Investing Project

Planning

What is this project about:

With this project the user will be able to access various data about some of the largest investments from the last 10-20 years. With this website you will be able to access data about the price of each of these investments and compare one another to find which would have been the best investment to have made. This website is for someone who wants a place to access historical data about stocks, crypto currencies in one place.

Design Notes:

Website needs to be easy to use for first time users, simple to understand and have a clear description of what each function of the website does.

Design Plan:

Font	Comic Sans
Nav bar and outlines	⬛
Background	⬜

Accent Colour	🟦

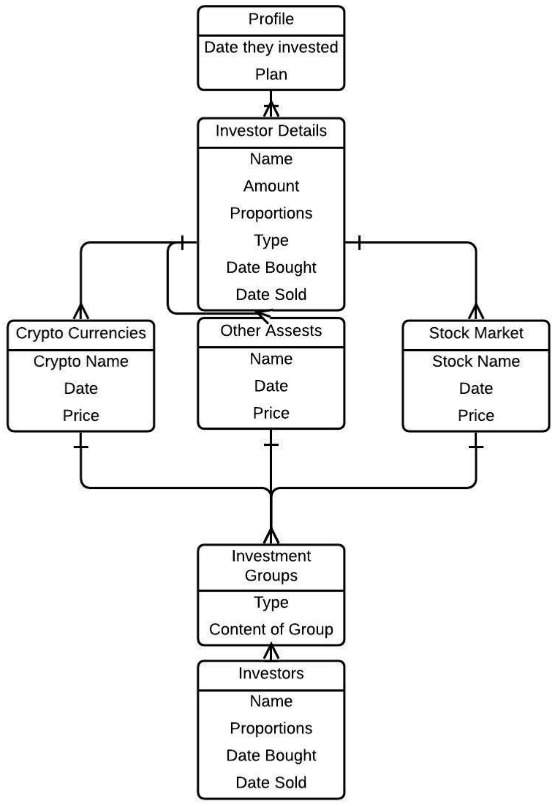
Possible source for data:

https://www.kaggle.com/mczielinski/bitcoin-historical-data

https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

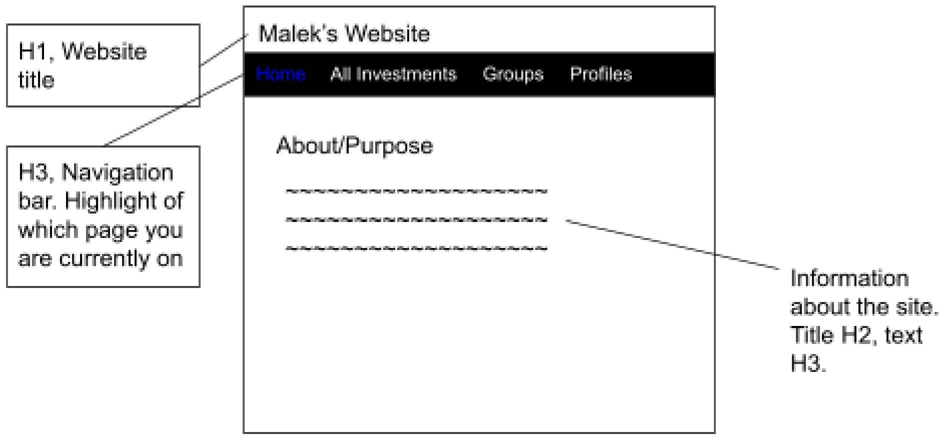
Layout of database:

This chart shows how I am planning to layout my database. There will be multiple profiles, these profiles will be of artificial people who have each invested into different things. The database will display the date they invested and what they have invested in. The next table will give details about that person. From there you will be able to see each specific investment that person has made e.g. Crypto, stocks or another investment. Each person will have a % of their money invested into a different group. These groups will have different risk margins. The most aggressive fund will contain more risky investments whereas a more balanced fund will have safer investments. Another table will show the gain of each person and also each individual investment.

