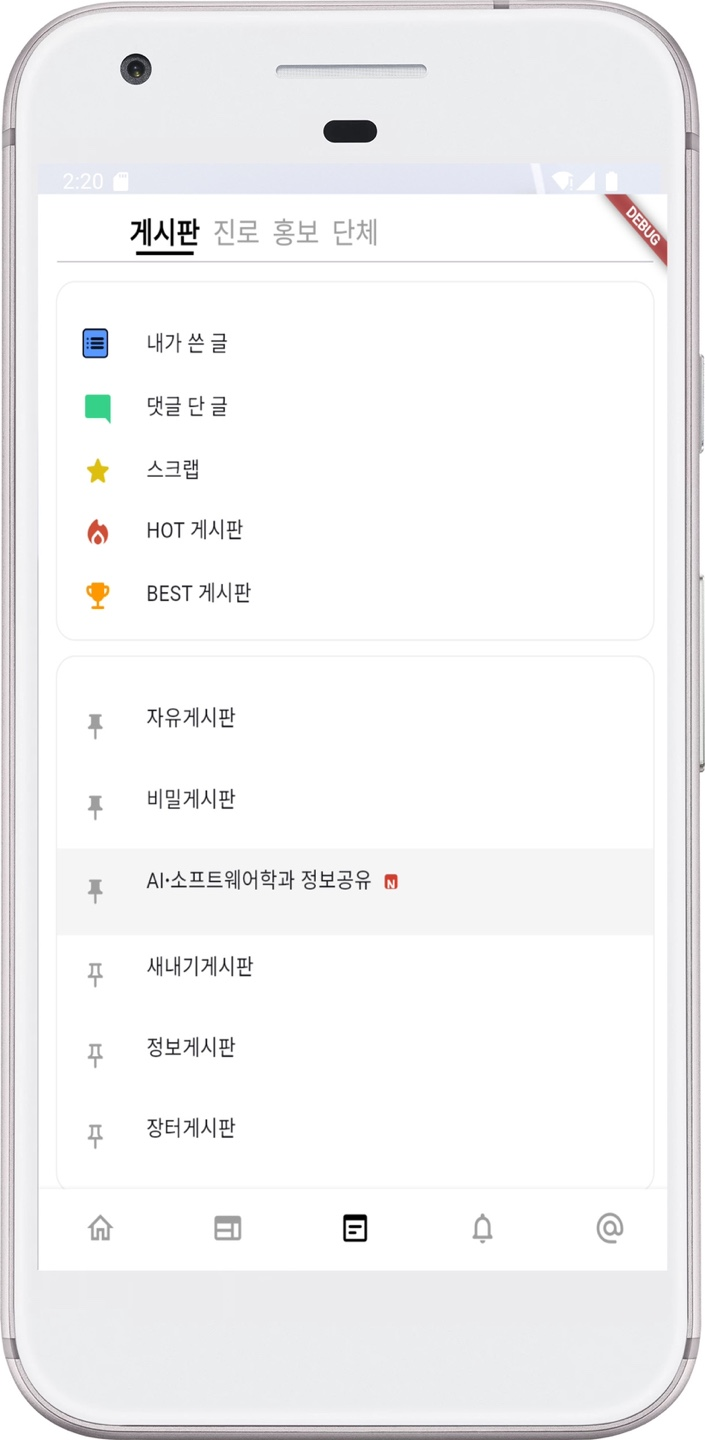
**게시판 view**

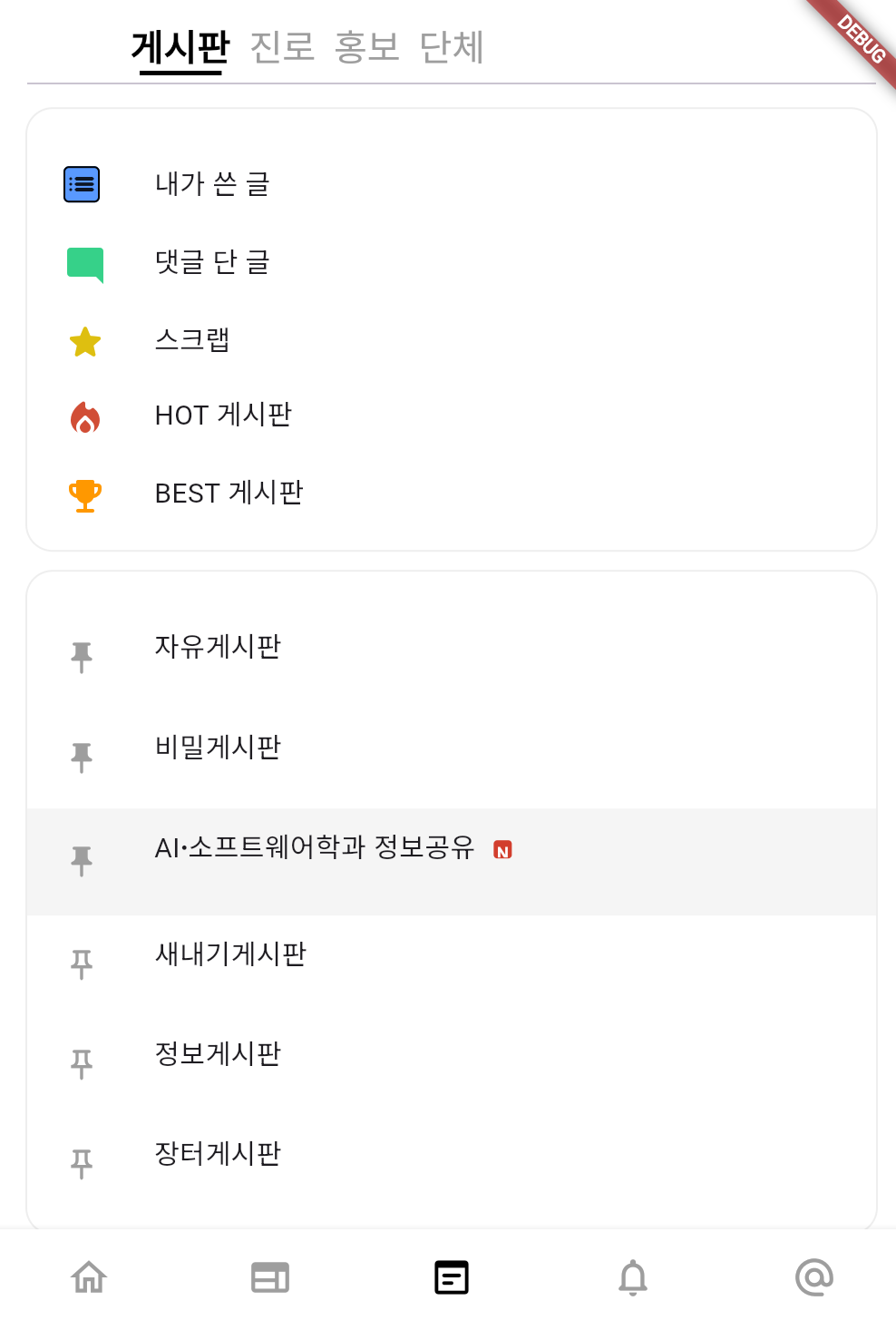


**Data needed to compose the view:**

1. 게시판 제목 2. 사용자가 고정을 누른 게시판 3. 각 게시판에 게시글이 당일에 나왔는지

**Kind of table needed:**

게시판sub(게시판id, 제목) , 게시판(게시판id, 제목), 게시글(게시글id, 게시판id, 글 쓴 날짜), 유저(유저id, 이름), 유저고정(유저id, 게시판id, 고정, 고정한 시간)



**1. 유저별로 고정 게시판 뷰 만들기 (강의 workshop3-2 마이피드 참조)**

- 관련 테이블 : User\_fixed\_bbs

- 유저별로 fix=1인 고정 게시판을 찾는다

- 최근 상태가 fix=0 이라면 고정 게시판이 아니다

- 최종 뷰, User\_fixed\_bbs\_top(i\_user, bbs\_id)

|  |
| --- |
| CREATE VIEW user\_fixed\_bbs\_top AS  SELECT ufb.i\_user, ufb.bbs\_id  FROM user\_fixed\_bbs ufb  JOIN(  SELECT i\_user, bbs\_id, max(update\_time) update\_time  FROM user\_fixed\_bbs  GROUP BY i\_user, bbs\_id  ) a on ufb.i\_user =a.i\_user  and ufb.bbs\_id =a.bbs\_id  and ufb.update\_time =a.update\_time  WHERE ufb.fix =1  ORDER BY ufb.i\_user, ufb.bbs\_id; |
|  |

