```
{

   "name": "server"

   version: "1.0.0"

   "description": "Node.js app that connect the react-pos app
to the Database

   "main" : "index.js",

   "scripts": {

      test": "echo \"Error: no test specified\ specified\" &&
exit 1"

},

"author": "Your Name",

"license": "MIT"

}
```

```
Is this ok?(yes) yes


app.all("/*", function(req, res, next) {
 // CORS headers
 res.header("Access-Control-Allow-Origin", "*"); // restrict
it to the required domain
 res.header("Access-Control-Allow-Methods",
"GET,PUT,POST,DELETE,OPTIONS");
 // Set custom headers for CORS
 res.header(
   "Access-Control-Allow-Headers",
   "Content-type,Accept,X-Access-Token,X-Key"
 );
 if (req.method == "OPTIONS") {
   res.status(200).end();
 } else {
   next();
 }
});




var app = require("express")();



var server = require("http").Server(app);



var bodyParser = require("body-parser");



var Datastore = require("nedb");



var async = require("async");




app.use(bodyParser.json());
```

```
module.exports = app;

// Creates  Database

var inventoryDB = new Datastore({

 filename: "./server/databases/inventory.db",

 autoload: true

});

// GET inventory

app.get("/", function(req, res) {

 res.send("Inventory API");

});
```

```
// GET a product from inventory by _id

app.get("/product/:productId", function(req, res) {

 if (!req.params.productId) {

   res.status(500).send("ID field is required.");

 } else {

   inventoryDB.findOne({ _id: req.params.productId },
function(err, product) {

     res.send(product);

   });

 }

});
```

```
// GET all inventory products

app.get("/products", function(req, res) {

 inventoryDB.find({}, function(err, docs) {

   console.log("sending inventory products");

   res.send(docs);

 });

});

// Create inventory product

app.post("/product", function(req, res) {

 var newProduct = req.body;
 inventoryDB.insert(newProduct, function(err, product) {

   if (err) res.status(500).send(err);

   else res.send(product);

 });

});

app.delete("/product/:productId", function(req, res) {

 inventoryDB.remove({ _id: req.params.productId },
function(err, numRemoved) {

   if (err) res.status(500).send(err);

   else res.sendStatus(200);

 });
```

```
});

// Updates inventory product

app.put("/product", function(req, res) {

 var productId = req.body._id;

 inventoryDB.update({ _id: productId }, req.body, {},
function(

   err,

   numReplaced,

   product

 ) {

   if (err) res.status(500).send(err);
   else res.sendStatus(200);

 });

});

app.decrementInventory = function(products) {

 async.eachSeries(products, function(transactionProduct,
callback) {

   inventoryDB.findOne({ _id: transactionProduct._id },
function(

     err,

     product

   ) {

     // catch manually added items (don't exist in inventory)
```

```
if (!product || !product.quantity_on_hand) {

  callback();

} else {

  var updatedQuantity =

    parseInt(product.quantity_on_hand) -

    parseInt(transactionProduct.quantity);

  inventoryDB.update(

    { _id: product._id },

    { $set: { quantity_on_hand: updatedQuantity } },

    {},
         callback

        );

      }

    });

  });

};

var app   = require('express')()

var server     = require('http').Server(app)

var bodyParser = require('body-parser')

var Datastore = require('nedb')


var Inventory = require('./inventory')
```

```
})

app.use(bodyParser.json())

module.exports = app

// Create Database

var Transactions = new Datastore({

     filename: './server/databases/transactions.db',

     autoload: true

})

app.get('/', function (req, res) {

     res.send('Transactions API')

})
```

```
// GET all transactions

app.get('/all', function (req, res) {

     Transactions.find({}, function (err, docs) {

          res.send(docs)

     })

})

// GET all transactions

app.get('/limit', function (req, res) {

     var limit = parseInt(req.query.limit, 10)

     if (!limit) limit = 5
     Transactions.find({}).limit(limit).sort({ date: -1
}).exec(function (err, docs) {

       res.send(docs)

     })

})

// GET total sales for the current day

app.get('/day-total', function (req, res) {

     // if date is provided

     if (req.query.date) {

          startDate = new Date(req.query.date)

          startDate.setHours(0,0,0,0)
          endDate = new Date(req.query.date)

          endDate.setHours(23,59,59,999)

     }

     else {

          // beginning of current day

          var startDate = new Date()

          startDate.setHours(0,0,0,0)

          // end of current day

          var endDate = new Date()

          endDate.setHours(23,59,59,999)
