CS319 – Object-Oriented Software Engineering

D5 Report

CAMPUS CONNECT®

Section 3

Team 4 – FACID

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

Table of Contents

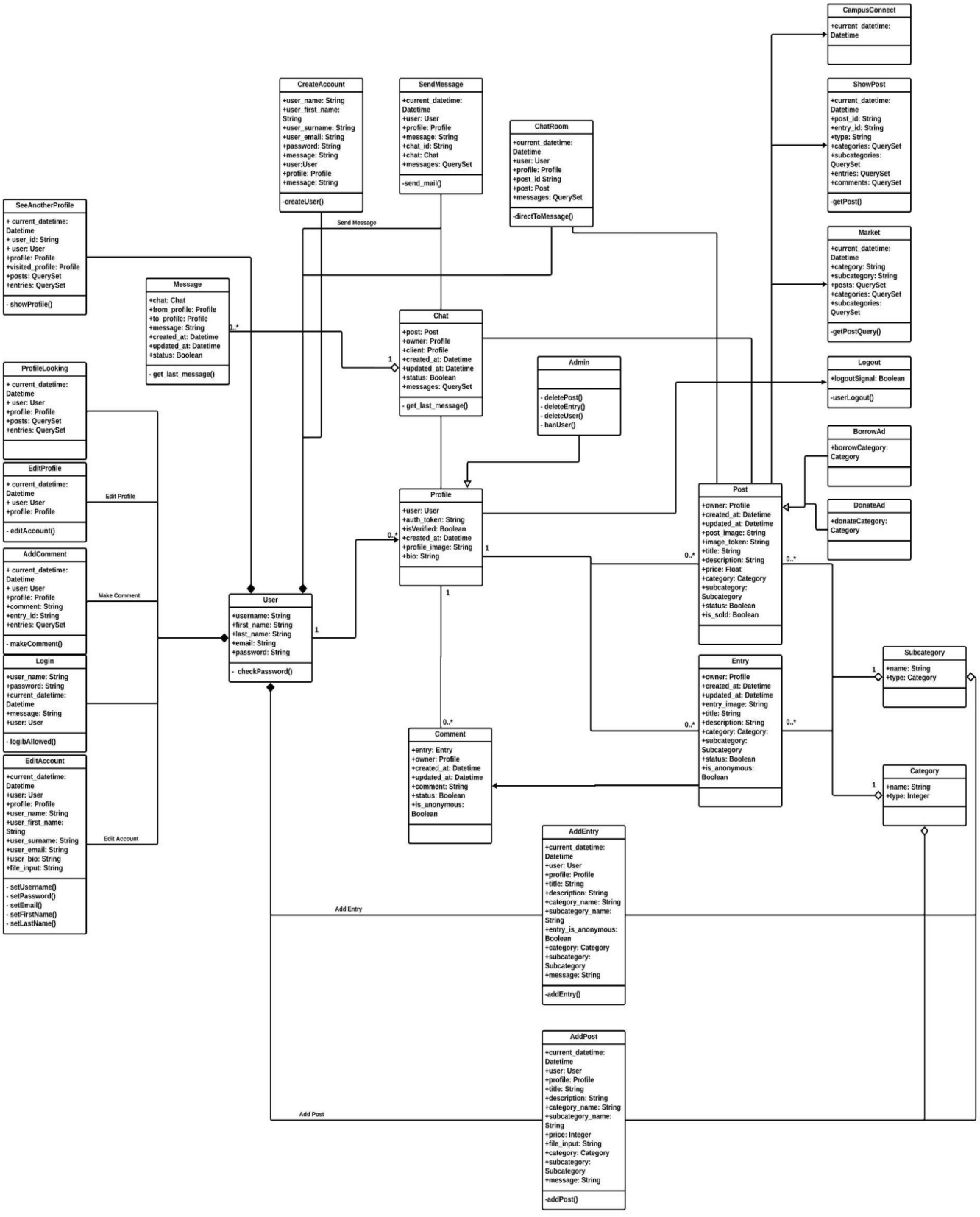
3. Low-level design

3.1 Object Design Trade-Offs

- **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.