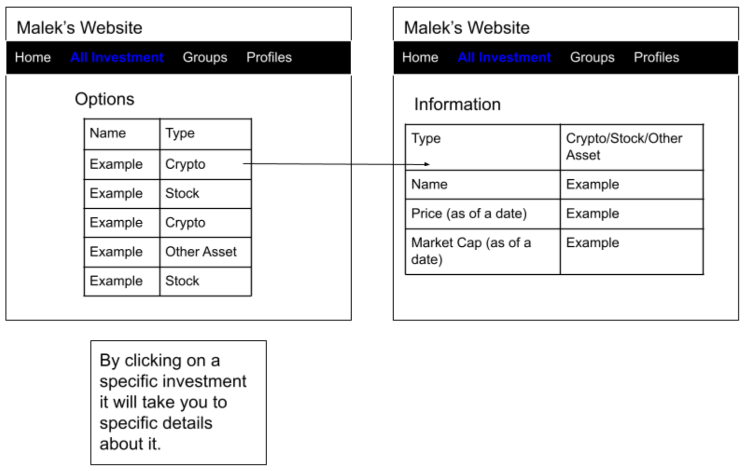


Page Layout and Plan:

Home Page: `@app.route('/Home')`


H1, Website title
H3, Navigation bar. Highlight of which page you are currently on
Malek's Website
Home
All Investments
Groups
Profiles
About/Purpose
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~
Information about the site. Title H2, text H3.

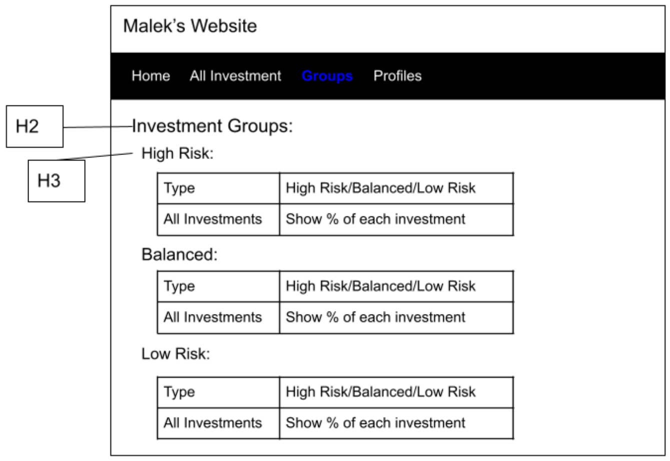
All Investments Page: @app.route('/All Investments')



Name	Type
Example	Crypto
Example	Stock
Example	Crypto
Example	Other Asset
Example	Stock

