This documentation mainly deals with the JAVA API’s and will help you provide a detailed step-by-step approach to run the Online Car Auction System.

Code base has been committed in the given GIT URL:

<https://github.com/saisashank/software-developer-coding-challenge.git>

Prior to starting the Services, Spring Config Server designed specifically for the application needs to be started. It will be running on the port 8888. This is mainly useful for the obtaining the Database connectivity.

After the Config Server has started running on the port 8888, we can pull in the Services code and start the Services as well.

Database used is an in-memory one (h2 Database). As the database schema is present within the Services code, when the application is up and running we can see the database schema created. We can find the Database schema at <http://localhost:8080/console>

Swagger implementation has been placed at: <http://localhost:8080/swagger-ui.html>

NOTE: If any of the application has been stopped or terminated by any chance complete data created prior to this will be lost and need to start from first.

There are 11 API’s listed below:

* biddingService/saveUserBid -> Creating or Updating a User Bid on a Car
* biddingService /getCarBidHistory -> Getting the Car Bid history based on car model
* biddingService /getWinningBidForCar -> Getting the current winning bid of a car
* biddingService /getAvailableCarForBid -> List of Available or unavailable car’s for bid
* biddingService /changeStatusOfCar -> change availability status of car
* updateCarDetails -> Update car Info (Odometer , Base price) or delete the record from table
* createCarForBid -> Creating a new Car for bid
* createUser -> Creating a new User
* updateUserDetails -> Updating user details (First name , last name, Phone number)
* udpateUserActiveStatus -> Update/delete User deatils
* getAllActiveUsers -> All Active or Inactive users

Before trying to create a bid for a car, user needs to make sure that he/she has created account for themselves within the System.

* For creating a user we can make use of this: localhost:8080/createUser (POST)

NOTE: When the user is created default active status is “Y”

Request Body:

{

"userId":"sashank",

"firstName":"Sai sashank",

"lastName":"Ravi",

"emailAddress":"sai@gmail.com",

"phoneNumber":"234-576-8987"

}

Response:

{

"response": "successfully created User"

}

If the user tries to create with the same Email address or same user id, account will not be created. User ID and Email Address needs to be unique for a given user.

Request:

{

"userId":"sai",

"firstName":"Sai sashank",

"lastName":"Ravi",

"emailAddress":"sai@gmail.com",

"phoneNumber":"234-576-8987"

}

Response:

{

"response": "Please check the UserId/Email given.... they already exist..."

}

If trying to recreate with the same user ID and email address as existing you will get the below:

Request:

{

"userId":"sashank",

"firstName":"Sai sashank",

"lastName":"Ravi",

"emailAddress":"sai@gmail.com",

"phoneNumber":"234-576-8987"

}

Response:

{

"response": "User trying to create already exists..."

}

* For updating a user we use: localhost:8080/updateUserDetails (PUT)

Request:

{

"userId":"sashank",

"firstName":"Sai",

"lastName":"Ravi",

"emailAddress":"sai@gmail.com",

"phoneNumber":"234"

}

Repsonse:

{

"response": "successful update of user details"

}

When trying to update the User which has not been created we get the following as response:

Request:

{

"userId":"sai",

"firstName":"Sai",

"lastName":"Ravi",

"emailAddress":"sai@gmail.com",

"phoneNumber":"234"

}

Response:

{

"Exception occured is: ": null

}

* For getting all the active user’s we use this: localhost:8080/getAllActiveUsers (POST)

To get Active users:

Request:

{

"userId":"",

"firstName":" ",

"lastName":"",

"emailAddress":"",

"phoneNumber":"",

"isActive":"Y"

}

Response:

{

"response": [

{

"userDetailsId": 1,

"userCarBidList": [],

"userId": "sashank",

"firstName": "Sai",

"lastName": "Ravi",

"emailAddress": "sai@gmail.com",

"phoneNumber": "234",

"isActive": "Y"

},

{

"userDetailsId": 2,

"userCarBidList": [],

"userId": "sai",

"firstName": "Sai sashank",

"lastName": "Ravi",

"emailAddress": "shashi@gmail.com",

"phoneNumber": "234-576-8987",

"isActive": "Y"

}

],

"header": "List of User who are Active..."