```

```
    }

    Transactions.find({ date: { $gte: startDate.toJSON(),
$lte: endDate.toJSON() } }, function (err, docs) {

        var result = {

            date: startDate

        }

        if (docs) {

            var total = docs.reduce(function (p, c) {

                return p + c.total

            }, 0.00)
```

```
               result.total =
parseFloat(parseFloat(total).toFixed(2))

               res.send(result)

          }

          else {

               result.total = 0

               res.send(result)

          }

     })

})

// GET transactions for a particular date
```

```
app.get('/by-date', function (req, res) {

    var startDate = new Date(2018, 2, 21)

    startDate.setHours(0,0,0,0)

    var endDate = new Date(2015, 2, 21)

    endDate.setHours(23,59,59,999)

    Transactions.find({ date: { $gte: startDate.toJSON(),
$lte: endDate.toJSON() } }, function (err, docs) {

        if (docs)

            res.send(docs)

    })
```

```
})

// Add new transaction

app.post('/new', function (req, res) {

    var newTransaction = req.body



    Transactions.insert(newTransaction, function (err,
transaction) {

        if (err)

            res.status(500).send(err)

        else {

            res.sendStatus(200)
Inventory.decrementInventory(transaction.products)

          }

     })

})

// GET a single transaction

app.get('/:transactionId', function (req, res) {

     Transactions.find({ _id: req.params.transactionId },
function (err, doc) {

          if (doc)

               res.send(doc[0])

     })
```

```
})

import React from "react";

import { render } from "react-dom";

import { BrowserRouter } from "react-router-dom";

import registerServiceWorker from "./registerServiceWorker";

import "./index.css";

import "bootstrap/dist/css/bootstrap.css";

import { makeRoutes } from "./routes";

import App from "./App";


render(


 <BrowserRouter>


   <App />
 </BrowserRouter>,

 document.getElementById("root")

);

registerServiceWorker();

import React from "react";

import { Switch, Route } from "react-router-dom";

import Inventory from "./Inventory";

import Pos from "./Pos";

import Transactions from "./Transactions";

import LiveCart from "./LiveCart";

const Main = () => (
 <main>

   <Switch>

     <Route exact path="/" component={Pos} />

     <Route path="/inventory" component={Inventory} />

     <Route path="/transactions" component={Transactions} />

     <Route path="/livecart" component={LiveCart} />

   </Switch>

 </main>

);

export default Main;

import React from "react";
import { Link } from "react-router-dom";

// The Header creates links that can be used to navigate

// between routes.

const Header = () => (

 <div className="text-center">

   <h1>

     <a href="/#/">Real Time Point POS</a>

   </h1>

   <ul className="nav-menu">

     <li className="lead">
```

```
  <Link to="/inventory">Inventory</Link>

</li>

<li className="lead">

  <Link to="/">POS</Link>

</li>

<li className="lead">

  <Link to="/transactions">Transactions</Link>

</li>

<li className="lead">

  <Link to="/livecart">LiveCart</Link>
```

```
      </li>

    </ul>

  </div>

);

import React, { Component } from "react";

import "./App.css";

import Header from "./Header";

import Product from "./Product";

import axios from "axios";

const HOST = "http://localhost:80";

class Inventory extends Component {
}

constructor(props) {

  super(props);

  this.state = { products: [] };

}

componentWillMount() {

  var url = HOST + `/api/inventory/products`;

  axios.get(url).then(response => {

    this.setState({ products: response.data });

  });

}
```

```
render() {
  var { products } = this.state;
  var renderProducts = () => {
    if (products.length === 0) {
      return <p>{products}</p>;
    }
    return products.map(product => <Product {...product} />);
  };
  return (
    <div>
```

```
<Header />

<div class="container">

  <a

    href="#/inventory/create-product"

    class="btn btn-success pull-right"

  >

    <i class="glyphicon glyphicon-plus" /> Add New Item

  </a>

  <br />

  <br />
```

```
<table class="table">

  <thead>

    <tr>

      <th scope="col">Name</th>

      <th scope="col">Price</th>

      <th scope="col">Quantity on Hand</th>

      <th />

    </tr>

  </thead>

  <tbody>{renderProducts()}</tbody>
```

```
          </table>

        </div>

      </div>

    );

  }

}

export default Inventory;

import React from "react";

import { Link } from "react-router-dom";

// The Header creates links that can be used to navigate

// between routes.
