🥕 전자제품 거래플랫폼 KKARROT 🥕

목차

1) Overview

2) 구조 소개

3) 시행착오 및 해결 방안

4) 더 추가할 (공부할) feature

Github

https://github.com/bw-99/kkarrot

1) Overview

전자제품 거래 플랫폼, KKARROT

1. 소개

실제 ML 모델을 배포하는 과정에서 생길 수 있는 다양한 이슈사항을 접하고 공부하기 위해 시작.

7월 21일 ~ 7월 29일 동안 ML 학습, ML model deploy (backend & frontend), 협업을 위한 docker image 배포 진행.

ML 모델은 Amazon eCommerce dataset (https://amazon-reviews-2023.github.io)을 이용해 학습.

2. 구현한 기능

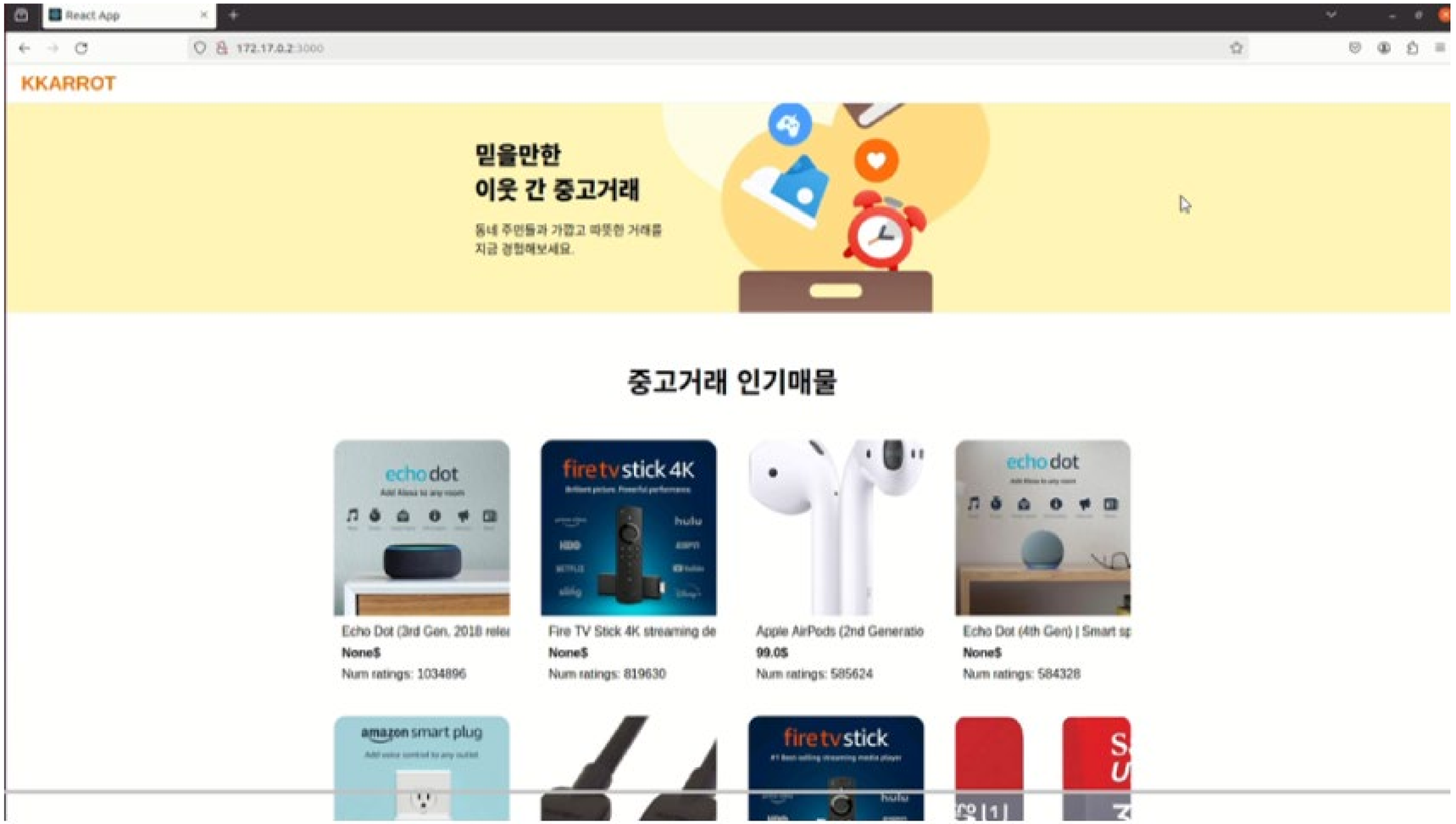
1. Machine Learning
    1. 홈 피드 추천 (deep neural network based collaborative filtering)
    2. 프로덕트 피드 추천 (session-based recommendation)