Type	Crypto/Stock/Other Asset
Name	Example
Price (as of a date)	Example
Market Cap (as of a date)	Example
~~~~~~~~~~~~~~~~~~~~

Investment Groups Page: @app.route('/Groups')


Malek's Website

Home All Investment Groups Profiles

Investment Groups:

High Risk:

Type	High Risk/Balanced/Low Risk
All Investments	Show % of each investment

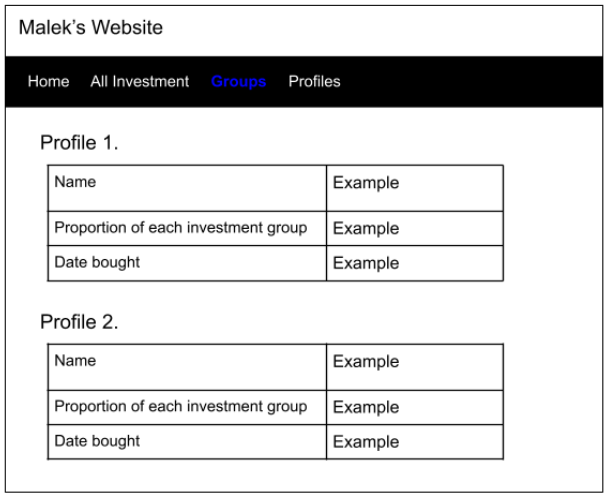
Balanced:

Type	High Risk/Balanced/Low Risk
All Investments	Show % of each investment

Low Risk:

Type	High Risk/Balanced/Low Risk
All Investments	Show % of each investment

Profiles Page: @app.route('/Groups')



Profile 1.

Name	Example
Proportion of each investment group	Example
Date bought	Example

Profile 2.

Name	Example
Proportion of each investment group	Example
Date bought	Example

Purpose for each page

Home:

The home page will tell the viewer the basic information about the website and what the purpose for it is.

All investments:

This page will show each investment that will be included in each of the investment pools. From there you will be able to click on each individual investment and find out more information about it. This info will include the name, type of investment, other details like price at a certain date as well as market cap.

Investment Groups:

The Investment groups are going to be made up of individual investments. There will be different investment groups. These include a high risk, balanced and low risk group. Each group will have different investments depending on how much risk they have. The risk will be determined by how they have done in the past.

Profile Page:

This will be a profile of an artificial person. Each profile will have a different % of each investment group. By looking at different profiles you will be able to see how each performed over the last 10 years. From this page you will be able to see each group the person is invested in as well as the individual investments in that group.

Queries:

Queries I expect to use for my website.

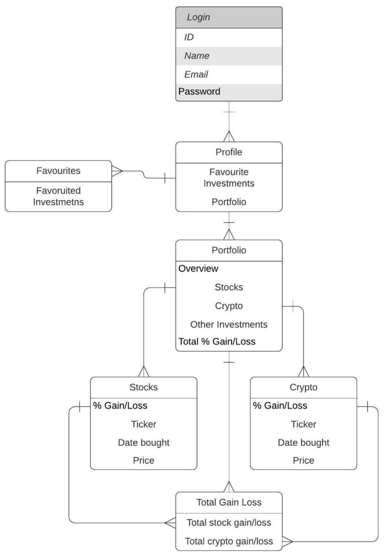
Page	Query	Expected Result

All Investments	SELECT * FROM Crypto_Currencies, Other_Assests, Stock_market	Show all different investments and details about it.
All Investments information	SELECT (Name) FROM (Type)	Show the individual investment and details about it.
Investment Groups Page	SELECT * FROM Investment_groups	Show all of the investment groups and content of the groups
Profiles Page:	SELECT * FROM Investor_Details	Show all information about each individual person.
Profiles Page individual person	SELECT (NAME) FROM Investor_Details	Show all details about the individual selected.

Database initial plan:

DBMS ER diagram (UML notation)

Malek Connor | June 9, 2021



Overview

Tools used for all standards:

The following tools were used to create my project:

1. **Visual Studio Code (VSC)** - This was used as the text editor to code the project. VSC was used to track the files required for this project. VSC was used across all of the components of the standards.
2. **SQLite studio -** This was used to both create and manage the database. This tool allowed easy creation of a database that could then be implemented into the website.
3. **Code languages -** The following programming languages were used; Python, HTML 5, CSS, and javascript. These were used across all components of the standards. Python was used for Flask and to create a link with the chosen API. Flask was used to connect the database to the website. HTML 5 was used for formatting and content of the website. CSS was used alongside HTML5 to ensure an aesthetically pleasing website. Javascript was used for add/remove (button changing on click) and graphing functionality (the stock page).
4. **Github** - Github was used to track improvements and change over time. This allowed visualisation and evidence of iterative progress of the project.

Iterative improvement:

https://github.com/MalekC01/12DTP

These are some of the biggest improvements/changes I have made throughout my project.

1. https://github.com/MalekC01/12DTP/commit/c7988116d940d3d7cc075aa845c306ceb816b3a2. This was my initial commit to github where I first established my files and started creating my early structure for the project.
