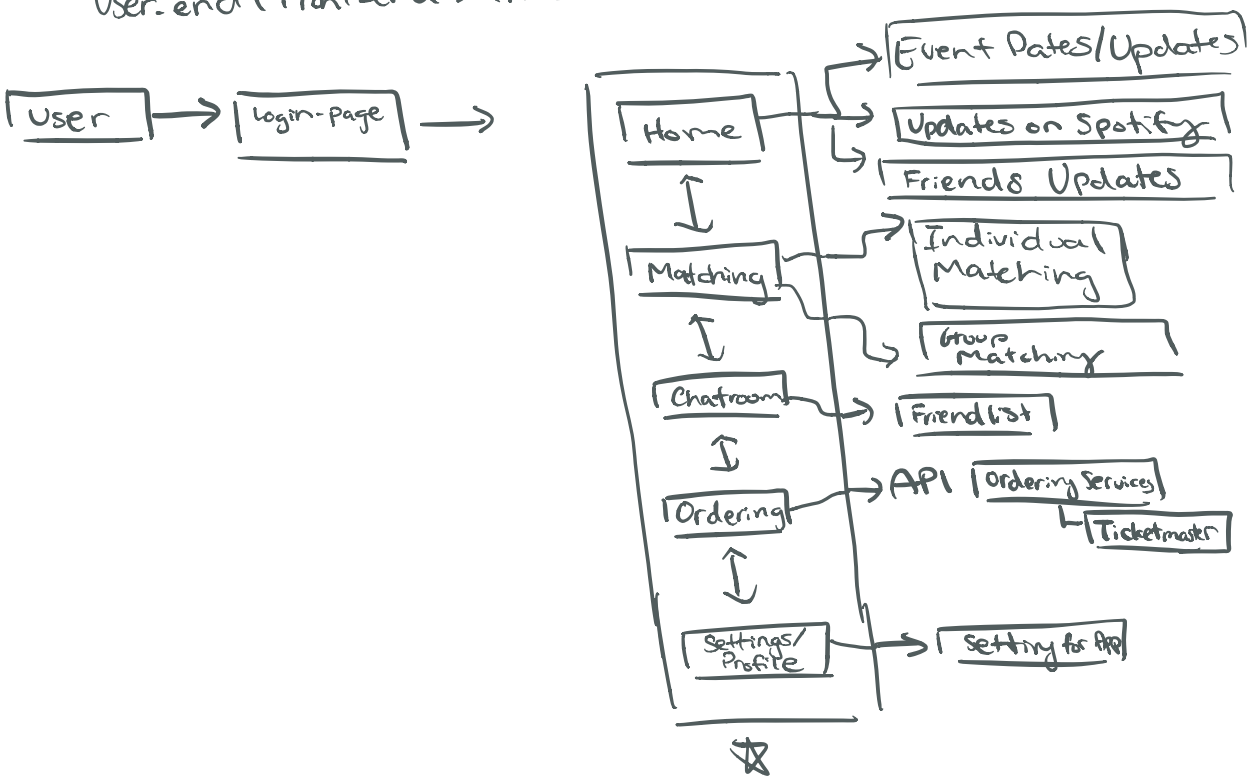
Figure 1 :