2. Backend & Frontend
    1. 로그인 세션 관리
    2. 비정상 request 핸들링

1) Overview

전자제품 거래 플랫폼, KKARROT

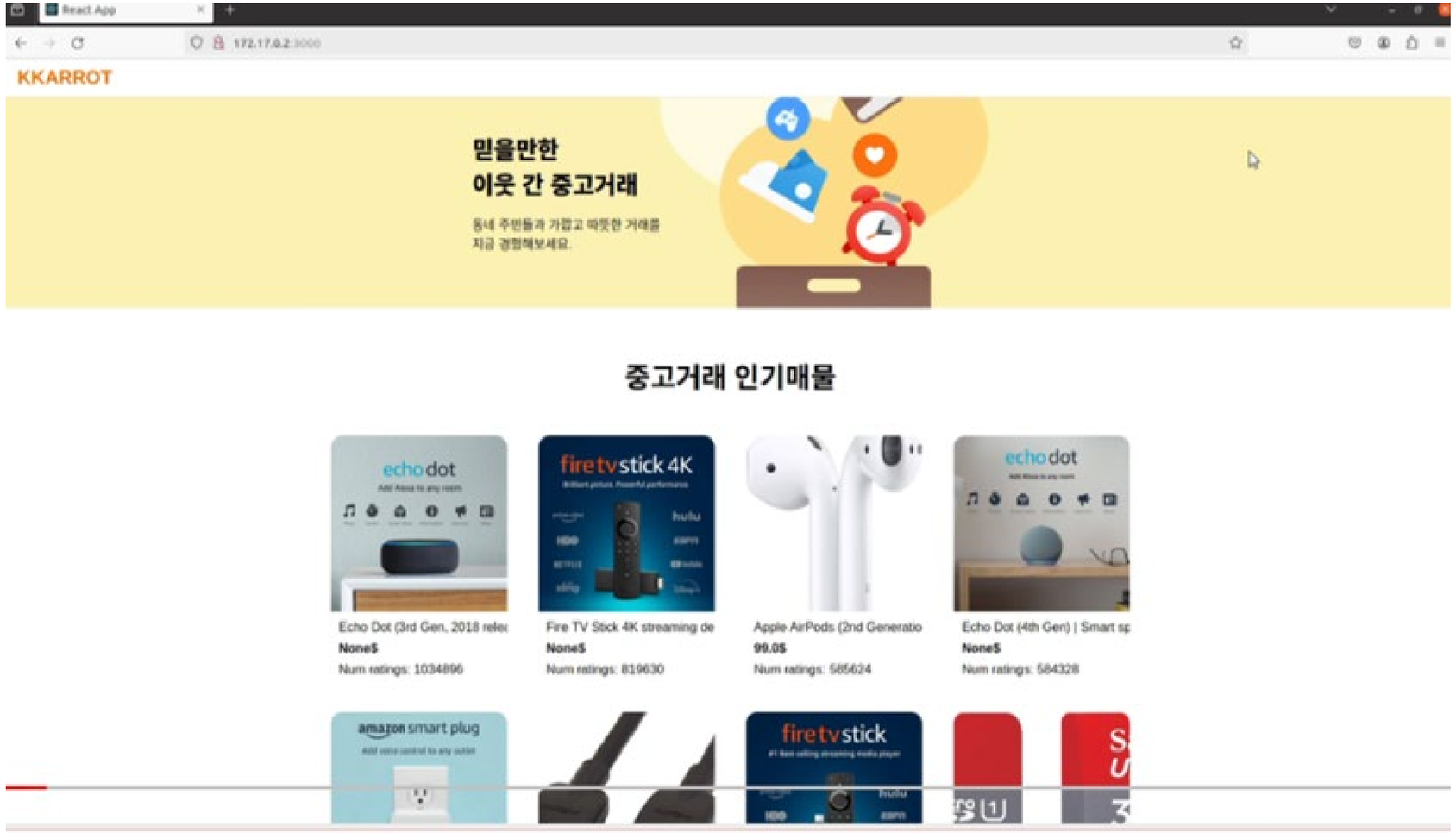
Demo (홈 피드 추천 (인기 매물 더보기))



