Desgin Document for Crypto Currency Calculator Application

Revision History

| Approver Role | Name | Signature |
| --- | --- | --- |
| Developer | Premkumar Natarajan  [premkumar502@gmail.com](mailto:premkumar502@gmail.com)  <http://www.linkedin.com/in/premn> | Version 1.0 |
|  |  |  |
|  |  |  |

Table of Contents

[1 Introduction 4](#_Toc8855149)

[2 Assumptions 4](#_Toc8855150)

[3 Software and Hardware 4](#_Toc8855151)

[4 Technical Architecture 5](#_Toc8855152)

[4.1 FrontEnd Design 5](#_Toc8855153)

[4.1.1 Setting Up the React Application 5](#_Toc8855154)

[4.2 BackEnd Design 6](#_Toc8855155)

[4.3 DB Design 6](#_Toc8855156)

[4.3.1 Configure the MongoDB in the PATH 7](#_Toc8855157)

[4.3.2 Start the MongoDB 7](#_Toc8855158)

[4.3.3 Create a Database in MongoDB 7](#_Toc8855159)

[5 Get Best Sell Time REST API service 9](#_Toc8855160)

[5.1 Get Best Sell Time and Profit Service – Backend 9](#_Toc8855161)

[5.2 Get Best Sell Time and Profit Service – FrontEnd 9](#_Toc8855162)

[6 Exchange REST API Service 12](#_Toc8855163)

[7 Scope for Enhancement and Open Issues 13](#_Toc8855164)

[7.1 Scope for Enhancement 13](#_Toc8855165)

[7.2 Open Issues 13](#_Toc8855166)

[8 Prerequisites 14](#_Toc8855167)

[9 Appendix 14](#_Toc8855168)

[9.1 Installation & Confgiraution – MongoDB, NodeJS, React 14](#_Toc8855169)

[9.1.1 Install Java and MongDB 14](#_Toc8855170)

[9.1.2 Install NodeJS 15](#_Toc8855171)

[9.1.3 Install ReactJS 15](#_Toc8855172)

[9.1.4 Install Visual Studio Code Editor 15](#_Toc8855173)

[10 References 15](#_Toc8855174)

# Introduction

Crypto Currecny calculator application is to process the historical price list of currecies provided by a currency exchange. This application helps the customer to identify the best time to sell the currecies and also it calculates the profit for the transaction.

Also additionally this application allows the user to perform the below REST API service on the exchange data in the backend.

**Exchange REST API Services:**

Service URL : /api/currencyExchange/

**Operations**  :

* GET : Get the all the records from the DB for the various crypto currency.
* GETByID : Get the crypto currency from the DB based on the record Id.
* POST : Inserts the crypt currency record in DB
* PUT : Updates the crypt currency record in DB based in the record Id.
* Delete : Deletes the crypt currency record in DB based in the record Id.

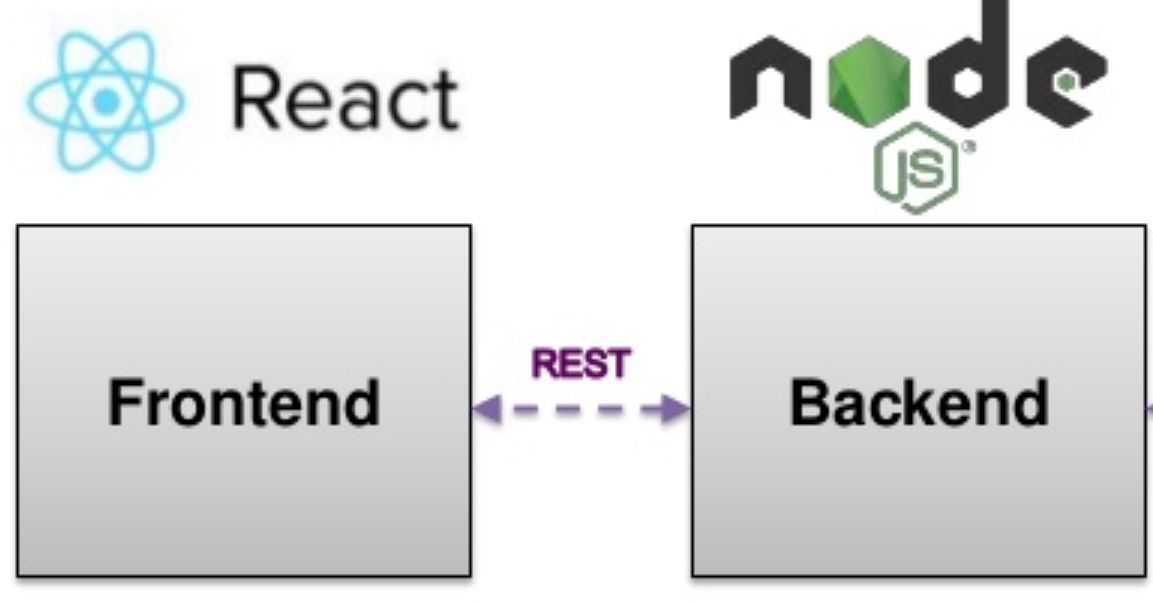
# Assumptions

* Below sample data been used for development of application. But still user can insert new records and application will be working for new records also.

# Software and Hardware

* Front End : ReactJS
* Back End : NodeJS, ES6
* Database : MongoDB
* Editor : Visual Studio Code (1.33.1)
* SCM & Repository : Git & GitHub
* Browser : Chrome
* OS : Windows 10

# Technical Architecture



MongoDB

Overall technical architecture illustrates the technologies adopted for developing this application.

