

# **CMPE 280 – Web UI Design and Development**

## **Cryptocurrency Portfolio Tracker**

### **Team Web Ninjas**

Ishwarya Varadarajan

Nishchay Madan

Sowmya Viswanathan

Suhel Mehta

## Table of Contents

<b>INTRODUCTION .....</b>	<b>3</b>
<b>ARCHITECTURE OVERVIEW .....</b>	<b>3</b>
MVC -Model, View, Controller .....	3
System Architecture.....	4
<b>APPLICATION FUNCTIONALITIES .....</b>	<b>6</b>
<b>UI DESIGN PRINCIPLES .....</b>	<b>11</b>
UI Design Pattern: Dashboard .....	12
UI Design Pattern: Fully Connected Navigation .....	13
UI Design Pattern: Grid of Equals .....	14
<b>DATA MODEL .....</b>	<b>15</b>
NoSQL Database - MongoDB .....	15
Collections.....	15
<b>DATA VISUALIZATION.....</b>	<b>17</b>
Features .....	18
Visualization for various periodicities using charts .....	19

## INTRODUCTION

Cryptocurrency Portfolio Tracker is an application to track and trade cryptocurrencies. Users are with option to trade 6 currencies – bitcoin, ripple, Ethereum, stellar, cardano, lite coin. Prices of the currencies are retrieved using an online API: <https://min-api.cryptocompare.com/data/price?fsym=BTC&tsyms=USD>

Cryptocurrencies are in demand today. In 2017, people who had bought bitcoins, would have been a lot richer as that crypto currency had a bull run last year. After that, there has been a constant increase in the usage of bitcoins and other currencies.

There are many web applications as well as mobile applications which handle the trading of these currencies. We got inspired from this online trading portal.

A user can check for the latest price of the currencies. And can also trade the currencies. He will also be presented with a dashboard showing him the weekly, monthly and yearly trends of the different currencies.

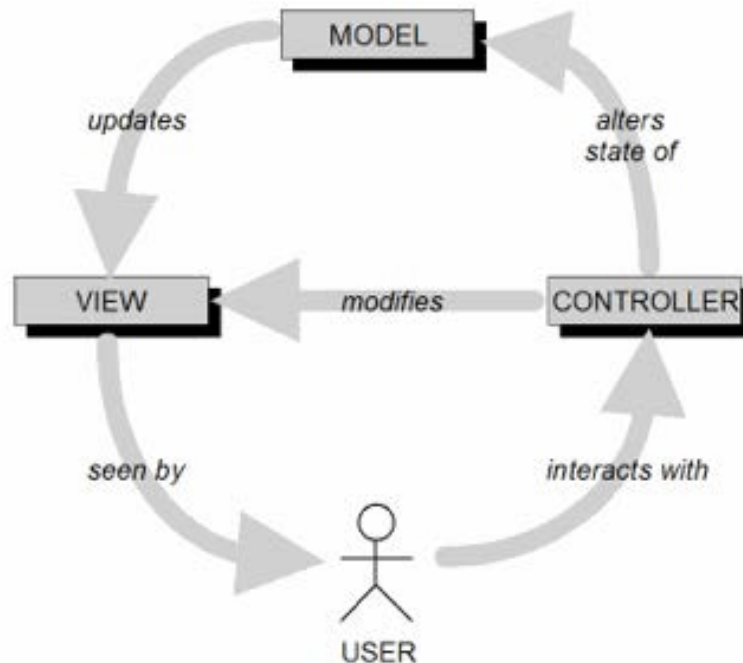
## ARCHITECTURE OVERVIEW

### MVC -Model, View, Controller

We implemented the MVC architecture for our application. This architecture has the following components: Model, View and Controller. This kind of model helps in decoupling the code, making it easier to maintain and debug. It also helps in code reusability. The MVC components and their functionality is described below:

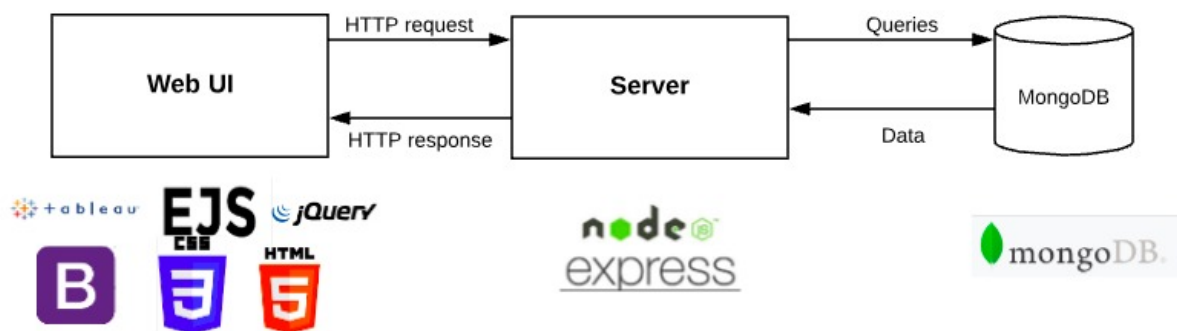
1. Model: It deals with the storing and retrieving of data. This is done with the directions issued by the controller code.
2. View: It is the representation of data. It deals with the changing of the view/representation based on the change in the model data.
3. Controller: Based on the input from the user, the controller routes the user to different pages.

MVC separates the application logic from that of the UI layer. Controller receives all the requests made by the user and works with model to change the view based on the request type for the user. The view then uses the data and presents a final UI output for the user. The MVC model can be graphically shown as below:



## System Architecture

The system architecture for our application is as show below:



We have implemented the MVC architecture for our application. The 'view' or the front-end has been developed using – HTML, CSS, jQuery, Bootstrap, EJS, and tableau. More focus and importance has been given to this component as the user experience is most important and holds

