**End to End Process:**

A Client will make request to the Internal component using below API.

**Component:**

● API: It will connect an external client to the stock exchange via the command line.

API should consist of following parameters:

**localhost.com/countrycode=US&Category=Automobile&BaseBid=10**

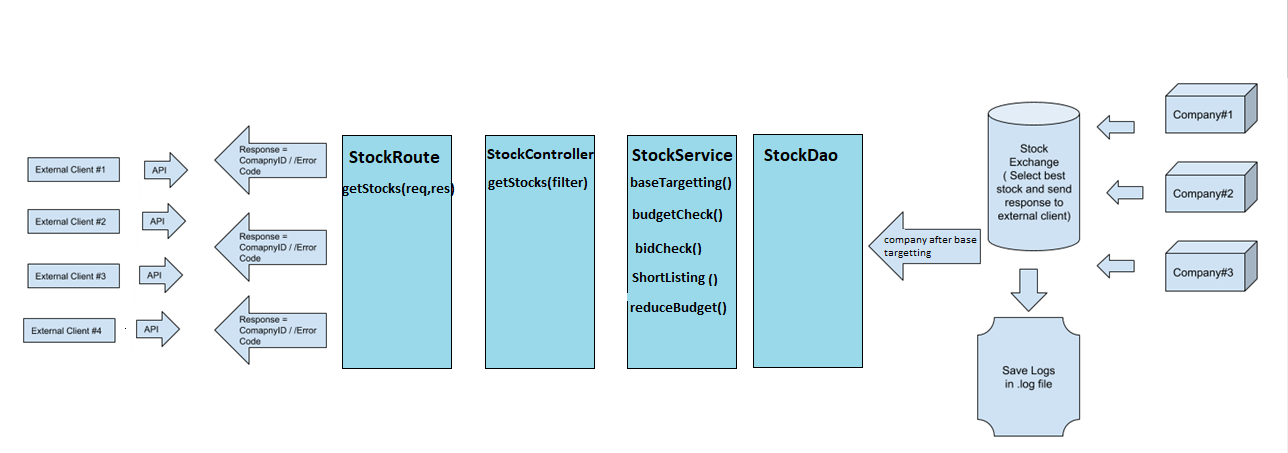
**where:**

**Country code: US, IN, FR**

**Category: Automobile, Finance, IT**

**Base Bid: External client needs stock more than this bid ex. 10 cent API Example**

**Process Flow:**



**API Internal component is divided in to various parts basically the n-tier architecture:**

* **StockRoute:** The Routing component decides which controller will be invoked on the basis of url path defined inside the component.
* **StockController:** The Controller class has responsibility to fetch all stocks on the basis of different conditions and service call.
* **StockService:** The Service Class contains all the required business logic(Exchange Internal Matching Logic) which calls data access layer and process on the basis of fetched data. It has several methods:

1. **baseTargeting():**

Match companies based on Country, Category. If no company passed from these filters then send response as “No Companies Passed from Targeting” to external client. Include logs like: BaseTargeting: {C1, Passed},{C2,Failed},{C1,Passed}

1. **budgetCheck():**

Check if Companies had some budget to sell stocks. If no companies passed from the filters then send response as “No Companies Passed from Budget” to external client. Include logs like: BudgetCheck: {C1, Passed},{C2,Failed},{C1,Passed}

1. **bidCheck():**

Check if the bid is more than the API base bid. If no companies passed from the filter then send response as “No Companies Passed from BaseBid check” to external client. Include in logs like: BaseBid: {C1, Passed},{C2,Failed},{C1,Passed}

1. **shortlisting():**

If more than one company passed from BaseBid check then select the highest one and send response = CompanyID. Include in logs like: Winner = CompanyID.

1. **reduceBudget():**

Once CompanyID is sent to external client then update Budget New = Budget ­ Bid and use Budget New for future transactions.

* **StockDao:** The Data Access Class has responsibility to perform several DB operations to the stock table. It has two methods:

1. **getAllStocks:** it get all the stock from the Database table stock.
2. **updateStock:** it update the stock.

* **End-to-End Process:**

When client send request using api with required search criteria. StockRouter process the request and call getStocks method with request and response parameters. Since it is a multithread application user can send parallel request for processing. StockRouter call the validation component StockValidator to validate the request is proper or not. If anything wrong with request parameters it will send Error response to the client in simple text otherwise it will call controller for further processing. StockController have responsibility to process the required condition with the help of service classes before sending the response to the StockRouter and it will send to the client. StockController will call the StockService methods to perform the required business logic defined for Internal Exchange Components which in turns call the StockDao method calls to access and update the required Stock data. Once the data will be accessed or updated by StockDao, it will send results to Stockservice and it will process the business logic on those data and sends the required output to the StockController. StockController will perform all the logics required and sends back the result to the StockRoute. StockRoute will send the Winner Company to the client.

* **Logs File:** Log files are getting generated under log folder inside the application main folder and Log manager is configured inside middleware folder.
* **Unit Test:**  Unit test are under test/unittest folder can be run using **npm test** inside the stockexchange directory from the command line.
* **Integration Test:** Integration tests are under test/integrationtest folder can be run using **npm test** inside the stockexchange directory from the command line.
* **Technology Used:**

Language:  Node.js

Database: Postgresql