## FrontEnd Design

Front end for the application been developed using ReactJS - A JavaScript front-end library for building user interfaces.

Refer the Appendix section for the Installation of the required softwares.

### Setting Up the React Application

1. Open the command prompt.
2. Check the Node been installed on the machine by checking the version of node.

**$ node -v**

1. Create the initial React project by using the create-react-appscript and the app name is **exchange-react-app**

**$ npx create-react-app exchange-react-app**

Inside this folder will find the default React project template with all dependencies installed. Change into the newly created folder:

$ cd mern-todo-app

1. Start the development web server by running the following command:

**$ npm start**

1. Install the bootstrap framework to the project as Bootstrap’s CSS classes will be used for developing the interfaces.

**$ npm install bootstrap**

1. Import the Bootstrap’s CSS file in the App.js which been created by deault as part of React application setup.

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

1. Set up the React router and its needed for adding the route configuration in the App.js file. Also import the react router in the App.js

**$ npm install react-router-dom**

**import { BrowserRouter as Router, Route, Link } from "react-router-dom";**

1. Embed the JSX code in a <Router></Router> element in the App.js for routing the requests.

**<Route path="/" component={CurrencyRate} />**

* Routes to the home page of the application.

**<Route path="/create" component={CreateForm} />**

1. Routes to the exchange form for submitting the data to get the best sell time for the provided currency, date and currency bought time. Also, it calculates the profit for the conversion.
2. To create the needed components in application, create a new directory src/components and create below new files:Components.

* **currency-rate.component.js**
  + Front End Home Page for the application.
* **exchange-create.component.js**
  + Exchange form for sending the parameters (currency, date and bought time) inorder to get the best sell time and profit.
  + This service been handled by the backend node service and its been exposed as REST service.

## BackEnd Design

NodeJS been used for developing the backend service and refer the appendix section for installation.

1. Server.js:

Iniation service file for starting the backend node server , monngoDB, for routing the api service and get best sell time service calls.

1. Model: app.model.js

Exchange API service operates based on this json schema and its been modelled for handling the service request. Using JOI npm modules, validation been handled for the model.

1. exchange-rate-api.js

Handles the Exchange API service call and for service details refer section : 6

1. exchange-routes.js

Handles the service request for fetching the best sell time and profit. This service been invoked from the frontend also.

1. config.js

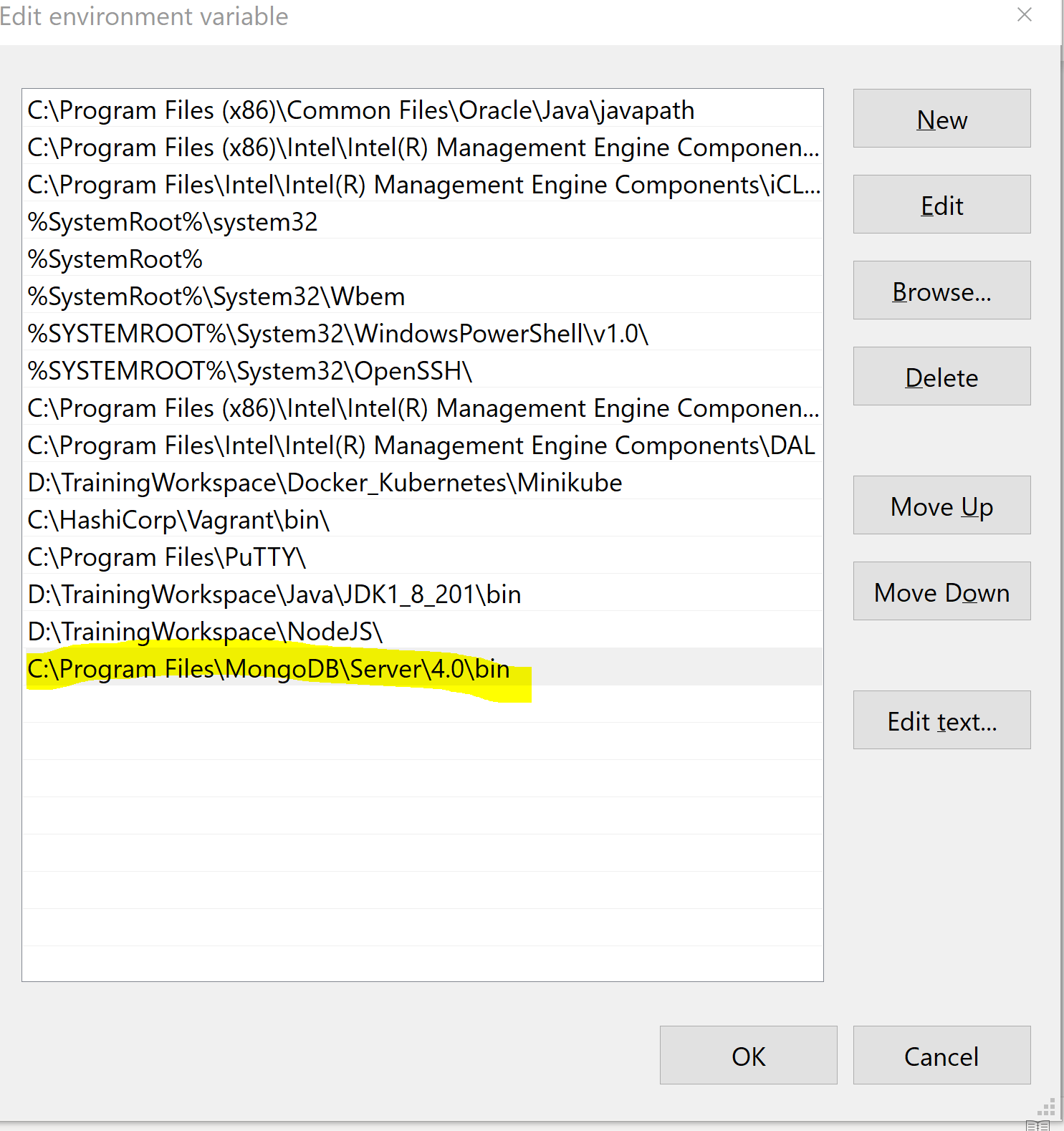
Holds the DB details and environment details.