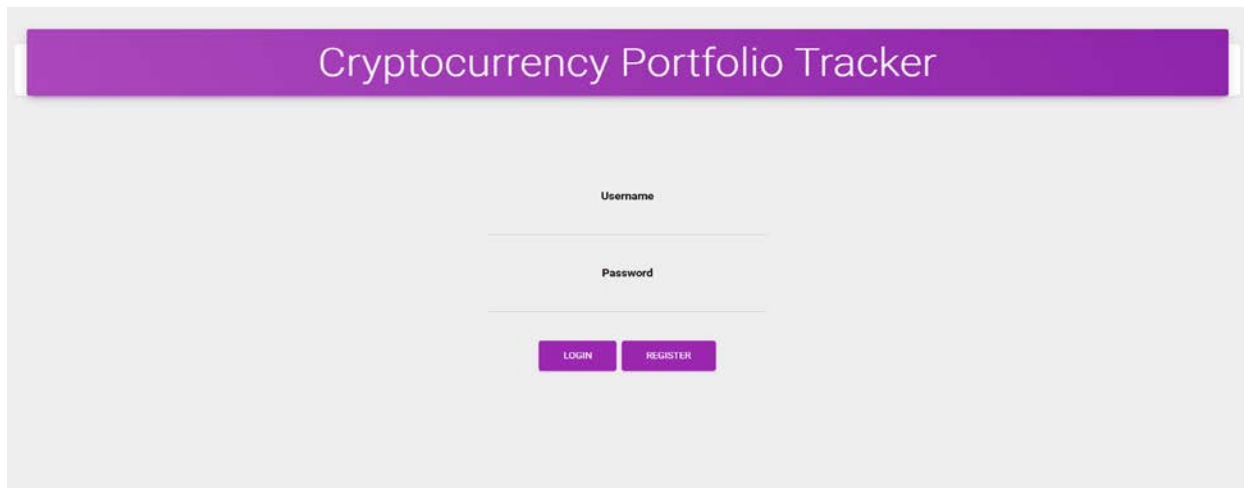
the highest priority. The ease of using this application has been achieved using the accessible screen layouts, appealing colors and simple web pages without much UI junk.

The controller components have been primarily developed using JavaScript. We have the node.js and the Express framework. Express is used to interact with the 'Model' component. It helps in routing a URL to the backend and stores all these routes in the 'routes' sub directory. A route can dynamically generate a webpage. EJS has been used to dynamically generate webpages. This helps in easier generation of HTML code with plain JavaScript. The EJS code can be found in the views sub directory. There is a separate controller module for every functionality in our application. We have all the code written in main.js.

The 'Model' components have been developed using MongoDB database. It is mainly for storing and retrieval of User information, currency information and transactions made by the user. Our model consists of collections for users, currencies and transactions. We are running the MongoDB on default port 27017. The user collections store the login details and the registration details of the users. The currencies collection stores the information about the currencies, to display it in the dashboard. The transactions collections store the details about users' trading of the currencies, what currency have they bought, how much and other details.

## APPLICATION FUNCTIONALITIES

- Login
  - The application starts with a login page which requires users to login to access the site.
  - The login page asks user their Username and Password by the user
  - If password and username doesn't match it will show a message "Invalid Login" on the same page



The screenshot displays the login interface for the 'Cryptocurrency Portfolio Tracker'. At the top, a purple header bar contains the title 'Cryptocurrency Portfolio Tracker' in white text. Below the header, the login form is centered on a light gray background. It features two input fields: the first is labeled 'Username' and the second is labeled 'Password'. Both labels are in a small, dark font above their respective input lines. At the bottom of the form, there are two purple buttons with white text: 'LOGIN' on the left and 'REGISTER' on the right.

- Register
  - New Users can register using this page
  - We are using validation to get accurate input from the users
  - After registering user can login to the site

# Register

Username

Username can contain any letters or numbers, without spaces

E-mail

xxx@xxx.com

Please provide your E-mail

Contact No.

(999) 999-9999

Please provide your Contact No.

Address

Please provide your address:

Password

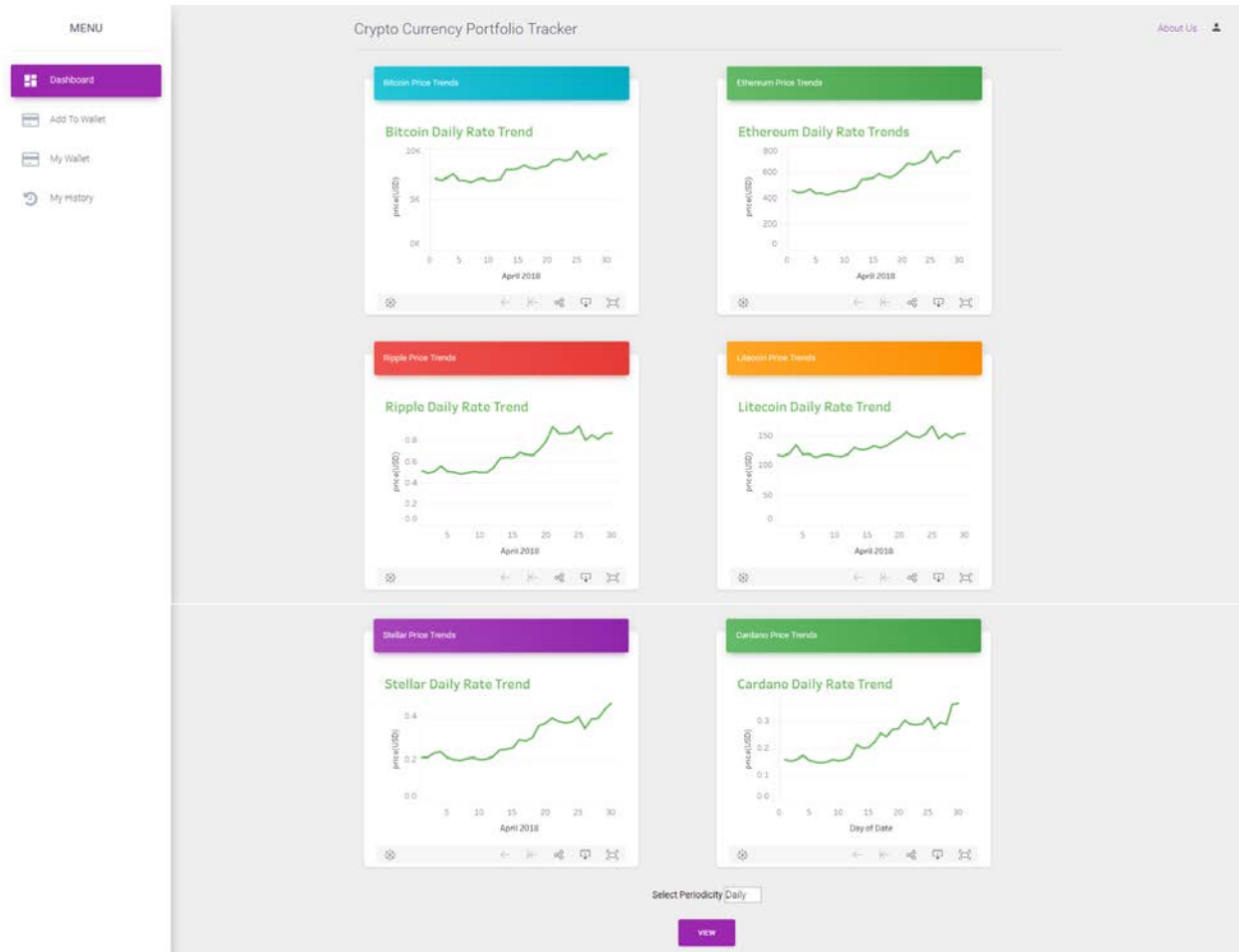
Password should be at least 4 characters

