Objective:
Your objective is to implement Ad server web application server side.
Some of the server project has already been implemented. Your goal is to complete the unimplemented parts of the server.

Implemented parts:

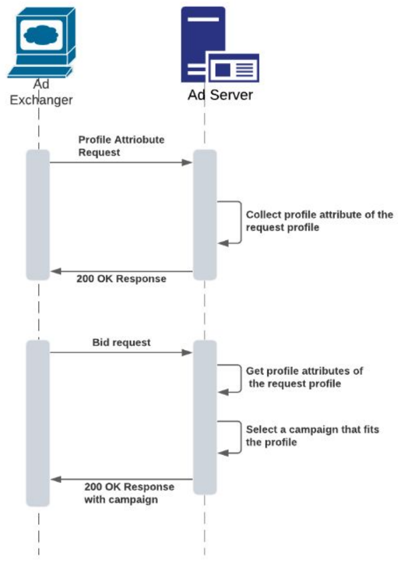
- **Basic Spring Boot servlet**
- **H2 DB connector**
- **H2 DB tables connectors**
    - **campaigns_conf (4.a)**
    - **campaigns (4.b)**
    - **profiles (4.c)**
- **H2 DB tables data initialized (campaigns, campaigns_conf)**

Unimplemented parts:

- **Support profile attribute request (2.a)**
- **Support bid request (2.b)**

Guideline:
In order to complete the project you are eligible to create new classes, data structures, and any Object oriented methodology to achieve your goal, you may also add additional DB tables if needed. You should take in consideration that the Adserver is a real time system, and is required to have the lowest possible response latency. Therefore, try to use the best **search algorithms/data structures**.
Keep in mind the Adserver serves a high volume of parallel incoming requests. This means multiple bid requests for the same profile is possible.

Background:
The Adserver goal is to collect surf history data which is called “profile attributes” using those profile attributes the Adserver searches for a matching campaign. A bid request is a request for getting web advertisement for some user based on its surf history. The Adserver collects surf history of users using rest API which will be explained later as well as the logic of how it matches a campaign to some user bid request, see the following diagram to better understand the process.


Ad Exchanger
Ad Server
Profile Attriobute Request
Collect profile attribute of the request profile
200 OK Response
Bid request
Get profile attributes of the request profile
Select a campaign that fits the profile
200 OK Response with campaign

1. **Overview**
   In the following assignment you are given an up-and-running Spring boot Java application that acts as a HTTP server. The server is called “RTB_Engine”.
   The “RTB_Engine” supports the following HTTP GET requests:
   a. Profile Attribute Request
   b. Bid Request

   And the following URL parameters:

   a. Action type = “act”
   b. Attribute ID = “atid”
   c. Profile ID = “pid”

2. **Server API**
   The server distinguish between the different HTTP requests using the URL parameter “act”:

   act=0 Profile Attribute Request
   act=1 Bid Request

   a. **Profile Attribute Request:**
      i. **Description:** It is used for collecting profile attributes into DB.
      ii. **URL Parameters:**
         1. “act=0”
         2. “pid”
         3. “atid”
      iii. **Expected response:** 200 OK
      iv. **Response body:** empty
      v. **Request URL example:** localhost:8080/api?act=0&pid=3&atid=456
   b. **Bid Request:**
      i. **Description:** It is used for finding a campaign based on the bid profile attributes, and campaign selection logic (3).
      ii. **URL Parameters:**

            1. "act=1"
            2. "pid"
        iii. **Expected response:** 200 OK
        iv. **Response body:**
        One of the following options:
            1. <campaign id> - the matching campaign ID
            2. unmatched – in case there is no matching campaign.
        v. **Request URL example:** localhost:8080/api?act=1&pid=3

3. **Bid request campaign selection logic:**
    a. **Campaign selection:**
    Each campaign has its own targeting attributes, the targeting attributes is a set of attributes which the bid profile must have in order to be eligible for this campaign.
    For example: if campaign #5 target attribute IDs are #1,#7,#9, and a bid request profile has attributes #1, #7, #9, #14, than campaign #5 fits this profile.
    But, if bid profile has attributes #1, #7, #65, #123, than campaign #5 does not fit for this profile (because it is missing attribute #9).
    Only profiles that have **ALL** the required campaign attributes, fit the campaign.
    b. **Campaign priority:** In case there are multiple campaigns that fit a profile, "RTB_Engine" selects the campaign with the highest priority.
    In case all the candidate campaigns have the same priority, the campaign with lowest ID is selected.

4. **Application DB tables:**
The "RTB_Engine" uses a H2 (in-memory) SQL DB.
It has the following tables:
    a. Table name: "campaign_config"
    Table columns:
        i. INTEGER campaign_id – KEY
        ii. INTEGER capacity
        iii. INTEGER priority
    b. Table name: "campaigns"
    Table columns:
        i. INTEGER campaign_id – KEY
        ii. INTEGER attribute_id - KEY
    c. Table name: "profiles"
    Table columns:
        i. INTEGER profile_id – KEY
        ii. INTEGER attribute_id - KEY

Part B:

A certain profile might have multiple campaigns that fit. The Adserver would like to publish a variety of campaigns to a profile, and wouldn't like to "blast" a profile with the same campaign over and over.
Therefore, every campaign has a capacity limit per profile:
If a campaign #4 has capacity set to 3, and profile #5 had 3 bids where campaign #4 was selected, then the profile #5 has reached the limit for campaign #4, and on the next bids on profile #5 campaign #4 will not be selected, even if it fits and has the highest priority out of all the campaigns that fit this profile.
Notice that campaign #4 can be selected for other profiles that did not exceed the cap.

1. **Server API modification:**

   Bid request Response body:
   One of the following options:

   1. <campaign id> - the matching campaign ID
   2. unmatched – in case there is no matching campaign.
   3. capped – in case all matching campaigns have reached their capacity.

2. **Scalability requirements:**

   1. The Adserver receives an average of 30,000 Bid requests per second.
   2. Ad bidding for a specific profile is sparse in time (a user is not online all the time), but once a profile becomes online (i.e. surf websites that show ads), the Adserver receives numerous bid requests on the profile in a short period of time, some may arrive simultaneously.

3. **Objective:**

   Support the campaign to profile frequency cap requirements.