Back End Design for Online vendor:

Application Designed by: Ashish Gupta

Contact No: 9582556333

Design Document:

Technology Used:

Db: Mongo Db with Mongoose ORM

Backend Technologies: Node Js, Express Js

URL: http://localhost:3000

TrackURL: ws://localhost:40510/

Assumptions:

1. Order can be placed only if delivery boy is available
2. Delivery boy is assigned only on the basis of location presence
3. By default order will be saved as pending status and will be added to list of pending orders of delivery boy
4. Anyone can track the order on the basis of orderId
5. Order should be placed only if available in stocks
6. User must be registered to place the order
7. To register a admin a key is to be used

DB Design:

DB name: Test

4 Schema used to define DB collections:

1. deliverSchema
    - Collection: deliveryBoys

Schema Design:

```
  name: String,

password: String,

completedOrders: [{ type: mongoose.Schema.ObjectId, ref: 'order' }],

pendingOrders: [{ type: mongoose.Schema.ObjectId, ref: 'order' }],

location: String
```

2. UserSchema
    - Collection: Users

Schema Design:

```
  UserName: String,

admin: Boolean,

password: String,

Address: [],

Orders: [{ type: mongoose.Schema.ObjectId, ref: 'order' }],

Mobile: String
```

3. OrderSchema
    - Collection: Orders

Schema Design:

```
  productId: { type: mongoose.Schema.ObjectId, ref: 'Product' },

qty: Number,
deliveryAddress: String,
orderDate: Date,
deliveryDate: Date,
status: String,
purchaser: { type: mongoose.Schema.ObjectId, ref: 'User' },
deliveryBoy: { type: mongoose.Schema.ObjectId, ref: 'deliveryBoy' }
```

    -

4. ProductSchema
    - Collection: Products

Schema Design:

```
ProductName: String,

Price: Number,
Seller: String,
Quantity: Number,
Desc: String,
Rating: Number
```

[redacted]

API EndPoints:

1. **To register a product:**

   API URL: http://localhost:3000/product/registerProduct
   Type: Post
   Request Body:
```
{
        "ProductName": "candles",
          "Price": 234,
          "Seller": "GS group",
          "Quantity": 120,
          "Desc": "fantastic",
          "Rating": 5

}
```

2. **To update Product details:**

   URL: http://localhost:3000/product/updateProductDetails
   Req Type: POST
   Req body:
```
    {
    "productId": "5c1295d15df99b32504dbb8b",
    "updates": {
            "Quantity":100,
            "Desc": "best",
            "qty": 1
    }
}
```

3. **Delete a product:**

   url: http://localhost:3000/product/deleteProduct

   req Type: DELETE

   body:
```
{
        "productId": "5c129604ba0af9031c7fcbd6"
}
```

4. **Place an Order:**

   URL: http://localhost:3000/order/placeorder

   Req Type: POST

   Body:
```
{
"productId": "5c1506aab00b4e331c70fdd5",
"purchaser": "Ashish",
"qty": 1,
"deliveryAddress": "Delhi"
}
```

5. **Cancel an Order:**

URL: http://localhost:3000/order/deleteorder/5c14e742002e772670bf0be6
Req Type: DELETE

6. Register A delivery Boy:
   url: http://localhost:3000/delivery/registerdeliveryboy
   Req Type: POST
   Req Body:
   ```
       {
       "name": "Ram",
     "password": "ram",
     "location": "Pune"
       }
   ```
7. Register a User:
   URL: http://localhost:3000/user/register
   Req Type: POST
   Body:
   ```
       {
               "username": "AshishGup",
               "password": "qwerty",
               "mobile": "9876778987"
       }
   ```
8. Register a Admin:
   URL: http://localhost:3000/user/register
   Req Type: POST
   Body:
   ```
       {
               "username": "Ashish",
               "adminKey": " qwghdfxzfkgc",
               "password": "qwerty",
               "mobile": "9876778987"
       }
   ```
9. Get User Details:
   URL: http://localhost:3000/user/userDetails/:username
   Req Type: GET

10. Get Order Details:
    url: http://localhost:3000/order/orderdetails/:orderId
    http://localhost:3000/order/orderdetails/5c150750b77a971990ef199d

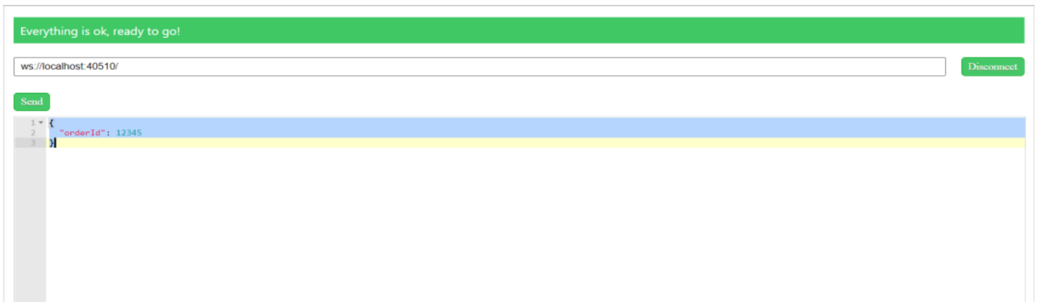
11. Track Agent live Location:

Steps:

a. Request to url: ws://localhost:40510/
b. Delivery boy and tracker will setup socketConnection using above URL
c. Tracker will register itself to the server by sending below message:

```
{
  "orderId": 12345
}
```

Request Screenshot:


Everything is ok, ready to go!

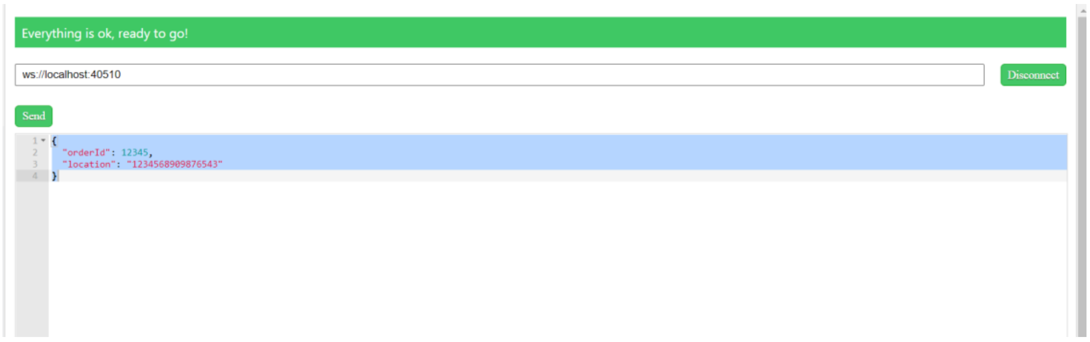
ws://localhost:40510/

```
{
  "orderId": 12345
}
```

d. Delivery Boy will send message to Server in following Format:

```
{
  "orderId": 12345,
  "location": "1234568909876543"
}
```

Request Screenshot:


Everything is ok, ready to go!

ws://localhost:40510

```
{
  "orderId": 12345,
  "location": "1234568909876543"
}
```

Result: On every change in location of the agent a message is sent to the client

Which will be transferred to Tracker.

For Reference: Postman collection is added to the GIT