1) Overview

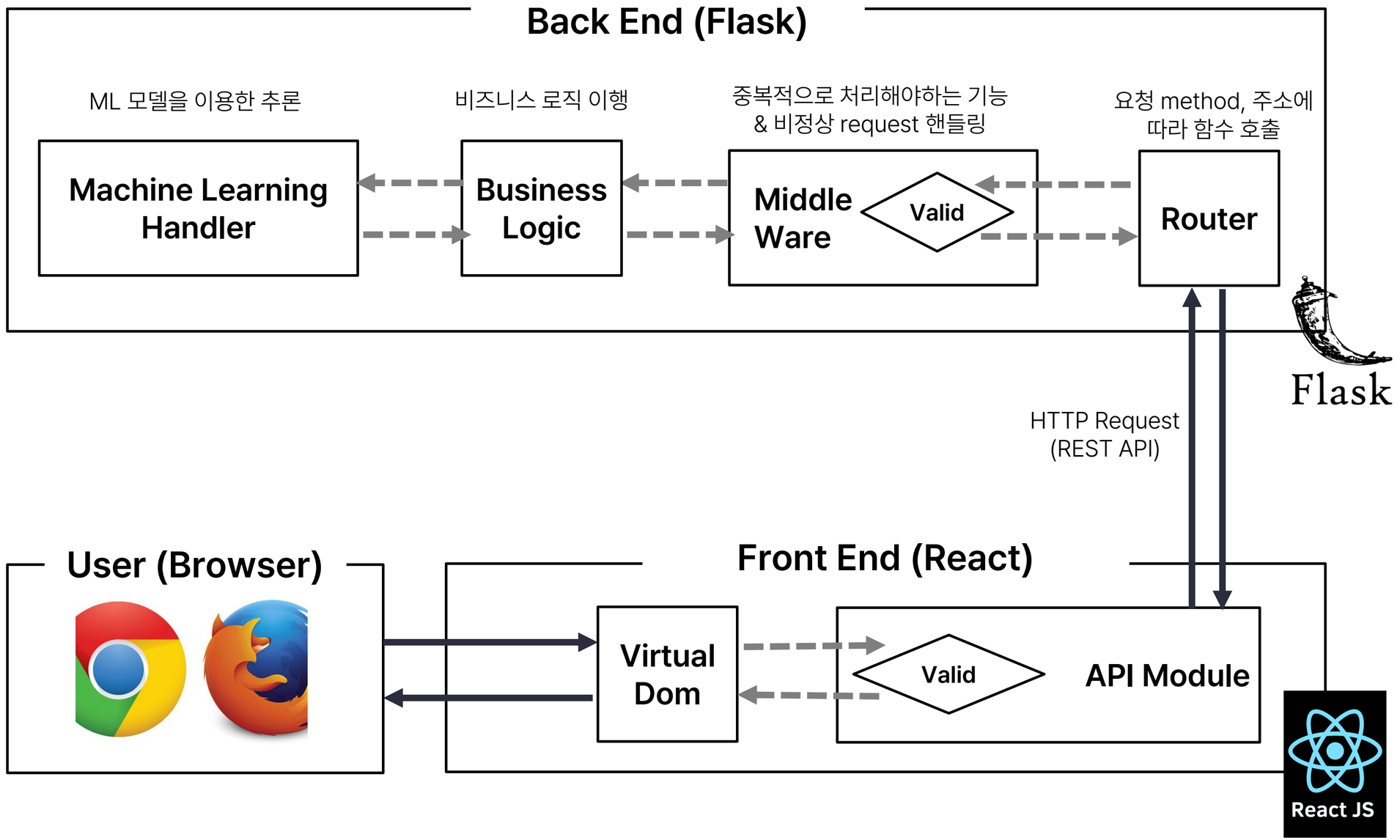
전자제품 거래 플랫폼, KKARROT

Demo (프로덕트 피드 추천 (당근 인기 중고 섹션))