## DB Design

Install the MongoDB as per the appendix section and configure as below.

### Configure the MongoDB in the PATH

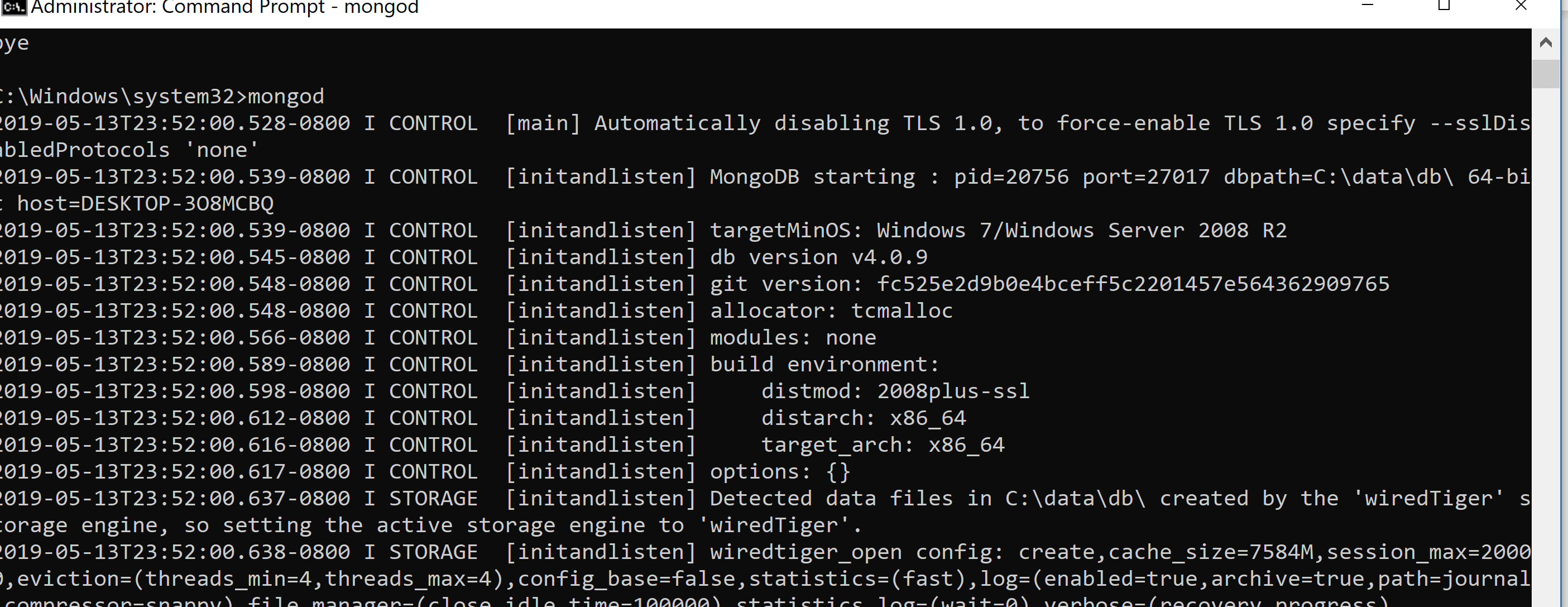
Add the installed mongoDB in the environment PATH.



