DASS PROJECT – Spring 2024

LEARNPERK

Team Members:

1. Modumudi Naga Sai Ritvik
2. Veerapuneni Keshava Kishora Nanda
3. Navdha Bansal
4. Paridhi Jain
5. Hari Chandana

Client:

1. J. Ramachandran Sir

**Technologies Used:**

**For UI:** Kotlin and Jetpack Compose Framework in Kotlin, Material 3

**For Backend:** Firebase, Kotlin for fetching from Firebase

1. **Things we built:**

All Screens are in the “ui” folder “main/java”. All the navigation is in the “Navigation” folder.

1. The Landing Page (MenuScreen.kt) – Serves as a menu to select the transit app you want to experience LearnPerk with.
2. Different Landing Apps (in their respective folders, we made: IRCTC, Uber, and RedBus) – Home Pages of their respective apps.
3. The User Selection Screen (Users.kt) – Two users are hard-coded in the DB; in this screen, we can select the user we want to login as.
4. LearnPerk Home Screen (LPScreen.kt) – Serves as the home screen, where we can search and select the video to watch. Videos are shown sorted according to the user's interests. Videos can be searched, and search history is stored.
5. The Video Player (Player.kt) – Used an open-source YouTube player found on GitHub which plays videos given the id of YouTube videos. Watching a video increases 10 coins and after a video is done, it shows a pop-up that redirects us back to the LP Home Screen.
6. E-Commerce Page (Ecommerce.kt) – Shows the list of products with their discounts sorted by user interests, with products listed from different sources. Implemented a basic logic that links the coins earned with discounts, and buying a product deducts a relevant number of coins and stores the transactions.
7. User Interests (Interests.kt) – A screen in which a user can view and modify their current interests.
8. Transaction History (LPTransactions.kt) – A screen in which the user can view their past purchase history (when tapped, give some extra details like time and date), as well as videos they watched before.
9. Backend (Backend.kt) – File containing all the relevant backend code.

On the server side in Firebase, we have db2 (collection) -> doc (doc). We have the following collection there:

1. users – Contains all the user docs. The id of this doc is the user ID.
2. content – Contains all the video docs with the ID being the YouTube ID of the video.
3. metadata – Contains the docs storing the information of various e-commerce sources.
4. products – Contains all the e-commerce product docs.
5. tsc - Contains all the transaction docs.
6. **Extensibility:**
7. Adding different Transit Apps.

* For POC (proof of concept) we can add some more landing pages of different transit apps.

1. Adding products from different sources.

- For POC (proof of concept) we can add some products from different sources.

1. Predictive and keyword search functionality.

* We can add predictive search functionality, and keyword suggestions based on the interests and actions of the user.

1. Integration with real E-commerce (along with the addition of discounts on products, and transactions).

* Fixing discounts for each product from different sources.
* About payment process (redirecting to third-party E-commerce app)

1. Integration with real Transit apps.

- Placement of LearnPerk button in their app.

- Colour scheming to follow when integrated with a real transit app.

1. Increasing micro-money based on the amount of time users watched videos.

* Fixing algorithm to Increase the micro-money-based duration of videos watched.
* Skipping video option. (whether to keep it or remove it)
* Some security concerns (like continuously watching videos, running a bot to play videos)

1. User Authentication.

* Do all users using the transit app have a default account in the LearnPerk system? We need to figure this out.
* Is user data shared between transit apps and the LearnPerk team? – We need to figure this out.
* How the Authentication process will be.

1. Recommended system.

Approach to recommendation system:

1. Stage-1:

Recommending using LLMs because we can have users from different demographics. Since LLMs have diverse knowledge. It is better to use them as a starting point.

How to use LLMs (Multi-Shot Prompting):

1. Give the list of products and their details to LLM.
2. Initially, interests are taken when a new user enters.
3. Store user profile along with interests.
4. Generate an order of products based on interests using LLMs.
5. After observing some actions of the user, update the interests in the profile of the user.
6. Again, generate an order of products based on interests using LLM’s.

Some prompts we tried with ChatGPT:

We generated some random data of products as shown below.

We generated some users from different demographics as shown below.

We have prompted ChatGPT initially as:  
“Here is list of products in the E-commerce app:\n” + list of products + “\n” +

“Profile of user:” + profile + “\n” + “Recommend the products for the User such that all products should be displayed but in the order of recommendation.”

Then it gave a list of products in the order. After that the prompting becomes easy.

“User Profile:\n” + profile + “Recommend”.

Then it starts giving the required order of the list of products based on the interests of the user and demographic.

1. Stage-2:

After we get a significant number of users, we can use a regular recommendation Algorithm (Collaborative filtering, content filtering, etc...)

We must decide on it in the future.

We must also keep in mind that the model we built must be verifiable. This can be done using collecting a dataset consisting of user demographics and purchases and seeing if the results are similar.

Also, the prompt can be tweaked to get more desirable answers from the LLM.