2. https://github.com/MalekC01/12DTP/commit/4f5dd40836faea415e1d59748dbb0e388421c5ec. Although I have now mostly changed all of the code in this python file it was where I began coding the API that would then be later on implemented into the website.
3. https://github.com/MalekC01/12DTP/commit/e7c0c1511c8ca5e964e14fbc78e9e859a7cbecd8. This was a major change I decided to make when instead of using my old design of the database as seen in planning I decided to change to a more user oriented website/database. This was to allow for more functionality within the project.
4. https://github.com/MalekC01/12DTP/commit/a8507e679604475d50de41bff8a69e304f764b2b. Creating the user login system is one of the most important parts of my project as it met the purpose of the website.
5. https://github.com/MalekC01/12DTP/commit/84532b19ab14727f6f61a1151b63eff847efd3cc/. After having the main functionality and structure for my website I then started to work on some of the visuals by using CSS.
6. https://github.com/MalekC01/12DTP/commit/9ab0830fbfeb5c970248041f39f8b76644c7b7ce. By combining the two large parts of my website together the project was starting to take shape and become functional.

7. https://github.com/MalekC01/12DTP/commit/ac99bc04866d9bf107acb1361d650a07ce04167c. This was a major part of the website that took me a long time to figure out. However once it worked it became much easier to test everything.
8. https://github.com/MalekC01/12DTP/commit/99beeb27c42c33628aa421f1846c353f4dae388a. Changing the navigation once logged in. This extra feature, although seemingly minor, significantly improved the user experience.
9. https://github.com/MalekC01/12DTP/commit/613903687e1b6a47e2ef66770a0b921516751ac6. Fixing the layout was a large change in the CSS component. However this resulted in a significant improvement in the user experience.
10. https://github.com/MalekC01/12DTP/commit/733afad1c5f6506e46c52dca60237f0b9beeeb52. Being able to add and remove from the favourites page was another big feature that once implemented greatly improved the project.
11. https://github.com/MalekC01/12DTP/commit/4a1632acfe8cc48e759070a7f2994146db5d8046. Adding the graph using google charts was one of the trickiest things in the whole website. This is because I was very unfamiliar with javascript.
12. https://github.com/MalekC01/12DTP/commit/e945a4f4a0c84c7587eb73300047260cc5922b8c. The 404 page made the website robust and less prone to breaking. This also improves the user experience.
13. https://github.com/MalekC01/12DTP/commit/0266a6b9bce4f6eeaf83a2b44363a32f19c7f6ce. This was by far the hardest and biggest change in the entire project. This is because it meant I had to rewrite all code and queries which were previously being used for the old API. I had to change the API because the old one was unreliable and didn't give me the information I was looking for.
14. https://github.com/MalekC01/12DTP/commit/01608761734f70f0e7daab183a36a7d22fa5a417. This was another big change to ensure the coding met the required standard. This meant rewriting almost all of the queries as the database had a new layout.

Database standard:

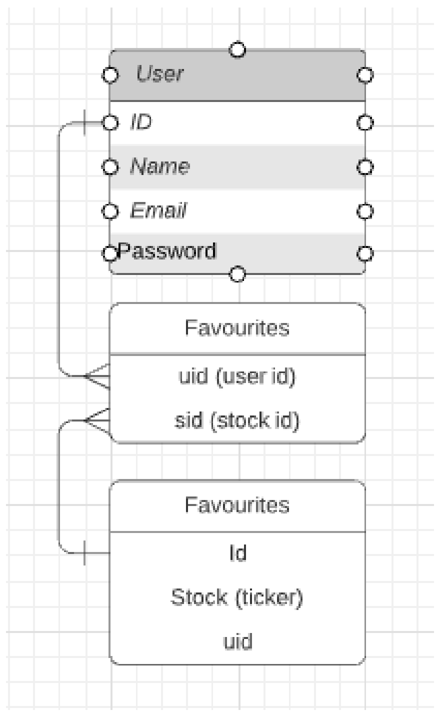
Problems that required pivoting:

1. After beginning the website I encountered problems that meant I had to adjust the design and purpose of the website. The purpose changed from tracking individual investments over time to viewing historical data about an individual stock. As a result of this pivot the database required restructuring. The redesign required creation of a user login so that users could create a bespoke view based on their selections. This allowed users to “favourite” certain stocks to be linked to their personal accounts.
2. Although my initial plan was to include diverse investments such as stocks, cryptocurrency and international currencies, the project was limited by the availability of free API (Yahoo finance). This limited the stocks that could be called to the US stock market. Although the website design is capable of surfacing other investment streams this will be dependent on API availability in the future.

Redesign

