Introduction

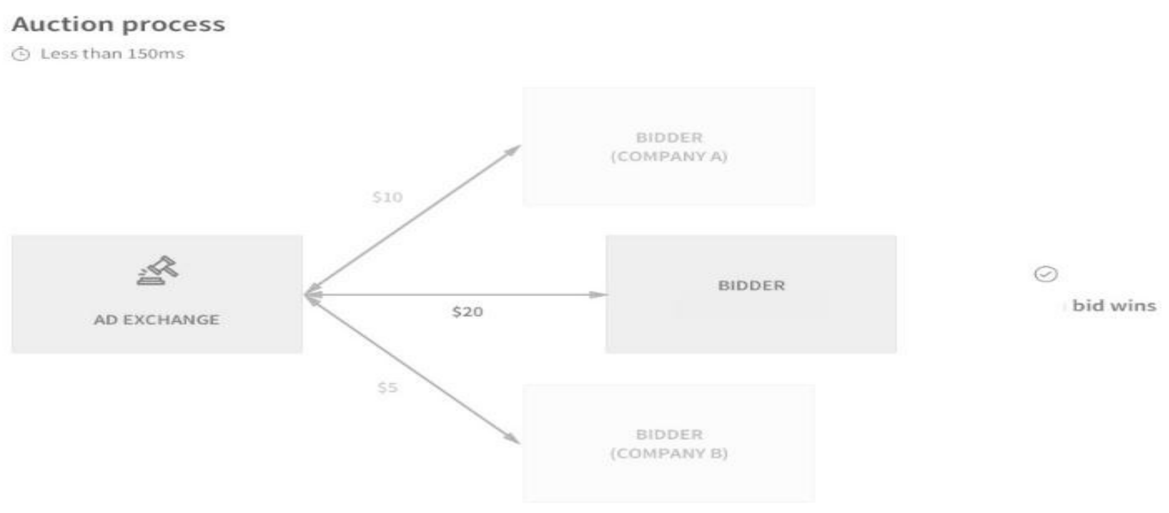
- Please read the instructions carefully before starting.
- **Please make sure you read the deliverables section before submitting your assignment.**
- The instructions below include the expected time of finishing this assignment. If you spend significantly more time than expected then please check if your implementation does more things than the assignment asks for.
- Please keep your solution as simple as possible! We are impressed by simple and elegant solutions.
- If something is not clear to you please do not hesitate to get in touch with us for clarifications. We'd love to help you out!

Objective

You are a software engineer for an advertising technology company called Project Agora. The company wants to enter the world of real-time bidding (RTB) and you are asked to implement a real-time bidder for mobile advertising campaigns.

What is a real-time bidder?

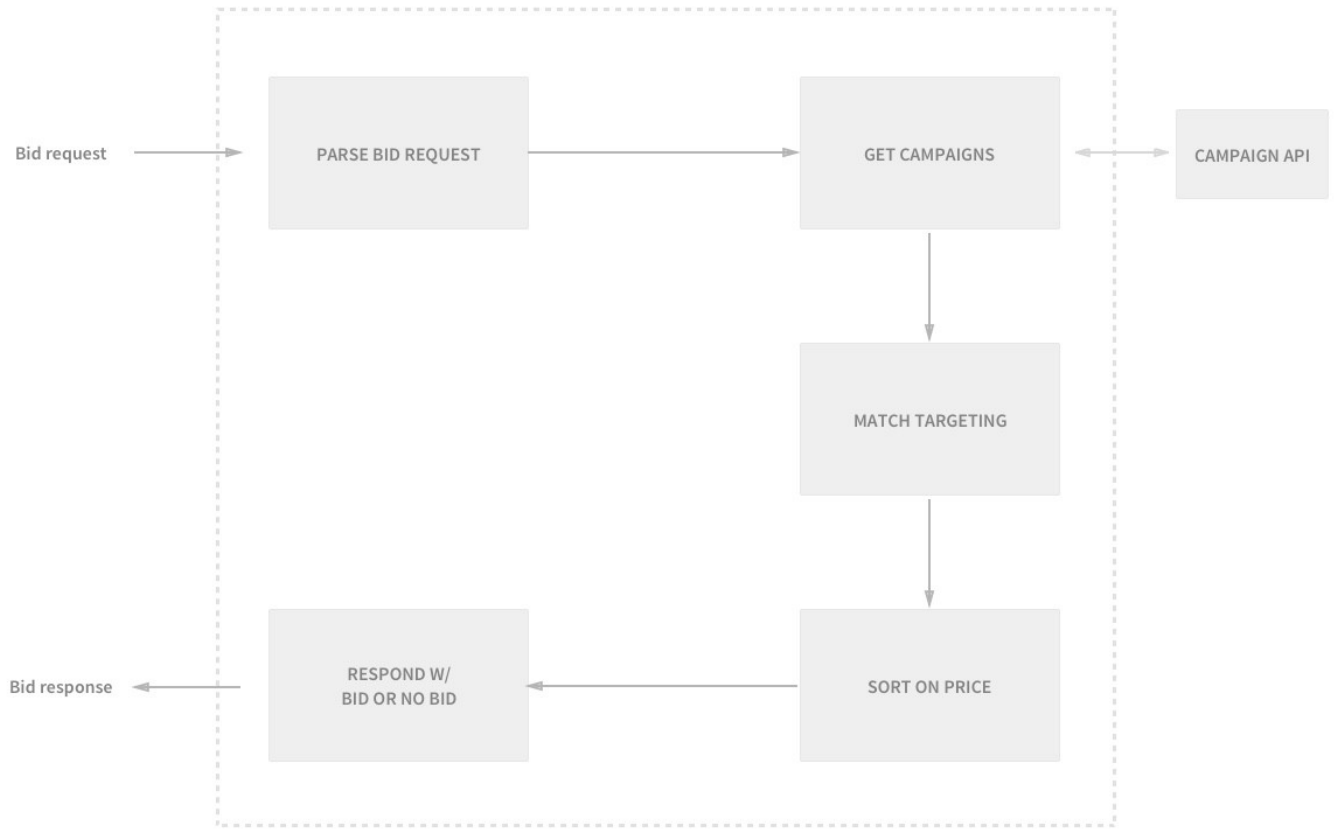
A bidder is simply a platform which allows advertisers to submit bids to buy mobile ad space in real-time. A bidder receives bid requests from 3rd-party ad exchanges and responds back with a bid response. This bid then competes with bids from other bidders in a real-time auction at the exchange. The highest bid (in terms of bid price) wins and gets to show its ad. You can see this process illustrated below.