```

```
const Header = () => (

 <div className="text-center">

   <h1>

     <a href="/#/">Real Time Point POS</a>

   </h1>

   <ul className="nav-menu">

     <li className="lead">

       <Link to="/inventory">Inventory</Link>

     </li>

     <li className="lead">
```

```
      <Link to="/">POS</Link>

    </li>

    <li className="lead">

      <Link to="/transactions">Transactions</Link>

    </li>

    <li className="lead">

      <Link to="/livecart">LiveCart</Link>

    </li>

  </ul>

</div>
```

```
);

import React, { Component } from "react";

import "./App.css";

import Header from "./Header";

import Product from "./Product";

import axios from "axios";


const HOST = "http://localhost:80";


class Inventory extends Component {


 constructor(props) {


   super(props);


   this.state = { products: [] };
```

```
}

componentWillMount() {

  var url = HOST + `/api/inventory/products`;

  axios.get(url).then(response => {

    this.setState({ products: response.data });

  });

}

render() {

  var { products } = this.state;

  var renderProducts = () => {
```

```
  if (products.length === 0) {

    return <p>{products}</p>;

  }

  return products.map(product => <Product {...product} />);

};

return (

  <div>

    <Header />

    <div class="container">

      <a
  href="#/inventory/create-product"
  class="btn btn-success pull-right"
>
  <i class="glyphicon glyphicon-plus" /> Add New Item
</a>
<br />
<br />
<table class="table">
  <thead>
    <tr>
```

```
          <th scope="col">Name</th>

          <th scope="col">Price</th>

          <th scope="col">Quantity on Hand</th>

          <th />

        </tr>

      </thead>

      <tbody>{renderProducts()}</tbody>

    </table>

  </div>

</div>
```

```
    );

  }

}

export default Inventory;

    import React, { Component } from "react";

    import "./App.css";

    import Header from "./Header";

    import io from "socket.io-client";

    import axios from "axios";

    import moment from "moment";

    import { Modal, Button } from "react-bootstrap";
    import LivePos from "./LivePos";

    const HOST = "http://localhost:80";

    let socket = io.connect(HOST);

    class Pos extends Component {

     constructor(props) {

       super(props);

       this.state = {

         items: [],

         quantity: 1,

         id: 0,
         open: true,

         close: false,

         addItemModal: false,

         checkOutModal: false,

         amountDueModal: false,

         totalPayment: 0,

         total: 0,

         changeDue: 0,

         name: "",

         price: 0
```

```
    };

    this.handleSubmit = this.handleSubmit.bind(this);

    this.handleName = this.handleName.bind(this);

    this.handlePrice = this.handlePrice.bind(this);

    this.handlePayment = this.handlePayment.bind(this);

    this.handleQuantityChange =
this.handleQuantityChange.bind(this);

    this.handleCheckOut = this.handleCheckOut.bind(this);

  }

  componentDidUpdate() {

    if (this.state.items.length !== 0) {
```

```
    socket.emit("update-live-cart", this.state.items);

  }

}

handleSubmit = e => {

  e.preventDefault();

  this.setState({ addItemModal: false });

  const currentItem = {

    id: this.state.id++,

    name: this.state.name,

    price: this.state.price,
    quantity: this.state.quantity

  };

  var items = this.state.items;

  items.push(currentItem);

  this.setState({ items: items });

};

handleName = e => {

  this.setState({ name: e.target.value });

};

handlePrice = e => {
    this.setState({ price: e.target.value });

  };

  handlePayment = () => {

    this.setState({ checkOutModal: false });

    var amountDiff =

      parseInt(this.state.total, 10) -
parseInt(this.state.totalPayment, 10);

    if (this.state.total <= this.state.totalPayment) {

      this.setState({ changeDue: amountDiff });

      this.setState({ receiptModal: true });

      this.handleSaveToDB();
      this.setState({ items: [] });
      this.setState({ total: 0 });
    } else {
      this.setState({ changeDue: amountDiff });
      this.setState({ amountDueModal: true });
    }
  };
  handleQuantityChange = (id, quantity) => {
    var items = this.state.items;
    for (var i = 0; i < items.length; i++) {
```

```
    if (items[i].id === id) {
      items[i].quantity = quantity;
      this.setState({ items: items });
    }
  }
};

handleCheckOut = () => {
  this.setState({ checkOutModal: true });
  var items = this.state.items;
  var totalCost = 0;
```

```
    for (var i = 0; i < items.length; i++) {

      var price = items[i].price * items[i].quantity;

      totalCost = parseInt(totalCost, 10) +
parseInt(price, 10);

    }

    this.setState({ total: totalCost });

  };

  handleSaveToDB = () => {

    const transaction = {

      date: moment().format("DD-MMM-YYYY HH:mm:ss"),

      total: this.state.total,