Password (Confirm)

Please confirm password

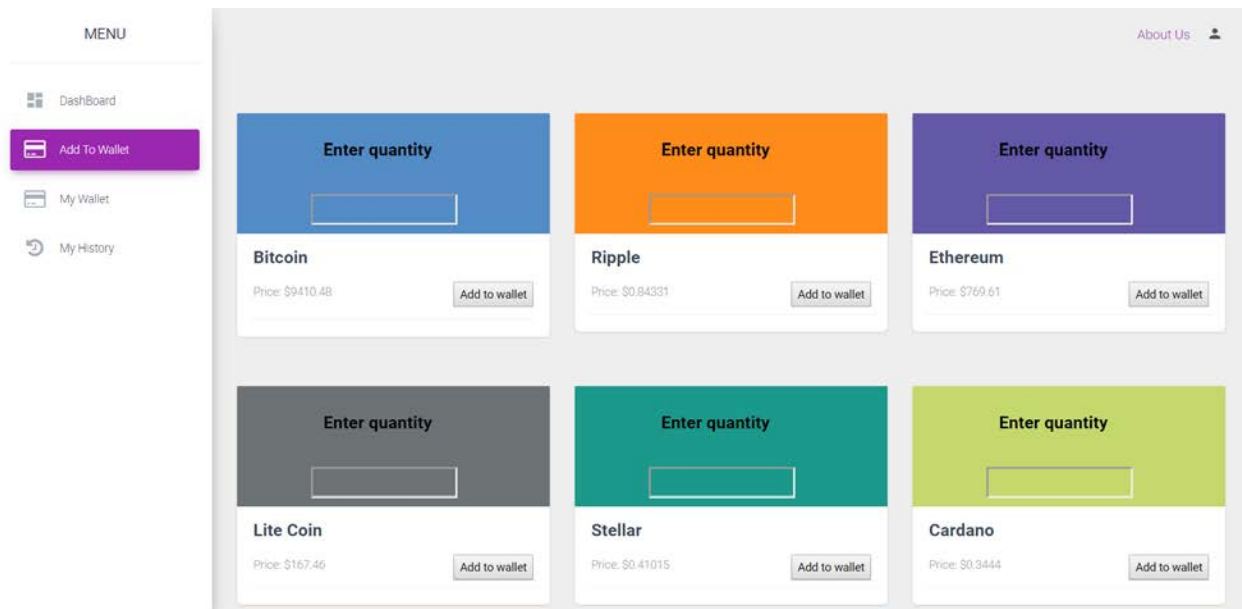
REGISTER

- Dashboard
  - After logging in, User lands on the dashboard
  - The Dashboard displays the daily trends of different currency which user can track and make a decision of which currency to buy
  - User can change the view to monthly or yearly to check the trends of the currencies

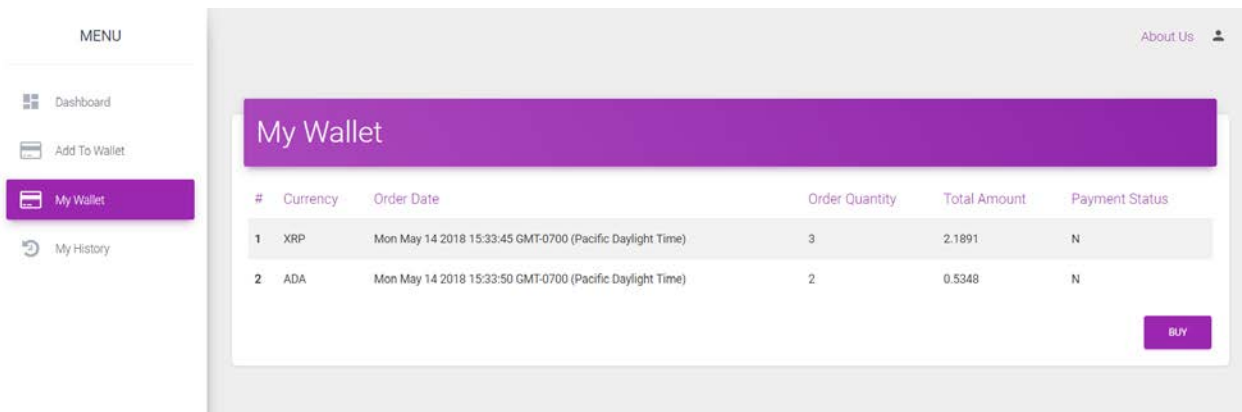


- Add to wallet
  - User can add different currencies to their wallet using add to wallet page
  - This page displays the amount of single coin for different currencies to help user to make a precise decision.

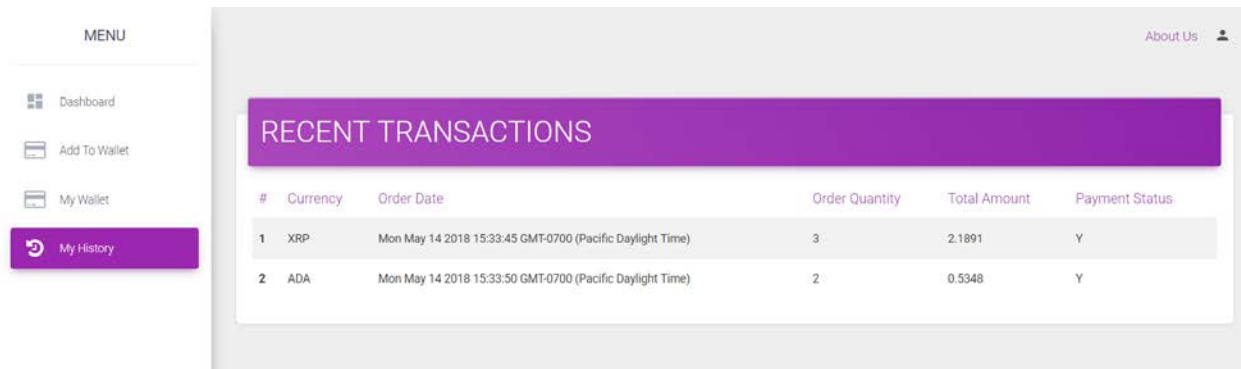




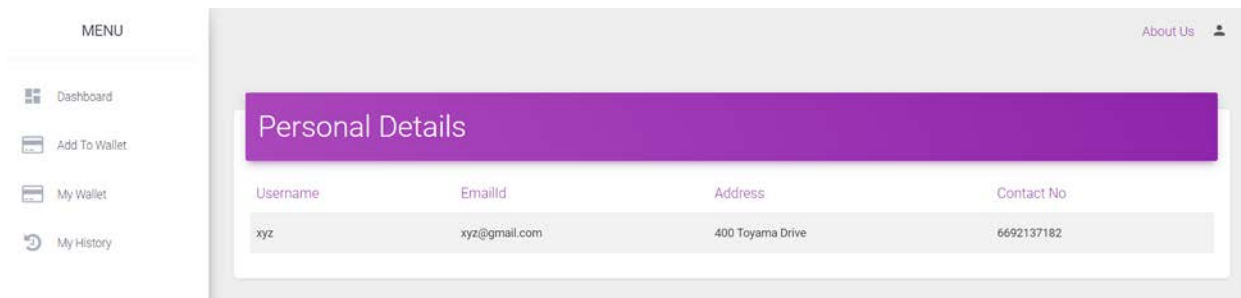
- My Wallet
  - After adding currencies to the wallet, User can buy the currency using My Wallet page
  - This page displays all the currencies user added to the wallet which hasn't been bought
  - User can click the buy button in the bottom right corner to buy the currencies