### Start the MongoDB

1. Start MongoDB by running the mongod command from a shell prompt.

**$ mongod --dbpath D:\Prem\MongoDB\data\db**



### Create a Database in MongoDB

1. Create the below Database and collection in the MongoDB using Mongo Compass or Shell.

**Database : app-db**

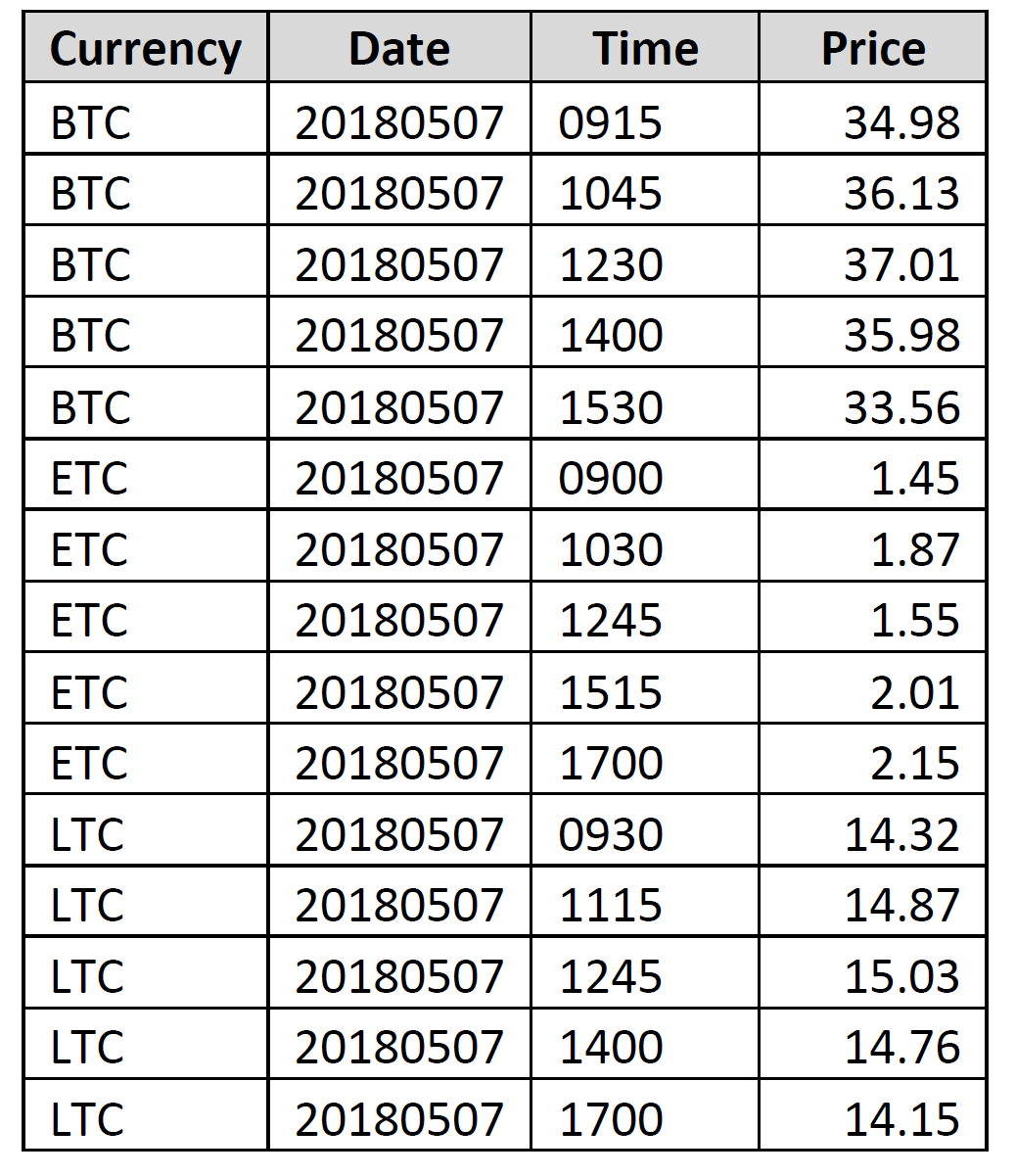
**Collection : appschemas**

1. Sample records provided for developing this application can be inserted in the DB using the backend API service or using the mongo CLI commands.

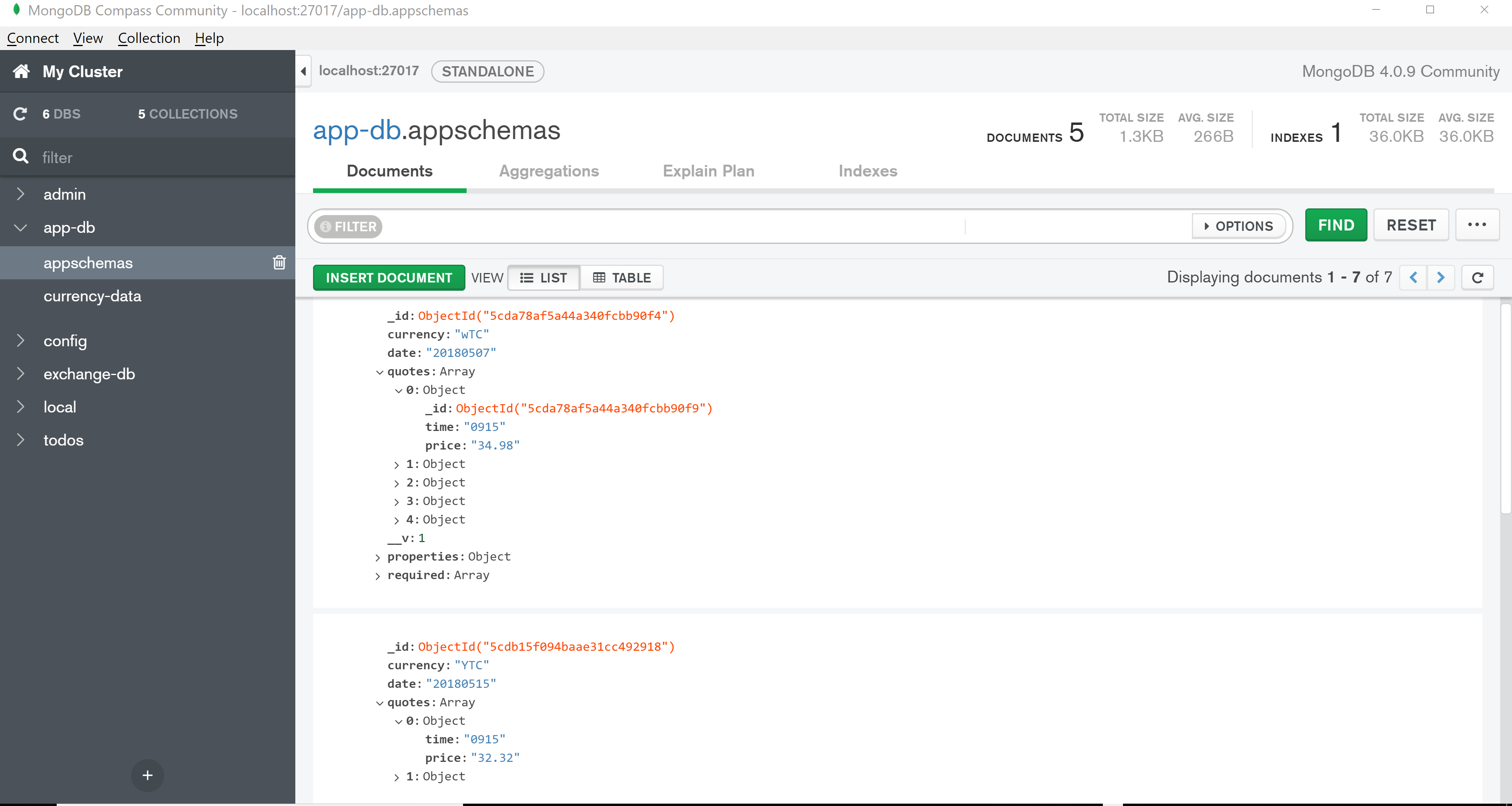
**$ mongoimport --db app-db –collection appschemas --file currency-data.json –jsonArray**

