Stock​​ Application

Test​​ Case ​​Plan

Derek​​ Brown,​​Alyssa ​​Drohan,​​Phillip​​ Gil-Perea,​​Isaiah ​​Johnson, Russell​​ Quao,​​Isaac​​ Silvious

December ​​9,​​2017

Table​​ of ​​Contents

Introduction……………………………………………………………………..pg​​3

Constraints………………………………………………………….…………..pg​​3

Test​ ​Environment​​&​​Tools………………………………………….…………..pg​​3

Approach………………………………………………………………………..pg​​4

Roles…………………………………………………………………………….pg​​4

Schedule………………………………………………………………………...pg​​4

References……………………………………………………………………....pg​​4

Test​​Case​​ID​​TC001​​ Register...…………..………………………….……….....pg​​5

Test​​Case​​ID​​TC002 ​​Login……….…………………………………………….pg​​7

Test​​Case​​ID​​TC003 ​​Logout...………………………………………………….pg​​8

Test​​Case​​ID​​TC004 ​​Search​​ for ​​Stocks​​………………………………………..pg​​9

Test​​Case​​ID​​TC005 ​​Reset​ ​Password………………………………………....pg​​10

Test​​Case​​ID​​TC006 ​​Start​ ​Application……………………………………..…pg​​11

Test​​Case​​ID​​TC007​​Add​​ to​ ​Stock​ ​Owned ​​List……………………………….pg​​12

**Introduction**

The​​ purpose ​​of ​​testing​ ​this ​​stock ​​application ​​is ​​for ​​a​​ user​​ to​​ be ​​able ​​to ​​view,​​track​​ and ​​add​ ​to their ​​stock ​​portfolio.​​The ​​user ​​can ​​add​ ​stocks ​​to​ ​their​​ stocks​​ owned​​ list​​ and ​​watch list​​ to​​ track their​​ performance.

**Constraints**

One​​constraint​​that​​we​​have​​is​​that​​the​​members​​of​​the​​Quality​​Control​​Team​​don’t​​have extensive​​programming​​experience​​with​​Node.js​​framework.​​Only​​two​​of​​the​​members,​​Isaac and​​Jarrett​​will​​be​​testing​​the​​Node.js​​framework.​​The​​rest​​of​​the​​team​​members​​will​​be​​testing the​​user​​inputs​​for​​the​​rest​​of​​the​​use​​cases.

**Test​​Items​​&​​Environment**

The​​bug​​tracking​​software​​we​​decided​​to​​use​​is​​called​​Axosoft.​​It​​is​​an​​easy​​to​​use​​website (https://bridgewater.axosoft.com)​​that​​allows​​the​​Quality​​Control​​Team​​and​​the​​Application​​Development team​​to​​easily​​track​​the​​bugs​​that​​are​​currently​​in​​the​​code.​​We​​are​​using​​Webstorm​​2017​​2.4​​for​​our​​IDE. We​​will​​also​​be​​using​​Node.js​​found​​at​​(​<https://nodejs.org/en/>​).​​We​​will​​be​​using​​version​​8.9.3.​​We​​will also​​be​​using​​version​​npm​​3.5.2