}

To get Inactive Users:

Request:

{

"userId":"",

"firstName":" ",

"lastName":"",

"emailAddress":"",

"phoneNumber":"",

"isActive":"N"

}

Response:

{

"response": [

{

"userDetailsId": 2,

"userCarBidList": [],

"userId": "sai",

"firstName": "Sai sashank",

"lastName": "Ravi",

"emailAddress": "shashi@gmail.com",

"phoneNumber": "234-576-8987",

"isActive": "N"

}

],

"header": "List of User who are Inactive..."

}

* For updating the status of the user whether he/she is Active currently we use this: localhost:8080/updateUserActiveStatus (PUT)

If we want to keep the User in the database, Active Status will be changed from “Y” to “N”.

Request:

{

"userId":"sai",

"firstName":"Sai sashank",

"lastName":"Ravi",

"emailAddress":"shashi@gmail.com",

"phoneNumber":"234-576-8987",

"userStatus":"Y"

}

Response:

{

"response": "updated the Active Status of user..."

}

If we want to delete the user from the DB then:

Request:

{

"userId":"sai",

"firstName":"Sai sashank",

"lastName":"Ravi",

"emailAddress":"shashi@gmail.com",

"phoneNumber":"234-576-8987",

"userStatus":"N"

}

Response:

{

"response": "User deleted from the DB permanently..."

}

If trying to update the status of user who is not present

Request:

{

"userId":"Sashank",

"firstName":"Sai sashank",

"lastName":"Ravi",

"emailAddress":"shashi@gmail.com",

"phoneNumber":"234-576-8987",

"userStatus":"N"

}

Response:

{

"response": "User intended to delete/update is not present..."

}

* For creating a car for the bid we use this: localhost:8080/createCarForBid (Post)

NOTE: Car Availability depends during the creation (Y or N)

Request:

{

"carCompany":"chevy",

"carModel":"2019 chvy",

"odometerReading":"20k",

"basePrice":"13420",

"carAvailability":"N"

}

Response:

{

"response": "successfully inserted new Car for Bid..."

}

If trying to recreate the same car then we get the below:

Request:

{

"carCompany":"chevy",

"carModel":"2019 chvy",

"odometerReading":"20k",

"basePrice":"13420",

"carAvailability":"N"

}

Response:

{

"response": "Requested car is already in the Auction... check for its availability status...",

"header": {

"carDetailsId": 6,

"userCarBidList": [],

"carCompany": "chevy",

"carModel": "2019 chvy",

"carAvailability": "N",

"basePrice": 13420,

"odometerReading": "20k"

}

}

* For updating the car details we use this: localhost:8080/updateCarDetails (PUT)

If we want to update the Odometer reading or Base Price of a car we use car Status as “Y”

Request:

{

"carCompany":"chevy",

"carModel":"2019 chvy",

"odometerReading":"21k",

"carStatus":"Y"

}

Response:

{

"response": "Successful Updation of Car's Details"

}

If we want to delete the Car from the database we use car status as “N”.

Request:

{

"carCompany":"chevy",

"carModel":"2019 chvy",

"odometerReading":"21k",

"carStatus":"N"

}

Response:

{

"response": "Car Details deleted from the DB permanently..."

}

When you try to update/delete a Car which is not created we get as follows:

Request:

{

"carCompany":"chevy",

"carModel":"2019 chvy",

"odometerReading":"21k",

"carStatus":"Y"

}

Response:

{

"response": "Cannot find the car trying to update..."