**Where:**

* currency-data.json – Sample records to be inserted in DB.
* Sample records for developing the app.

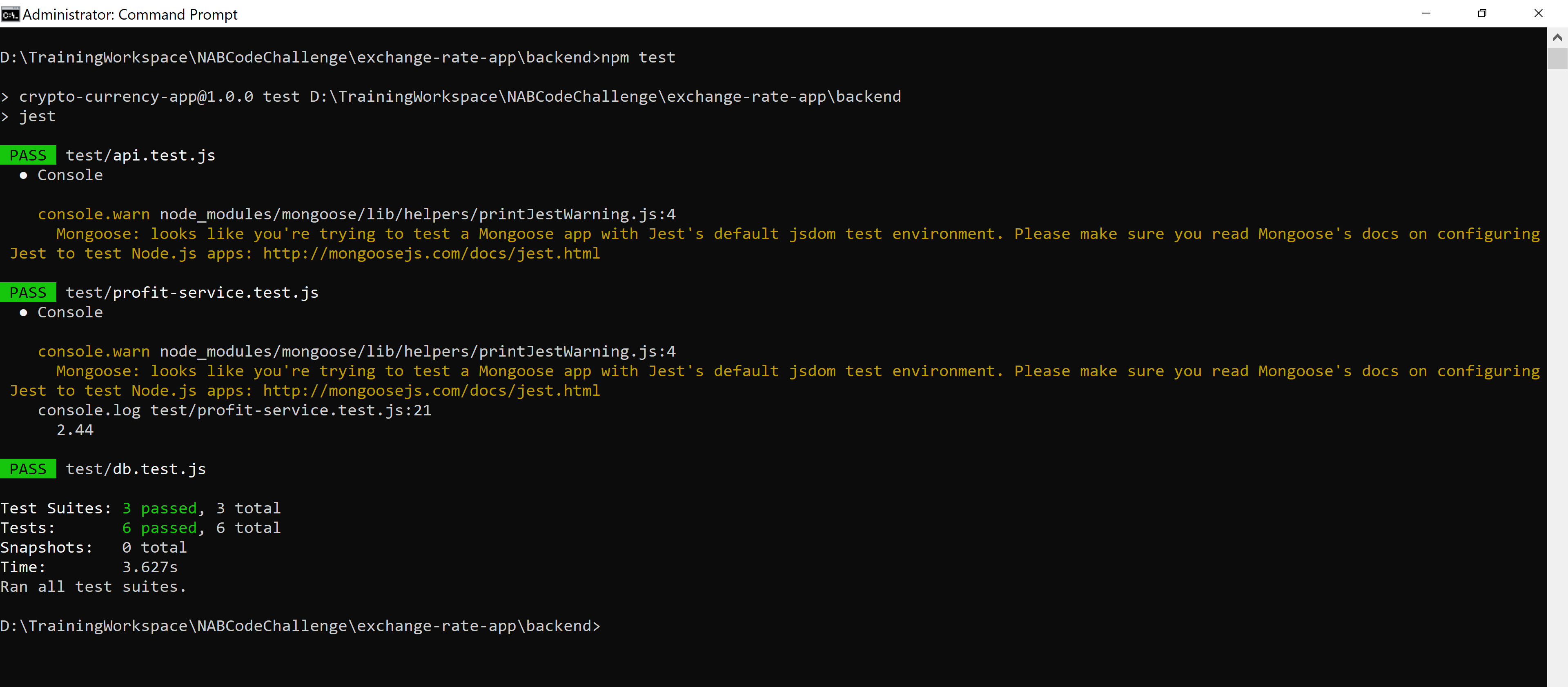


1. After insertion of records, it can verify by the MongoDB compass UI as shown below.



## Test Case

Prepared unit test cases for the backend services using Jest and test cases available in the folder. Backend/tests.



# Get Best Sell Time REST API service

These services are developed using React and NodeJS and exposed as REST API. It provides the below operations:

* Get the best Sell time for currency and profit.