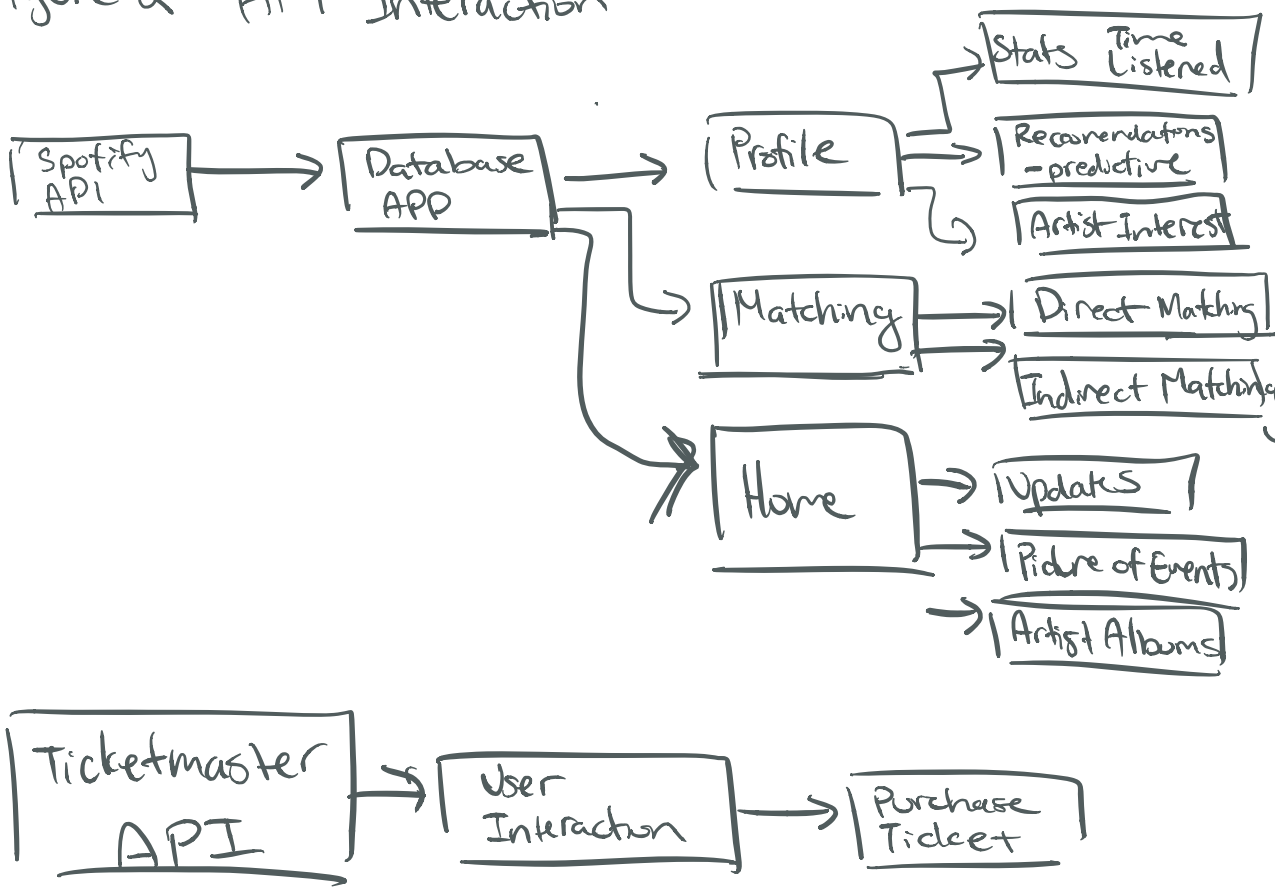
User-end (Front-end) interactions



☆ - generic-layered architecture

This first User front-end architecture serves to basically display the functions and services provided to the user from the application perspective. The User will have a login page where it authenticates them into their appropriate account. Then after logging in the User will have a bundle of web-server platforms/services provided. The home page is responsible for displaying the main event dates/updates from spotify API that notifies user of new albums, new drops, and friends that display an interest in artist or event.
Then the Matching is responsible for two function which is Individual Matching , Group Matching, (and an potential additional function: Event Broadcast for Big Groups/Events)
Chatroom will just be a basic messaging room. that is linked with a personal authenticated friend list.
Ordering will link API to ordering services such as Ticketmaster and will allow you to order through the app.
Settings/Profile will set up the user's profile for interests/artists/event interests and settings is just for app configuration.