I decided to redesign my database to a different format. It is now based around having a user login. The advantage of this is that it does not need to store large amounts of price data from the different investments. Thai helps with future-proofing of the database. This keeps the database very compact and maintains both the currency and integrity of rapid cycling data.

ER Diagram:



Database testing:

Testing:

Part	Input	Expected result	Actual result
Creating Users	Name: Example Email: example@gmail.com Password: example	Information to be added into the users table inside the database. Take the user to login once	As expected. Noticed redirect takes user home instead of login.

		signed up.	
Make sure email is valid	Name: Example Email: example/gmail.com Password: example	Websites should show an error that email is invalid. Should not be added to the database.	As expected.
Make sure special characters are added correctly.	Name: Example Email: example@gmail.com Password: example!123/*	Should add as expected.	As expected.
Check stocks are being added/removed from favourites as wanted. While logged in.	On the apple stock page pressing add to favourites.	Adds apple stock to the database with the user's id.	Adds to the database however the button always starts with remove even if stock is already in the database.
Check stocks are being added/removed from favourites as wanted. While logged in.	On the apple stock page pressing remove to favourites.	Removes apple stock to the database with the user's id.	Removes from the database however the button always starts with remove even if stock is already in the

			database.
Check stocks are being added/removed from favourites as wanted. While not logged in.	On the apple stock page pressing remove to favourites.	Button should not remove anything to the database as there is no user to link it to.	As expected however the release button should not show up as there is nowhere to remove it from. However, it should tell the user to login.
Check stocks are being added/removed from favourites as wanted. While not logged in.	On the apple stock page pressing adding to favourites.	Button should not add anything to the database as there is no user to link it to.	As expected however the release button should not show up as there is no where to add it to. However, it should tell the user to login.

*All numbers used in queries are examples as I use session data to run queries

Queries	Input	Expected result	Successful
Finding users favourite stocks	SELECT stock_ticker FROM Favourites WHERE id IN (SELECT sid FROM UserFav WHERE uid	To display all users stocks they have favourited	No. No result given

	= 152);		
	SELECT stock_ticker FROM Favourites WHERE id IN (SELECT sid FROM UserFav WHERE uid = 13);		Yes. Was using an incorrect id (stock id instead of user's id).
Deleting users favorite stocks.	DELETE FROM Favourites WHERE id IN (SELECT uid FROM UserFav WHERE uid = 13 AND sid = 132);	For the user's stock to be deleted.	No.
	DELETE FROM UserFav WHERE uid = 13 AND sid = 152;		Yes
Insert stock into user fav	INSERT INTO UserFav (uid, sid) VALUES (?, ?);	To insert searched stock into database	Yes
Find id for stock to then be added to session. Will be later	SELECT sid FROM UserFav WHERE uid = (?)		No. Broke the entire site because of a bracket.

used for the other queries.			
	SELECT sid FROM UserFav WHERE uid = (?))	To find the stock id and then add to session	Yes

After testing my database I have found multiple problems to fix:

- Once a registered user is redirected to the home page instead of the login page.
    - To fix this I will change the redirect to the login page.
        - The cause of this issue is that I was not running a fetchall but actually a fetch one.
        - `stocks = do_query(find_id, (session['uid']))`

          `Solution:`

          ```
          stocks = do_query(find_id, (session['uid'],), True)
          ```

- Add/remove button always shows up as remove when first loaded up regardless whether stock is in favorites or not.
    - Use a query of the favourites to check what button to display.
        - To fix this issue I added another check for the button to appear. This extra check was by adding logged_in == True.
        - add/remove from the favourites button not displaying the correct one.

    ```
    {%if in_fav == True%}

           <script>
    ```