## Get Best Sell Time and Profit Service – Backend

Backend URL: [http://localhost:4000/ api/getRates/](http://localhost:4000/api/currency-excahnge){currency}/{date}/{time}

Method : GET

Sample Request and Response from Backend:

URL : <http://localhost:4000/api/getRates/BTC/20180507/0915>

Response :

{

"currency": "BTC",

"date": "07-May-18",

"buyTime": 915,

"buyPrice": 34.98,

"sellTime": 1230,

"sellPrice": 37.01,

"profit": "2.03"

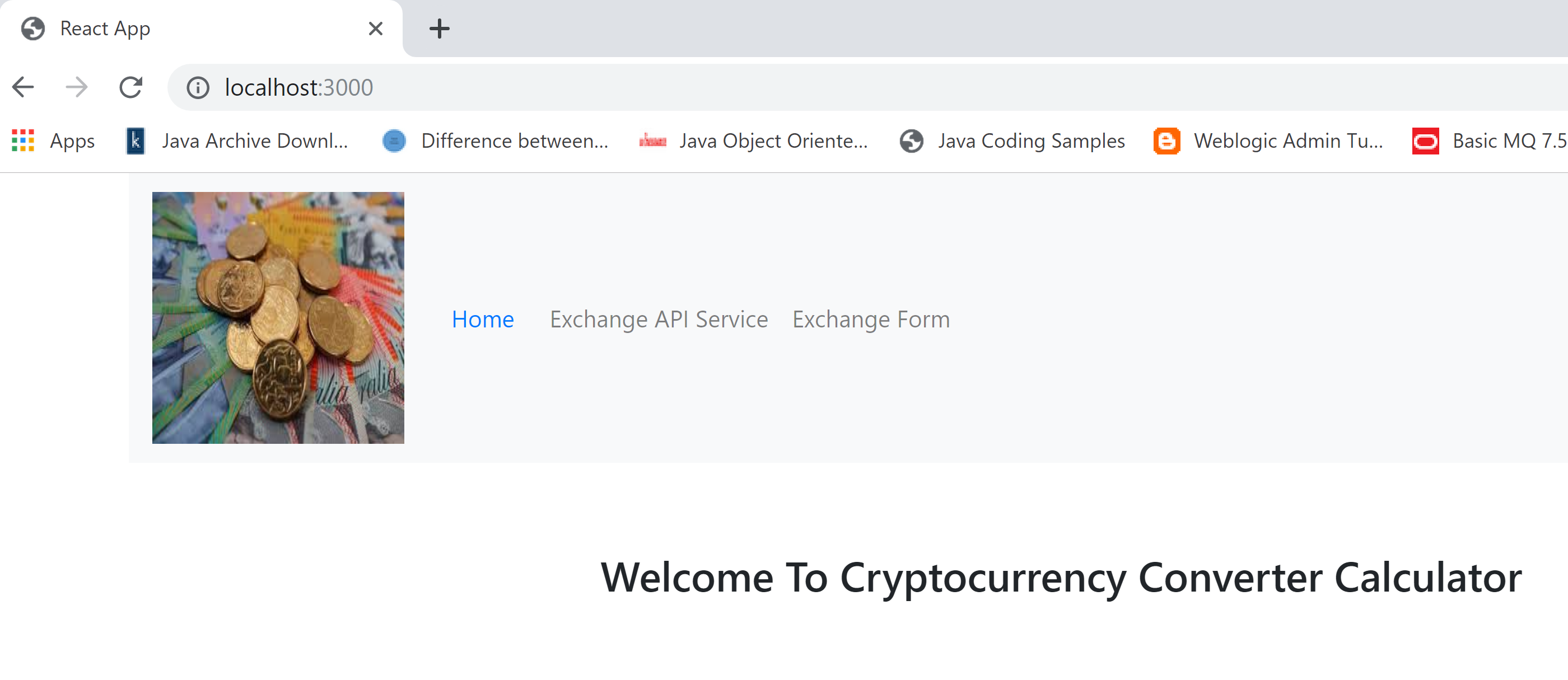
}

## Get Best Sell Time and Profit Service – FrontEnd

Sample Request and Response from FrontEnd:

FrontEnd URL: <http://localhost:3000/create>

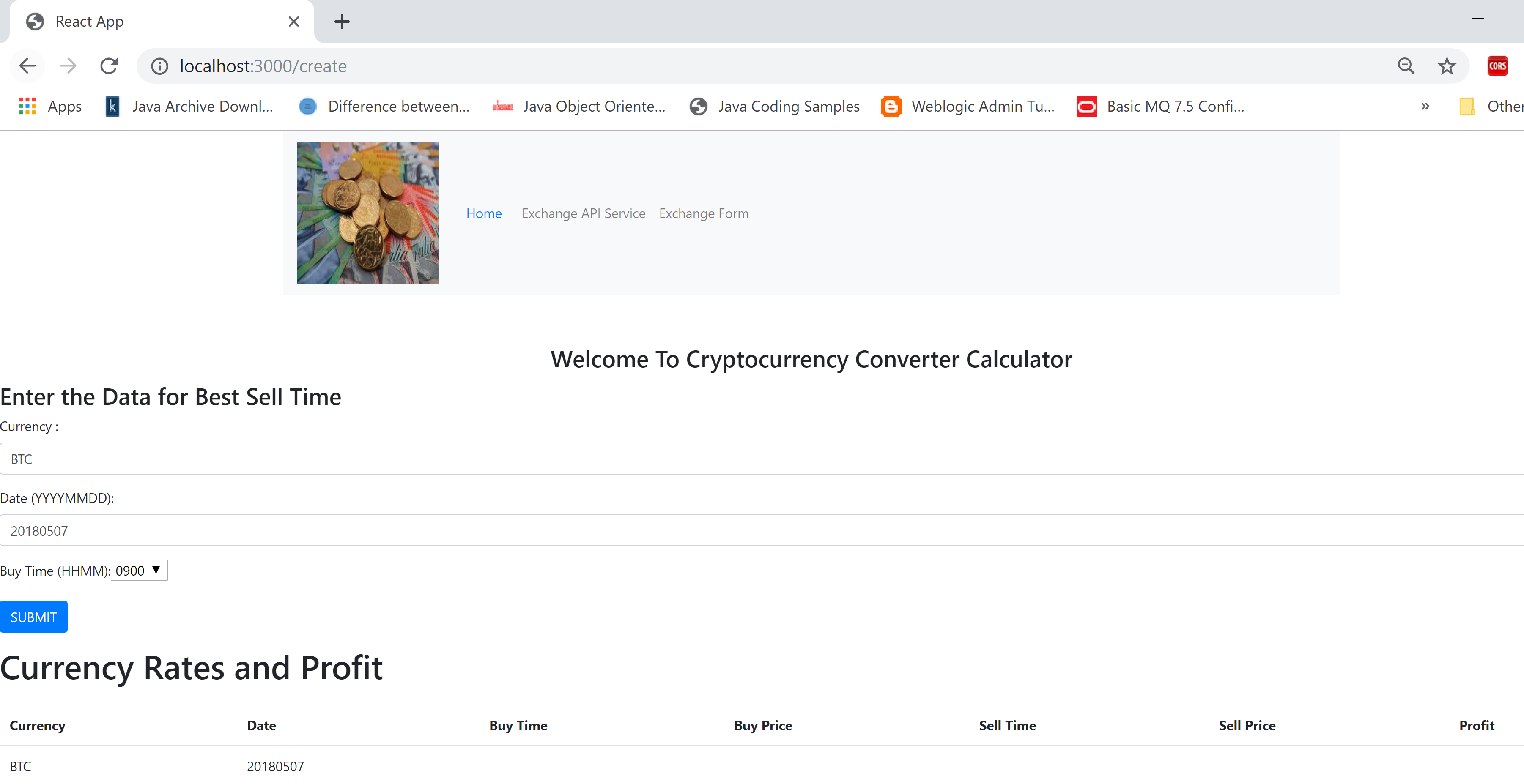
**Home Page: (http://localhost:3000/)**



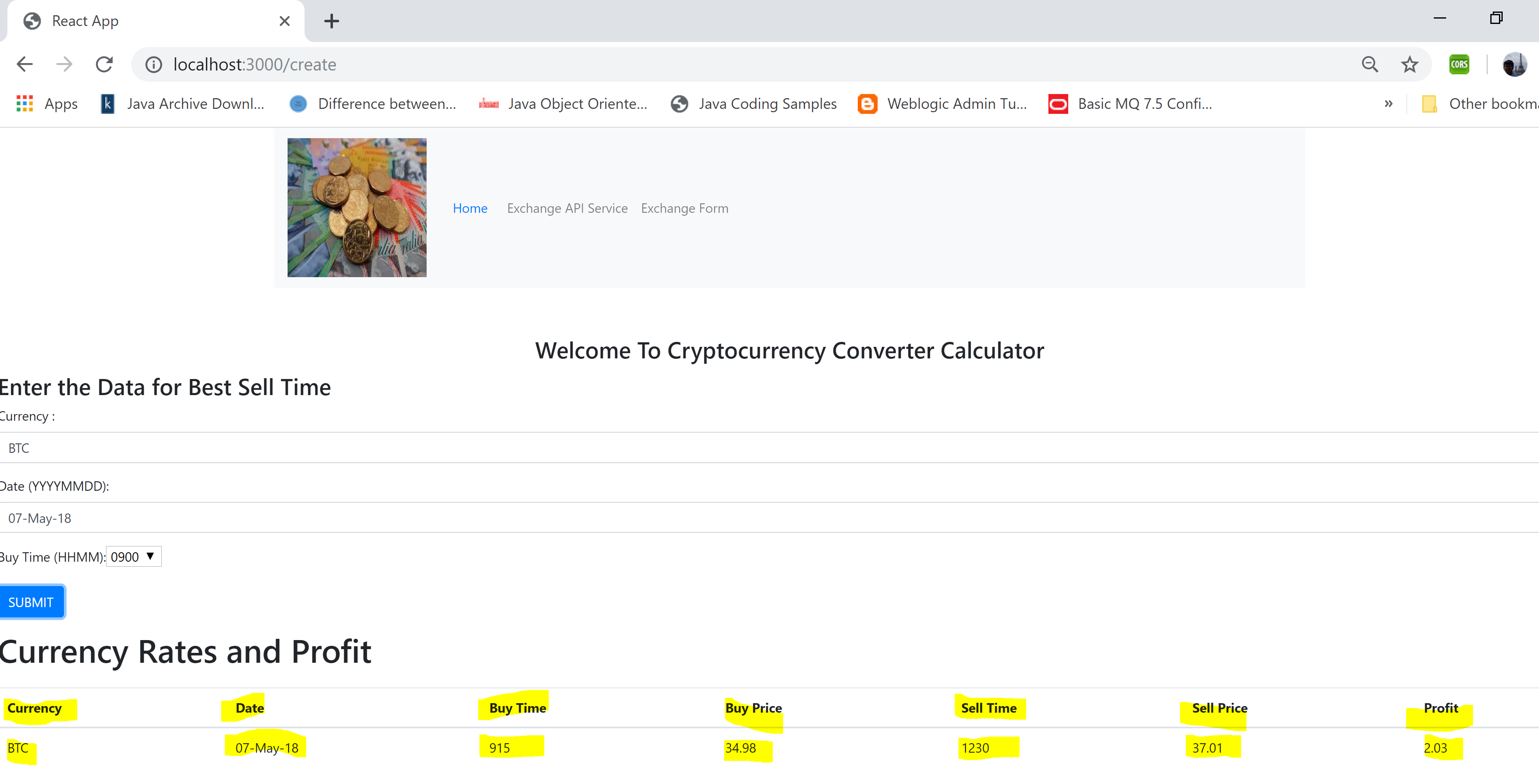
**Exchange Form: (http://localhost:3000/create)**

Mandtory fields for this service are currency code, date and exact time for currency bought.