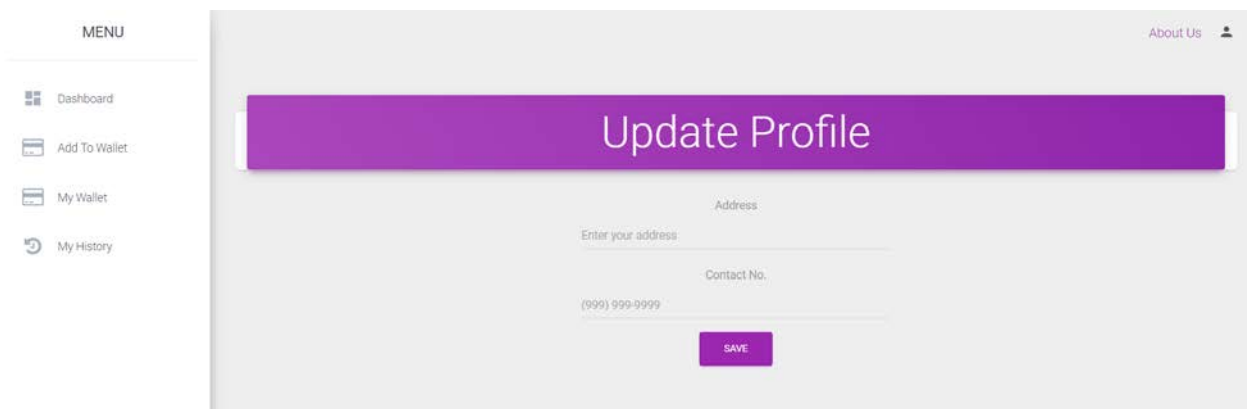
- My History
  - After buying the currencies, User can see their recent transactions i.e. showing all currencies they have previously bought.



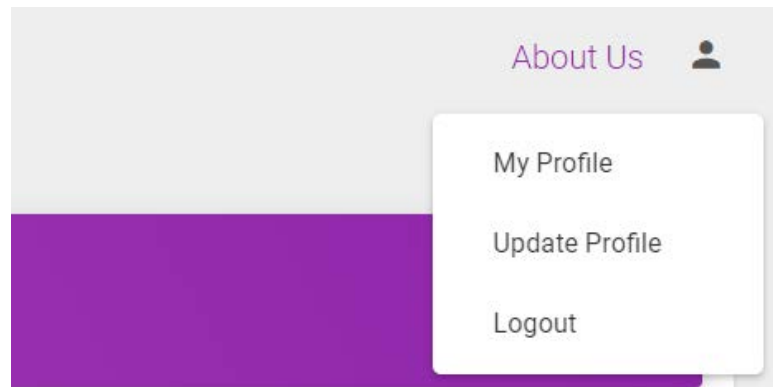
- My Profile
  - My Profile page displays the information of the user who is currently logged in



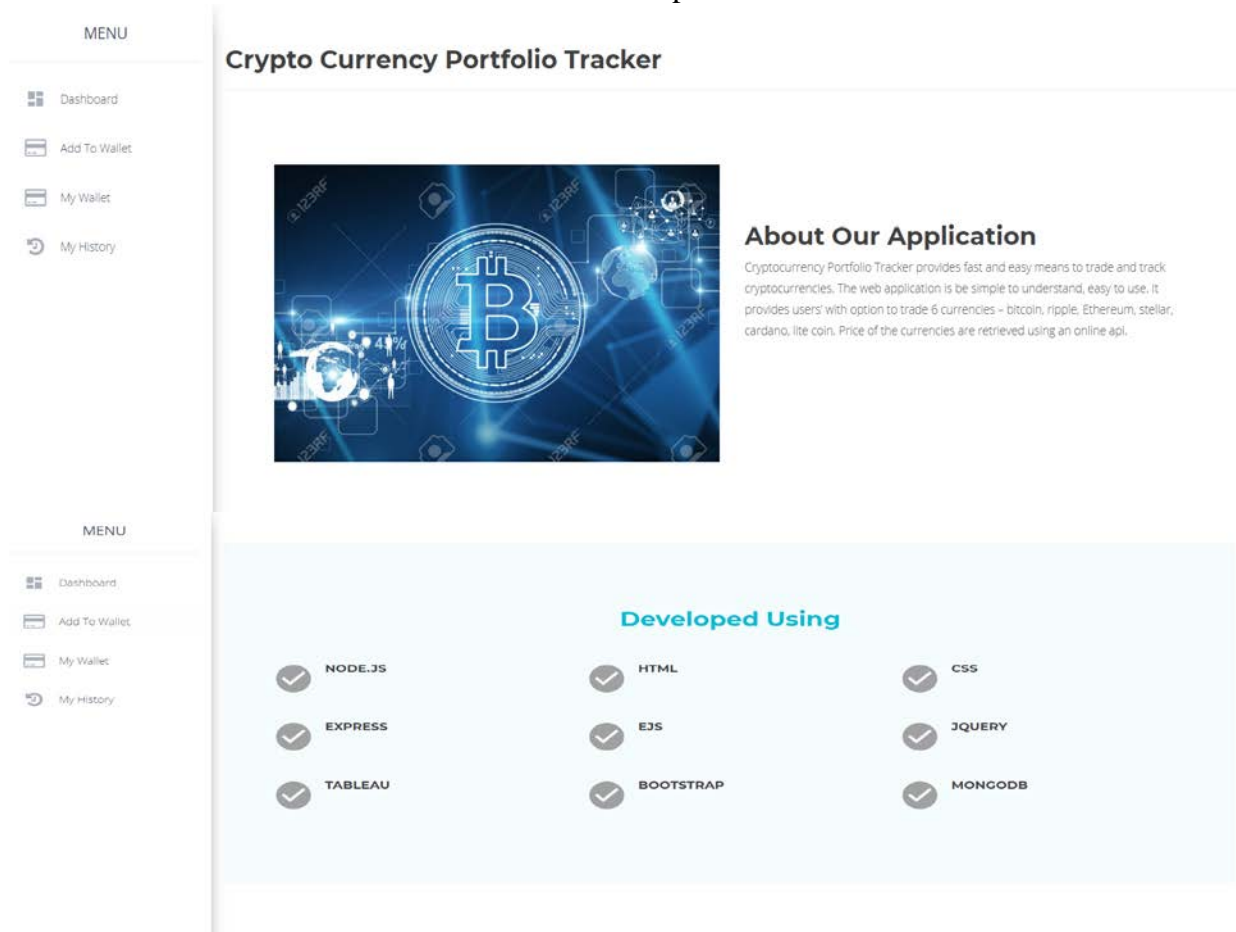
- Update Profile
  - Users can update their profile information on this page
  - When user click on submit button, the database gets updated.



