SYSTEM ARCHITECTURE

The platform ‘DonHub’ will be used by students and faculty to buy, sell, or exchange academic resources, or even products. Based on that, here are proposed system architectures:

1. **Client-Server Architecture:** At its core, a university marketplace is a web application where users (clients) interact with a centralized platform (server). Clients make requests to the server, which processes the requests and sends back appropriate responses. This model provides a straightforward way to maintain the central repository of listings, user accounts, transaction records, etc.
2. **Client:**

**- Students and Staff:** Verified PFW students and staff access the ‘DonHub’ application through web browsers. These clients provide user interfaces for interacting with the various features of ‘DonHub’, such as browsing products, posting items for sale, viewing notifications, and managing user profiles.

1. **Server:**

**- User Management:** The central server handles user authentication and authorization processes. When a user logs in, the server verifies their credentials and grants appropriate access rights based on their role (students, staff, or admin). This ensures that only verified users can access DonHub’s features.

**- Product Management:** The server manages product listings, including posting new items, updating existing products, marking items as sold, and removing listings. It stores product details, seller information, and transaction status in database.

**- Notifications:** The server sends notifications to users based on their preferences and activities. For example, users receive notifications when someone expresses interest in their product, when a product is sold, or when there are important updates related to their listings.

**- Chats and Communication:** It serves as a real-time communication platform that enables users to exchange messages with a other users, inquire about products, negotiate prices and terms will sellers or buyers and discuss transaction logistics, such as meeting locations and times for in-person exchanges.

**- Admin Access:** The server provides a separate administrative interface accessible only to authorized admins. Admins have the authority to revoke user privileges, moderate listings, and handle exceptional cases, ensuring smooth operation and adherence to the platform’s policies.

1. **Communication between Client and Server:**

**- HTTP Requests:** Clients communicate with the server via HTTP requests. For instance, when a user posts a new product, the client sends a request to the server, including the product details. The server processes this request, validates the data, and stores it in the database.

**- Real-Time Updates:** The server employs technologies like WebSockets to provide real-time updates to clients. For example, when a new message or offer is received, the server pushes the update to the respective user’s client interface, ensuring timely notifications and interactions.

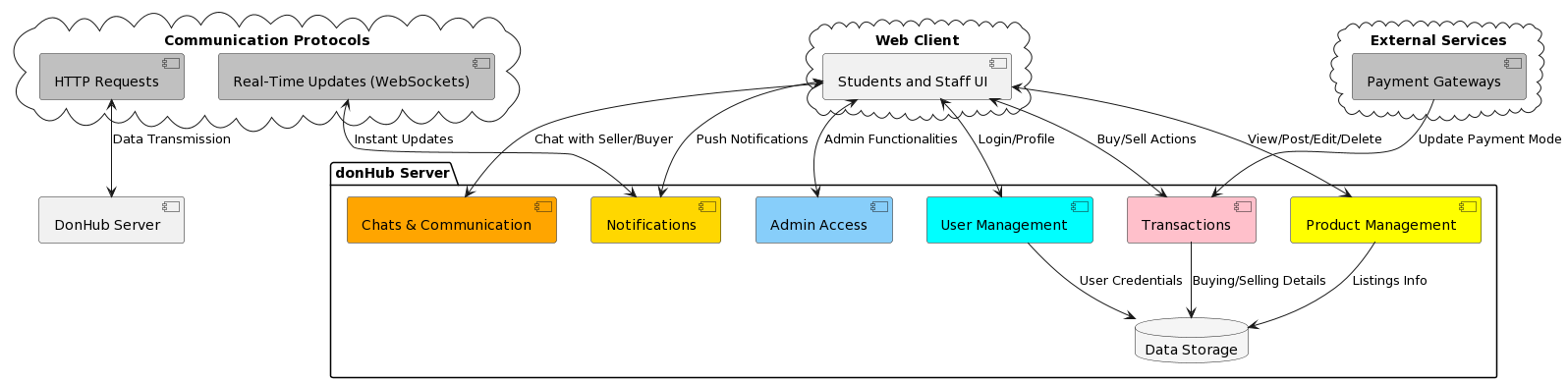
1. **Database:**

**- Data Storage: ‘**DonHub’ utilizes a database system (e.g., MySQL, MongoDB) to store user profiles, product listings, transaction history, and other relevant data. The server interacts with the database to retrieve or store information based on user requests.

1. **Security Measures:**

**- Data Encryption:** The client-server communication is encrypted to ensure that sensitive information, such as user credentials and transaction details, remains secure during transit.

**- Authentication Tokens:** Upon successful login, the server issues authentication tokens to clients. These tokens are used for subsequent requests, allowing the server to identify and authenticate users without requiring them to repeatedly enter their credentials.



1. **Event-Driven Architecture:** It is a software architectural pattern in which the flow of the system is determined by events. In EDA, events are occurrences or notifications that represent changes in the state of a system or the environment. These events are used as triggers to initiate actions, processes, or communications between different parts of a software system or between multiple systems. EDA is particularly useful in scenarios where real-time responsiveness, scalability, and decoupling of components are essential. EDA can be implemented in ‘DonHub’ by identifying the key events, event producers and event consumers.
2. **Key Events:** The critical events that drive interactions within the application need to be identified. These events could include user actions such as posting a listing, sending a message, making an offer, updates to listings, user interactions (e.g., posting comments), and system events such as notifications, recommendations.
3. **Event Producers:** The event producers are responsible for generating events based on user actions, system updates, or external data integration. These producers need to be implemented throughout the platform. For instance, when a user posts a new listing, an event is generated.
4. **Event Consumers:** Each consumer is designed to perform particular actions or updates based on the events they receive. For example:

* A listing detail page might subscribe to events related to the specific listing and update its view when changes occur.
* A recommendation system might subscribe to user interaction events to refine its recommendations.

1. **Event Broker/Bus:** Implements an event broker or event bus that acts as a central hub for managing and routing events. This component facilitates event distribution from producers to consumers efficiently.
2. **Asynchronous Processing:** Event driven interactions should be asynchronous, allowing components to operate independently. Events are published and consumed without waiting for immediate responses, improving system responsiveness. For example, when listings are created or updated, their corresponding data in the search index can be updated asynchronously. This ensures that search results remain up to date without impacting the user experience.
3. **Microservices or Service Components:** Utilize microservices or service components that can act as both event producers and consumers. These components can generate events based on their functionality and subscribe to events relevant to their responsibilities.
4. **Data Synchronization:** Leverage events to maintain data consistency across different parts of the application. For instance, when a user updates their profile information, related events can trigger updates in messaging, notifications, and search indexes.

