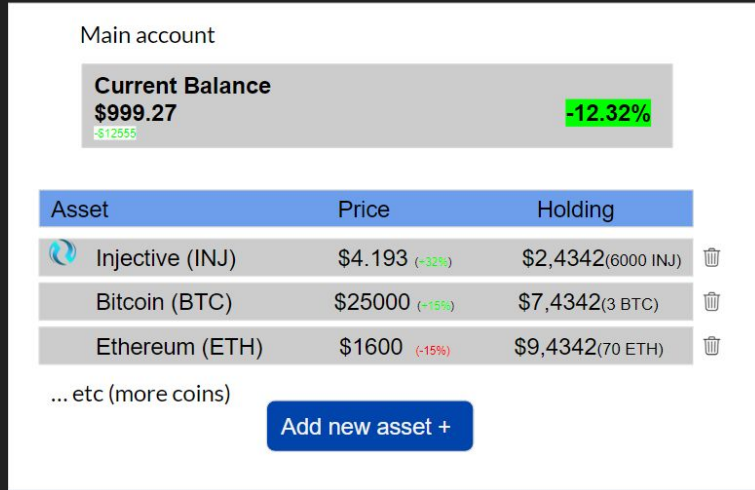


CoinSpy Documentation

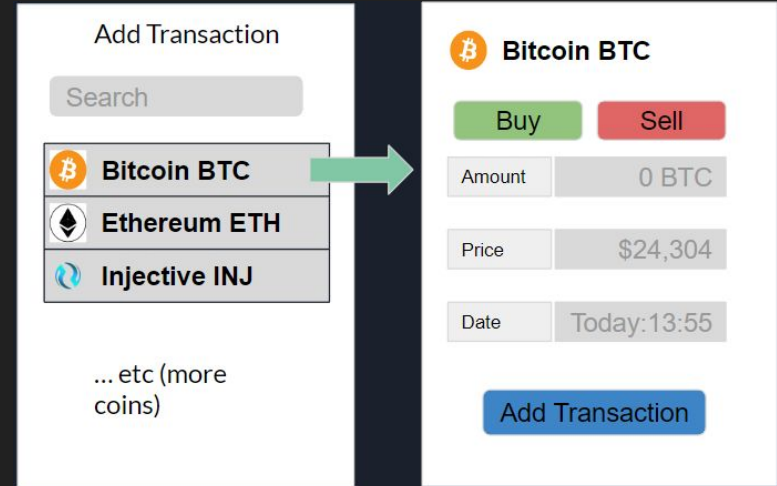
Vlad Borungel Final project

Wireframes



Main Portfolio page

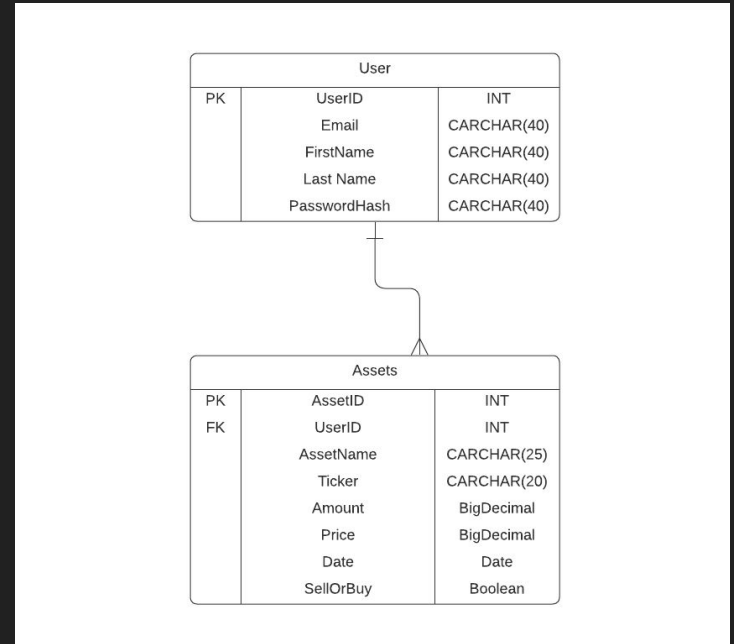
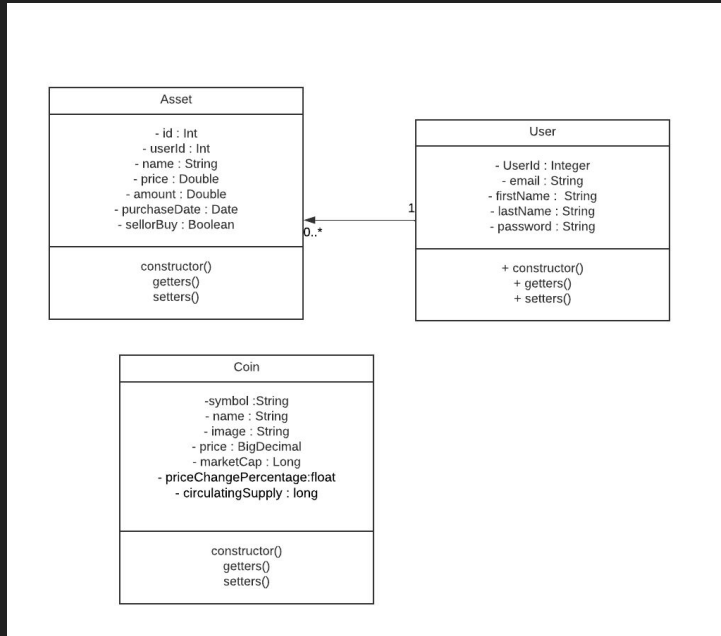
- Display total balance
- Display asset information



Adding Asset

- Add asset information using a “modal”

UML and EDR diagrams



Register and Login

Register and Login functionality was sourced from Code JAVA

<https://www.codejava.net/frameworks/spring-boot/user-registration-and-login-tutorial>

API output

The data from the API is received in the format showed.

Useful data needs to be manipulated and retrieved using JSON.

The data is received in the format
[`{coin}{coin}{coin}{coin}`], therefore we need to access the JSON Array first then the JSON Object

Coin price is retrieved in a similar way , there is no JSON array so you I had to access JSON Objects x2

```
[
  {
    "id": "bitcoin",
    "symbol": "btc",
    "name": "Bitcoin",
    "image": "https://assets.coingecko.com/coins/images/1/large/bitcoin.png?1547033579",
    "current_price": 28089,
    "market_cap": 543204940339,
    "market_cap_rank": 1,
    "fully_diluted_valuation": 590234334286,
    "total_volume": 20200997390,
    "high_24h": 28854,
    "low_24h": 27531,
    "price_change_24h": 266.91,
    "price_change_percentage_24h": 0.95933,
    "market_cap_change_24h": 5112843086,
    "market_cap_change_percentage_24h": 0.95018,
    "circulating_supply": 19326737,
    "total_supply": 21000000,
    "max_supply": 21000000,
    "ath": 69045,
    "ath_change_percentage": -59.17137,
    "ath_date": "2021-11-10T14:24:11.849Z",
```

```
{
  "bitcoin": {
    "usd": 28174
  }
}
```

API request call

```
//Public API URL generated using the website filters and variables
URL url = new URL(spec:"https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&order=market_cap_desc&per_page=100&page=1");
try (InputStream input = url.openStream()) {

    //reading the output
    InputStreamReader isr = new InputStreamReader(input);
    BufferedReader reader = new BufferedReader(isr);
    StringBuilder json = new StringBuilder();
    int c;
    while ((c = reader.read()) != -1) {
        json.append((char) c);
    }

    //the data is retrieved as an JSON array because it is in []
    JSONArray jsonArr = new JSONArray(json.toString());

    for (int i = 0; i < jsonArr.length(); i++) {
        //The arraylist stores JSON objects
        System.out.println(jsonArr.getJSONObject(i).get(name:"id"));

        String symbol = jsonArr.getJSONObject(i).get(name:"symbol").toString();
        String name = jsonArr.getJSONObject(i).get(name:"name").toString();
        String image = jsonArr.getJSONObject(i).get(name:"image").toString();
        String price = jsonArr.getJSONObject(i).get(name:"current_price").toString();
        String marketCap = jsonArr.getJSONObject(i).get(name:"market_cap").toString();
        String priceChangePercentage = jsonArr.getJSONObject(i).get(name:"price_change_percentage_24h").toString();
        String circulatingSupply = jsonArr.getJSONObject(i).get(name:"circulating_supply").toString();

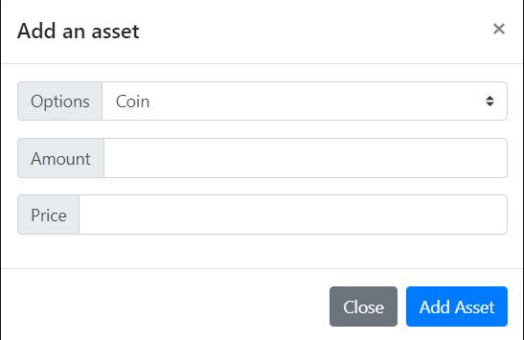
        Coin currentCoin = new Coin(symbol.toUpperCase(), name, image, price, marketCap, priceChangePercentage, circulatingSupply);
        coinList.add(currentCoin);
    }

    model.addAttribute(attributeName:"coinList", coinList);
}
```

Inputting data (Modal)

For inputting data from the user I decided to use bootstrap modal to have a popup come up and input the data.

Ajax is then used to retrieved this data from the modal and send a POST https request to the controller .



The image shows a Bootstrap modal window titled "Add an asset" with a close button (X) in the top right corner. The modal contains three input fields, each with a label on the left and a text input area on the right:

- The first field is labeled "Options" and contains the text "Coin" with a dropdown arrow icon on the right.
- The second field is labeled "Amount" and is an empty text input.
- The third field is labeled "Price" and is an empty text input.

At the bottom right of the modal, there are two buttons: a grey "Close" button and a blue "Add Asset" button.

Ajax function to send modal data and controller

```
$(document).ready(function() {  
  
    // when the "Add Asset" button is clicked  
    $("#addAssetBtn").on("click",function(event){  
        // get the form data  
        event.preventDefault();  
        modalData = {  
            "coin": $("#name").val(),  
            "amount": $("#amountName").val(),  
            "price": $("#priceName").val()  
        }  
  
        $("#addAssetBtn").prop("disabled", true);  
  
        console.log(modalData);  
  
        $.ajax({  
            type: "GET",  
            url: "/addAsset",  
            data: modalData,  
            success: function(result){  
                console.log("In success")  
                location.reload()  
            },  
            error: function(jqXHR, textStatus, errorThrown) {  
                // handle error response from the server  
                console.log("not In success")  
                console.log(textStatus + ": " + errorThrown);  
            }  
        })  
    })  
})
```

Data coming in as parameters in the controller function

```
@GetMapping("/addAsset")  
public ResponseEntity<> addAsset(@RequestParam("coin") int coin, @RequestParam("amount")  
    System.out.println(x:"asdasdasdsadsa");  
  
    Asset newcoin = new Asset();  
    if (coin == 1){  
        newcoin.setName(name:"bitcoin");  
    }else if(coin == 2){  
        newcoin.setName(name:"ethereum");  
    }else if(coin == 3){  
        newcoin.setName(name:"tether");  
    }else {  
        newcoin.setName(name:"bitcoin");  
    }  
  
    newcoin.setPrice(price);  
    newcoin.setAmount(amount);  
    newcoin.setEmail(principal.getName());  
  
    dao.addAssert(newcoin);  
  
    return new ResponseEntity<>(body:"asdas",HttpStatus.OK);  
}
```