Like​​previous​​years,​​we​​will​​rely​​on​​Axosoft​​for​​testing​​purposes.​​Tutorial​​on​​Axosoft:​​The​​Axosoft​​site is​​fairly​​simple​​to​​use​​and​​easy​​to​​understand.​​In​​the​​Organize​​Panel,​​there​​are​​four​​sections;​​Projects, Releases,​​Users​​&​​Teams,​​and​​Customers.​​The​​Projects​​panel​​shows​​the​​projects​​that​​are​​currently​​being worked​​on.​​The​​user​​can​​add​​or​​edit​​their​​projects​​by​​clicking​​on​​the​​“Add”​​or​​“Edit”​​buttons​​at​​the​​top​​of the​​panel.​​When​​the​​user​​clicks​​the​​“Add”​​button​​a​​box​​shows​​up​​that​​asks​​you​​to​​enter​​the​​name​​of​​your new​​project.​​The​​Release​​panel​​shows​​the​​projects​​you​​have​​access​​to.​​In​​the​​Users​​&​​Teams​​panel,​​users can​​add​​users​​and​​also​​create​​teams​​within​​the​​panel.​​To​​add​​or​​edit​​these​​users,​​click​​on​​the​​“Add”​​or “Edit”​​buttons​​at​​the​​top​​of​​the​​panel.​​When​​adding​​a​​user,​​you​​simply​​enter​​the​​user's​​e-mail​​address​​into the​​box​​and​​Axosoft​​sends​​them​​an​​email​​explaining​​to​​them​​how​​to​​confirm​​that​​they​​have​​been​​added. In​​the​​Customer's​​panel,​​you​​can​​add​​or​​edit​​customers​​but​​clicking​​the​​“Add”​​or​​“Edit”​​buttons​​at​​the​​top of​​the​​panel.​​When​​adding​​a​​customer,​​it​​will​​ask​​you​​to​​type​​the​​company/customer​​name​​and​​also​​a URL.​​The​​main​​panel​​on​​the​​page​​is​​where​​you​​can​​add​​bugs​​and​​you​​add​​them​​by​​clicking​​the​​“Add” button​​at​​the​​top​​of​​the​​panel.​​When​​you​​click​​the​​“Add”​​button,​​it​​will​​show​​you​​different​​options​​you can​​set.​​You​​add​​a​​title,​​assign​​due​​dates​​for​​when​​you​​want​​the​​bugs​​to​​be​​tested​​by​​and​​assign​​who should​​test​​it.​​You​​can​​also​​assign​​priority​​to​​the​​bugs​​and​​the​​severity​​of​​the​​bugs.

**Approach**

The​​type​​of​​testing​​we​​are​​using​​for​​this​​stock​​application​​is​​manual​​black​​box​​testing.​​Black​​box​​testing is​​used​​to​​test​​the​​input/output​​behavior​​of​​each​​test​​case.​​All​​of​​the​​test​​cases​​are​​found​​below​​for​​the stock​​application.

**Roles**

Jarrett​​Horton:​​Manager​​of​​the​​team​​and​​in​​charge​​of​​writing​​up​​everything​​but​​the​​test​​cases​​for​​this document​​(Introduction,​​Constraints,​​Test​​Items​​&​​Environment,​​Approach,​​Roles,​​Schedule,​​and References).​​Jarrett’s​​role​​in​​the​​testing​​process​​is​​to​​manage​​the​​team​​and​​assign​​code​​to​​members​​to​​test.

Isaac​​Silvious:​​Team​​member​​in​​charge​​of​​writing​​the​​other​​half​​of​​the​​test​​cases​​for​​this​​document (Splash​​Screen,​​Create​​Account,​​Login,​​View​​Main​​Screen,​​Load​​Excursion,​​Start​​Recording​​Route,​​and Add​​Observation).​​Paul’s​​role​​in​​the​​testing​​process​​is​​to​​test​​the​​code​​that​​was​​passed​​down​​from​​the​​App development​​team.

Isaiah​​Johnson:​​Team​​member​​in​​charge​​of​​writing​​half​​of​​the​​test​​cases​​for​​this​​document​​(View Observation,​​Edit​​Observation,​​Stop​​Recording​​Route,​​Edit​​Excursion,​​Save​​Excursion,​​Logout,​​and Exit).​​He​​was​​also​​responsible​​for​​finding​​a​​free​​bug​​tracking​​software.​​Brandon’s​​role​​in​​the​​testing process

Russell​​Quao:Team​​Member​​in​​charge​​of​​writing/formatting​​Use​​Cases(calculate​​price​​alert,​​calculate trending​​stocks,​​search​​for​​stocks).​​Participated​​in​​designing​​the​​search​​Gui​​for​​the​​application.

Derek​​Brown:Team​​member​​in​​charge​​of​​creating​​all​​of​​the​​Gui’s​​and​​the​​Add​​to​​Stock​​Owned​​List, Remove​​from​​Stock​​Owned​​List

Phillip​​Gil-Perea:​​Responsible​​for​​use​​cases.​​(Edit​​user​​settings,​​Log​​out,​​Forgot​​Password)

Alyssa​​Drohan:​​Team​​member​​is​​responsible​​for​​the​​part​​of​​the​​black​​box​​testing.