```
          document.getElementById("add_to_favourites").style.visibility =
'hidden';
       </script>
       {%elif in_fav == False%}
       <script>
        document.getElementById("remove_from_favourites").style.visibility
= 'visible';
       </script>
       {%endif%}
```

Relevant Implications:

1. Privacy: Privacy is a critical feature for a database especially one that is holding users emails and passwords. The database must be secure because in the case that it were to be hacked the user should be assured that even if there was a leak their data will be safe. This can be done by using password hashing. Password hashing is when a mathematical algorithm is used to scramble input from the user and stored in a way that is no longer plain text. This is then decoded again once put back into the website.

   How I have incorporated privacy into my website: To ensure user passwords are safe I have hashed them. Hashing is when the plain text input from the user is run through an algorithm that encodes it to a unique value. This means that in the case of a leak or hack the user's password is still secure.

2. Usability - Usability is a feature that is important while ensuring that it is easy to find the data needed by being stored in an efficient way. The database must be usable so that all functions can be used well.

   I have incorporated usability into my database when instead of storing each user's favourites separately they are all stored in a table together where they are then linked to the user, this saves space by not doubling up on the same data being inputted twice. By storing users' email I am then able to query for the uid, that uid is then linked to all of that users favorites stocks which can then be queried using the uid.

Programming standard:

The programming part of my project is the most complicated because it is where all my data comes from. This happens by using an API that gathers the data on command. This is beneficial because it means I do not have to store large amounts of rapidly expanding data (stock price changes every minute) inside my database. Python also allows me to use a login system for my website.

Changes:

1. The biggest change that I had to make throughout programming my project was that I had to change which API I was using. This was due to the one I had originally used had many restrictions. These restrictions included a limited amount of calls as well as limited time spans available. To fix these issues I had to find a new API and change almost all of my code as almost all of my Python is to do with my API.