2) 구조 소개

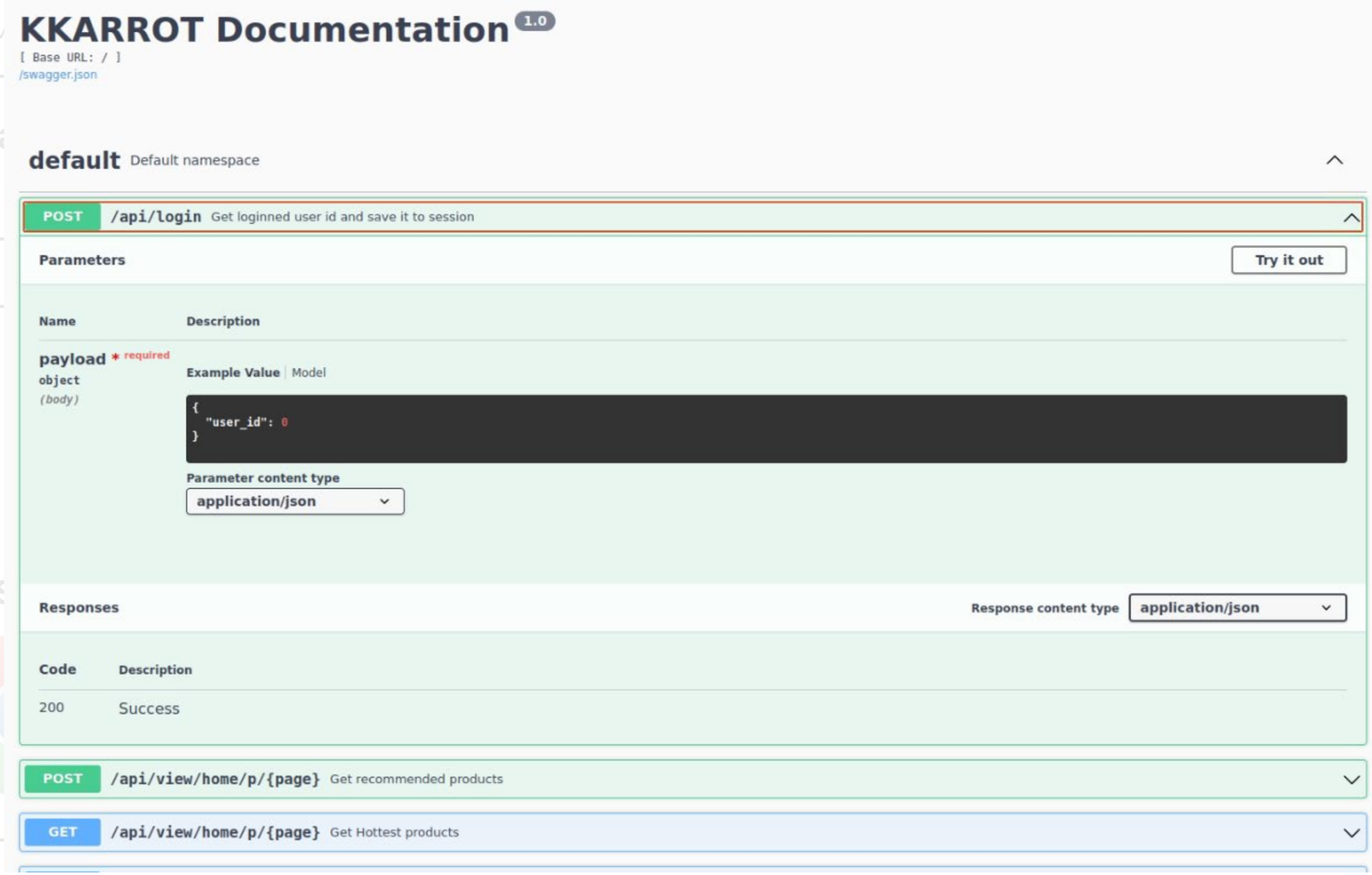
전자제품 거래 플랫폼, KKARROT



2) 구조 소개

전자제품 거래 플랫폼, KKARROT

효율적인 Frontend 개발을 위한 Swagger 문서 작성



3) 시행착오 및 해결 방안

1) 홈 피드 추천에서 지나치게 많이 소요되는 Inference Time

1. GNN Link Prediction을 이용한 Home Feed 추천 시도

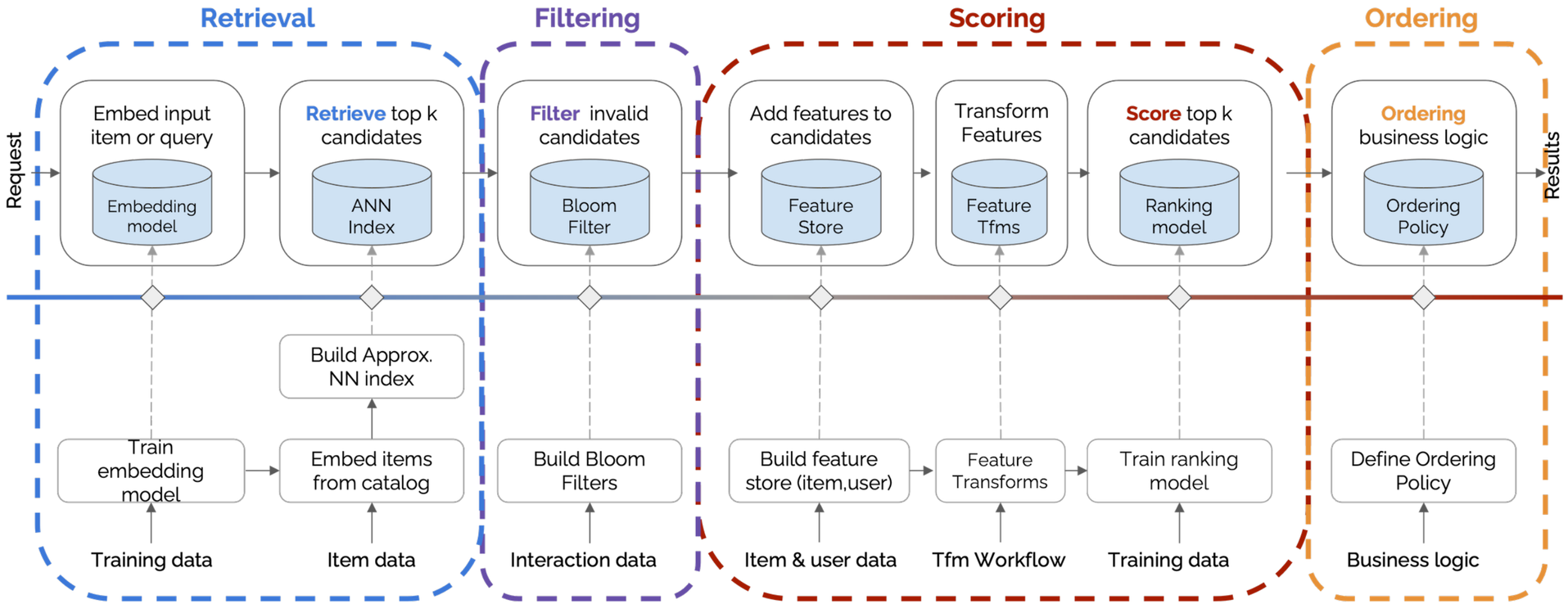
2. 이럴 경우 한 API request 당 평균 **1.3초 소요**

3. Jacob Nielsen (HCI 연구자)에 따르면 **1초 ~ 10초가 소요**되면 유저는 **느리다고 느끼며 다른 생각**을 할 수 있다고 함

=> **Mutli Stage Recommendation** 방법론 도입

3) 시행착오 및 해결 방안

1) 홈 피드 추천에서 지나치게 많이 소요되는 Inference Time



1) Two Tower Model

2) 중복 아이템 제거
(이미 많이 노출된)