* **Currency Code** : It should be 3 characters representing the crypto currency. Example : BTC
* **Date** : Its the date for which the best sell time and profit to be calculated. Format for this field is YYYYMMDD. Example : 20180507.
* **Time**: It’s the currency bought time and format is HHMM (Example : 0915). Currently service retireves the records only for the exact time and it can be enhanced fiurhter to find the best sell time within the range.



**Response :**



# Exchange REST API Service

These services are developed using NodeJS and exposed as REST API. It provides the below operations:

* Get Exchange Records
* Get Exchange Records based on Id
* Insert Exchange Records
* Update Exchange Records
* Delete Exchange Records

1. **Create Record:**

URL : <http://localhost:4000/api/currency-excahnge>

Method : POST

Sample Payload :

{

"currency": "LTC",

"date": "20180507",

"quotes": [

{

"time": "0930",

"price": "14.32"

},

{

"time": "1115",

"price": "14.87"

},

{

"time": "1245",

"price": "15.03"

},

{

"time": "1400",

"price": "14.76"

},

{

"time": "1700",

"price": "14.15"

}

]

}

1. **GetRecordById**

URL : <http://localhost:4000/api/currency-excahnge>/{Id}

Method : GET

1. **Delete Record**

URL : <http://localhost:4000/api/currency-excahnge>/{Id}

Method : DELETE

1. **Update Record**

URL : <http://localhost:4000/api/currency-excahnge>

Method : PUT

{

"currency": "LTC",

"date": "20180507",

"quotes": [

{ "time": "0930", "price": "14.32"

},

{ "time": "1115", "price": "14.87"

},

{ "time": "1245", "price": "15.03"

},

{"time": "1400", "price": "14.76"

},

{"time": "1700", "price": "14.15"

}

]

}

# Scope for Enhancement and Open Issues

## Scope for Enhancement

Application can be enhanced further using the below features:

1. Exchange REST API service which can be accessed from the back end service and it can be exposed through front end.
2. Front End can be enhaced by developing addiona widgets and calendar feature for input the date.
3. Can create docker images of this application and upload in the docker hub.
4. Can deploy it in AWS EC2 instance and also can adopt more features from AWS.

## Open Issues

1. Select option for selecting the currency bought time need to be refreshed on the page for reflecting the option values.
2. NPM package – CORS need the below extension to be enabled in the browser inorder to view the response from backend.

Enable Cross Origin resource sharing - Using the extension Allow-Control-Expose-Headers

# Prerequisites

1. Install MongoDB – version 3.4
2. NodeJS and dependent Modules
   1. Express
   2. Mongoose
   3. Joi
   4. body-parser
   5. winston
   6. cors
3. React JS and dependent Modules
   1. react-router-dom
   2. react

# Appendix

## Installation & Confgiraution – MongoDB, NodeJS, React

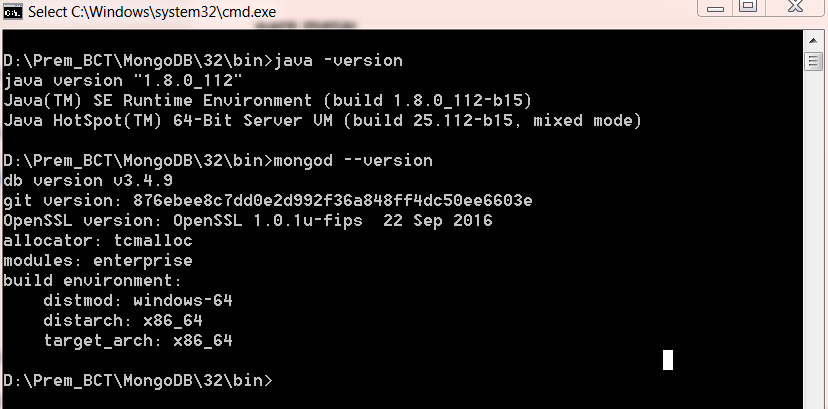
### Install Java and MongDB

Java Installation: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

MongoDB Installation: <https://docs.mongodb.com/manual/installation/>

Follow the instructions as mentioned in above URL’s for the specific operating system and make sure that their binaries are actually executable (so they are in your PATH env variable).

To check Java and MongoDB, execute the following commands and should get something like the below (output might vary depending on Java version and your OS):



### Install NodeJS

1. Download the nodejs from the url <https://nodejs.org/en/download/> as per the OS requirement.
2. Double click on the downloaded .msi file to start the installation. Click the Run button in the first screen to begin the installation.
3. And follow the screen and once it been installed successfully, verify it by checking the node version using the command : node -v

### Install ReactJS

Refer the section 6.1.1 for setting up the REACT JS application

### Install Visual Studio Code Editor

<https://code.visualstudio.com/>

# References

1. <https://reactjs.org/docs/forms.html>
2. [https://mongodb.github.io](https://mongodb.github.io/)
3. <https://nodejs.org/en/download/>
4. <https://www.npmjs.com/>