**Schedule**

Schedule​​can​​be​​found​​at

([​https://github.com/StockAppSoftwareEng450/Stock-App/blob/master/rad.pdf](https://github.com/StockAppSoftwareEng450/Stock-App/blob/master/rad.pdf)​)​​on​​page​​31.

​**References**​​

The​​Requirements​​Analysis​​Document​​(RAD)​​is​​available​​on​​the​​stock​​application​​website​​and​​other project​​information:​​(​<https://github.com/StockAppSoftwareEng450/Stock-App/blob/master/rad.pdf>​).​​​For more​​information​​on​​how​​to​​use​​Axosoft​​link:​​(https://www.youtube.com/watch?v=32x-UseZRyg).

**Test​​Cases**

**Test​​Case​​ID​​TC001​​“Register​​Account”**

**Summary:​​​**Verify​​a​​user​​can​​create​​an​​account​​as​​long​​as​​the​​user​​uses​​a​​valid​​email​​thatis​​not​​in​​the​​database​​and​​the​​Password​​and​​Confirm​​Password​​fields​​are​​the​​same.

**Prerequisite:​​​**Application​​has​​the​​Register​​screen​​loaded​​and​​a​​account​​with​​thefollowing​​data​​has​​been​​created.

First​​Name:​​John

Last​​Name:​​Doe

Email:​​Johndoe@gmail.com

Password:​​password1

**Instructions:​​​**For​​each​​test,​​enter​​the​​following​​data​​into​​the​​correct​​fields,​​then​​modifyeach​​field​​according​​to​​the​​test​​instructions.

First​​Name:​​Bob

Last​​Name:​​Dylon

Email:​​Bobdylon@gmail.com

Password:​​password2

Confirm​​Password:​​password2

**Test​​Data​​and​​Expected​​Result**

1. Change​​the​​Email​​field​​to​​“bobdylongmail.com”​​and​​press​​the​​Create​​Account​​button.
2. Change​​the​​Email​​field​​to​​“@gmail.com”​​and​​press​​the​​Create​​Account​​button.
3. Change​​the​​Email​​field​​to​​“[​bobdylon/@gmail.com](mailto:bobdylon/@gmail.com)​”​​and​​press​​the​​Create​​Account​​button.
4. Change​​the​​Email​​field​​to​​“bobdylon”@gmail.com”​​and​​press​​Create​​Account​​button.

RESULT:​​System​​should​​display​​a​​prompt​​saying​​the​​email​​is​​not​​valid.

1. Change​​the​​Email​​field​​to​​“Johndoe@gmail.com”​​and​​press​​the​​Create​​Account​​button.
2. Change​​the​​Email​​field​​to​​“​[admin@gmail.com](mailto:admin@gmail.com)​”​​and​​press​​the​​Create​​Account​​button.

RESULT:​​System​​should​​display​​a​​prompt​​saying​​the​​email​​is​​already​​in​​use.

1. Change​​the​​Password​​field​​to​​“password,”​​but​​leave​​the​​Confirm​​Password​​field “password2”​​and​​press​​the​​Create​​Account​​button.
2. Change​​the​​Password​​field​​to​​“password2,”​​but​​leave​​the​​Confirm​​Password​​field “password”​​and​​press​​the​​Create​​Account​​button.
3. Change​​the​​Password​​field​​to​​“Password!,”​​but​​leave​​the​​Confirm​​Password​​field “password!”​​and​​press​​the​​Create​​Account​​button.

RESULT:​​System​​should​​display​​a​​prompt​​saying​​the​​passwords​​do​​not​​match.