3) GNN Link Prediction

4) 휴리스틱하게 계산한 Hottest score 추가 합산

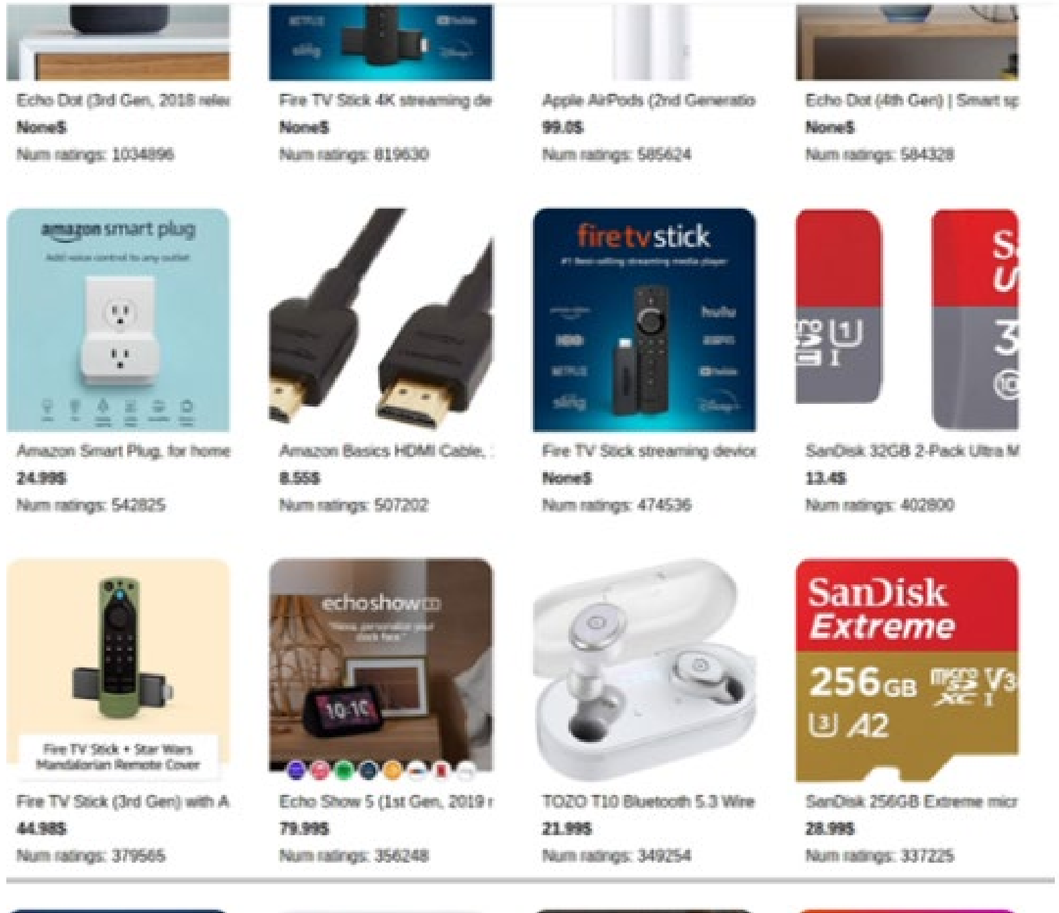
https://github.com/NVIDIA-Merlin/publications/blob/main/tutorials/RecSys22tutorial/04-Building-multi-stage-RecSys.ipynb

3) 시행착오 및 해결 방안

1) 홈 피드 추천에서 지나치게 많이 소요되는 Inference Time

	1 회차 (s)	2 회차 (s)	3 회차 (s)	4 회차 (s)	5 회차 (s)	**Avg. (s)**
Before (GNN)	1.4149	1.2909	1.2868	1.3272	1.3260	**1.3292**
After (Multi Stage)	0.1290	0.1431	0.1223	0.1381	0.1352	**0.1103 (92% 단축)**

적용 전 (Before)



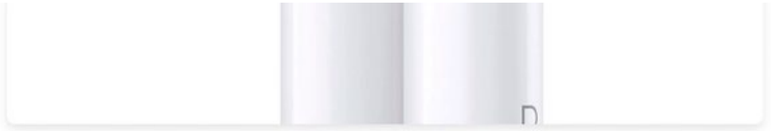
적용 후 (After)

중고거래 인기매물

3) 시행착오 및 해결 방안

2) 프로덕트 피드 추천에서의 Long Tail Problem

1. GRU를 이용한 sequential prediction으로 추천 시도
2. 성능은 Full ranking evalutaion 기준 **Recall@20: 7%**
3. 그러나 아래 사진처럼 대부분 저장장치, HDMI 같은 제품만 추천됨



Apple AirPods (2nd Generation) Wireless Earbuds with Lightning Charging Case Included. Over 24 Hours of Battery Life, Effortless Setup. Bluetooth Headphones for iPhone
Apple Products
99.0

Echo Dot (2nd Generation) - Smart speaker with Alexa - White
Amazon Devices
39.99

WD Blue internal hard drives deliver reliability for office and web applications. They are ideal for use as primary drives in desktop PCs and for office applications. With a range of capacities and cache sizes, there's a WD Blue internal hard drive that's just right for you.