}

* For Changing the Availability Status of a car we use this: localhost:8080/biddingService/changeStatusOfCar (PUT)

Request:

{

"userId":"",

"bidAmount":"",

"car":"Benz",

"carModel":"2019 Mercedes-Benz C-Class",

"createNewOne":false,

"carAvailability":"N"

}

Response:

{

"Status": "Successful Updation of Car's Status",

"header": "Successful Updated Status of a Car..."

}

When we try to change the status of a car which has not been created we get

Request:

{

"userId":"",

"bidAmount":"",

"car":"Benz",

"carModel":"2019 Mercedes",

"createNewOne":false,

"carAvailability":"Y"

}

Response:

{

"header": "Intended car is not available"

}

* For getting the list of cars which are either Available or Unavailable for Bid we use this:

localhost:8080/biddingService/getAvailableCarForBid (POST)

If we want the list of car’s which are available then car availability should be “Y”

Request:

{

"userId":"",

"bidAmount":"",

"car":"",

"createNewOne":false,

"carAvailability":"Y"

}

Response:

{

"carBidHistoryList": [

{

"carDetailsId": 2,

"userCarBidList": [],

"carCompany": "Benz",

"carModel": "2019 Mercedes-Benz C-Class",

"carAvailability": "Y",

"basePrice": 20000,

"odometerReading": "19k"

},

{

"carDetailsId": 3,

"userCarBidList": [],

"carCompany": "Nissan",

"carModel": "2018 Nissan GT-R",

"carAvailability": "Y",

"basePrice": 15000,

"odometerReading": "35k"

},

{

"carDetailsId": 4,

"userCarBidList": [],

"carCompany": "Honda",

"carModel": "2018 Honda CR-V",

"carAvailability": "Y",

"basePrice": 14000,

"odometerReading": "45k"

},

{

"carDetailsId": 5,

"userCarBidList": [],

"carCompany": "Hyundai",

"carModel": "2019 Hyundai Santa Fe",

"carAvailability": "Y",

"basePrice": 12000,

"odometerReading": "78k"

}

],

"header": "Available car's for Bidding..."

}

If we want the list of car’s unavailable then we pass car availability as “N”

Request:

{

"userId":"",

"bidAmount":"",

"car":"",

"createNewOne":false,

"carAvailability":"N"

}

Response:

{

"carBidHistoryList": [

{

"carDetailsId": 1,

"userCarBidList": [],

"carCompany": "Toyota",

"carModel": "2019 Toyota C-HR",

"carAvailability": "N",

"basePrice": 10000,

"odometerReading": "57k"

}

],

"header": "Unavailable car's for Bidding..."

}

If we send the car availability as “ ”

Request:

{

"userId":"",

"bidAmount":"",

"car":"",

"createNewOne":false,

"carAvailability":""

}

Response:

{

"carBidHistoryList": [],

"header": "Sorry, currently no car's are available for bidding..."

}

* For creating a bid on the car we use this:

localhost:8080/biddingService/saveUserBid (PUT)

NOTE: For creating new bid’s on a car createNewOne should be true. Will be false when updating the existing bid. And also when the bid is created current status will be IN-PROGRESS, until the winning bid is reached where it changes to BID CLOSED.

If we want to create a new car bid on a car and if user has not created an account

Request:

{

"userId":"sashank",

"bidAmount":"10",

"car":"Nissan",

"carModel":"2018 Nissan GT-R",

"createNewOne":true,

"emailAddress":"sai@gmail.com",

"phoneNumber":"3627-383-4592"

}

Response:

{

"header": "Sorry, Requested Car/ User is not available for Auction...",

"userCarBidVO": null

}

Once the account has been created and when tried to bid the car below the base price

Request:

{

"userId":"Sashank",

"bidAmount":"10",

"car":"Nissan",

"carModel":"2018 Nissan GT-R",

"createNewOne":true,

"emailAddress":"sai@gmail.com",

"phoneNumber":"3627-383-4592"

}

Response:

{

"header": "Sorry, Bidding Price is less than the base price set...",

"userCarBidVO": {

"userId": "Sashank",

"car": "Nissan",

"carModel": "2018 Nissan GT-R",

"emailAddress": "sai@gmail.com",

"phoneNumber": "3627-383-4592",

"bidAmount": 10,

"createNewOne": true,

"carAvailability": null

}

}

When the bid is more than the base price then we get as such

Request:

{

"userId":"Sashank",

"bidAmount":"15001",

"car":"Nissan",

"carModel":"2018 Nissan GT-R",

"createNewOne":true,

"emailAddress":"sai@gmail.com",

"phoneNumber":"3627-383-4592"

}

Response;

{

"header": "successful insertion"

}

Once user creates a bid and tries to update with the same amount and also same will be the case when different user tries to create with the same bid then

Request:

{

"userId":"Sashank",

"bidAmount":"15001",

"car":"Nissan",

"carModel":"2018 Nissan GT-R",

"createNewOne":false,

"emailAddress":"sai@gmail.com",

"phoneNumber":"3627-383-4592"

}

Response:

{

"header": "unsuccessful insertion/updation as the bid is less than or equal to the highest bid"

}

When user updates the bid amount higher than the current amount

Request:

{

"userId":"sai",

"bidAmount":"15005",

"car":"Nissan",

"carModel":"2018 Nissan GT-R",

"createNewOne":false,

"emailAddress":"shashi@gmail.com",

"phoneNumber":"3627-383-4592"

}

Response:

{

"header": "successful updation"

}

* For getting the car bidding history we use this: localhost:8080/biddingService/getCarBidHistory (POST)

When NO Bid has been initiated on a car

Request:

{

"userId":"",

"bidAmount":"",

"car":"Benz",

"carModel":"2019 Mercedes-Benz C-Class",

"createNewOne":false

}

Response:

{

"carBidHistoryList": [],

"header": "No Bidding history registered on this car..."

}

When a car has a bidding history

Request:

{

"userId":"",

"bidAmount":"",

"car":"Nissan",

"carModel":"2018 Nissan GT-R",

"createNewOne":false

}

Response:

{

"carBidHistoryList": [

{

"userCarBidId": 2,

"userId": "sashank",

"emailAddress": "shashi@gmail.com",

"phoneNumber": "3627-383-4592",

"carName": "Nissan",

"carModel": "2018 Nissan GT-R",

"bidAmount": 15005,

"auctionStatus": "IN-PROGRESS"

},

{

"userCarBidId": 1,

"userId": "sai",

"emailAddress": "sashank@gmail.com",

"phoneNumber": "3627-383-4592",

"carName": "Nissan",

"carModel": "2018 Nissan GT-R",

"bidAmount": 15004,

"auctionStatus": "IN-PROGRESS"

}

],

"header": "Car's Bidding History"

}

* For getting winning bid of the current car we use this:

localhost:8080/biddingService/getWinningBidForCar (POST)

Request:

{

"userId":"",

"bidAmount":"",

"car":"Nissan",

"carModel":"2018 Nissan GT-R",

"createNewOne":false,

"carAvailability":"N"

}

Response:

{

"header": "Winning Bid for a car",

"userBidDetails": {

"userCarBidId": 2,

"userId": "sashank",

"emailAddress": "shashi@gmail.com",

"phoneNumber": "3627-383-4592",

"carName": "Nissan",

"carModel": "2018 Nissan GT-R",

"bidAmount": 15005,

"auctionStatus": "IN-PROGRESS"

}

}

When there is no bid initiated on the car or car not in the auction or database is given

Request:

{

"userId":"",

"bidAmount":"",

"car":"Benz",

"carModel":"2019 Mercedes-Benz C-Class",

"createNewOne":false,

"carAvailability":"N"

}

Response:

{

"header": "No Bid has been made on the car...",

"userBidDetails": null

}