- Logout
  - The User can choose to logout of the session whenever they want
  - After clicking on logout button, the user session ends and the login page of the site will be displayed.



- About Us
  - This page displays details about our application and the technologies we used to develop it.
  - It also shows details about our developers



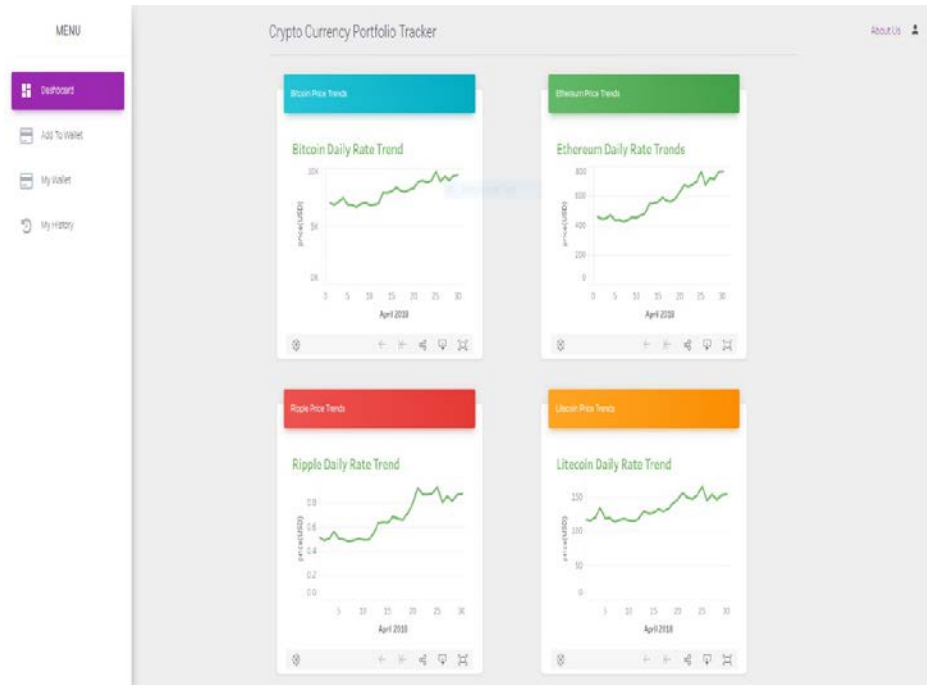
## UI DESIGN PRINCIPLES

There are certain UI design principles we had in mind when we set out to build this application, which we picked up as we kept on learning in the class. Some of these principles are discussed below:

- **Seen All at Once, At a Glance:**  
For a dashboard to be called a dashboard, this principle must be followed. It's not a dashboard if you have to scroll it.
- **Have user inputs only as much required:**  
Don't annoy users by making them enter redundant information. The main goal was to make all forms user friendly.
- **Most Important Information:**  
Have the most important information readily and easily available. So that the viewer can quickly absorb what's needed to know.
- **Snapshot is what clicks:**  
The information need not necessarily be updated in real time. Sometimes only a snapshot is needed.
- **Consistency in Views:**  
The screen should look the same from day to day without alteration. Except for changes in the data. Therefore, we kept UI of all pages as similar as possible to have a uniform look of the application.

### UI Design Pattern: Dashboard

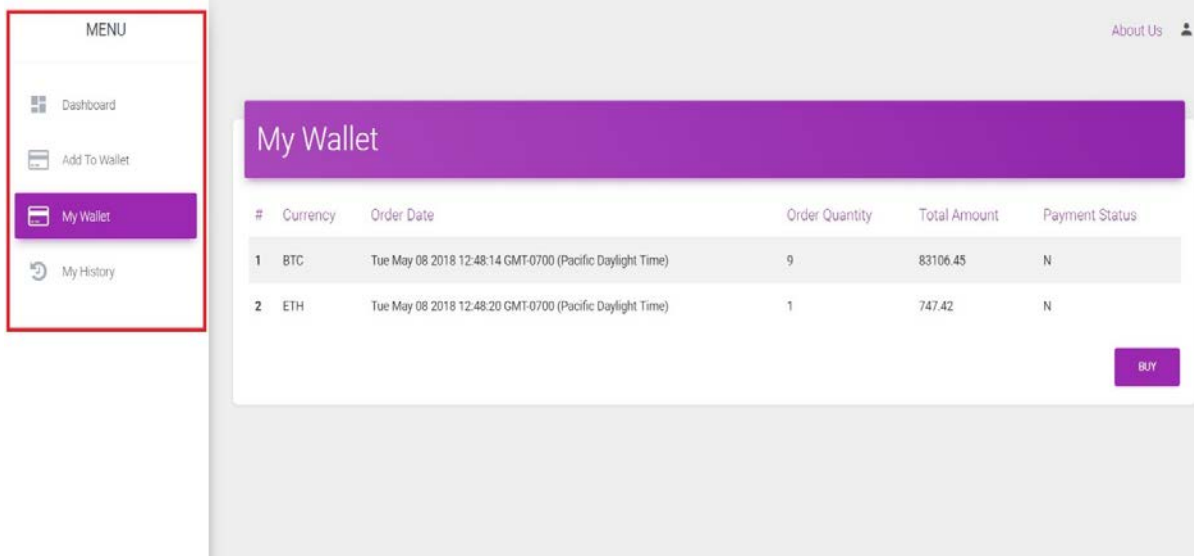
- A dashboard is a single information dense page containing many information widgets which are constantly updated.
- The Dashboard design pattern is used when there is a constant flow of incoming information.
- Dashboard design pattern is useful when a user needs to monitor the information and needs to glance at only on those parameters that are of utmost importance.
- In our application, we used Dashboard design pattern to show all the cryptocurrencies metrics which makes the user to see and analyze the stability of the currency before buying it.
- It also shows the current price so that user can get all the necessary information at one place before making its final decision.
- Therefore, we have made our web application more intuitive and user friendly by using this organizational UI design pattern.