For this assignment we will focus on building a bidder so you do not need to worry about what happens at the ad exchange. If we zoom into what a bidder does then the typical flow is outlined below:

- Bidder receives and parses a bid request from an ad exchange.
- Bidder retrieves all the available campaigns from its campaign pool.
- Bidder filters out campaigns that do not match the targeting criteria.
- If there are matching campaigns then the bidder finds the highest paying campaign and submits a bid for that campaign to the ad exchange.
- Otherwise the bidder submits an empty response with no bid.

Bidder process



Task

Objective

You will need to implement the basic flow of a bidder which was described above.

You are given the following

1. The doc for the bidder API (**Bidder_API.pdf**) which specifies the protocol between our bidder and the ad exchange.
2. The doc for the campaign API (**Campaign_API.pdf**) which can be used to retrieve the available campaigns.
3. For the targeting matching you should use the country info contained in the bid request and check it against the list of *targetedCountries* for each campaign.
4. For the campaign ranking you should use the *price* info of the campaign.

Mocking the Campaign API

Your bidder implementation depends on an external service which exposes the **Campaign API**. There is no instance of this service running, therefore in your tests, we expect to see this dependency mocked to simulate the real production functionality.

You are free to use any approach you like to implement mocking.

End-to-end test cases

In order for the task to be completed your solution should implement automated tests for the following test cases. You are free to use any testing library or framework you feel most comfortable in.

Description	Expected input	Expected output	Parameters/Mocks
Bidder should respond with a bid for the highest paying campaign that matches the targeting criteria.	Use this for the expected input(**test-case-1-input.json**)	Use this for the expected output(**test_case_1_expected_output.json**)	Use this response for the Campaign API mock. (**test_case_1_campaignb_api_mock.json**)
Bidder should respond without a bid if there no available or matching campaigns.	Use this for the expected input(**test-case-2-input.json**)	Bid response with no body and status code 204 as specified in Bidder_API.docx.	Use this response for the Campaign API mock. (**test_case_2_campaignb_api_mock.json**)

Deliverables

In your uploaded solution you should include:

1. The complete working code (so that we can run your solution on our machines).
2. Step-by-step instructions on how to run your code and tests (how to compile, how to install dependencies, how to run code etc).
3. Any code that is needed to run your passing end-to-end test cases listed in the assignment.

Notes

- If your code depends on external libraries and frameworks please do not include the source code for those in your submission. Please document the steps to install all the dependencies.
- Send your solution as a zip file.
- Please keep your solution as simple as possible.

Assessment criteria

Below you can find the list of assesment criteria we use to judge your code along with their importance.

Dimension	Description	Importance
Code architecture	Is separation of concerns clear in your code? Is it split into modules/components or is it a big monolith?	**HIGH**
Simplicity	Does your code fulfils the specifications in the simplest way possible? Did you use complicated code structure or complex data structures that are an overkill for our problem?	**HIGH**
Reusability	Do you follow the DRY or the WET principle? Did you break your code down into reusable services/components?	**HIGH**
Extensibility	Is your code structured in a way that future changes would require minimal effort?	**HIGH**
Solution Completness	Did you complete the implementation of the solution according to the specifications?	MEDIUM
Test coverage	Did you covered most of your code's functionality with tests?	MEDIUM
Performance	Did you make an effort to optimize the performance of your code?	LOW