subcategory: Stores the subcategory of post.
        - status:
        - is_sold: Stores whether content of post is sold or not.
- **Entry:**
    - Attributes:
        - owner: Stores the owner profile of entry.
        - created_at: Stores the creation time of entry.
        - updated_at:
        - entry_image: Stores the Google Firebase image link of entry images.

- title: Stores the title of entry.
- description: Stores the title of description
- category: Stores the category of entry.
- subcategory: Stores the subcategory of entry.
- status:
- is_anonymous: Stores whether entry is anonymous or not.