*Dashboard showing currency trends along with current values*

### UI Design Pattern: Fully Connected Navigation

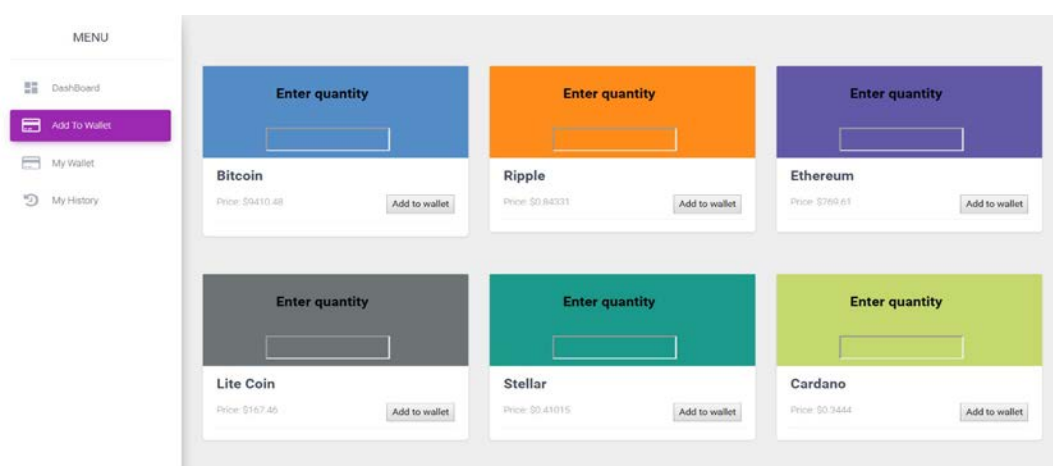
- In fully connected navigation, there is a link from all pages to all other pages in a website or web app.
- This is highly useful if the application is modular and a user frequently needs to go from one module to another.
- We have used side menu bar to incorporate this UI design pattern in our application which makes it easier for the new users to handle navigation.
- Once the user has logged in, the user can navigate to any subsection of the application from any webpage.
- Using menu side bar also helped in keeping our web UI neat and uniform that attracts users.



*Snapshot of Menu Sidebar showing the links for navigation*

## UI Design Pattern: Grid of Equals

- In this design pattern, we arrange all the items in a grid which follows a common template and weight for each item also needs to be similar.
- We implemented this design pattern in our AddToWallet page which enables the user to preview and select one of those items at once.
- One of the other reasons for using Grid of Equals was to show the viewer the live rates of the currency that would help him make his selection.
- The add to wallet button in each grid takes the user to another web page where user can view the items he has added in his own cart.
- Overall, grid looks neat, ordered and user-friendly that suits the style of our application.



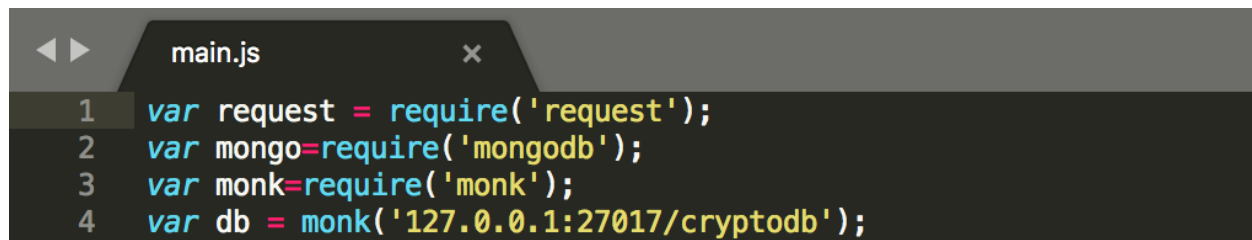
*Snapshot of AddtoWalletPage using GridOfEquals*

## DATA MODEL

### NoSQL Database - MongoDB

We have used MongoDB, a NoSQL database to support data storage and management operations for our web application. MongoDB is a document-oriented database which stores data in Binary JSON (BSON) format. It does not require a schema to be defined and created in advance for storing data and schema changes dynamically, as data changes. Since our application is built using Node.js and Express.js for the server-side framework, which provides runtime environment applications built using JavaScript, it is easy to integrate MongoDB with it. We use Monk, a middleware to connect MongoDB to the application which also provides more improved features in performing database operations.

'Cryptodb' is the database that we have used in our application. MongoDB which is installed locally in the system is connected to the application using the middleware Monk as below and the script is written in the main.js (controller) file.



```
1 var request = require('request');
2 var mongo=require('mongodb');
3 var monk=require('monk');
4 var db = monk('127.0.0.1:27017/cryptodb');
```

### Collections

The 'cryptodb' database has collections in which various application data are stored. The collections are as follows:

1. Transaction:
  - This collection stores users' transaction data, when a user adds cryptocurrencies to his/her wallet or purchases them.
  - It consists of the following columns
    - Username
    - Currency\_code
    - Current\_price
    - Tot\_amt
    - Order\_date
    - Order\_qty
    - Payment\_status
  - When the user adds the items to their wallet, data is inserted into the database with value for 'payment\_status' = 'N'
  - When the user purchases the items in the wallet using the 'checkout' option, the 'payment\_status' is set to 'Y'.

Below is a snapshot of the 'Transaction' collection

```
"_id" : ObjectId("5af127aa58f45b1a184ade11"),
"username" : "tUser",
"currency_code" : "ADA",
"current_price" : 0.342,
"order_date" : "Mon May 07 2018 21:29:30 GMT-0700 (Pacific Daylight Time)",
"order_qty" : "22",
"tot_amt" : 7.524000000000001,
"payment_status" : "Y"

"_id" : ObjectId("5af1fefe33e9fe1afc15ac90"),
"username" : "tUser",
"currency_code" : "BTC",
"current_price" : 9234.05,
"order_date" : "Tue May 08 2018 12:48:14 GMT-0700 (Pacific Daylight Time)",
"order_qty" : "9",
"tot_amt" : 83106.45,
"payment_status" : "Y"
```

## 2. Users

- This collection stores the personal data of users who register with our application
- It consists of the following columns
  - Username
  - Password
  - Email
  - Address
  - mobile
- Data of users are inserted and updated into this collection, when a new user registers and an existing user attempts to update his/her personal information

Below is a snapshot of the 'users' collection

```
"_id" : ObjectId("5af00777b3080225f876c091"),
"username" : "nish",
"password" : "1234",
"mailid" : "nishchay03@gmail.com",
"address" : "201, South 4th street, apt 534",
"mobile" : "6692137182"
```

## 3. Trends

- This collection stores the currency trend data for the cryptocurrencies offered by the application for trade, namely Bitcoin, Litecoin, Ethereum, Ripple, Stellar & Cardano.
- It consists of the following columns
  - Currency\_Name
  - Date (MM/DD/YY)



- txVolume
- txCount
- Marketcap
- Price
- exchangeVolume
- Fees
- generatedCoins
- This collection is mainly used for data visualization of currency trends over daily, monthly and annual time periods
- The collection is updated with data from the year of 2015 for currencies that were available from that year of time.