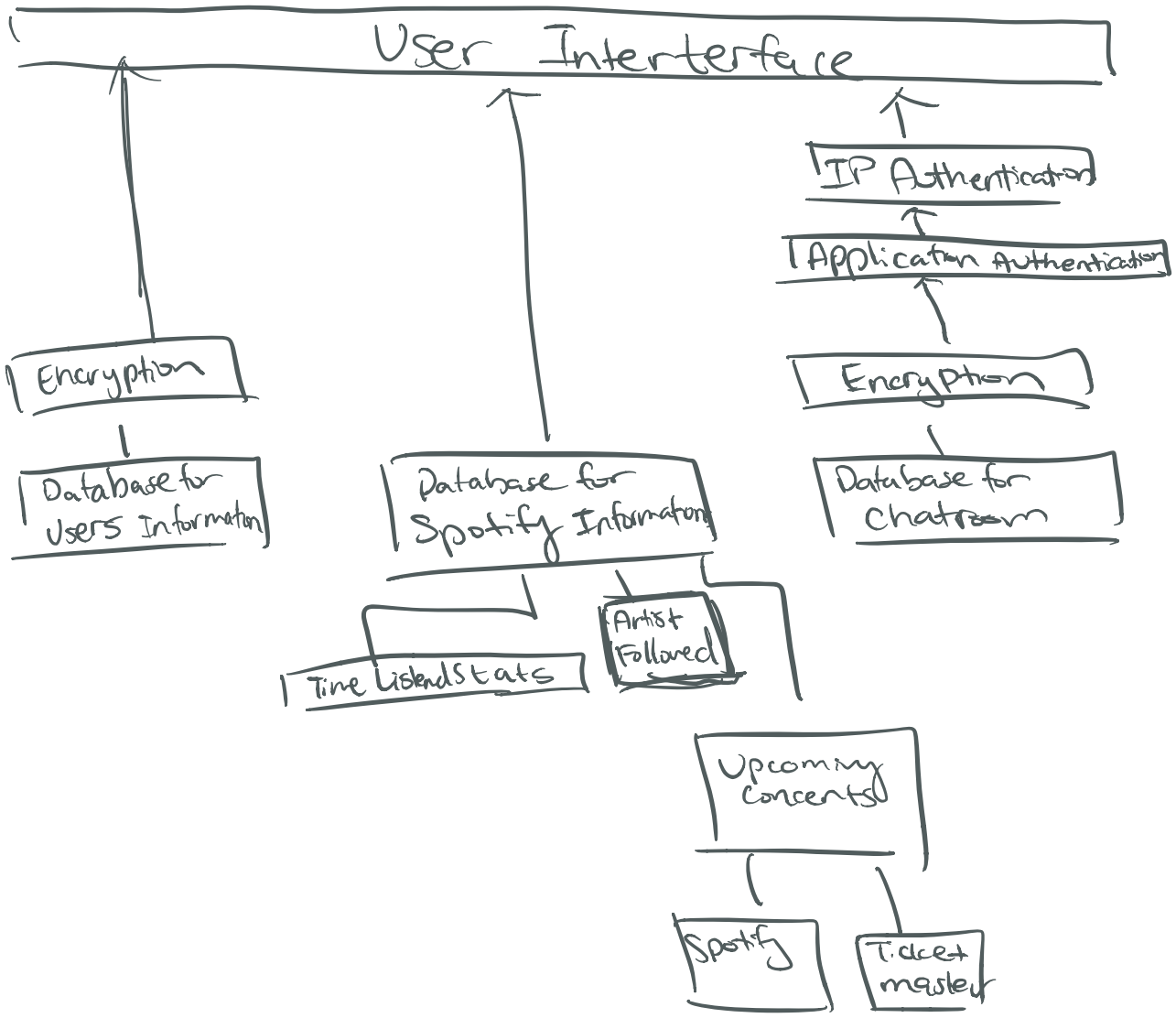
Figure 2 : API Interaction



This API Interaction page will just include two APIs. Spotify API for collecting data and sending it the to Jamify APP and this will affect some attributes in their profiles such as the ones listed like stats, recommendation that are predictive, as well as artists interest. Matching will be direct matching and indirect matching. Direct matching will be matching that is matching solely based on same/similar artists and song taste. Indirect matching will be basically matching based on predicative and past behavior that matches people not on similar taste but on what the algorithm thinks two people with certain different listening behavior would match up well together. Home will be updated with events occurring and artists album updates.

