Test Cases

GET METHOD :

CASE 1 : To get All the Stock Items stored in the DB

1) Sample Pathname : localhost:8080/stocks (Valid path)

Expected output : we will get all the stocks that were stored in the DB in the form of JSON


```
[
 {
    "categoryId": 101,
    "price": 10000.0,
    "stockName": "Tata steels"
  },
  {
    "categoryId": 103,
    "price": 10000.0,
    "stockName": "brila steels"
  }
]
```


2) Sample Pathname : localhost:8080/stock (invalid path)

Expected output : It should return JSON which determines the exception.


```
  {
        "timestamp": "2021-05-29T18:22:50.110+00:00",
        "status": 404,
        "error": "Not Found",
        "path": "/stock"
}
```

CASE 2 : To get the required stock item if it is stored in the DB

Sample Pathname : localhost:8080/stocks/101

Expected output : we will get all the stocks that were stored in the DB in the form of JSON


```
[
 {
     "categoryId": 101,
     "price": 10000.0,
     "stockName": "Tata steels"
  },
]
```


CASE 3 : To get the required stock item if it is stored in the DB by validating the user

1) Sample Pathname : localhost:8080/categoryid/101/login/anil

Expected output : we will get all the stocks that were stored in the DB in the form of JSON


```
[
 {
     "categoryId": 101,
     "price": 10000.0,
     "stockName": "Tata steels"
  },
]
```


2) Sample Pathname : localhost:8080/categoryid/101/login/kumar (invalid userid)

Expected output : Blank screen

3) Sample Pathname : localhost:8080/categoryid/201/login/kumar (invalid categoryid)

Expected output : Blank screen

4) Sample Pathname : localhost:8080/categoryd/101/login/kumar (Invalid Path)

Expected output : Json which defines the Exception


```
{
  "timestamp": "2021-05-29T18:18:34.379+00:00",
  "status": 404,
  "error": "Not Found",
  "path": "/categoryd/101/login/kumar" }
```

POST METHOD:

Case 1: To insert the data into table

Sample path : localhost:8080/stock

User input : Json which include the required data to insert into the table


```
{
      "categoryId": 107,
      "price": 1000000.0,
      "stockName": "Amazon"
   }
```


Expected output : Data to be inserted into the table and return the entry that got inserted


```
{
      "categoryId": 107,
      "price": 1000000.0,
      "stockName": "Amazon"
 }
```


Case 2 : To insert data without primary key

Sample path : localhost:8080/stocks

User input : : Json which include the required data to insert into the table other than primary key


```
{
      "categoryId": ,
      "price": 1000000.0,
      "stockName": "Amazon"
}
```


Expected output : We can insert the data without primary key


```
{
   "timestamp": "2021-05-29T18:27:44.798+00:00",
   "status": 400,
   "error": "Bad Request",
   "path": "/stocks"
}
```

PUT METHOD:

Case 1: To update the data in the table

Sample path : localhost:8080/stock

User input : Json which include the required data to update an entry in the table


```
{
    "categoryId": 107,
    "price": 1000000.0,
    "stockName": "Amazon Prime"
  }
```


Expected output : Data to be updated into the table and return the status


```
{
    "categoryId": 107,
    "price": 1000000.0,
    "stockName": "Amazon Prime"
}
```


Case 2 : To update the data in table without primary key

Sample path : localhost:8080/stocks

User input : : Json which include the required data to insert into the table other than primary key


```
{
    "categoryId": ,
    "price": 1000000.0,
    "stockName": "Amazon"
}
```


Expected output : We can update the data without primary key


```
{
  "timestamp": "2021-05-29T18:27:44.798+00:00",
  "status": 400,
  "error": "Bad Request",
  "path": "/stocks"
}
```