**2. 게시판 정보 나열하기**

- 관련 테이블 : bbs, contents, user\_fixed\_bbs\_top,

- 게시판 이름, 고정 게시판, 게시판 별 새로운 게시글 유무 정보 나열

- ex) i\_user=1인 사람이 앱을 켰다고 가정한다 (후에 postman에서 i\_user 입력받음)

- 게시판 별로 ‘2024-11-20’일에 올려진 게시글에 한정을 둔다

- ‘2024-11-20’일에 올려진 게시글이 있으면 1표시 없으면 0표시

- 새로운 게시글이 없는 게시판도 나열해야한다

- bbs\_id가 1,2,3,4,5면 가장 먼저 출력

- 고정 게시판 뷰에 있는 게시판들을 그다음 정렬한다. (고정 게시판들 사이에서는 게시판id 오름차순 정렬)

- 나머지 게시판도 게시판id에 대해 오름차순 정렬한다.

SELECT

bbs.bbs\_name,

IF(

EXISTS (

SELECT 1

FROM contents

WHERE contents.bbs\_id = bbs.bbs\_id AND DATE(contents.bbs\_date) = '2024-11-20'

),

1,

0

) AS has\_new\_posts

FROM

bbs

LEFT JOIN user\_fixed\_bbs\_top ufbt

ON bbs.bbs\_id = ufbt.bbs\_id AND ufbt.i\_user = 1

ORDER BY

case

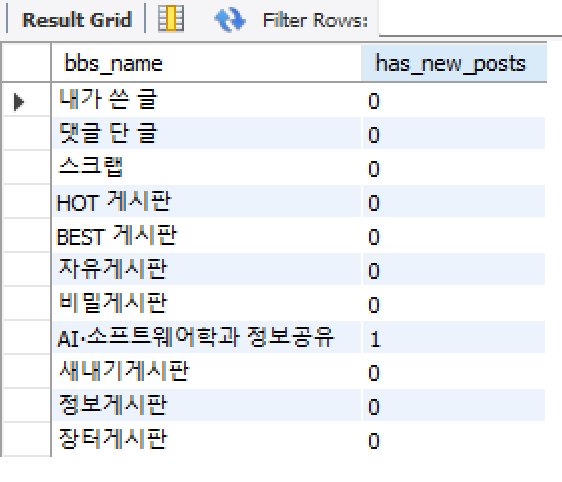
when bbs.bbs\_id in (1,2,3,4,5) then 0

when ufbt.bbs\_id IS NOT NULL then 1

else 2

end,

bbs.bbs\_id;