      items: this.state.items

    };

    axios.post(HOST + "/api/new", transaction).catch(err
=> {

      console.log(err);

    });

  };

  render() {

    var { quantity, modal, items } = this.state;

    var renderAmountDue = () => {

      return (
        <Modal show={this.state.amountDueModal}>

          <Modal.Header closeButton>

            <Modal.Title>Amount</Modal.Title>

          </Modal.Header>

          <Modal.Body>

            <h3>

              Amount Due:

              <span
class="text-danger">{this.state.changeDue}</span>

            </h3>

            <p>Customer payment incomplete; Correct and
Try again</p>
```

```
        </Modal.Body>

        <Modal.Footer>

          <Button onClick={() => this.setState({
amountDueModal: false })}>

            close

          </Button>

        </Modal.Footer>

      </Modal>

    );

  };

  var renderReceipt = () => {
    return (

      <Modal show={this.state.receiptModal}>

        <Modal.Header closeButton>

          <Modal.Title>Receipt</Modal.Title>

        </Modal.Header>

        <Modal.Body>

          <h3>

            Total:

            <span
class="text-danger">{this.state.totalPayment}</span>

          </h3>
```

```
           <h3>

             Change Due:

             <span
class="text-success">{this.state.changeDue}</span>

           </h3>

         </Modal.Body>

         <Modal.Footer>

           <Button onClick={() => this.setState({
receiptModal: false })}>

             close

           </Button>

         </Modal.Footer>
```

```
      </Modal>

    );

  };

  var renderLivePos = () => {

    if (items.length === 0) {

      return <p> No products added</p>;

    } else {

      return items.map(

        item => (

          <LivePos {...item}
onQuantityChange={this.handleQuantityChange} />
```

```
      ),
      this
    );
  }
};
return (
  <div>
    <Header />
    <div class="container">
      <div class="text-center">
```

```
           <span class="lead">Total</span>

           <br />

           <span class="text-success
checkout-total-price">

             ${this.state.total}

             <span />

           </span>

           <div>

             <button

               class="btn btn-success lead"

               id="checkoutButton"
               onClick={this.handleCheckOut}

             >

               <i class="glyphicon
glyphicon-shopping-cart" />

               <br />

               <br />

               C<br />

               h<br />

               e<br />

               c<br />

               k<br />
```

```
  o<br />

  u<br />

  t

</button>

<div className="modal-body">

  <Modal show={this.state.checkOutModal}>

    <Modal.Header closeButton>

      <Modal.Title>Checkout</Modal.Title>

    </Modal.Header>

    <Modal.Body>
```

```
                   <div ng-hide="transactionComplete"
class="lead">

                     <h3>

                       Total:

                       <span class="text-danger">
{this.state.total} </span>

                     </h3>

                     <form

                       class="form-horizontal"

                       name="checkoutForm"

                       onSubmit={this.handlePayment}

                     >
```

```
                      <div class="form-group">

                        <div class="input-group">

                          <div
class="input-group-addon">$</div>

                          <input

                            type="number"

                            id="checkoutPaymentAmount"

                            class="form-control
input-lg"

                            name="payment"

                            onChange={event =>

                              this.setState({
                                totalPayment:
event.target.value

                              })

                            }

                            min="0"

                          />

                        </div>

                      </div>

                      <p class="text-danger">Enter
payment amount.</p>

                      <div class="lead" />

                      <Button
```

```
                        class="btn btn-primary btn-lg
lead"

                        onClick={this.handlePayment}

                      >

                        Print Receipt

                      </Button>

                    </form>

                  </div>

                </Modal.Body>

                <Modal.Footer>

                  <Button
```

```
                     onClick={() => this.setState({
checkOutModal: false })}

                   >

                     Close

                   </Button>

                 </Modal.Footer>

               </Modal>

             </div>

           </div>

         </div>

         {renderAmountDue()}
         {renderReceipt()}

