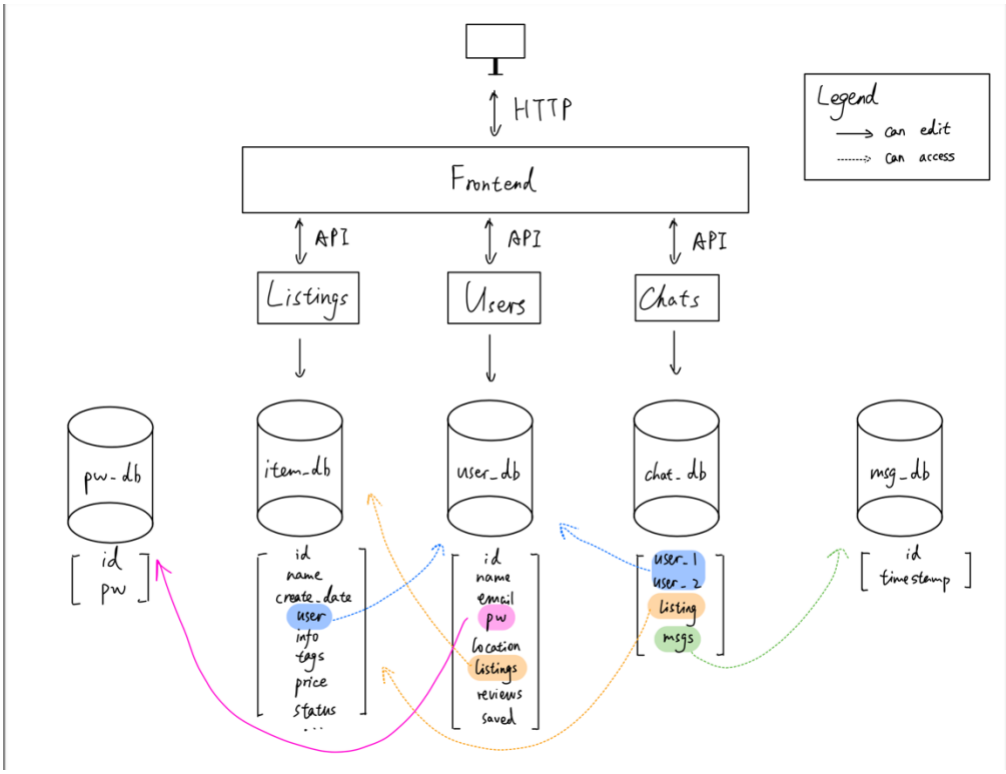
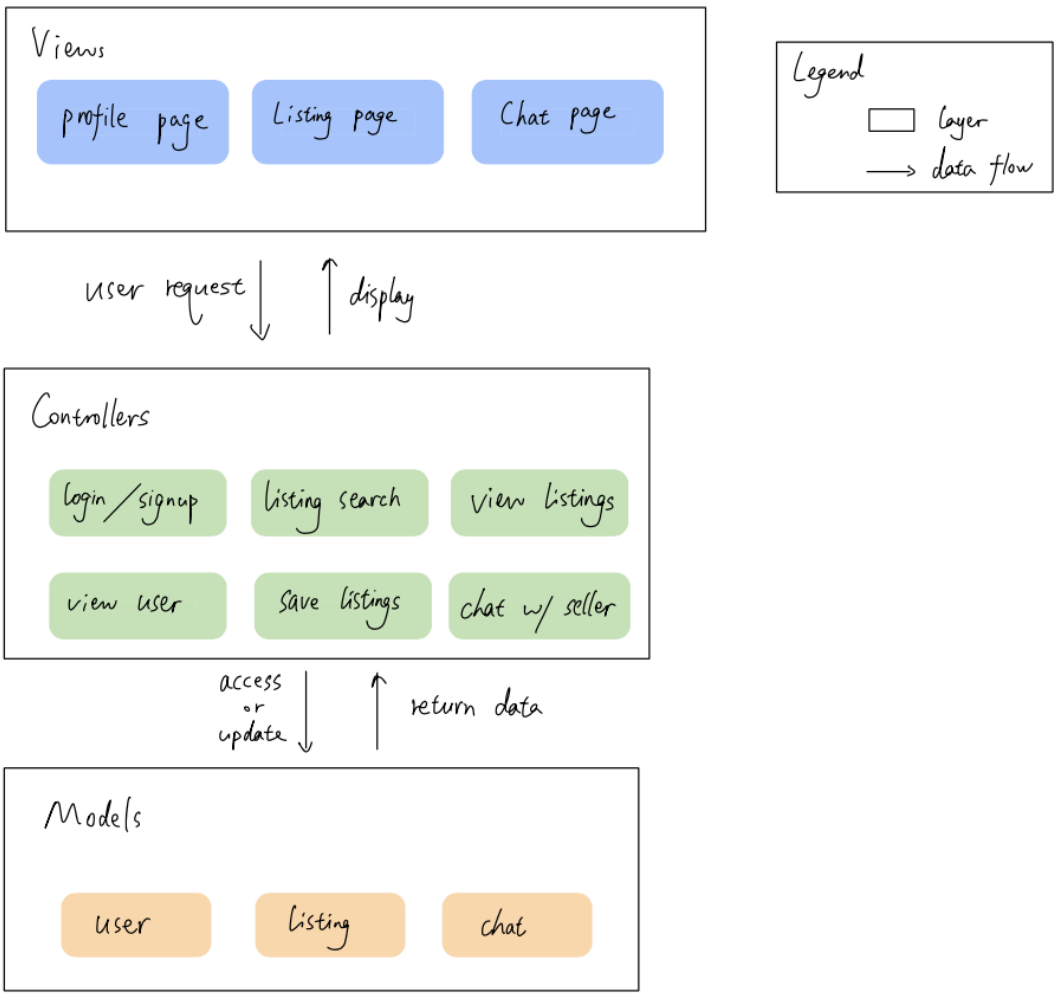
Microservice structure:



- The system is separated into three main modules:
    - Listings
    - User info
    - Chats

  Frontend uses API calls to communicate with service modules and interact with databases.
- User service handles any actions related to user info, including login info and created listings. Password management is handled by another database that has limited access to improve security. Info regarding created listings will be communicated to the listing management service.
- Listing management service handles any actions related to listings, including viewing and editing. This service is interconnected with the user service. Users' detailed listings can be viewed by calling the listing management service, and creator of a listing can be found by calling the user service
- Chat service is more independent in the sense that it can only be called & accessed by the two users in a chat. Each chat session will refer to a specific listing, which can be accessed by calling the listing management service. The messages in a chat session will be stored separately in a message database, which will be viewable only to the two users in the chat, improving privacy and security of sensitive info.

Monolithic structure:



When users make a request, corresponding controller is called and data access/modify call is then sent to databases through the models. Retrieved info is then displayed to the frontend pages.

Comparison:

- Monolithic:
  Since the entire service will have one centralized database, it is easier to manage and control access to important data and universal policies can be applied. However, an attack can affect the entire system due to lack of isolation. As the number of users increase in the future, scaling will be a challenge.
- Microservices:

With multiple services in place, better security isolation can be achieved. The risk of full system compromise is reduced, but the implementation and maintenance complexity will increase due to multiple databases and services that need to be individually secured. Moreover, multiple services and databases introduces more potential points of attack.