# Auction System

## SETUP

**Docker**: The system uses docker environment for deployment purpose. Containerizing applications gives huge amount flexibility without worrying too much about the environment.

**Travis CI**: The code when pushed is almost instantly built by Travis CI and is pushed to docker hub container registry (Ready to be deployed). The application builds in not more than 2 to 3 mins right from the moment the code is pushed

**Java + Play Framework**: The services are written using Play framework large due to the familiarity which was important to ensure timely delivery of the required application.

**Testing Setup**: *JUnit, Mockito and TestContainers frameworks* were used for testing the pipeline. SBT Plugin Jacoco was used to evaluate the code coverage of **auction-web** web service. Can be checked using command **sbt jacoco** from within the **auction-web** project

**Nginx**: Very easy to use (L7 Application Layer) based load balance + reverse proxy. Helps the platform in scaling out.

**MongoDB**: One of the most popular NoSQL databases. Flexible indexing constructions, ability to support sharding, replication and integrations with popular tools like Kafka, Spark; mongo is the go-to choice to create a flexible yet scalable platform.

## RUN

Ensure docker is installed on the system

1. Checkout the code
2. Run command: docker-compose up –d
3. The services are hosted at http://localhost:8080/

**Refer to Swagger UI**

http://localhost:8088/

## MAJOR APIs

### 1. Auctions List

Description: Lists all the auctions in the system in paginated way

Request

```
curl --location --request GET 'localhost:8080/auction?count=3&status=RUNNING'
```

Response

```
{"auctions":[{"itemCode":"item1","stepRate":5,"winningPrice":100}]}
```

### 2. Create a user

Description: Creates a user in the system

Request

```
curl --location --request POST 'localhost:8080/user' \
--header 'Content-Type: application/json' \
--header 'Host: localhost' \
--data-raw '{
  "name" : "Ankit Jain",
  "email" : "ankit@google.com",
  "passwd" : "password"
}'
```

Response

```
{"userId":"87eae1fe-517a-41af-be7b-cf1f35e25928","name":"Ankit Jain"}
```

## 3. Get User Info

Description: Gets only the very basic information of the user

Request

```
curl --location --request GET 'localhost:8080/user/87eae1fe-517a-41af-be7b-cf1f35e25928'
```

Response

```
{"userId":"87eae1fe-517a-41af-be7b-cf1f35e25928","name":"Ankit Jain"}
```

## 4. Create an auction

Description: Creates an auction on which the bid can be placed

Request

```
curl --location --request POST 'localhost:8080/auction' \
--header 'Content-Type: application/json' \
--data-raw '{
   "itemCode" : "item1",
   "basePrice" : 30,
   "stepRate" : 5
}
```

Response

```
{
   "itemCode": "item1",
   "winningPrice": 30,
   "stepRate": 5
}
```

## 5. Generate token for bid participation

Description: Helps a user in generating token which can be used for bidding

Request

```
curl --location --request POST 'localhost:8080/user/authenticate' \
--header 'Content-Type: application/json' \
--data-raw '{
  "email" : "ankit@google.com",
  "passwd" : "password"
}'
```

Response

```
{"expiry":1604121954432,"userId":"87eae1fe-517a-41af-be7b-
cf1f35e25928","token":"c07d5777-b7b7-4f94-ac43-1841eaeef9fe"}
```

## The Bidding

Description: User can place bids against auctions using token

Request

```
curl --location --request POST 'localhost:8080/auction/item1/bid' \
--header 'token: c07d5777-b7b7-4f94-ac43-1841eaeef9fe' \
--header 'userId: 87eae1fe-517a-41af-be7b-cf1f35e25928' \
--header 'Content-Type: application/json' \
--data-raw '{
  "bidPrice" : 110
}'
```

Response

```
Valid Bid
```