당근 인기중고

Amazon Basics HDMI Cable, 1
8.55$
Num ratings: 507202

Western Digital 4TB WD Blue
82.5$
Num ratings: 53764

SanDisk Ultra 32GB UHS-I/Cla:
8.29$
Num ratings: 47325

당근 인기중고

Amazon Basics HDMI Cable, 1
8.55$
Num ratings: 507202

SanDisk Ultra 32GB UHS-I/Cla:
8.29$
Num ratings: 47325

SanDisk Ultra 32GB microSDH
8.99$
Num ratings: 115781

당근 인기중고

Amazon Basics HDMI Cable, 1
8.55$
Num ratings: 507202

Western Digital 4TB WD Blue
82.5$
Num ratings: 53764

SanDisk Ultra 32GB microSDH
8.99$
Num ratings: 115781

3) 시행착오 및 해결 방안

2) 프로덕트 피드 추천에서의 Long Tail Problem

1. 2020 RecSys에 publish된 Long-Tail Session based Recommender System (TailNet) **논문 내용 토대로 구현**
2. 간단 요약: 모델의 성능을 **해치지 않는 선**에서, Recitification factor를 학습하여 **부드럽게 tail item을 추천할** 수 있도록 하는 방법

Methods	30MUSIC				
	MRR@20	Recall@20	Coverage@20	Tail_Coverage@20	Tail@20
POP	0.18	0.69	0.01	0	0
S-POP	8.20	18.98	15.52	10.55	20.16
Item-KNN	15.71	37.68	75.14	79.04	54.60
FPMC	9.17	14.47	84.37†	97.45†	60.00†
BPR-MF	6.97	12.25	49.62	86.95	19.94
GRU4REC	20.45	**39.12**†	36.60	26.18	18.64
NARM	20.19	36.68	45.51	36.99	30.84
STAMP	13.12	23.34	14.10	3.81	12.95
RepeatNet	18.01	33.01	32.24	24.49	23.06
SR-GNN	27.28	37.86	40.59	28.71	20.48
TailNet without PM	28.62	39.00	34.53	21.33	11.56
TailNet-propotion	27.91	37.29	38.55	25.37	19.85
TailNet	**28.70**†	38.34	**47.86**	**40.10**	**32.13**

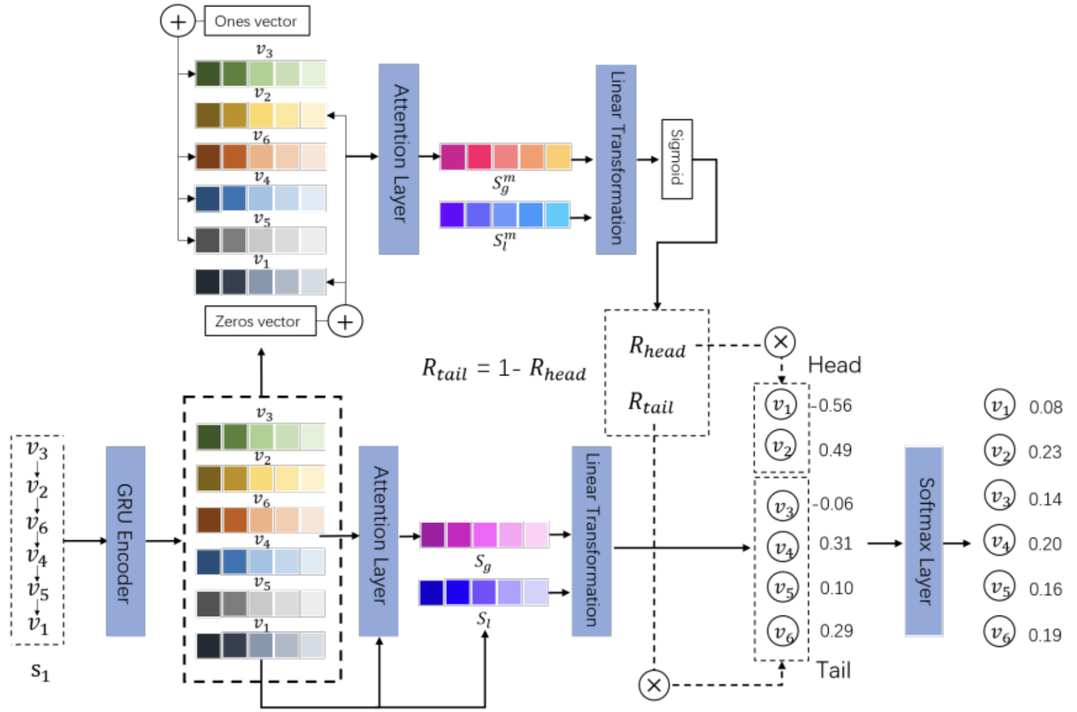


Figure 1: The framework of TailNet. We first encode the session by a GRU layer and get the session's latent representation V. Then it is sent to the *preference mechanism* and the attention layer. *Preference mechanism* uses V to generate *rectification factors* R_{head} and R_{tail}, which softly adjust the mode of TailNet. Next, the global session embedding S_g is generated by the attention layer. After that, the global embedding S_g and local embedding S_l (the last clicked item's representation) are concatenated and sent to a linear transformation. Finally, we predict the probability of each item with the adjusting of R_{head} and R_{tail}.