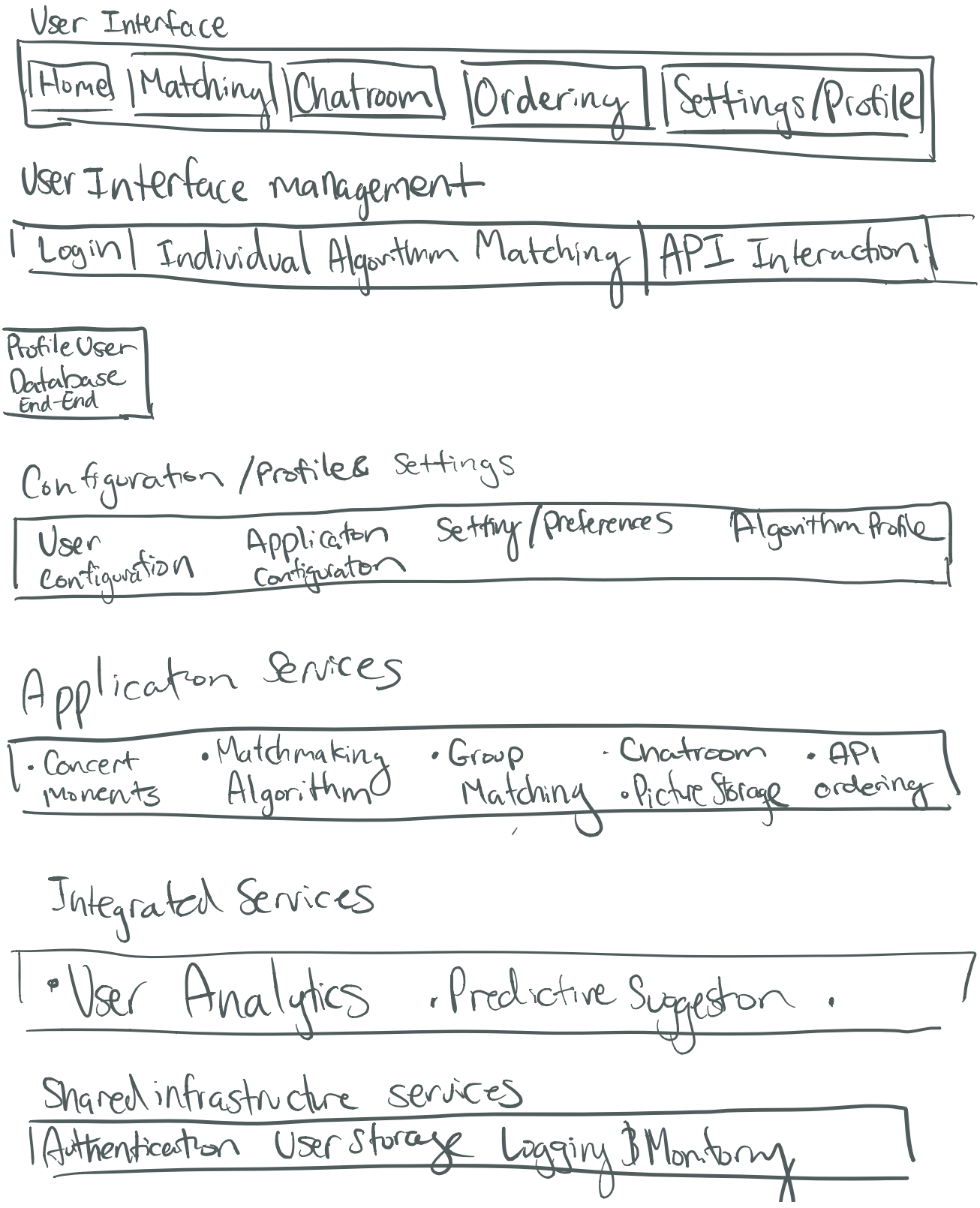
Ticketmaster API is basic user interaction with payment authentication and purchasing ticket if the API exists if it doesn't, the app will send user over to the ticketmaster page responsible for the event.