Below is a snapshot of the 'trends' collection

```
> db.trends.find().pretty()
{
  "_id" : ObjectId("5af0db1ca7ab576b350ba9d9"),
  "Currency Name" : "Stellar",
  "date" : "5/5/18",
  "txVolume(USD)" : 72316.72543,
  "txCount" : 1448,
  "marketcap(USD)" : "$8,008,540,000.00 ",
  "price(USD)" : 0.431216,
  "exchangeVolume(USD)" : 44096200,
  "generatedCoins" : 0,
  "fees" : 0.01726
}
{
  "_id" : ObjectId("5af0db1ca7ab576b350ba9da"),
  "Currency Name" : "Stellar",
  "date" : "5/3/18",
  "txVolume(USD)" : 117795.9461,
  "txCount" : 882,
  "marketcap(USD)" : "$8,153,360,000.00 ",
  "price(USD)" : 0.439016,
  "exchangeVolume(USD)" : 83481300,
  "generatedCoins" : 0,
  "fees" : 0.01259
}
{
  "_id" : ObjectId("5af0db1ca7ab576b350ba9db"),
  "Currency Name" : "Stellar",
  "date" : "5/1/18",
  "txVolume(USD)" : 132207.2474,
  "txCount" : 3673,
  "marketcap(USD)" : "$7,908,680,000.00 ",
  "price(USD)" : 0.425853,
  "exchangeVolume(USD)" : 152906000,
  "generatedCoins" : 0,
  "fees" : 0.04018
}
```

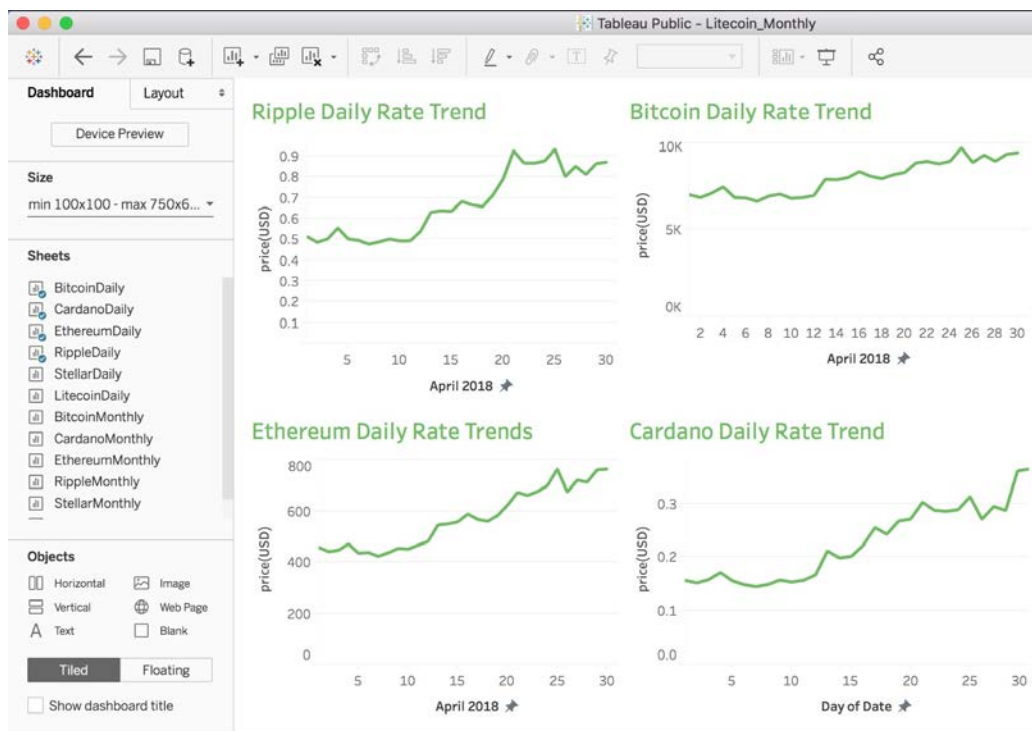
## DATA VISUALIZATION

The application offers users attractive charts to visualize trends of cryptocurrencies offered to aid in the trade of the currencies. Visualization is created using bar, line and area graphs to aid users to see trends more easily and clearly. We chose appropriate display media to visualize

trends for different periodicities, in order to have good span of control over a lot of information. Tableau Public is used to create different visualization media for the dashboard. It is a free software that connects to a data source to provide customized visualization of data.

By default, the dashboard page loads the daily price trends of the 6 cryptocurrencies. Users can choose to view other periodic trends such as monthly and yearly, by selecting the periodicity using 'Select Periodicity' option and clicking 'View' as in the below screenshot.