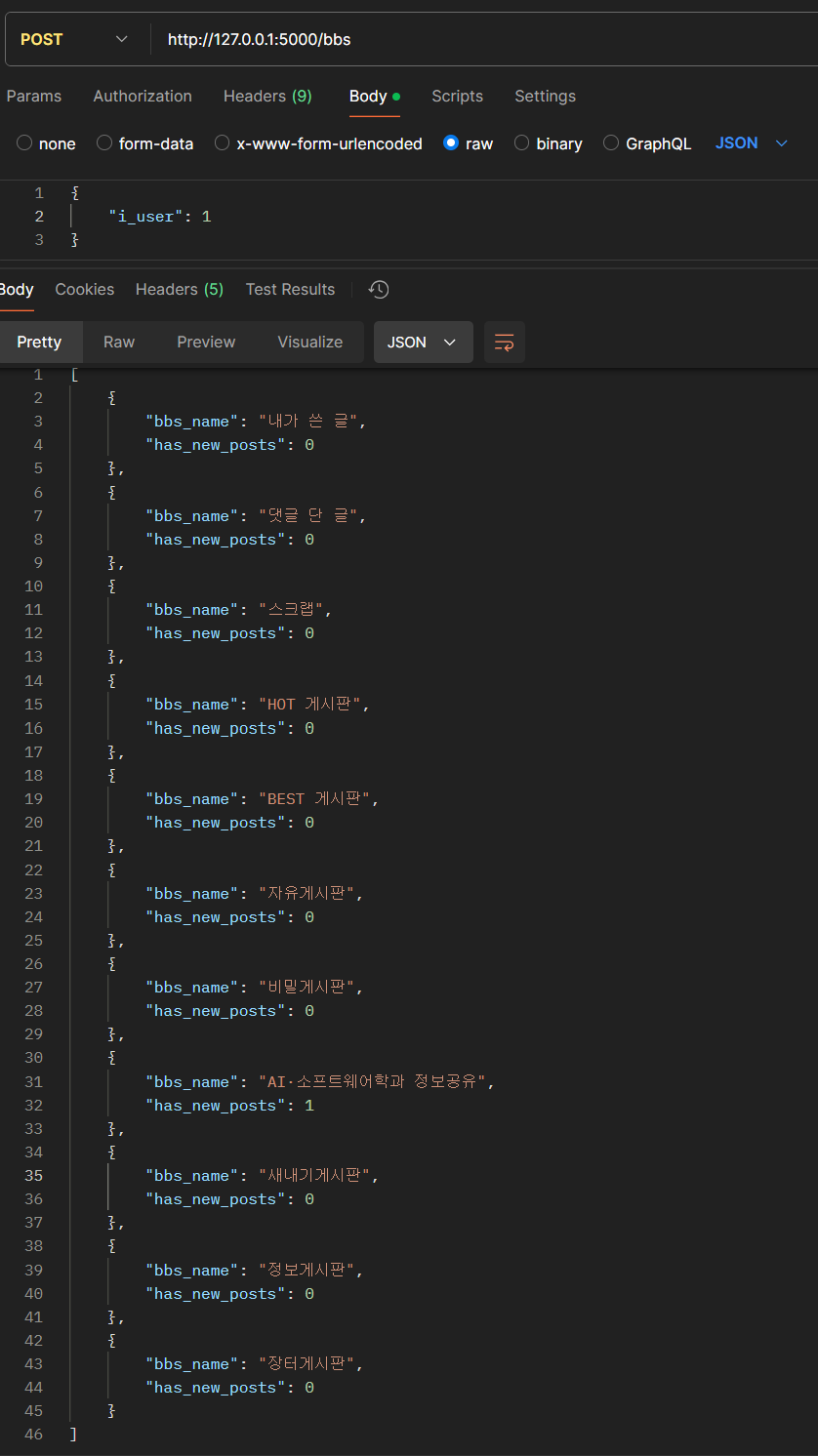
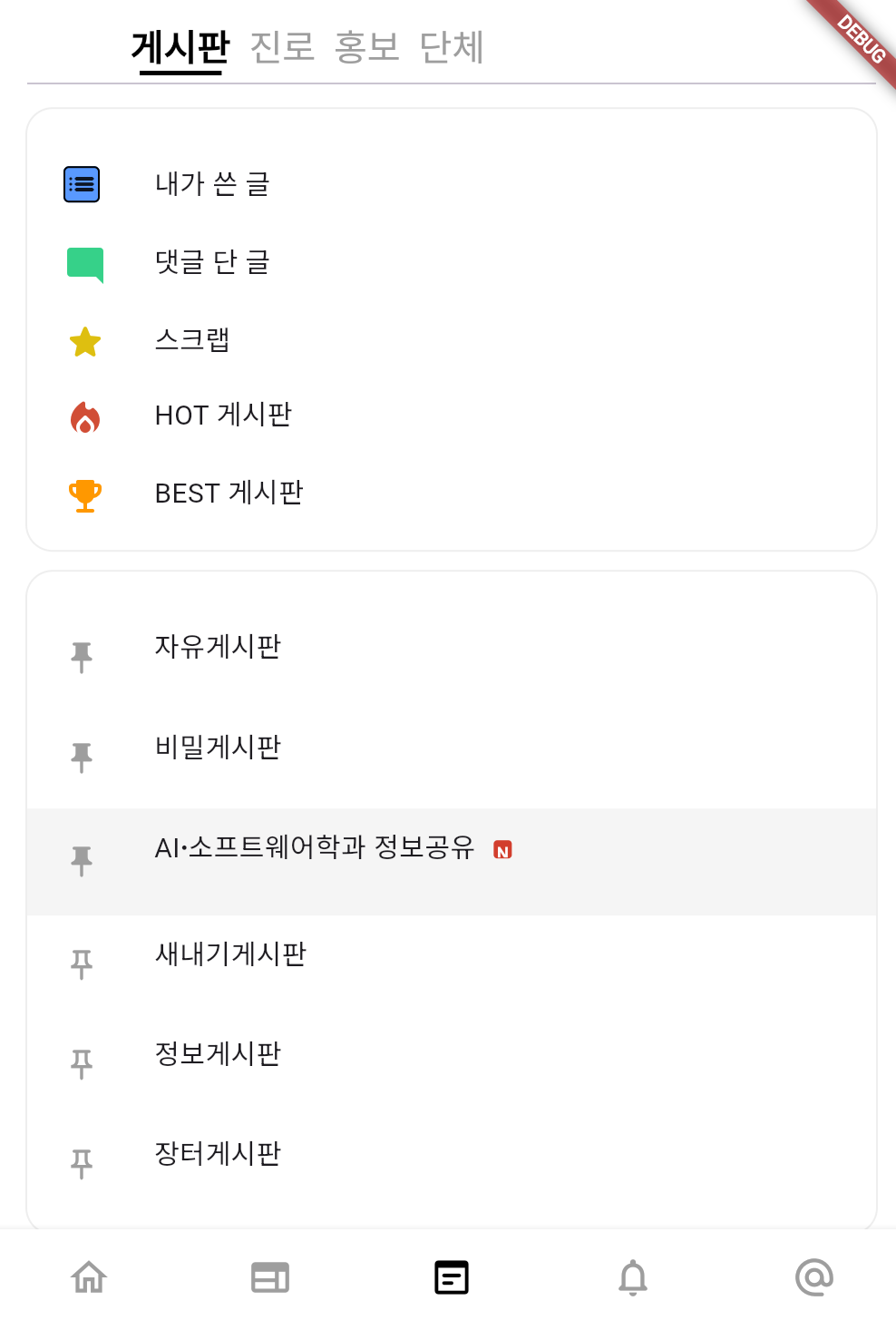
****