         <table class="pos table table-responsive
table-striped table-hover">

           <thead>

             <tr>

               <td colspan="6" class="text-center">

                 <span class="pull-left">

                   <button

                     onClick={() => this.setState({
addItemModal: true })}

                     class="btn btn-default btn-sm"

                   >
```

```
                     <i class="glyphicon glyphicon-plus"
/> Add Item

                   </button>

                 </span>

                 <Modal show={this.state.addItemModal}
onHide={this.close}>

                   <Modal.Header closeButton>

                     <Modal.Title>Add
item(Product)</Modal.Title>

                   </Modal.Header>

                   <Modal.Body>

                     <form

                       ref="form"
```

```
                      onSubmit={this.handleSubmit}

                      class="form-horizontal"

                    >

                      <div class="form-group">

                        <label class="col-md-2 lead"
for="name">

                          Name

                        </label>

                        <div class="col-md-8
input-group">

                          <input

                            class="form-control"
                            name="name"

                            required

                            onChange={this.handleName}

                          />

                        </div>

                      </div>

                      <div class="form-group">

                        <label class="col-md-2 lead"
        for="price">

                          Price

                        </label>
```

```
                         <div class="col-md-8
input-group">

                           <div
class="input-group-addon">$</div>

                           <input

                             type="number"

                             step="any"

                             min="0"

                             onChange={this.handlePrice}

                             class="form-control"

                             name="price"

                             required
```

```
                              />

                            </div>

                          </div>

                          <p class="text-danger">Enter price
for item.</p>

                        </form>

                      </Modal.Body>

                      <Modal.Footer>

                        <Button
onClick={this.handleSubmit}>Add</Button>

                        <Button

                          onClick={() => this.setState({
addItemModal: false })}
```

```
        >

          Cancel

        </Button>

      </Modal.Footer>

    </Modal>

  </td>

</tr>

<tr class="titles">

  <th>Name</th>

  <th>Price</th>
```

```
          <th>Quantity</th>

          <th>Tax</th>

          <th>Total</th>

          <th />

        </tr>

      </thead>

      <tbody>{renderLivePos()}</tbody>

    </table>

  </div>

</div>
```

```
    );

  }

}

export default Pos;

import React, { Component } from "react";
import "./App.css";

import io from "socket.io-client";

import Header from "./Header";

import axios from "axios";

import RecentTransactions from "./RecentTransactions";

import LiveTransactions from "./LiveTransactions";

import moment from "moment";

const HOST = "http://localhost:80";

var url = HOST + `/api//day-total/`;

class LiveCart extends Component {

 constructor(props) {
   super(props);

   this.state = { transactions: [], liveTransactions: []
};

 }

 componentWillMount() {

   // console.dir(socket);

   axios.get(url).then(response => {

     this.setState({ transactions: response.data });

     console.log("response", response.data);

   });

   var socket = io.connect(HOST);
   socket.on("update-live-cart-display", liveCart => {

     this.setState({ liveTransactions: liveCart });

   });

 }

 componentWillUnmount() {

   // socket.disconnect();

   // alert("Disconnecting Socket as component will
unmount");

 }

 render() {

   var { transactions, liveTransactions } = this.state;
   var renderRecentTransactions = () => {
     if (transactions.length === 0) {
       return <p>No recent transactions available</p>;
     } else {
       return transactions.map(transaction => (
         <RecentTransactions {...transaction} />
       ));
     }
   };
   var renderDate = () => {
     return moment().format("DD-MMM-YYYY HH:mm:ss");

   };

   var renderLiveTransactions = () => {

     if (liveTransactions.length === 0) {

       return (

         <div>

           <div class="col-md-5 pull-right">

             <div>

               <div class="alert alert-warning
text-center" role="alert">

                 <strong>Not Active:</strong> No items
added at the moment.