https://arxiv.org/pdf/2007.12329

3) 시행착오 및 해결 방안

2) 프로덕트 피드 추천에서의 Long Tail Problem

공개된 소스 코드가 없어, **논문 내용 바탕으로 직접 개발 및 학습** 후 모델 서빙 진행

```python
class TailNet(nn.Module):
    def __init__(self, num_items, embedding_dim, hidden_dim, head_mapping_lst):
        super(TailNet, self).__init__()
        self.head_mapping_lst = head_mapping_lst

        self.head_idx = self.head_mapping_lst == 1
        self.tail_idx = self.head_mapping_lst == 0

        self.hidden_dim = hidden_dim
        self.embedding = nn.Embedding(num_items, embedding_dim)
        self.gru = nn.GRU(embedding_dim, hidden_dim, num_layers=2, batch_first=True)

        self.attn_key = nn.Linear(hidden_dim, hidden_dim, bias=True)
        self.attn_query = nn.Linear(hidden_dim, hidden_dim, bias=True)
        self.attn_fc = nn.Linear(hidden_dim, 1, bias=True)
        self.last_fc = nn.Linear(hidden_dim*2, num_items)

        self.adjust_attn_key = nn.Linear(hidden_dim, hidden_dim, bias=True)
        self.adjust_attn_query = nn.Linear(hidden_dim, hidden_dim, bias=True)
        self.adjust_attn_fc = nn.Linear(hidden_dim, 1, bias=True)
        self.adjust_last_fc = nn.Linear(hidden_dim*2, 1)

    def forward(self, item_idx):
        x = self.embedding(item_idx)
        item_emb, _ = self.gru(x)

        # * rectification factors
        item_types= self.head_mapping_lst[item_idx].view(item_idx.shape[0], item_idx.shape[1],
        item_types = item_types.repeat(1, 1, self.hidden_dim)

        adjust_item_emb = item_emb + item_types
        adjust_last_item_emb = adjust_item_emb[:, -1, :].unsqueeze(dim=1)
```

TailNet 개발

```python
@rest_api.route('/api/view/product/<int:item_id>')
class Product(Resource):
    @item_check
    @login_check
    def get(self, item):
        """Get detailed description of the product."""
        if(not request.is_logined):
            return {}, 401
        user_history[f"user_id{request.user_id}"].append(item["item_id"].tolist()[0])
        click_history = user_history[f"user_id{request.user_id}"]
        if(len(click_history) < 5):
            print("render hottest items")
            feed_lst = meta_df.iloc[0:FETCH_UNIT].to_json(orient="records")
        else:
            print("render recommended items")
            click_history = click_history[-5:]
            user_history[f"user_id{request.user_id}"] = click_history
            top_pred_idx = sequence_recommend(session_rec, click_history)
            top_items = pd.merge(pd.DataFrame(top_pred_idx, columns=['item_id']),meta_
            feed_lst= top_items.to_json(orient="records")

        return {"success": True,
                "item": item.to_json(orient="records"),
                "feed_lst": feed_lst,
                }, 200
```

```python
def sequence_recommend(model:TailNet, history):
    with torch.no_grad():
        preds = model(torch.LongTensor(history).view(1, -1).cuda()).cpu().squeeze()
        pred_idx = torch.argsort(preds, descending=True)[:SESSION_TOPK].cpu()
    return pred_idx
```