Figure 3: User Databases



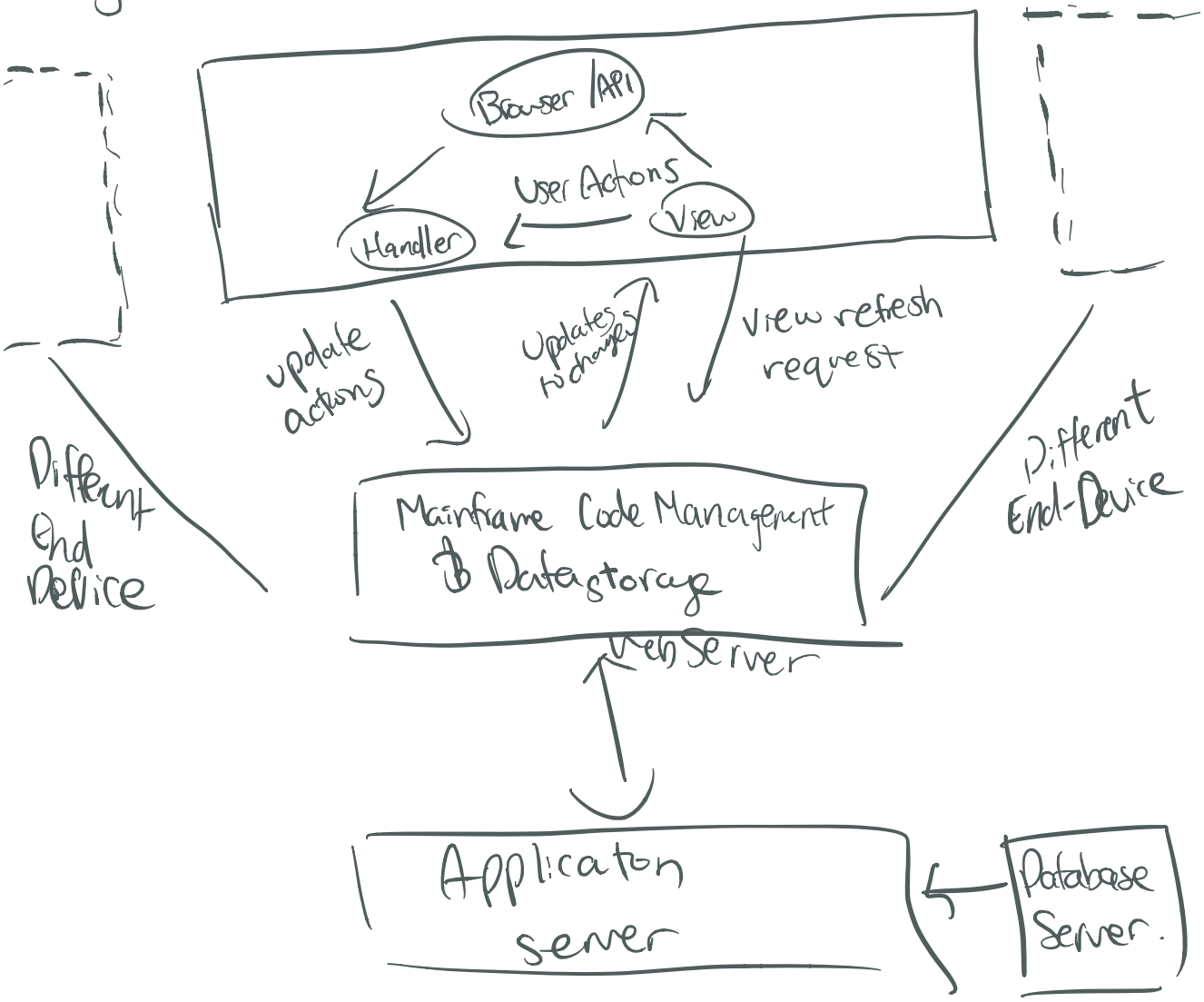
This architecture is a basic overview of User interface interacting with the encryption of Database Users Information, Database for Spotify Information, as well as Database for user information and chatrooms. This architecture will have addition authentication for whenever there is a link to sensitive information and will ensure to design the appropriate encryption for each layer.

Figure 4:



This architecture is a basic layered overview on the whole Jamify Application and the different services that is needed to be programmed and placed in. The setting and profiles will be an important part on how it interacts with the other services.

Figure 5 Client Server Architecture



This is the client-server architecture will be interacting. There may be a better architecture upon further research on how the concurrent most popular forms of the dating apps are functioning. The basic architecture will be the user device performing user actions on handler that checks with the Web Server/Mainframe on whether the update actions are viable and then will check with the application server which is linked to the database server to interact with the web server and thus return the refresh request for response.