1. Change​​the​​First​​Name​​field​​to​​“Bob#”​​and​​press​​the​​Create​​Account​​button.
2. Change​​the​​First​​Name​​field​​to​​“Bob14”​​and​​press​​the​​Create​​Account​​button.
3. Change​​the​​First​​Name​​field​​to​​“Bob\*”​​and​​press​​the​​Create​​Account​​button.
4. Change​​the​​First​​Name​​field​​to​​“Bob​​”​​and​​press​​the​​Create​​Account​​button.
5. Change​​the​​First​​Name​​field​​to​​“​​Dylon”​​and​​press​​the​​Create​​Account​​button.

RESULT:​​System​​should​​display​​a​​prompt​​saying​​names​​may​​only​​contain​​alphabetic​​symbols.

1. Change​​the​​Last​​Name​​field​​to​​“Dylon1”​​and​​press​​the​​Create​​Account​​button.
2. Change​​the​​Last​​Name​​field​​to​​“Dylon#”​​and​​press​​the​​Create​​Account​​button.
3. Change​​the​​Last​​Name​​field​​to​​“Dylon\*”​​and​​press​​the​​Create​​Account​​button. Change​​the​​Last​​Name​​field​​to​​“Dylon​​”​​and​​press​​the​​Create​​Account​​button.
4. Change​​the​​Last​​Name​​field​​to​​“​​Dylon”​​and​​press​​the​​Create​​Account​​button.

RESULT:​​System​​should​​display​​a​​prompt​​saying​​names​​may​​only​​contain​​alphabetic​​symbols.

1. Keep​​all​​information​​correct​​and​​press​​the​​Create​​Account​​button. RESULT:​​System​​should​​create​​an​​account​​and​​redirect​​to​​the​​login​​screen.

**Test​​Case​​ID​​TC002​​“Login​​Screen”**

**​​Summary​​​**Verify​​a​​user​​can​​log​​in.

**Prerequisite​​​**Application​​has​​the​​Login​​screen​​loaded​​and​​a​​account​​with​​the​​followingdata​​has​​been​​created.

First​​Name:​​Bob

Last​​Name:​​Dylon

Email:​​Bobdylon@gmail.com

Password:​​password2

**Instructions​​​**Perform​​the​​following​​instructions

**Test​​Data​​and​​Expected​​Result**

1. Enter​​“Bobdylon@gmail.com​​”​​in​​the​​Email​​field​​and​​“password1”​​in​​the​​password​​field and​​press​​the​​Login​​button.

RESULT:​​System​​should​​display​​an​​error​​message​​stating​​the​​username​​or​​password​​is​​incorrect.

1. Enter​​“fakeemail@hotmail.com​​”​​in​​the​​Email​​field​​and​​“password2”​​in​​the​​password field​​and​​press​​the​​Login​​button.

RESULT:​​System​​should​​display​​an​​error​​message​​stating​​the​​username​​or​​password​​is​​incorrect.

1. Enter​​“Bobdylon@gmail.com​​”​​in​​the​​Email​​field​​and​​“password2”​​in​​the​​password​​field and​​press​​the​​Login​​button.

RESULT:​​System​​should​​log​​into​​John’s​​account​​and​​the​​portfolio​​use​​case​​should​​be​​displayed.

**Test​​Case​​ID​​TC003​​“Log​​out”**

**​​Summary​​​**Verify​​that​​the​​user​​can​​log​​out.

**Prerequisite​​​**The​​user​​is​​logged​​in​​and​​on​​the​​main​​screen.

**Instructions​​​**Perform​​the​​following​​instructions.

**Test​​Data​​and​​Expected​​Result**

1. On​​the​​navigation​​slider,​​click​​the​​logout​​button

**Test​​Case​​ID​​TC004​​“Search​​for​​stock”**

**​​Summary​​​**Verify​​that​​the​​search​​function​​works.

**Prerequisite​​​**The​​user​​is​​logged​​in​​and​​on​​the​​main​​screen.

**Instructions​​​**Perform​​the​​following​​instructions.

**Test​​Data​​and​​Expected​​Result**

1. Enter​​“AAPL!”​​into​​the​​search​​field. RESULT:​​The​​system​​should​​display​​no​​stocks
2. Enter​​“AAPL”​​into​​the​​search​​field.

RESULT:​​The​​system​​should​​display​​“Apple​​Inc.”