**※ BBS python code.**

i\_user는 postman에서 입력 받기

|  |
| --- |
| from flask import Flask, request, jsonify  import pymysql  import pandas as pd  app = Flask(\_\_name\_\_)  @app.route('/bbs', methods=['POST'])  def get\_bbs\_info():  request\_json = request.get\_json()  user\_id = request\_json['i\_user']    conn = pymysql.connect(  host='localhost',  port=3306,  user='root',  password='1234',  db='everytime'  )  sql\_query = f"""  SELECT  bbs.bbs\_name,  IF(  EXISTS (  SELECT 1  FROM contents  WHERE contents.bbs\_id = bbs.bbs\_id AND DATE(contents.bbs\_date) = '2024-11-20'  ),  1,  0  ) AS has\_new\_posts  FROM  bbs  LEFT JOIN user\_fixed\_bbs\_top ufbt  ON bbs.bbs\_id = ufbt.bbs\_id AND ufbt.i\_user = {user\_id}  ORDER BY  CASE  WHEN bbs.bbs\_id IN (1, 2, 3, 4, 5) THEN 0  WHEN ufbt.bbs\_id IS NOT NULL THEN 1  ELSE 2  END,  bbs.bbs\_id;  """  try:  df\_bbs\_info = pd.read\_sql\_query(sql\_query, conn)  return df\_bbs\_info.to\_json(orient="records")  except Exception as e:  return jsonify({"error": str(e)})  finally:    conn.close()  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |
|  |

**※ BBS api :**



**(선택) 게시판 view에 필요한 테이블 및 컬럼 상세 설명:**

**1. Bbs** (bbs\_id, bbs\_name)

- bbs\_id : 게시판 고유 번호

- bbs\_name : 게시판 제목

**2. Contents** (contents\_id, bbs\_id, time)

- contents\_id : 게시글 고유 번호

- bbs\_id : 게시판 고유 번호

- time : 게시글을 작성한 년월일시

**3. User\_fixed\_bbs** (i\_user, bbs\_id, fix, update\_time)

- i\_user : 사용자 고유 번호

- bbs\_id : 게시판 고유 번호

- fix : fix =1이면 고정, 0이면 고정 해제

- update\_time : 고정/고정해제를 누른 일시