     }

               </div>

             </div>

           </div>

         </div>

       );

     } else {

       return liveTransactions.map(liveTransaction => (

         <LiveTransactions {...liveTransaction} />

       ));

     }
```

```
    };

    return (

      <div>

        <Header />

        <div class="livecart">

          <div class="col-md-5 pull-right">

            <div class="panel panel-primary">

              <div class="panel-heading text-center
lead">{renderDate()}</div>

              <table class="receipt table table-hover">

                <thead>
                  <tr class="small">

                    <th> Quantity </th>

                    <th> Product </th>

                    <th> Price </th>

                  </tr>

                </thead>

                <tbody>{renderLiveTransactions()}</tbody>

              </table>

            </div>

          </div>
```

```
        <div class="col-md-5 pull-left">

          <div class="panel panel-default">

            <div class="panel-heading lead text-center">

              Recent Transactions

            </div>

            <div class="panel-body">

              <div class="text-center">

                <span>Today's Sales</span>

                <br />

                <span class="text-success
checkout-total-price">
```

```
            $<span />

          </span>

        </div>

        <table class="table table-hover
table-striped">

          <thead>

            <tr>

              <th>Time</th>

              <th>Total</th>

            </tr>

          </thead>
```

```
<tbody>{renderRecentTransactions()}</tbody>

               </table>

             </div>

           </div>

         </div>

       </div>

     </div>

   );

 }

}
```

```
 import React, { Component } from "react";

import "./App.css";

import Header from "./Header";

import CompleteTransactions from
"./CompleteTransactions";

import axios from "axios";

const HOST = "http://localhost:80";

const url = HOST + `/api/all`;

class Transactions extends Component {

 constructor(props) {

   super(props);
```

```
  this.state = { transactions: [] };

}

componentWillMount() {

  axios.get(url).then(response => {

    this.setState({ transactions: response.data });

    console.log("response:", response.data);

  });

}

render() {

  var { transactions } = this.state;
  var rendertransactions = () => {

    if (transactions.length === 0) {

      return <p>No Transactions found</p>;

    }

    return transactions.map(transaction => (

      <CompleteTransactions {...transaction} />

    ));

  };

  return (

    <div>
```

```
<Header />

<div class="text-center">

  <span class="">Today's Sales</span>

  <br />

  <span class="text-success checkout-total-price">

    $ <span />

  </span>

</div>

<br />

<br />
```

```
<table class="table table-hover table-striped">

  <thead>

    <tr>

      <th>Time</th>

      <th>Total</th>

      <th>Products</th>

      <th>Open</th>

    </tr>

  </thead>

  <tbody>{rendertransactions()}</tbody>
```

```
          </table>

        </div>

     );

  }

}

export default Transactions;
```

On the `componentWillMount` all transactions and retrieved from the database.

The `rendertransactions` function displays all the transactions using the `CompleteTransactions` presentional component. See source code for more on 'CompleteTransactions.