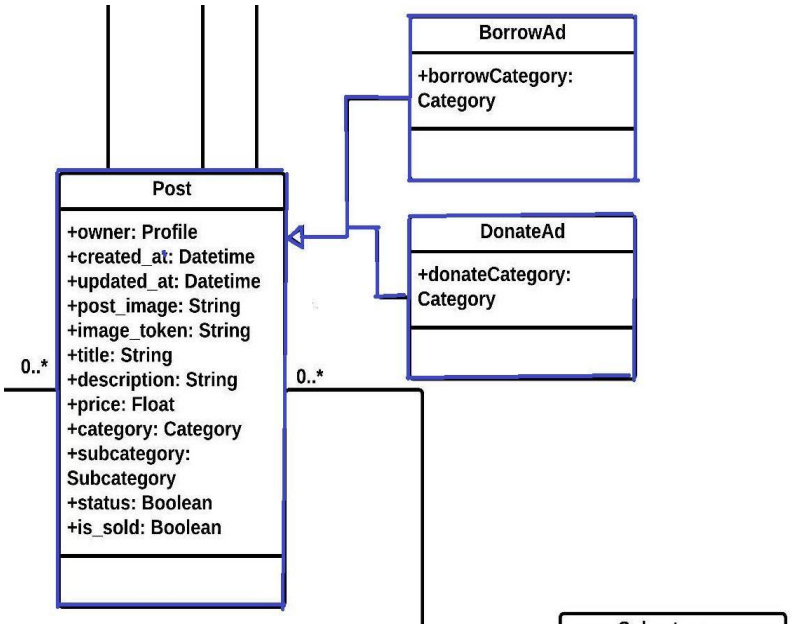
- **Security vs. Usability:** In CampusConnect website, trade-off between security and usability has high importance. We used mail control system at registration stage in our code segments to improve website security. If mail of the user is not a Bilkent mail, registration application of a user is declined. 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.

3.2 Final Object Design

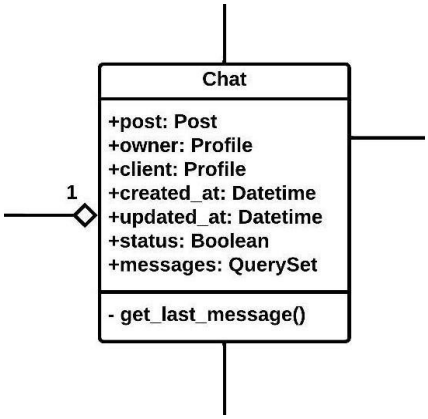


3.3 Design Patterns

- **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.



- **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.



3.4 Packages

3.4.1 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.

3.4.2 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.

3.5 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: Determine which user is visited.
        - profile: Visited user's profile object.
        - 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: Determine the registered user.
        - profile: Determine the registered user's profile object.
    - Methods:
        - EditAccount(): Provides the edited changes are sent to the database.
- **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: Determine the registered user.
        - profile: Determine the registered user's profile object.
    - 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: Message string to show if any error occurred.
        - user: Determine the registered 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: Determine the registered user.
    - profile: Determine the registered user's profile object.
    - 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.
    - visited_user: Determine the visited user.
    - user: Determine the registered user.
    - profile: Determine the registered user's profile object.
    - visited_profile: Determine the visited user's profile object.
    - 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: Determine the registered user to create.
    - profile: Determine the registered user's profile object to create.
    - message: message string if any error occurred.

- 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: Determine the registered user.
    - profile: Determine the registered user's profile object.
    - message: Stores the message which will be sent.
    - chat_id: Stores the id number of chat.
    - chat: determine the chat object in which user use.
    - 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: Determine the registered user.
    - profile: Determine the registered user's profile object.
    - 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(): Shows the information about selected message.
- **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: Determine whether it is post or entry.
    - 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: Stores that whether user wants to logout from CampusConnect.
    - 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: Determine whether the subcategory belongs to posts or entries.
- **Category:**
    - Attributes:
        - name: Stores the name of category.
        - type: Determine whether the category belongs to posts or entries.
- **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: Determine the registered user.
        - profile: Determine the registered user's profile object.
        - 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: Message string to show if any error occurred.
    - 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: Determine the registered user.
        - profile: Determine the registered user's profile object.
        - 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: Message string to show if any error occurred.
    - Methods:
        - addPost(): Adds the new post to the CampusConnect.
- **Message:**
    - Attributes:
        - chat: Chat object that the message is belong
        - 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: If the message is updated, new datetime is put.
        - status: Determine whether the message is active or not.
    - Methods:

        - get_last_message(): Get last message to show last message in preview interface.
- **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: If the chat is updated, new datetime is put.
        - status: Determine whether the chat is active or not.
        - messages: The messages list inside the chat.
    - 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(): Checks whether the password is correct or not when user logs in to CampusConnect.
- **Profile:**
    - Attributes:
        - user: Determines the registered user.
        - auth_token: Determine whether the profile is authenticated or not.
        - 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: Stores the time when the last update happened.
        - comment: Stores the content of comment.
        - status: Determine whether the comment is active or not.
        - 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: Stores the time when the last update happened.
        - 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: Determines whether the post is deleted or not.
        - 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: Stores the time when the last update happened.
        - 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: Determines whether the post is deleted or not.
        - is_anonymous: Stores whether entry is anonymous or not.