모델 서빙

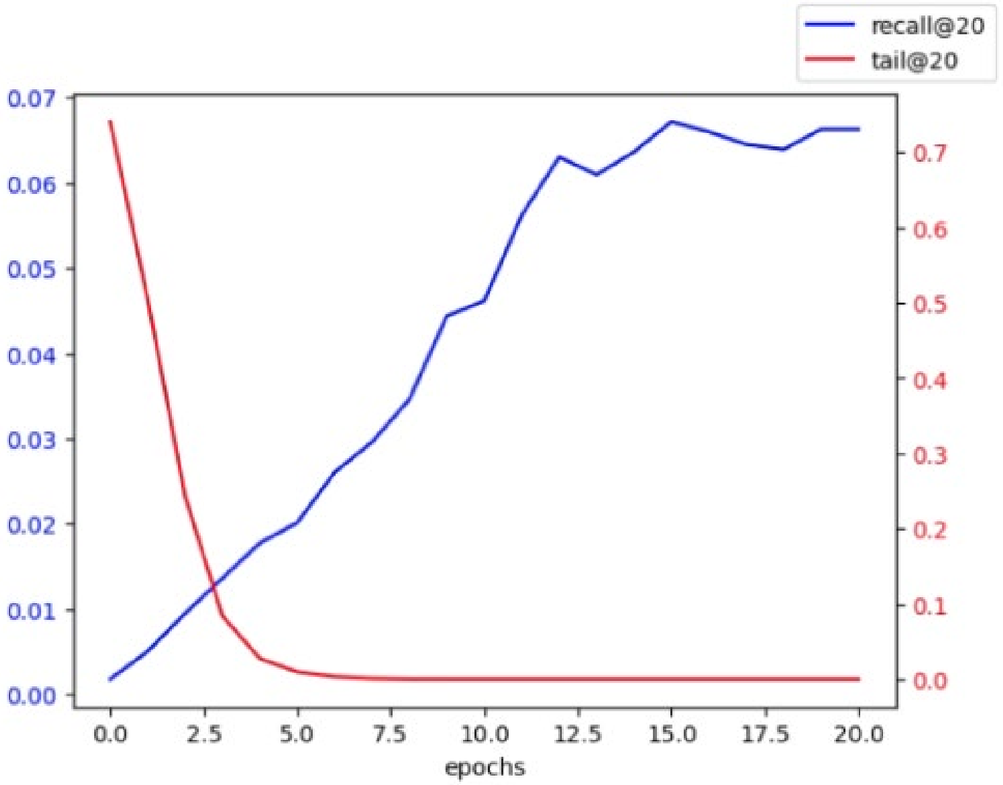
3) 시행착오 및 해결 방안

2) 프로덕트 피드 추천에서의 Long Tail Problem

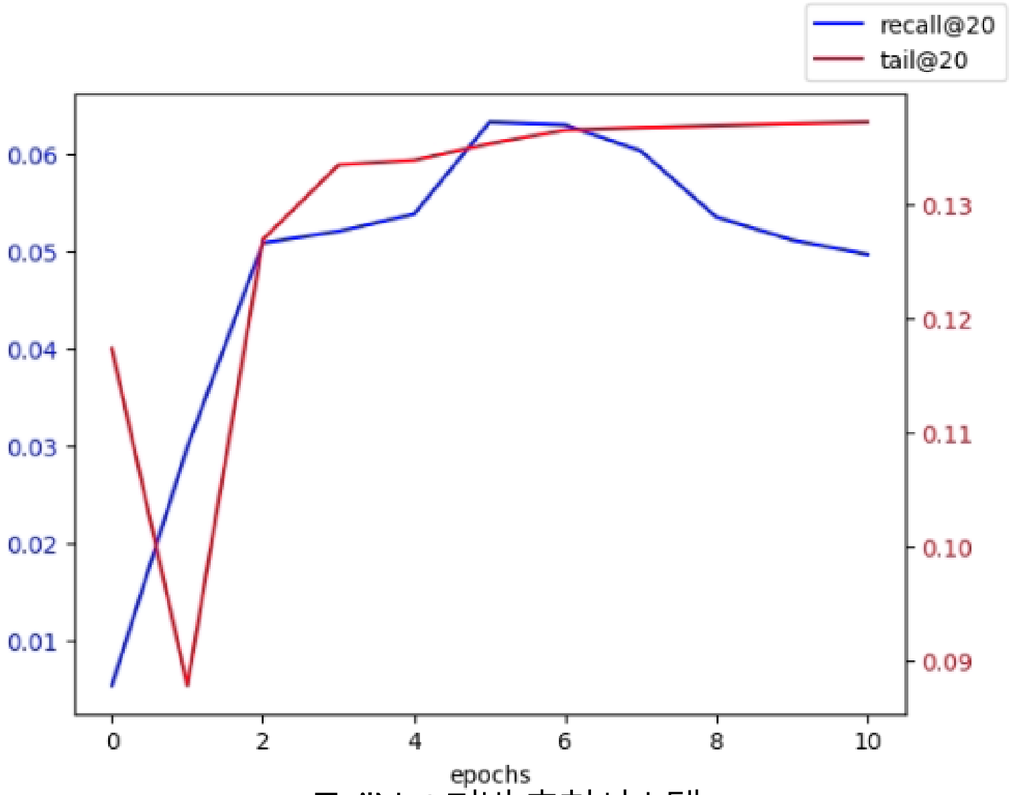
GRU 기반 (좌측) 추천시스템은 학습이 거듭될 수록 **tail이 0에 수렴되는 현상** 발견

TailNet 기반 (우측) 추천시스템은 recall 지표와 **함께 tail이 상승되는** 경향성 발견

(recall@20: 전체 정답 개수 대비 Top-20 아이템 중 정답 개수, tail@20: Top-20 아이템 중 long tail item 개수)



GRU 기반 추천시스템



TailNet 기반 추천시스템

3) 세션 관리의 필요성

1. Session based Recommendation

1. SBR에선 user의 click history를 기반으로 next click item을 예측 후 추천
2. 이때, 각 user마다의 click 기록을 실시간으로 database에 I/O 작업을 하는 것은 비효율적
3. 로그인한 user 세션에 click 기록을 in-memory 형태로 관리하여 효율적으로 추천

2. 로그인 user / 비로그인 user 구별

1. 현재 KKARROT은 Home Feed는 비로그인 유저도 접근이 가능함
2. 하지만 이외 제품 상세 페이지 (SBR), 제품 더보기 (CF) 기능은 로그인 유저만 가능
3. 한번 로그인한 user는 session을 통해 session이 끊길 때까지 계속 개인화 추천 기능을 제공받을 수 있음
4. 비로그인 user가 접근할 경우, session에 user_id가 있는지 검사 후 없으면 401 Error 전송 (Backend) -> 로그인 페이지 렌더링 (Frontend)

=> Session 관리 적용

4) 더 추가할(공부할) feature

1: Continual Learning

1. 모델이 학습한 **데이터(오프라인)**와 서비스 중 추가로 적재되는 **데이터(온라인)**의 불일치 이슈 존재

2. 추가로 적재된 데이터를 학습하는 과정에서, 과거의 지식을 잃어버리는 현상 방지를 위한 Continual Learning 도입의 필요성

3. Knowledge Distillation에 기반한 **Continual Learning에 대한 지식 쌓기 + 이후 개발**