**Test​​Case​​ID​​TC005​​“Reset​​Password”**

**​​Summary​​​**Creating​​a​​new​​password​​if​​the​​user​​gets​​locked​​out​​of​​their​​account.

**Prerequisite​​​**The​​user​​is​​connected​​to​​the​​internet.

**Instructions​​​**Perform​​the​​following​​instructions.

**Test​​Data​​and​​Expected​​Result**

1. User​​observes​​reset​​password​​page​​which​​includes​​email​​address​​field. RESULT:​​The​​user​​enters​​email​​address

​​​2.​​​Password​​Server​​sends​​user​​email​​with​​reset​​password​​link.

RESULT:​​User​​clicks​​email​​and​​goes​​to​​reset​​Password.”

​​​3.​​​User​​observes​​change​​password​​page.

RESULT:​​User​​changes​​password​​and​​presses​​“save”.​​Passwords​​are​​checked​​to​​make​​sure​​they are​​the​​same,​​and​​the​​password​​is​​updated​​in​​the​​database

**Test​​Case​​ID​​TC006​​“Start​​Application”**

**​​Summary​​​**Will​​determine​​if​​the​​user​​has​​logged​​on​​to​​the​​website​​previously.

**Prerequisite​​​**The​​user​​is​​connected​​to​​the​​internet​​and​​has​​access​​to​​the​​firebase​​database.

**Instructions​​​**Perform​​the​​following​​instructions.

**Test​​Data​​and​​Expected​​Result**

1. Application​​checks​​browser​​for​​the​​user’s​​credentials​​through​​the​​browser’s​​cookies.

RESULT:​​The​​system​​is​​able​​to​​see​​if​​the​​user​​has​​been​​logged​​in​​to​​the​​website​​previously​​or not.

​​​2.​​​The​​user​​clears​​the​​cache​​of​​the​​browser​​to​​see​​if​​the​​website​​will​​stayed​​logged​​in. RESULT:​​The​​system​​is​​able​​to​​see​​that​​the​​user​​has​​been​​logged​​into​​that​​device​​and​​logs​​on.”

​​​3.​​​The​​user​​clears​​the​​history​​of​​the​​device​​to​​see​​if​​the​​website​​will​​remember​​the​​user​​and stay​​logged​​in.

RESULT:​​The​​system​​is​​able​​to​​save​​the​​user’s​​data​​inside​​their​​database​​and​​the​​user​​is​​able​​to stay​​logged​​in.

​​​4.​​​The​​user​​clears​​the​​password​​of​​the​​device​​to​​see​​if​​the​​website​​will​​remember​​the​​user and​​stay​​logged​​in.

RESULT:​​The​​system​​is​​not​​able​​to​​save​​the​​user’s​​data​​inside​​their​​database​​and​​the​​user​​is​​not able​​to​​stay​​logged​​in.

**Test​​Case​​ID​​TC007​​“Add​​to​​Stock​​Owned​​List”**

**​​Summary​​​**Verify​​a​​user​​can​​add​​a​​stock​​to​​their​​owned​​list.

**Prerequisite​​​**Application​​has​​to​​be​​connected​​to​​the​​firebase​​database,​​and​​be​​connectedto​​the​​internet

**Instructions​​​**Perform​​the​​following​​instructions

**Test​​Data​​and​​Expected​​Result**

* **Purchase​​Price**

1. User​​enters​​in​​‘abc’​​in​​purchase​​price​​location.

RESULT:​​System​​should​​display​​an​​error​​message​​stating​​the​​purchase​​price​​needs​​to​​be​​in decimal​​form

​​​2.​​​User​​enters​​in​​‘24’​​in​​purchase​​price​​location

RESULT:​​System​​should​​display​​an​​error​​message​​stating​​the​​purchase​​price​​needs​​to​​be​​in decimal​​form

​​​3.​​​User​​enters​​in​​‘24.abc’​​in​​purchase​​price​​location

RESULT:​​System​​should​​display​​an​​error​​message​​stating​​the​​purchase​​price​​needs​​to​​be​​in decimal​​form​​and​​only​​consisted​​of​​numbers