Below is a screenshot of data visualization created using Tableau Public



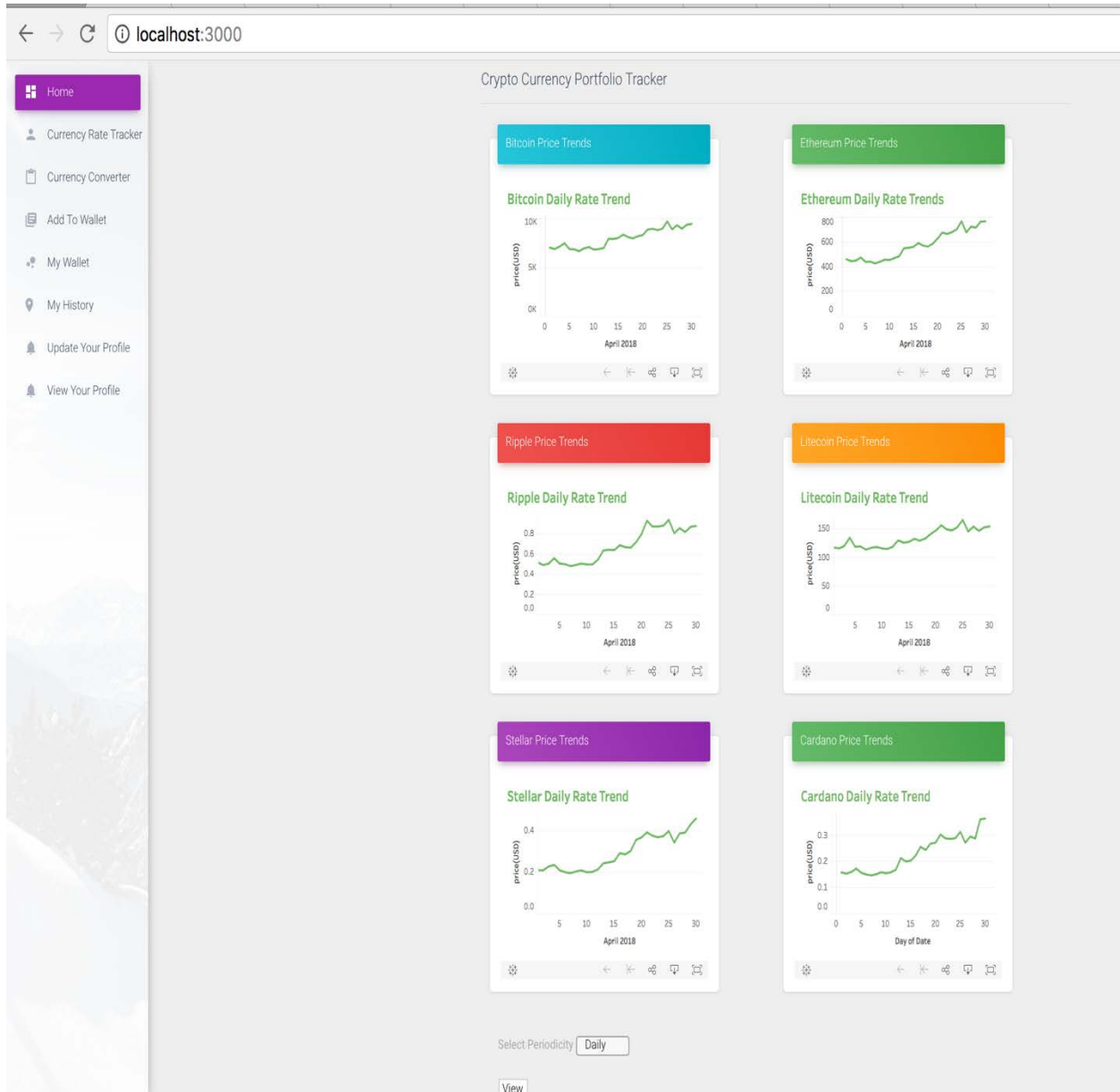
Organization design pattern, one of the UI design patterns has been chosen to create the HTML web page for displaying various graphs and charts.

## Features

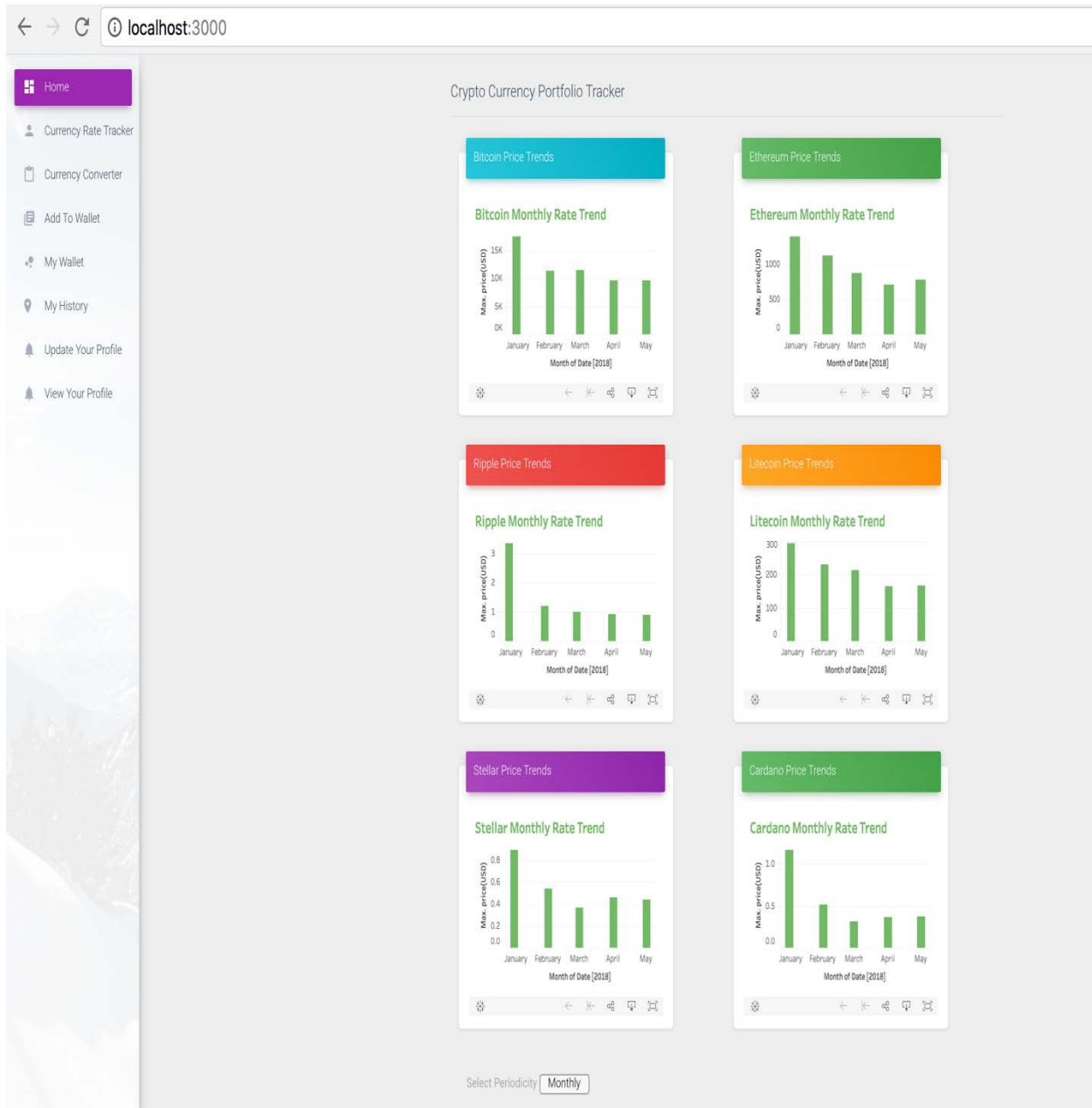
- “Seen all at Once” pattern is used to visualize all of the data at one glance in order for the user not to lose track of information, when he/she scrolls down the page.
- Bar graphs have been used to aid in proper visualization of data, since human eyes are sharper in detecting difference in the heights of the bars in the graph.
- We have also used line graphs which are more popular display media for visualizing price trends of stocks, cryptocurrencies etc.

Visualization for various periodicities using charts

## Dashboard for Daily Price Trends



## Dashboard for Monthly Price Trends



Dashboard for Yearly Price Trends

