**Idei Sistem de notificare a clientilor cu privire la reducerile de pret a unor produse alese**

-interfata frumoasa pe mobile (android/ios)

-sistem de inregistrare+log in cu criptare de parola

-user-ul poate cauta produsele dorite direct din aplicatie, si sa le adauge la favorite, acesta fiind notificat cu privire la reducerea preturilor

-notificare pe mail/whatsapp/push notification

- partea de back end am vzt ca se poate face simplu din python

– folosim price scraping pentru a prelua datele cu privire la produse de pe site uri

-baza de date in care sa storeuim datele despre useri si produse

- pot sa ma focusez de exemplu pe site uri de fashion, ca sa nu fie prea mare plaja de cautare

-la cautarea produselor din aplicatie ar fi fain ca user-ul sa poata sa adauge niste filtre de pret/categorii

-folosesc git pt a salva codul

Aplicatii de price alert:

1)Notify Me

Ai posibilitate sa te loghezi/inregistrezi, apoi ca sa te “abonezi” la notificari, completezi un form, practice adaugi un site la care vrei sa-i urmaresti schimbarile, si ai optiunea sa fi notificat doar cu privire la anumite cuvinte, pe care le selectezi in form. Notificarile se trimit pe mail sau pe discord, in general verifica site-urile odata la 2 ore si se trimite o poza highlightuita cu verde pt text adaugat, rosu pt text sters, si galben pt text modificat.

-Discount Reminder

-The Watchlyst-Price Alerts

-Price Drop FREE

-Alert Me! Price Alerts

Structura App:

**Server**:

* **Database**: Store information about items, prices, user profiles, wish lists, and historical price data. You can use a relational database or NoSQL database, depending on your needs.
* **Price Tracking Service**: Implement a service that regularly checks prices of tracked items from various sources (e-commerce websites, APIs, etc.), and stores this data in your database.
* **User Management**: Handle user registration, authentication, and profile management.
* **Notification Service**: Send notifications to users when discounts are detected.
* **API**: Expose APIs to interact with your server, allowing the mobile and web apps to communicate with it.

**Mobile App** (for Android and iOS):

* **User Interface**: Create a user-friendly interface for users to set their preferences, search for items, manage wish lists, and receive notifications.
* **Notification Receiver**: Receive and display push notifications from the server.
* **Local Storage**: Store minimal data on the user's device, such as preferences and cached item information for offline use.
* **User Interaction**: Allow users to interact with the app, mark items as purchased, share deals, and provide feedback.
* **Web App** (Optional):
* A web app can complement your mobile app, providing users with access from desktop and other devices.
* It may include features for users to manage their profiles, wish lists, and preferences.
* It can be a simpler version of your mobile app for users who prefer a web interface.
* **Backend**:
* Develop the server-side logic using a backend programming language or framework. Popular choices include Node.js, Python (Django/Flask), Ruby on Rails, or Java.
* Host your server on a cloud platform like AWS, Azure, Google Cloud, or a managed platform like Heroku.
* **Database Management**:
* Choose a database management system (DBMS) that fits your needs, such as MySQL, PostgreSQL, MongoDB, or Firebase Realtime Database.
* Implement database backup and data retention policies.
* **Authentication and Security**:
* Implement secure authentication mechanisms to protect user data and privacy.
* Use encryption for data in transit and at rest.