

Test Plan Document

InvestiMapp

Contents

Unit Testing	3
1.1 Request Session.....	3
1.2 Add Stock Card.....	4
Use Case Testing	4
2.1 Login Use Case.....	4
2.2 Stock Manipulation Use Case.....	5
2.3 Dashboard Updates Use Case.....	6
2.4 Notifications Use Case.....	7
Acceptance Testing.....	7

1. Unit Testing

1.1 Test for requesting a new user session:

```
/*@Test
public void testCreateServer(){
    //user visits web application URL
    req = user.session;

    assertNotNull(req);
}
*/
http.createServer(function (req, res) {
    if (req.url.indexOf('/img') != -1) {
        var filePath = req.url.split('/img')[1];
        fs.readFile(__dirname + '/public/img' + filePath, function (err, data) {
            if (err) console.log(err);
            res.writeHead(200, {'Content-Type': 'image/svg+xml'});
            res.write(data);
            res.end();
        });
    } else if (req.url.indexOf('/js') != -1) {
        var filePath = req.url.split('/js')[1];
        fs.readFile(__dirname + '/public/js' + filePath, function (err, data) {
            if (err) console.log(err);
            res.writeHead(200, {'Content-Type': 'text/javascript'});
            res.write(data);
            res.end();
        });
    } else if (req.url.indexOf('/css') != -1) {
        var filePath = req.url.split('/css')[1];
        fs.readFile(__dirname + '/public/css' + filePath, function (err, data) {
            if (err) console.log(err);
            res.writeHead(200, {'Content-Type': 'text/css'});
            res.write(data);
            res.end();
        });
    } else {
        fs.readFile(__dirname + '/public/index.html', function (err, data) {
            if (err) console.log(err);
            res.writeHead(200, {'Content-Type': 'text/html'});
            res.write(data);
            res.end();
        });
    }
})
```

1.2 Test for adding a new stock card:

```
/*
@Test
public void addStockTEST(res, rej){
    assertNotNull(res);
    assertNull(rej);
}
*/
function promiseAddStock() {

    let fetchStocks = new Promise((resolve, reject) => {
        symbol = $("#modalStockSymbol").val();

        const stocksData = fetch("https://api.robinhood.com/quotes/?symbols=" + symbol);
        stocksData
            .then(data => data.json())
            .then(data => {
                askPrice = data.results[0]["ask_price"];
                equity = askPrice * userSharesOwned;
                totalInvested = userSharesOwned * userPricePerShare;
                saleValue = equity - totalInvested;
                resolve();
            })
            .catch(err => console.log(err));
    });
};
```

2. Use Case Testing

2.1 Login Use Case

Actor: User

Main Scenario:

- 1) User clicks "Login"
- 2) User fills out form with credentials
- 3) User is signed into their account

Alternatives:

- 2a) User clicks "Login"
- 2b) User fills out form with incorrect credentials
- 2c) User is returned to form with a message stating they could not be signed in.
- 3a) User clicks "Create Account"
- 3b) User fills out form with information
- 3c) User verifies an email sent to the address provided
- 3d) User clicks "Login"

Test Situations:

- 1) User successfully signs in with their credentials
- 2) User fails sign-in with invalid credentials
- 3) User creates a new account

Test Coverage:

- Base: Number of main and alt scenarios = 3
- Test Situations cover all 3 cases
- 100% coverage of use case

Status: Not implemented (no screenshot)

2.2 Stock Manipulation Use Case

Actor: User

Main Scenario:

- 1) User clicks “Add/Remove Stock”
- 2) User fills out a form and clicks “Submit”
- 3) The page will redirect to their dashboard with their updated stock ownership.

Alternatives:

- 2a) User clicks “Cancel” or closes out the form window
- 2b) User is brought back to their dashboard, with no changes to stock ownership
- 3a) User had entered an invalid stock symbol and/or quantity
- 3b) User is brought back to their dashboard, with not change to stock ownership

Test Situations:

- 1) User successfully adds a stock
- 2) User successfully removes a stock
- 3) User begins the add/remove process, but terminates it before completion
- 4) User attempts to add an invalid stock
- 5) User attempts to remove a stock they don’t have
- 6) User attempts to add/remove an invalid number of stock (i.e., -1)

Test Coverage:

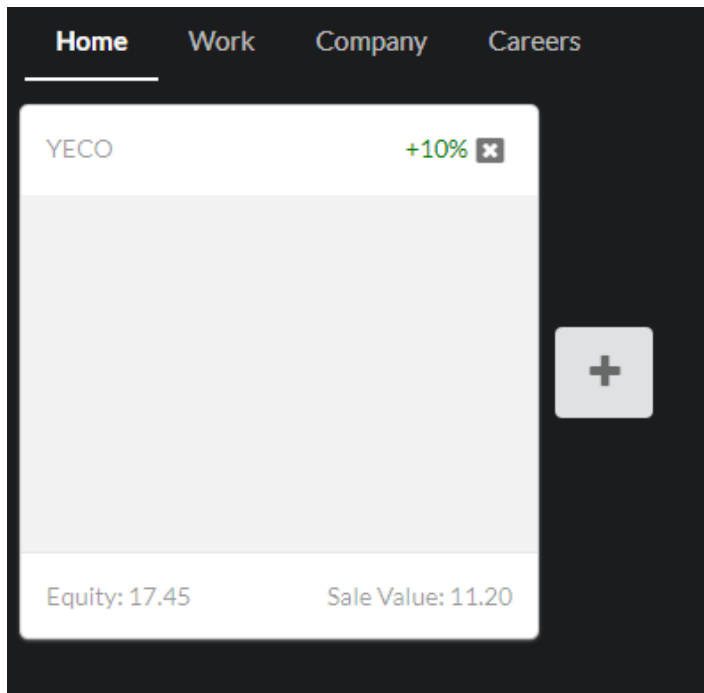
- Base: Number of main and alt scenarios = 3
- Test Situations cover all 3 cases
- 100% coverage of use case

Status: Implemented (screenshots below)

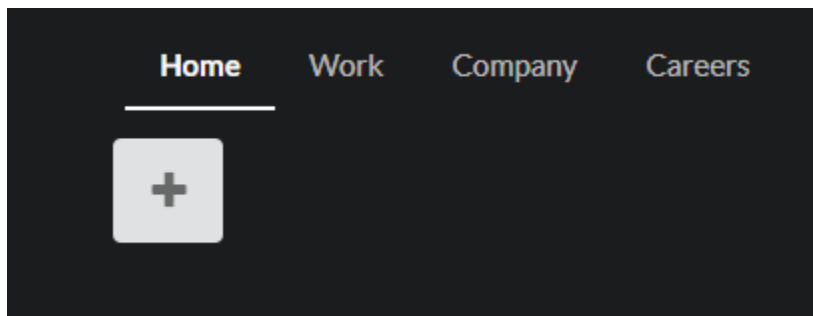
1) Adding a Stock

A screenshot of a web form for adding or removing stock. The form has three input fields: the first contains 'YECO', the second contains '5', and the third contains '1.25'. Below the input fields is a light gray bar. At the bottom right of the form are two buttons: 'Cancel' and 'OK'.

2) Stock has been added



3) Stock was removed



2.3 Dashboard Updates Use Case

Actor: User

Main Scenario:

- 1) User signs into, or is redirected to their homepage.
- 2) User views the most recent information on stock prices, and how they compare to price of purchase

Alternatives:

- 2a) User has poor/no internet connection, and the most recent prices could not be fetched

Test Situations:

- 1) User successfully views dashboard
- 2) User unsuccessfully views dashboard

Test Coverage:

- Base: Number of main and alt scenarios = 2

- Test Situations cover all 2 cases
- 100% coverage of use case

Status: Not yet implemented (no screenshots)

2.4 Notifications Use Case

Actor: System

Main Scenario:

- 1) Stock Market data is updated and loaded in the system
- 2) System sends notification based on user preferences

Alternatives:

- 2a) User has notifications turned off
- 3a) User has invalid contact methods in the system

Test Situations:

- 1) User with valid contact methods in system, and notifications turned on
- 2) User with invalid contact methods in system, and notifications turned off
- 3) User with notifications turned off

Test Coverage:

- Base: Number of main and alt scenarios = 3
- Test Situations cover all 3 cases
- 100% coverage of use case

Status: Not yet implemented (no screenshots)

3. Acceptance Testing

Current Status: Incomplete

The system is not yet complete, thus black-box testing is not expected to be verified. However, we do have about half of our functionality, which can be seen in the screenshot below:

