Play-to-Earn Financial Betting Application

Individual Coursework for Software Engineering and Product Management
(2767 words excluding Appendix)

Product Vision

Vision: Gain more accurate financial insight into capital markets			
Target Group: Anyone who is directly involved in the capital markets. e.g. Financial analysts, fund managers, traders Customers could also be investment banks wanting to implement competitive scoring in house.	**Needs:** Gain more qualitive data about market fluctuations through successful players within the space. Provides a more accurate data stream about sentiment within financial institutions about certain stocks and companies	**Product:** A play-to-earn online application that allows users to place predictions into market fluctuations. A betting application built on top of the prediction appliance to hedge your bets against other players with public leader boards of who currently has the most accurate predictions of the markets. The betting money can be deposited individually or taken the of next years bonus	**Business Goals:** Get access to insightful knowledge from some of the most successful players in the sector.
Competitors: Regular trading sites i.e. eToro. Any internal products firms may have for competitive prediction.	**Revenue Streams:** Application of the data provided by the prediction app for investment into the capital market. Commission taken on the betting app,	**Cost Factors:** Play to earn factor, development and upkeep of the application.	**Channels:** We will market to users as a competitive app to play against peers within the firm and industry whilst earning tokens for playing. Also give them access to the betting application

Project Management

The project management of this application has primarily been implemented in Trello. Within Trello, we have created a TODO checklist that can be slowly worked through to achieve the completion of this report. The TODO checklist gives us a more high-level management style for this project with some of the tasks on the checklist being substantial. Considering the time we have on this project our main objective is to deliver a MVP to a user that can be deployed on the cloud, this has lead to restructuring of our priorities to deliver a product that works, albeit skinny on features. For our app implementation, we have used different lists within Trello to define the different development stages. The most obvious is our backlog, the backlog contains a list of cards that have user stories and Epics within that can allow us to begin implementing the features. When we want to implement a feature we move a card from the backlog into our ‘Cards in Sprint’ list where we will implement the features all at once in a coding sprint. We have used the MOSCOW technique within Trello using labels to define the importance of a feature, with our ‘Must Have’s being the features that allow us to have a minimal viable product to launch. We have aimed to complete all must and should ‘haves’ within our development, however, our designs change over time, especially when taking an agile approach to our development. Furthermore, as we develop our features over time more requirements have arisen that have fractured the initial work plan with our initial time estimations being significantly off the mark. An example of this is the backend development on AWS, seeming as we have very little experience in AWS we have had to adapt our timeline to sufficiently implement the minimal required technological support needed for our app.

Requirements Gathering

As we have no ‘client’ I will be acting as our client for the requirements gathering. This has its benefits as I can personalise the application to my tastes however getting real time feedback, which is essential for agile software development, has proven a challenge with no client. The following functional and non-functional requirements are required to create a commercial product.

Functional Requirements:

- User shall be able to register and login to the system using email and password credentials
- Users must be able to create a profile containing relevant information
- User must be able to place multiple predictions on financial securities
- Users must be able to view their predictions
- The system must display leader boards of the most successful users predictions score
- System must calculate how well a user predicted a future price of security and add it as a score
- Users can place bets on their predictions for monetary gain
- Users can link their bank accounts to the application for withdrawal and depositing of monies
- User credentials must be stored in secure location

Non-Functional Requirements:

- System should be hosted on the cloud
- User predictions should be securely stored
- Application should run in under 2s
- Application framework should be scalable to add more features
- Application should be accessed from all web devices
- System backend should be capable to handling up to 1000 concurrent users on deployment
- System should be able to handle 10000 concurrent users by the end of the first year of deployment
- Application shouldn’t have geolocational restrictions to access

- System should be ready for integration into investment banking frameworks
- System should comply with any and all local laws regarding finance and the relation it has with our product

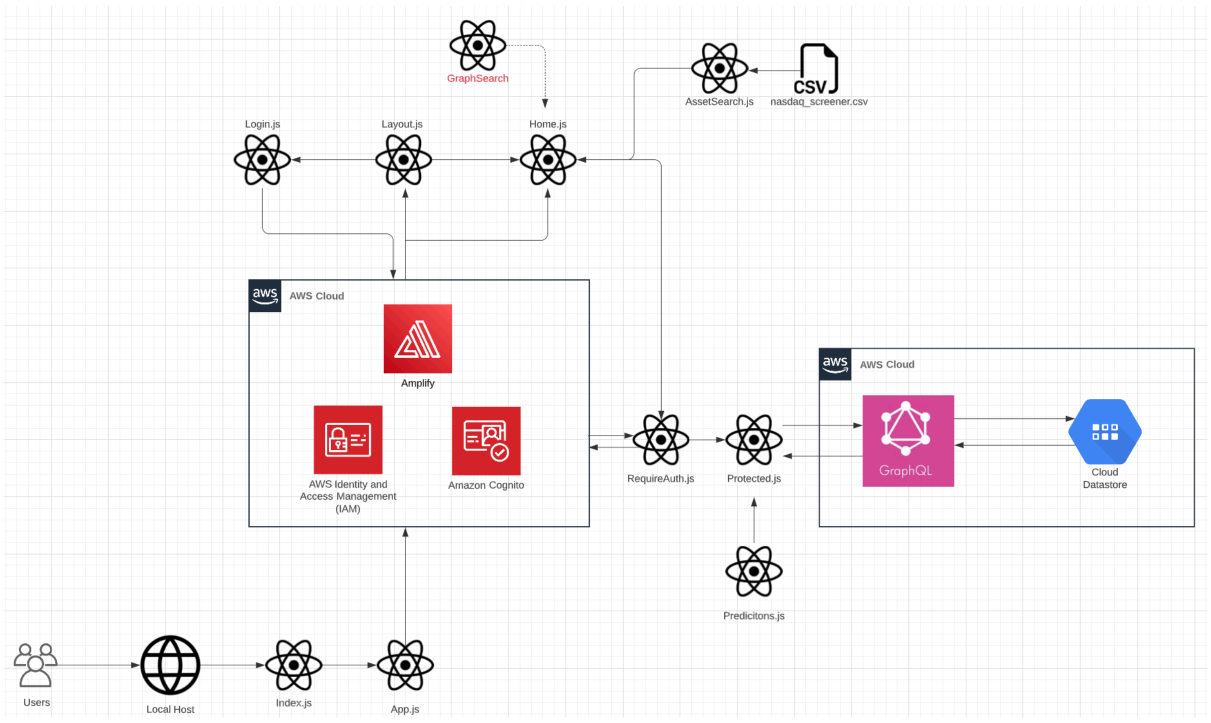
Our user stories are implemented in Trello as outlined in our project management section. Each card that contains user stories also contains an overview of the feature to be implemented, the business motivation behind the feature, time estimation of how long I think it will take to implement to the system and the MOSCOW importance scale. Here is the link to the Trello Document with all user stories:
https://trello.com/b/cp17wOrt/product-roadmap-ucl-wm
User Stories are also printed in appendix should link not work.

Analysis and Design

The analysis and design section of this report will focus on the data flow of the application. As we have chosen to use AWS as our backend with AWS Amplify as our front end this has allowed more automation within the AWS system and multiple features of AWS have been touched upon. Furthermore we use node and react.js for our front end and import ui-components from our Amplify folder stored locally.

Design Explanation

When we run our application our index.js file routes to the App function in App.js. Our index.js also configures our AWS exports through Amplify. The App.js file has two functions within it, MyRoutes and App. The App function calls the MyRoutes function whilst encapsulated in an Authenticator. The MyRoutes function is our main function that we use for routing to different react pages. To explain more clearly, users who are not logged in still have access to certain pages like the home page. However they don’t have access to a protected page which holds the users personal data, in this case their stock predictions. Layout.js defines our nav bar with different buttons that are used to navigate to different pages. If a user tries to select their profile page and they are not logged in they will be redirected to a login authentication page. We use Amplify’s Authentication Boilerplate code with some adjustments for our login and register functions. Once a user is logged in they have access to the My Profile page, Protected.js. This page interacts with our AWS server to retrieve information about their predictions they have made, the predictions are stored in a DataStore in AWS that is routed from a GraphQL table. A form is also available to input new predictions and once submitted it interacts with the DataStore to create new information.



Analysis of Framework
The framework we have designed is overkill for our product we have created, however it has been design with further expansion in mind. The AWS cloud framework and IAM management will allow us to have multiple users logging in and out simultaneously as well as accessing the data within the DataStore without a problem. Furthermore it allows each of our users to make predictions and update the DataStore with their own information accessed only by them. In retrospect my page flow and design could be cleaned up and will be in further design iterations however the current framework delivers an MVP to a client and it satisfies the major user requirements.

Design of Front End
The design of the front end was done in Figma, with my designs starting in low fidelity wireframes completed on my iPad then taking one design further into a high fidelity prototype of what I would like my end product to look like. All designs are annotated in the Appendix. My high fidelity design encapsulates what a polished finished product would look like with all my user stories completed which is highly unlikely. With my very limited front end development background there is a good chance I will begin to design the prototype according to one of my low fidelity designs with adaptability and portability in mind to change the design as more stories are developed. Figma links are:
https://www.figma.com/community/file/1192448357652023573
https://www.figma.com/community/file/1192448582236683542

Prototype and Implementation
In this section I am going to dive deeper into the content that is delivered in the prototype code. Our two main pages that we can access are the home page and the profile page, we will also touch upon the login and register forms as well. The home page delivers

available stocks a user can look up with scrolling list of over 8000 stocks, these stocks and their stock symbols are stored within a csv file. When the page loads the prototype parses these stocks into a table within a scrolling view. Furthermore, the user can search for a stock of their choosing, provided it is with the CSV file. This works by using hooks within the prototype and when a user begins to search for a stock the table automatically updates upon change showing only stocks that the user is searching for. The next section of the home page is not fully implemented, this is due to time restriction and primarily my stock API that I was using, the same API that provided the csv file of stocks available, requires that I pay for the subscription and when testing the prototype I ran out of credits. The stock API I was using was https://site.financialmodelingprep.com/developer/docs/stock-api/. This was an unfortunate development however being an agile development project I decided to move on and try different implementations. However, the code is still available in GraphSearch.js. Within the homepage, below the stock search, is an input box that requires the user to input a stock symbol of their choosing. When the user submits their stock symbol via the submit button the prototype redirects to the GraphSearch.js script. This script creates an API request to retrieve stock data about the stock they have inputted. Throughout my development process of this prototype I designed several different ways to implement this functionality, my initial design is to have buttons embedded in the scroll view table that when clicked would retrieve the information about the given stock. This however, I later learned, was not efficient on the client side as it would take time to load in every button and lag occurred. I opted out of this idea and instead chose our search field. My next challenge was to convert this data pulled from the API into a graph using react-graphs, I was unable to do this and decided it is not of high enough importance and would be implemented in the next iteration.

To access the profile page you must log in. The login forms were provided by Amplify and I used an amplify tutorial to import them into my prototype. The forms are directly liked to Amplify IAM and when logged in the user has access to the profile page. On the backend once a user is created AWS requires them to confirm their email they used and all users can be seen by the root user.

Once logged in the profile page displays a welcome message by pulling the current authenticated users data and placing it inside a hook. This hook then displays the message in the return section. The create prediction form was designed in Amplify console, AWS user interface to help with all things backend. I created a GraphQL table with the prediction information I wanted to store and subsequently linked this to a datastore. I chose to use datastore not a DynamoDB due to datastore allowing me to directly link a UI form with my GraphQL table. Furthermore access to the GraphQL API was through API keys. The ability to link both form and table allowed me to focus more on other aspects of the prototype and not worry about backend development and coding where I have no experience bar the AWS tutorial. I imported the form into my front end and added it to my Protected.js script. Once a user submits a prediction form the data is displayed on the other side of the page. This required me to create a request query to access the users data in the datastore and subsequently create a hook and parse that data into a table component on my UI.

Although this satisfies the minimal requirements there is error handling that needs to be implemented in further development. Currently on the subscription form a user can chose when they make the prediction. This should be the current time not a time of the users choosing. Furthermore there is no scoring system implemented for the predictions.
Links:

https://github.com/wmacpherson/amplify-app
https://main.d1lfplbomztxo6.amplifyapp.com/

Testing

My main testing strategy was to do unit testing as I developed the application. This is primarily due to my lack of experience in front end development and unit testing allowed me to test as I went along to make sure that the prototype had core functionality. This ranged from doing simple console.log commands for making sure variables had been correctly hooked, or data had successfully been retrieved from API's. You will find in my code there are instances where I have left console.log commands in to show developers of the status of a feature, in the Protected.js script I log the message from the authentication. In Graph_Search.js I log a 'running' message to confirm the submission of a form. Furthermore for external sources there is error handling within the code, for API's this is especially important as the user may be unaware that the API request has failed or that the application keeps crashing due to failed API requests. In the Predicitons.js script there are try and catch handling for requesting data from the GraphQL API and in the Graph_Search.js script there is also an if statement questioning if an error was thrown. I have not coded specific test for all my functions as there are obvious flaws that are due to other features not implemented yet, as an example we can't see the leaderboard as of yet or the scoring system so until that is implemented we cannot amend the issues with the predictions date system outlined in our Implementation section above.

When I have sufficiently programmed the application to a point nearing deployment, I would like to use the AWS provided Cypress tests on my application. As it is an Amplify app the Cypress testing system will allow me to design and run end-to-end tests from the Amplify console. This would allow me to test how the application runs as a whole and not just individual components that you get with unit testing. Cypress tests would allow me to test the login flow as well as the data flow from client to server when regarding predictions.

Future Iterations

Touched upon briefly in this report the future iterations of this project are expansive. Within Trello you can see future ideas to add the product, i.e. blockchain implementation of a play-to-earn scheme, NLP algorithms quantifying user scores and their prediction reasoning. In the short term however, there are key feature implementations that would take this product towards commercial deployment. In the next iteration I would like to add the scoring system to the product that allows users to get rewarded for their predictions, furthermore I would like to add the leaderboard to the main page that displays the public scores of certain users. I believe both the leaderboard and scoring system would provide the competitive aspect to the product that would attract customers to the application as outlined in my product vision. Moreover some basic clean-up of the code is required, i.e. distinction between components and pages, and completing the implementation of the graph search using a credible API. This brings me to the current cost of the application. As it is not deployed commercially AWS is not charging me for their services as of yet furthermore all my research on stock API's has told me to create a commercial product I will need to pay for that data.

Appendix

Trello

https://trello.com/b/cp17wOrt/product-roadmap-ucl-wm
User Stories as Text:
Login and Register Page
###Overview of the feature
Create an initial login and registration page for our application
###Business motivation
We need to give people access to our app and have the first level of security. We also need to identify who is using our app so we can begin to build a database of users and build more features later on
###User stories
As a customer, I want to be able to login so that I can gain access to the app
As a customer, I want to be able to register so that I can create an account personalised to me
As a customer, I want to add my firm, names, and position to my account so that it reflects my position when shown to the public
Time Estimation
4 days

View My Predictions
###Overview of the feature
View all my past and present predictions
###Business motivation
Allowing the users to look at their past predictions helps to gauge their own involvement in the application as well as reminding them of the predictions they have made
###User stories
As a customer, I want to see all my past predicitons so that I can see how well I have historically performed on certain asset classes
As a customer, I want to see the predictions I have made for the future so I can track how successful I think I am going to be.
Time Estimation
3 days

Make A Prediction
###Overview of the feature
Allow the user to make a prediction of the price of a security.
###Business motivation
This prediction mechanic is a core feature of the play to earn design of the application.
###User stories
As a customer, I want to make a prediction on the price of a security so that I can get a higher prediction score
As a customer, I want to give a reason for my prediction of a security price so I get extra credit when my predictions are correct.
Time Estimation
3 days

Create the Dashboard
###Overview of the feature

A dashboard is presented to the user when logged in. This is the home page which all our other features will grow from.
###Business motivation
The dashboard is the most crucial feature of our application. This is the home page and is the core of our application
###User stories
As a customer, I want to see a home page so that I can explore new features and view past actions
As a customer, I want to see financial markets so that I can think of different financial predictions I will make
As a customer, I want to see my prediction score so I know how accurate my predictions have been
Time Estimation
4 days

Update the Prediction Score
###Overview of the feature
Updates the prediction score of a user when a prediction of a future bet has reached maturity
###Business motivation
A feature that shows how well a user is performing can be used for the leaderboard
###User stories
As a user, I want my prediction score best represent my prediction performance so I can move up the public leaderboard
As a user, I want to show off my prediction score to peers so that I can boast about how good I am at predicting financial markets
Time Estimation
1 days

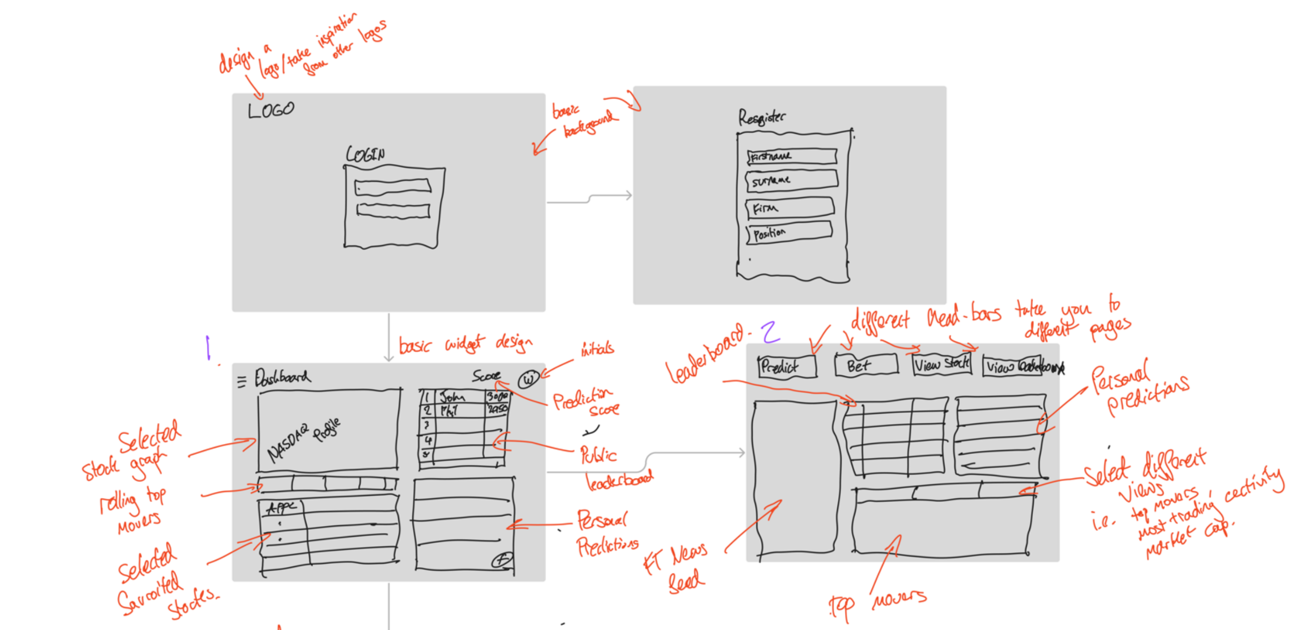
View the Leaderboard
###Overview of the feature
A leaderboard of all the users prediction scores
###Business motivation
A leaderboard adds a competitive aspect to the prediction mechanic motivating people to make predictions
###User stories
As a customer, I want to see who is the best at making market predictions so that I have a benchmark to beat.
Time Estimation
1/2 days

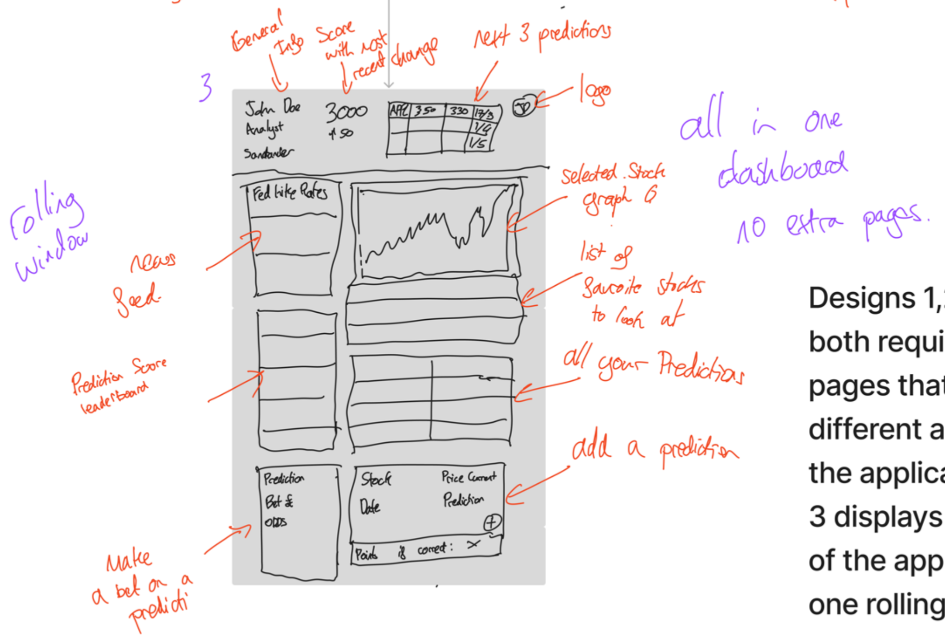
Change Account Details
###Overview of the feature
Allow access to change the account details of a user.
###Business motivation

If a user changes job, name, etc. the application should have the most up to date information about them. It should also allow users to set whether their information is public or private
###User stories
As a customer, I want to change my job title as I have changed jobs recently
As a customer, I want to change my viewing status to private so nobody can see my prediction score.
If I change my status to private I cannot make bets on predictions or see other people's prediction scores.
Time Estimation
2 days

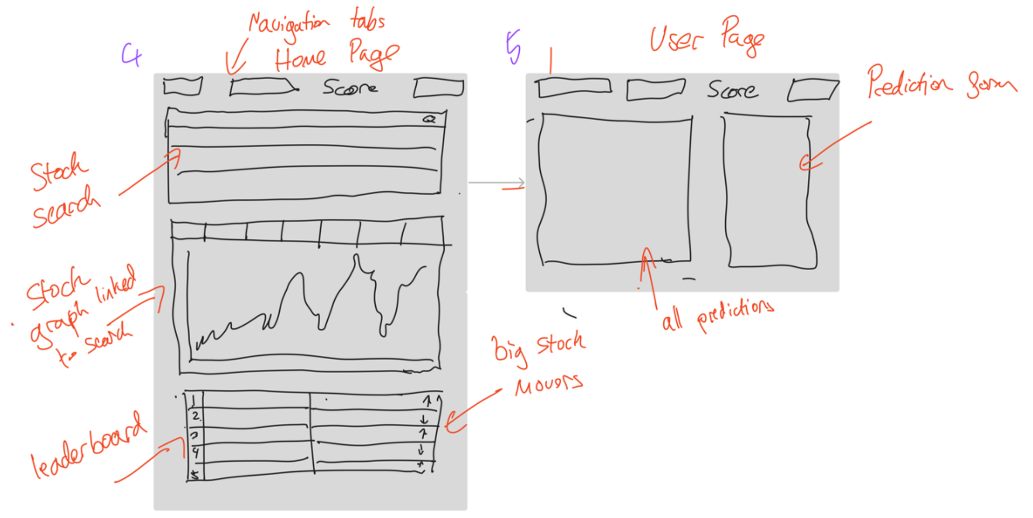
Link Bank Account
###Overview of the feature
Linking a bank account to the users account
###Business motivation
Adding a removing money for the betting mechanic
###User stories / Jobs To Be Done
As a user, I want to add a bank account so I can remove money from my account to fund bets and cash out on my bets
Time Estimation
2 days

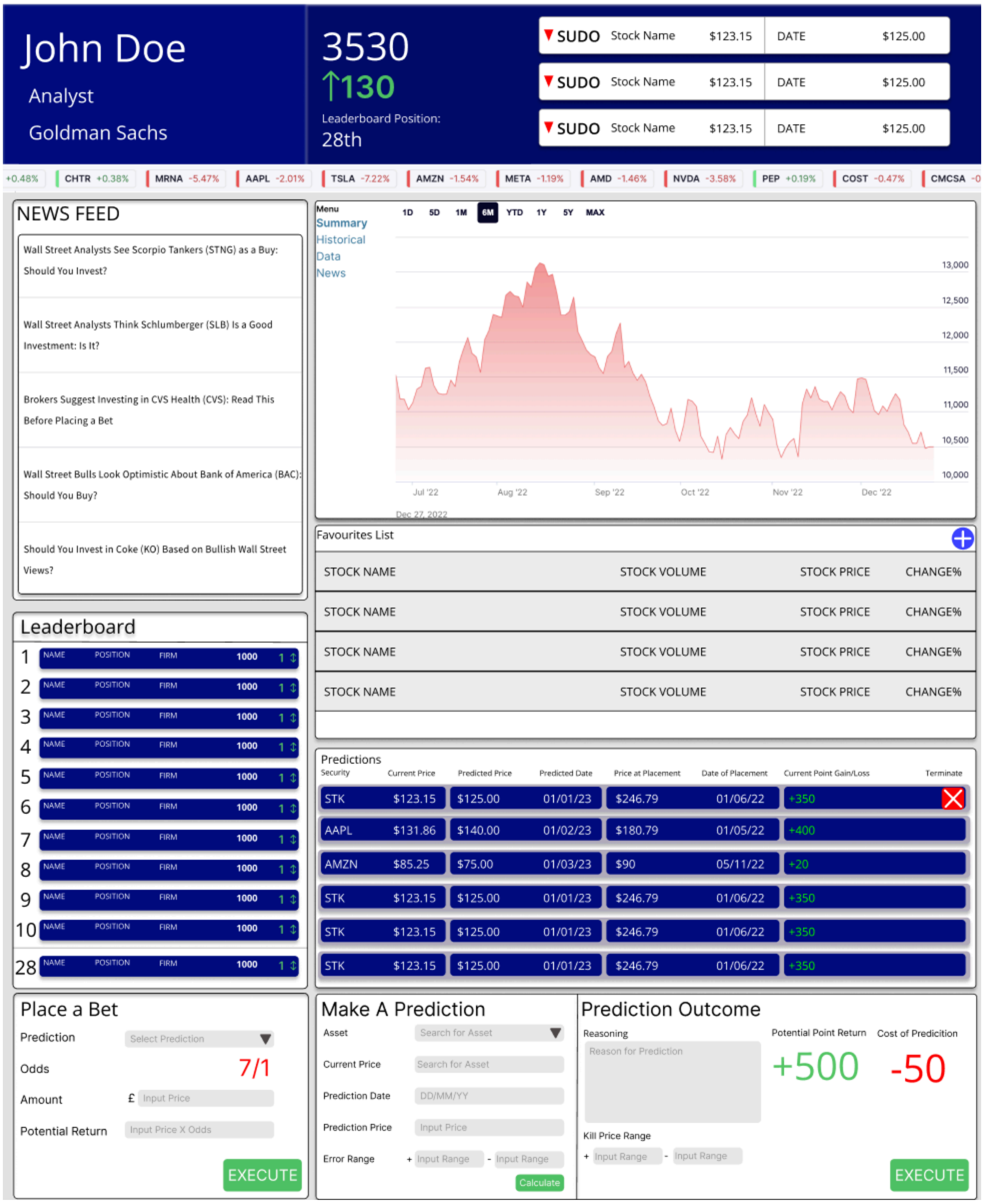
Place a bet on prediction
###Overview of the feature
Users can place real money bets on predictions they make based on automatically generated odds
###Business motivation
Allowing users to place bets creates a revenue stream for the application
###User stories / Jobs To Be Done
As a user, I want to be able to place a bet on a prediction I have made so that I can make money when my prediction is correct.
Time Estimation
3 days





Designs 1,2 and 4,5 both require multiple pages that display the different aspects of the application. Design 3 displays all aspects of the application on one rolling window




John Doe
Analyst
Goldman Sachs

3530
↑130
Leaderboard Position:
28th

▼ SUDO	Stock Name	$123.15	DATE	$125.00
▼ SUDO	Stock Name	$123.15	DATE	$125.00
▼ SUDO	Stock Name	$123.15	DATE	$125.00

+0.48% | CHTR +0.38% | MRNA -5.47% | AAPL -2.01% | TSLA -7.22% | AMZN -1.54% | META -1.19% | AMD -1.46% | NVDA -3.58% | PEP +0.19% | COST -0.47% | CMCSA -0

NEWS FEED

Wall Street Analysts See Scorpio Tankers (STNG) as a Buy: Should You Invest?

Wall Street Analysts Think Schlumberger (SLB) Is a Good Investment: Is It?

Brokers Suggest Investing in CVS Health (CVS): Read This Before Placing a Bet

Wall Street Bulls Look Optimistic About Bank of America (BAC): Should You Buy?

Should You Invest in Coke (KO) Based on Bullish Wall Street Views?

Favourites List

STOCK NAME	STOCK VOLUME	STOCK PRICE	CHANGE%
STOCK NAME	STOCK VOLUME	STOCK PRICE	CHANGE%
STOCK NAME	STOCK VOLUME	STOCK PRICE	CHANGE%
STOCK NAME	STOCK VOLUME	STOCK PRICE	CHANGE%

Leaderboard

1	NAME	POSITION	FIRM	1000	1 ↕
2	NAME	POSITION	FIRM	1000	1 ↕
3	NAME	POSITION	FIRM	1000	1 ↕
4	NAME	POSITION	FIRM	1000	1 ↕
5	NAME	POSITION	FIRM	1000	1 ↕
6	NAME	POSITION	FIRM	1000	1 ↕
7	NAME	POSITION	FIRM	1000	1 ↕
8	NAME	POSITION	FIRM	1000	1 ↕
9	NAME	POSITION	FIRM	1000	1 ↕
10	NAME	POSITION	FIRM	1000	1 ↕
28	NAME	POSITION	FIRM	1000	1 ↕

Predictions

Security	Current Price	Predicted Price	Predicted Date	Price at Placement	Date of Placement	Current Point Gain/Loss	Terminate
STK	$123.15	$125.00	01/01/23	$246.79	01/06/22	+350	❌
AAPL	$131.86	$140.00	01/02/23	$180.79	01/05/22	+400	
AMZN	$85.25	$75.00	01/03/23	$90	05/11/22	+20	
STK	$123.15	$125.00	01/01/23	$246.79	01/06/22	+350	
STK	$123.15	$125.00	01/01/23	$246.79	01/06/22	+350	
STK	$123.15	$125.00	01/01/23	$246.79	01/06/22	+350	

Place a Bet

Prediction	Select Prediction ▼
Odds	7/1
Amount	£ Input Price
Potential Return	Input Price X Odds

EXECUTE

Make A Prediction

Asset	Search for Asset ▼
Current Price	Search for Asset
Prediction Date	DD/MM/YY
Prediction Price	Input Price
Error Range	+ Input Range - Input Range

Calculate

Prediction Outcome

Reasoning
Reason for Prediction

Potential Point Return +500
Cost of Predicition -50

Kill Price Range
+ Input Range - Input Range

EXECUTE

Code, full code is also available on GitHub
https://github.com/wmacpherson/amplify-app

./components/Asset_Search.js
```
import React, { useState, useEffect } from 'react';
import { Card, ScrollView, TextField, Table, TableHead, TableCell, TableRow,
TableFoot} from '@aws-amplify/ui-react';
import * as Papa from 'papaparse';
import "../css/Asset_Search.css"
import { GraphSearch } from './Graph_Search';

function Asset_Search(){
    const [items, setItems] = useState([]);
    const [q, setQ] = useState("");
    const [searchParam] = useState(["Name", "Symbol"]);

    useEffect(() => {
        fetch( './nasdaq_screener.csv' )
        .then( response => response.text() )
        .then( responseText => {
            Papa.parse(responseText, {
                    header: true,
                    complete: results => {
                    setItems(results.data)
                },
            });
        });
    }, []);

    function search(items) {
        return items.filter((item) => {
            return searchParam.some((newItem) => {
                return (
                    item[newItem]
                        .toString()
                        .toLowerCase()
                        .indexOf(q.toLowerCase()) > -1
                );
            });
        });
    }
    return (
        <Card variation="elevated">
            <div>
                <label htmlFor="search-form">
                    <TextField
                        type="search"
                        name="search-form"
                        id="search-form"
                        className="search-input"
                        placeholder="Search for Stocks..."
                        value={q}
                        onChange={(e) => setQ(e.target.value)}
                    />
                </label>
            </div>
                <ScrollView className="my-scrollview" >
                <Table caption="" highlightOnHover={true} variation="elevated">
                    <TableHead>
                        <TableRow>
                        <TableCell as="th">Symbol</TableCell>
                        <TableCell as="th">Name</TableCell>
                        </TableRow>
                    </TableHead>
                    {search(items).map((item)=> (
                        <TableRow>
                            <TableCell width={"100px"}>{item.Symbol}</TableCell>
                            <TableCell>{item.Name}</TableCell>
                        </TableRow>
                    ))}
                    <TableFoot></TableFoot>
                </Table>
                </ScrollView>
            <GraphSearch />
        </Card>
    );
    }

export default Asset_Search
```

./components/Protected.js

```
import { useAuthenticator, Flex, Heading, Grid, Divider, Card } from '@aws-
amplify/ui-react';
import { Auth } from 'aws-amplify';
import React, { useState } from 'react';
import { Predicition} from '../ui-components';
import {ShowPredictions} from './Predictions'
import "../css/Protected.css"

export function Protected() {
  const [currentUser, setUser] = useState("");
  const { route } = useAuthenticator((context) => [context.route]);
  const message =
    route === 'authenticated' ? 'Your Profile' : 'Loading...';
  console.log(message)
  Auth.currentAuthenticatedUser({
    bypassCache: false
    })
      .then((user) => {
        setUser(user.username);
        console.log(user.username);
    })
  return(
    <Grid>
      <Heading level={2} row={1}>
        Your Profile
      </Heading>
      <Heading level={3} column={1} row={2}> Welcome {currentUser}</Heading>
      <Heading level={5} column={1} row={3}> View your Predictions Below</Heading>
      <Heading level={5} column={2} row={3}> Make a Prediction</Heading>
      <Card className='createPrediction' column={2} row={4} variation="elevated">
        <Predicition />
      </Card>
      <Card column={1} row={4}>
        <ShowPredictions />
      </Card>
    </Grid>
  );
}
```

./components/Predicitions.js

```
import { Card, Table, TableCell, TableHead, TableRow, TableFoot } from "@aws-
amplify/ui-react";
import { DataStore } from '@aws-amplify/datastore';
import { Prediction } from '../models';
import React, { useState } from 'react';

export function ShowPredictions(){
    const [items, setItems] = useState([]);
    async function GetPredictions(){
        try {
            const posts = await DataStore.query(Prediction);
            setItems(posts)
          } catch (error) {
            console.log("Error retrieving posts", error);
          }
    }
    GetPredictions()
    return(
        <Card className="prediciton-card">
            <Table variation="elevated">
                <TableHead>
                    <TableRow>
                        <TableCell>Stock Name</TableCell>
                        <TableCell>Stock Symbol</TableCell>
                        <TableCell>Initial Price</TableCell>
                        <TableCell>Initial Date</TableCell>
                        <TableCell>Prediciton Price</TableCell>
                        <TableCell>Prediciton Date</TableCell>
                    </TableRow>
                </TableHead>
                {items.map((data, key) => (
                    <TableRow key={key}>
                        <TableCell>{data.stock} </TableCell>
                        <TableCell>{data.stock_ticker}</TableCell>
                        <TableCell>{data.initial_price}</TableCell>
                        <TableCell>{data.initial_date}</TableCell>
                        <TableCell>{data.prediction_price}</TableCell>
                        <TableCell>{data.prediction_date}</TableCell>
                    </TableRow>
                ))}
                <TableFoot></TableFoot>
            </Table>
        </Card>
    )
}
```

./components/Login.js

```
import { useEffect } from "react";
import { Authenticator, useAuthenticator, View } from '@aws-amplify/ui-react';
import '@aws-amplify/ui-react/styles.css';

import { useNavigate, useLocation } from 'react-router';

export function Login() {
  const formFields = {
    signUp: {
      username: {
        placeholder: 'Enter Your Username Here',
        isRequired: true,
        label: 'Username:'
      },
      firm: {
        placeholder: 'Enter Your Firm Name',
        isRequired: false,
        label: 'Firm',
      },
      job: {
        placeholder: 'Enter your Position within the Firm',
        isRequired: false,
        lable: 'Job',
      }
    },
  }
  const { route } = useAuthenticator((context) => [context.route]);
  const location = useLocation();
  const navigate = useNavigate();
  let from = location.state?.from?.pathname || '/';
  useEffect(() => {
    if (route === 'authenticated') {
      navigate(from, { replace: true });
    }
  }, [route, navigate, from]);
  return (
    <View className="auth-wrapper">
      <Authenticator formFields={formFields}></Authenticator>
    </View>
  );
}
```

./components/Layout.js

```
import React from 'react';
import { Outlet, useNavigate } from 'react-router-dom';
import { useAuthenticator, Button, View} from '@aws-amplify/ui-react';
import '../css/layout.css';

export function Layout() {
  const { route, signOut } = useAuthenticator((context) => [
    context.route,
    context.signOut,
  ]);
  const navigate = useNavigate();

  function logOut() {
    signOut();
    navigate('/login');
  }
  return (
    <>
      <nav>
        <Button row={1}column={1} onClick={() => navigate('/')}>Home</Button>
        <Button row={1}column={2} onClick={() => navigate('/protected')}>
        Your Profile
        </Button>
        {route !== 'authenticated' ? (
        <Button row={1}column={10} onClick={() =>
navigate('/login')}>Login</Button>
        ) : (
        <Button row={1}column={10} onClick={() => logOut()}>Logout</Button>
        )}
        <View row={1}column={9}>
        {route === 'authenticated' ? 'You are logged in!' : 'Please Login!'}
      </View>
      </nav>
      <Outlet />
    </>
  );
}
```

./components/Home.js

```
import { Heading, Divider, Card, Text, Grid } from '@aws-amplify/ui-react';
import Asset_Search from './Asset_Search';
import * as React from 'react';
export function Home() {

  return (
    <Grid>
      <Heading level={3} row={1}>
        Home Page
      </Heading>
      <Divider
        size="small"
        orientation="horizontal"
        row={2}/>
      <Card row={3}>
        <Heading level={5}>Stock Search</Heading>
        <Asset_Search />
      </Card>
    </Grid>
  );
}
```

./components/Graph_Search.js:

```
// import { LineChart, Line, CartesianGrid, XAxis, YAxis, Tooltip } from
'recharts';
import React, { useState } from 'react';
import { TextField, Button, Grid, Card, Tabs, TabItem } from '@aws-amplify/ui-
react';
import "../css/Asset_Search.css"
import axios from 'axios'

export function GraphSearch(){
    var stockData
    const [symbol, setSymbol] = useState("");
    const [error, setError] = useState(null);
    const handleSubmit = (event) => {
        event.preventDefault();
        console.log("running")
        console.log(symbol)

axios('https://query1.finance.yahoo.com/v10/finance/quoteSummary/'+symbol+'?modules
=earningsHistory')
            .then((response) => {
                stockData=(response.data);
                setError(null);
            })
            .then(setSymbol(""))
            .then(console.log(stockData))
            .catch(setError);
        if (error) return <p>An error occurred</p>
        return(<p>{stockData}</p>)
      }

    return (
        <Grid>
        <form onSubmit={handleSubmit} >
            <label>Retrieve Stock Data:</label>
            <TextField  variation="elevated"
                type="text"
                placeholder="Input Stock Symbol as defined above..."
                value={symbol}
                onChange={(e) => setSymbol(e.target.value)}
            />
            <Button type='submit'>Load Stock Data</Button>
        </form>

        <Card className='stock-img'  variation="elevated">
        <Tabs
        justifyContent="flex-start">
        <TabItem title="Day">
            Display Daily Data
        </TabItem>
        <TabItem title="Month">
            Display Monthly Data
        </TabItem>
        <TabItem title="Year">
            Display Yearly Data
        </TabItem>
        </Tabs>
        </Card>
        </Grid>
      );}
```

./components/Asset_Search.js:

```
import React, { useState, useEffect } from 'react';
import { Card, ScrollView, TextField, Table, TableHead, TableCell, TableRow,
TableFoot} from '@aws-amplify/ui-react';
import * as Papa from 'papaparse';
import "../css/Asset_Search.css"
import { GraphSearch } from './Graph_Search';

function Asset_Search(){
    const [items, setItems] = useState([]);
    const [q, setQ] = useState("");
    const [searchParam] = useState(["Name", "Symbol"]);

    useEffect(() => {
        fetch( './nasdaq_screener.csv' )
        .then( response => response.text() )
        .then( responseText => {
            Papa.parse(responseText, {
                    header: true,
                    complete: results => {
                    setItems(results.data)
                },
            });
        });
    }, []);

    function search(items) {
        return items.filter((item) => {
            return searchParam.some((newItem) => {
                return (
                    item[newItem]
                        .toString()
                        .toLowerCase()
                        .indexOf(q.toLowerCase()) > -1
                );
            });
        });
    }
    return (
        <Card variation="elevated">
            <div>
                <label htmlFor="search-form">
                    <TextField
                        type="search"
                        name="search-form"
                        id="search-form"
                        className="search-input"
                        placeholder="Search for Stocks..."
                        value={q}
                        onChange={(e) => setQ(e.target.value)}
                    />
                </label>
            </div>
                <ScrollView className="my-scrollview" >
                <Table caption="" highlightOnHover={true} variation="elevated">
                    <TableHead>
                        <TableRow>
                        <TableCell as="th">Symbol</TableCell>
                        <TableCell as="th">Name</TableCell>
                        </TableRow>
                    </TableHead>
                    {search(items).map((item)=> (
                        <TableRow>
                            <TableCell width={"100px"}>{item.Symbol}</TableCell>
                            <TableCell>{item.Name}</TableCell>
                        </TableRow>
                    ))}
                    <TableFoot></TableFoot>
                </Table>
                </ScrollView>
            <GraphSearch />
        </Card>
    );
    }

export default Asset_Search
```

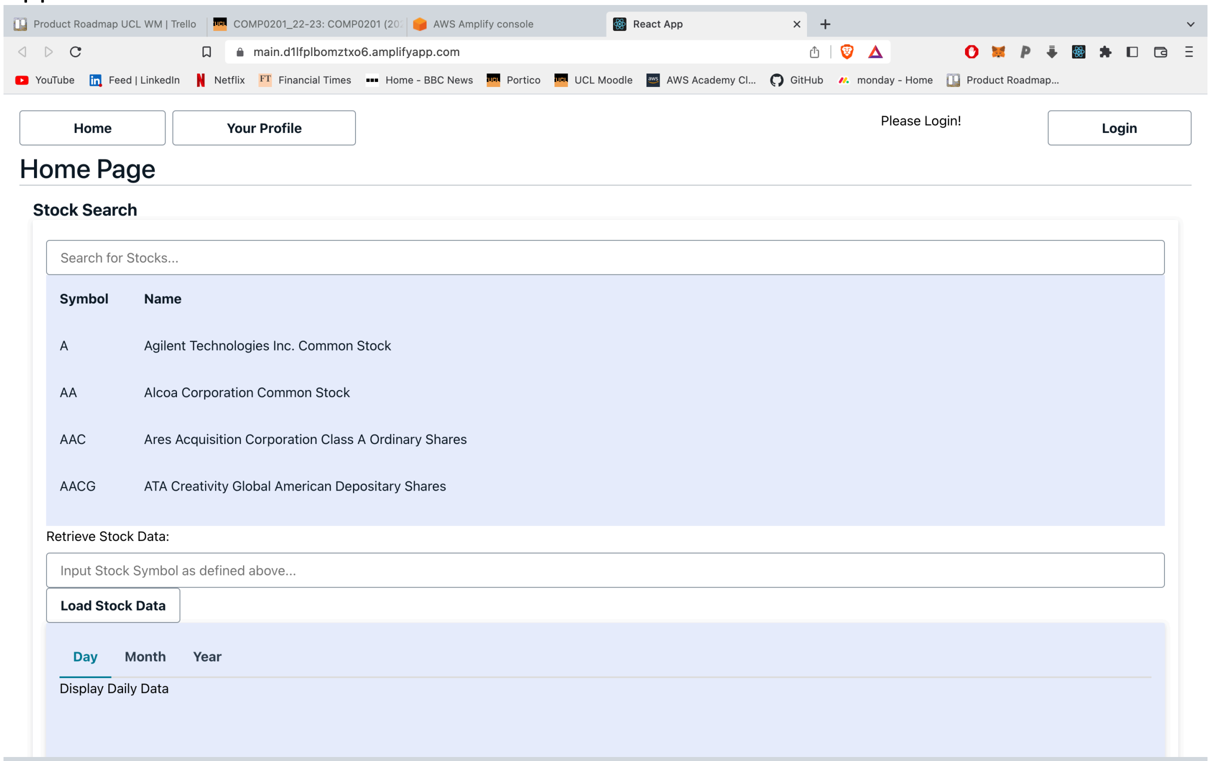
Index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './css/index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import awsExports from "./aws-exports";
import "@aws-amplify/ui-react/styles.css";
import { Amplify } from 'aws-amplify'
Amplify.configure(awsExports);

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

reportWebVitals();
```

App.js:

```
import { Authenticator } from '@aws-amplify/ui-react';
import { Protected } from './components/Protected';
import { RequireAuth } from './RequireAuth';
import { Login } from './components/Login';
import { Home } from './components/Home';
import { Layout } from './components/Layout';

import { BrowserRouter, Routes, Route } from 'react-router-dom';

import './css/App.css';

function MyRoutes() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route
            path="/protected"
            element={
              <RequireAuth>
                <Protected />
              </RequireAuth>
            }
          />
          <Route path="/login" element={<Login />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

function App() {
  return (
    <Authenticator.Provider>
      <MyRoutes />
    </Authenticator.Provider>
  );
}

export default App;
```

App views if code doesn't run



Please Login!

Home Page

Stock Search

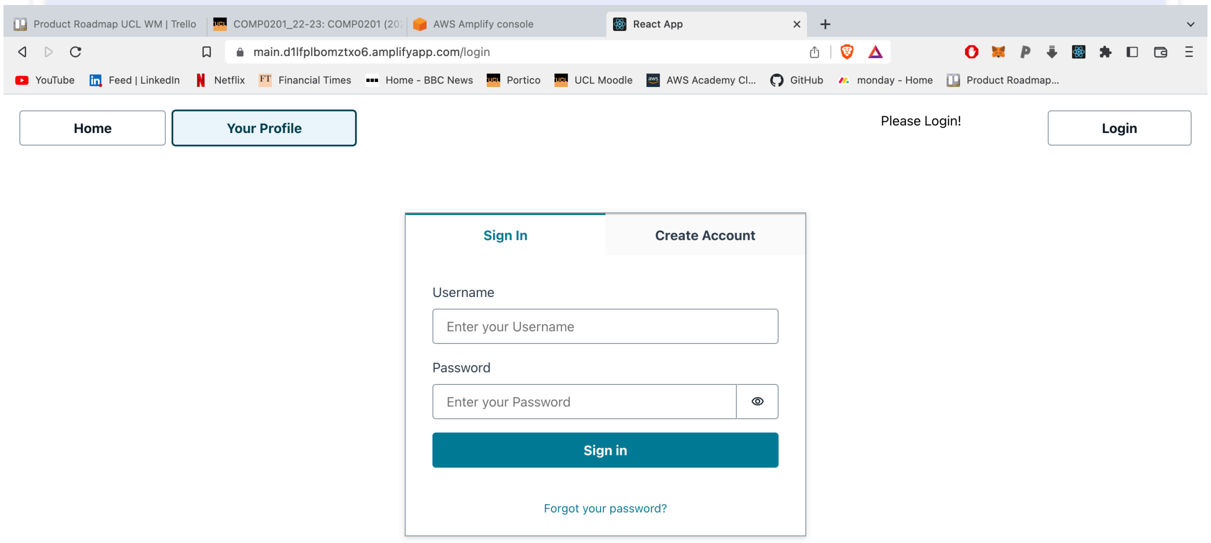
Search for Stocks...

Symbol	Name
A	Agilent Technologies Inc. Common Stock
AA	Alcoa Corporation Common Stock
AAC	Ares Acquisition Corporation Class A Ordinary Shares
AACG	ATA Creativity Global American Depositary Shares

Retrieve Stock Data:

Input Stock Symbol as defined above...

Display Daily Data



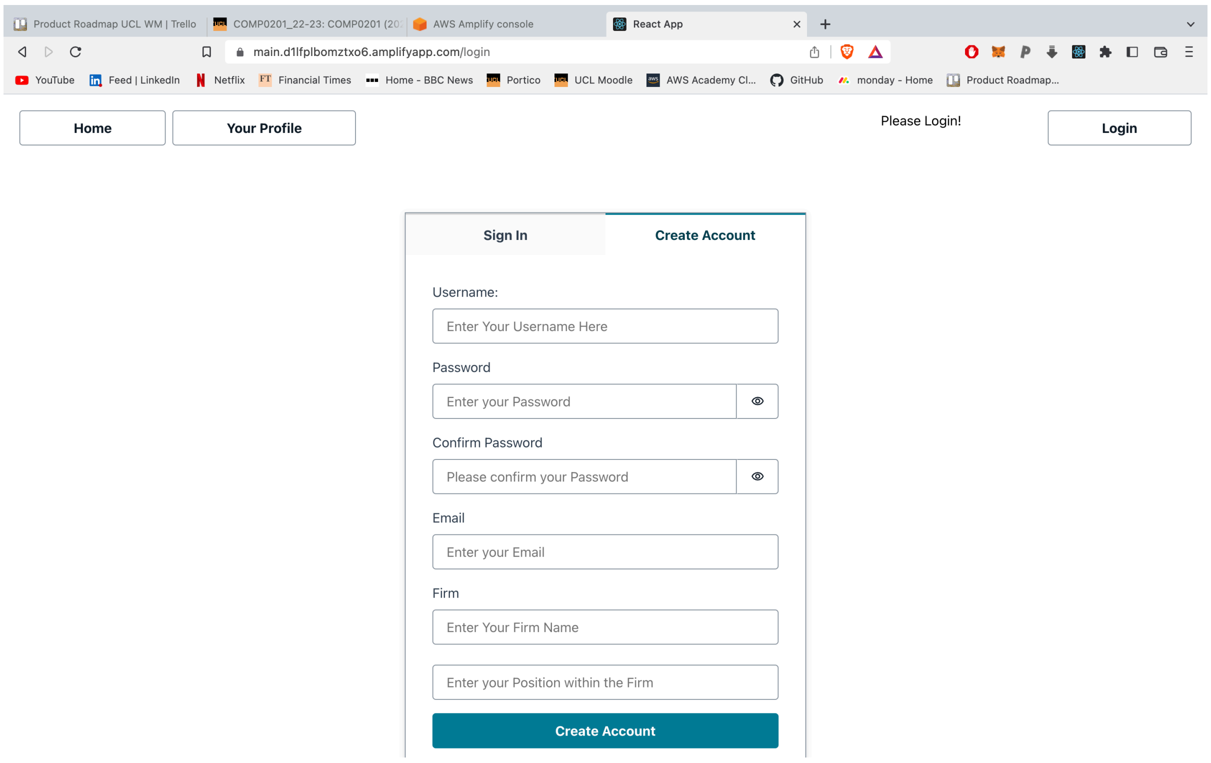
Please Login!

Username

Enter your Username

Password

Enter your Password



Product Roadmap UCL WM | Trello
COMP0201_22-23: COMP0201 (20
AWS Amplify console
React App

main.d1lfplbomztxo6.amplifyapp.com/login

YouTube · Feed | LinkedIn · Netflix · Financial Times · Home - BBC News · Portico · UCL Moodle · AWS Academy Cl... · GitHub · monday - Home · Product Roadmap...

Home · Your Profile · Please Login! · Login

Sign In · Create Account

Username:
Enter Your Username Here

Password
Enter your Password

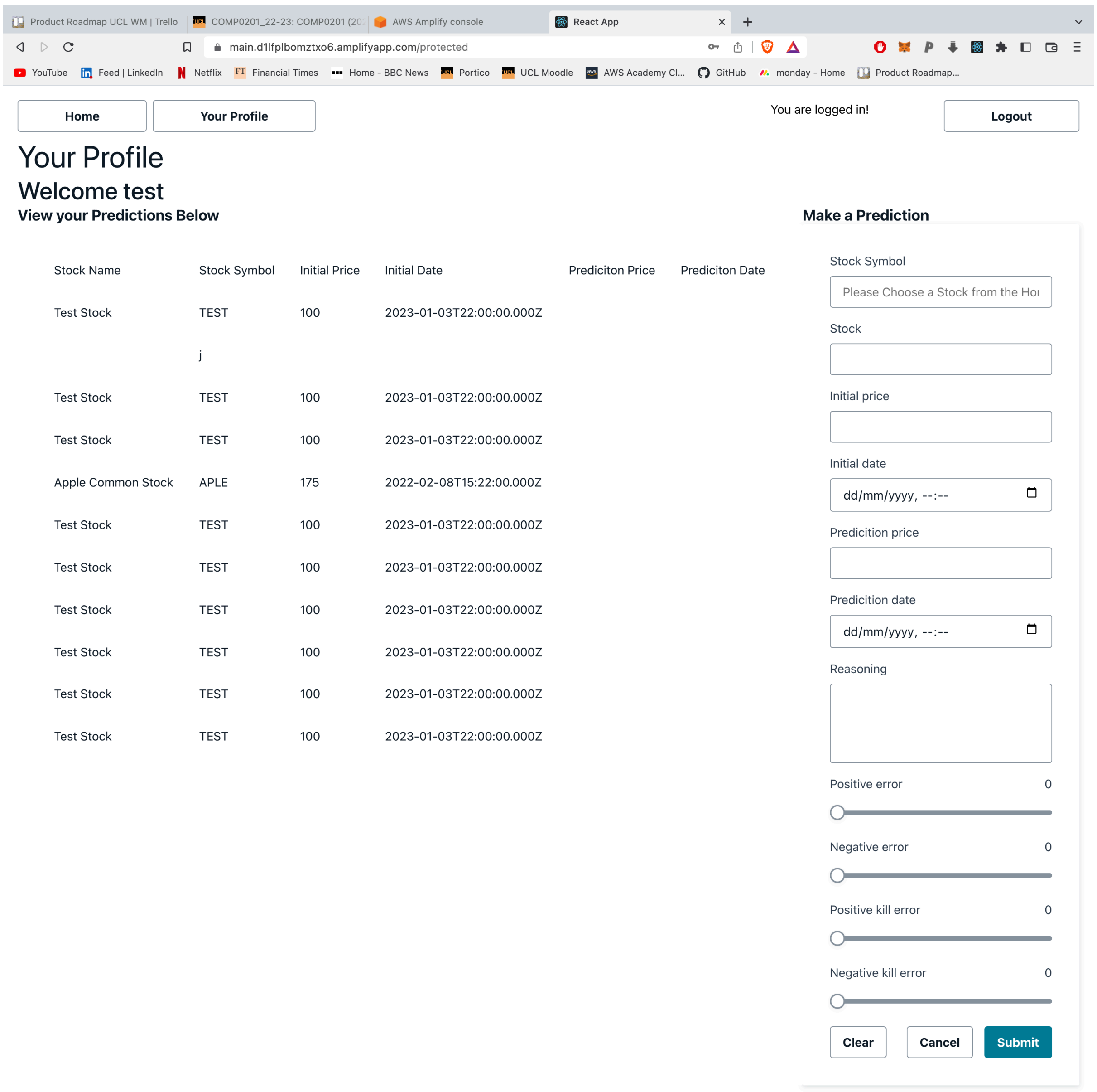
Confirm Password
Please confirm your Password

Email
Enter your Email

Firm
Enter Your Firm Name

Enter your Position within the Firm

Create Account



Product Roadmap UCL WM | Trello | COMP0201_22-23: COMP0201 (20 | AWS Amplify console | React App

main.d1lfplbomztxo6.amplifyapp.com/protected

YouTube | Feed | LinkedIn | Netflix | Financial Times | Home - BBC News | Portico | UCL Moodle | AWS Academy Cl... | GitHub | monday - Home | Product Roadmap...

Home | Your Profile | You are logged in! | Logout

Your Profile

Welcome test

View your Predictions Below

Stock Name	Stock Symbol	Initial Price	Initial Date	Prediciton Price	Prediciton Date
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
	j				
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
Apple Common Stock	APLE	175	2022-02-08T15:22:00.000Z		
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		
Test Stock	TEST	100	2023-01-03T22:00:00.000Z		

Make a Prediction

Stock Symbol: Please Choose a Stock from the Hor

Stock:

Initial price:

Initial date: dd/mm/yyyy, --:--

Predicition price:

Predicition date: dd/mm/yyyy, --:--

Reasoning:

Positive error 0

Negative error 0

Positive kill error 0

Negative kill error 0

Clear | Cancel | Submit