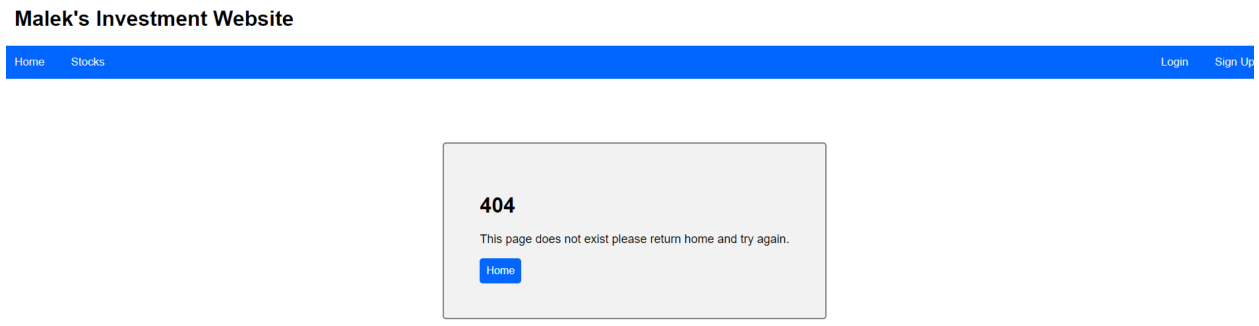
Making my project robust and robust:

Robust:

I have developed a few ways to ensure that my program is robust and will stay reliable. This is an important feature to have to make sure users are not able to break your website.

First way:

The first way I have used to ensure the site is robust is by creating a 404 page:



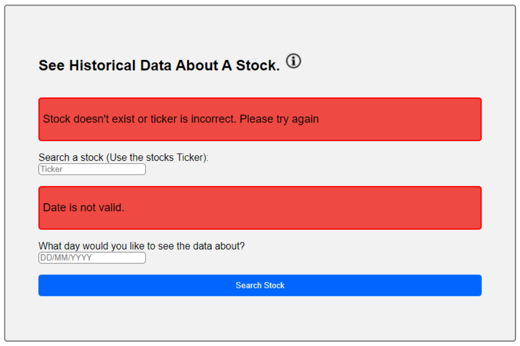
404

This page does not exist please return home and try again.

Should any additional errors arise that have not been caught, the 404 page will allow the user to redirect to the home page. In doing this it tells the user this page is not available and gives them the option to return home and try again.

Second way:

I have methods in place to give the user error messages if their input is unsuccessful. These error messages ensure the user is aware both what is wrong and that they can try again.

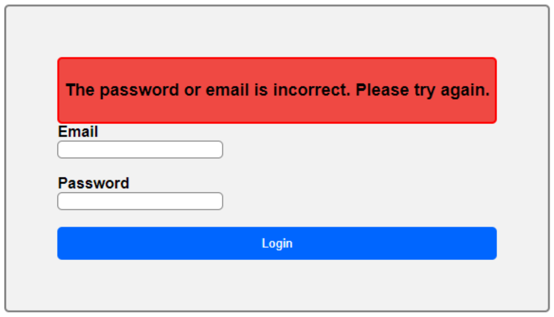

See Historical Data About A Stock.

Stock doesn't exist or ticker is incorrect. Please try again

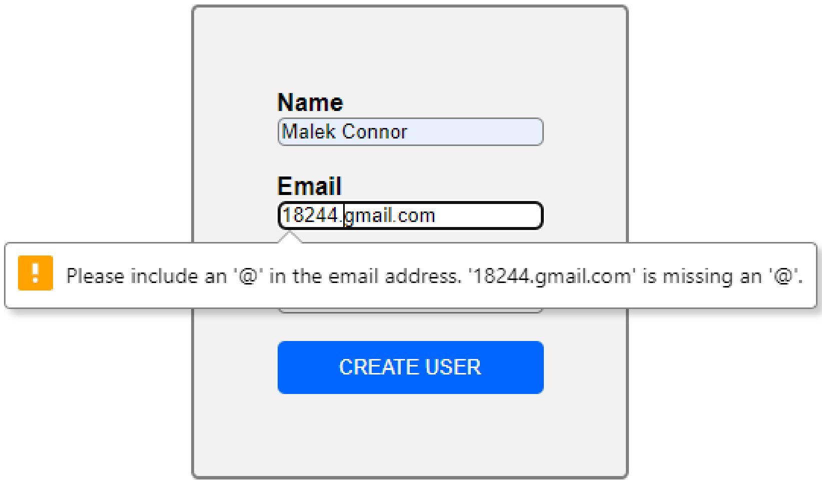
Search a stock (Use the stocks Ticker):

Date is not valid.

What day would you like to see the data about?


The password or email is incorrect. Please try again.

This also helps with usability as the user is always aware of what is happening and why something has not worked.


Name: Malek Connor

Email: 18244.gmail.com

Please include an '@' in the email address. '18244.gmail.com' is missing an '@'.

Third way: This error message occurs if the user has input an invalid email address when signing up for the account. This prevents accidental register issues which could lead to further problems later on. These further problems could include the user being unable to log back into their account later on but it also stops unnecessary accounts being added to the database.

Flexible:

By using an API means I do not have to update or change any data to keep relevance. Because of this I am relying on the API to be able to handle any new data e.g. new price and new stocks being added to the exchange. This means that the website's data stays current and up to date assuming the API stays running.

Programming and Website Testing:

To make sure all aspects of the website are functional I went through and tested each individual part of the website. To test my program works I must use my website. Therefore I will combine my testing of the website with the testing for the programming standard.

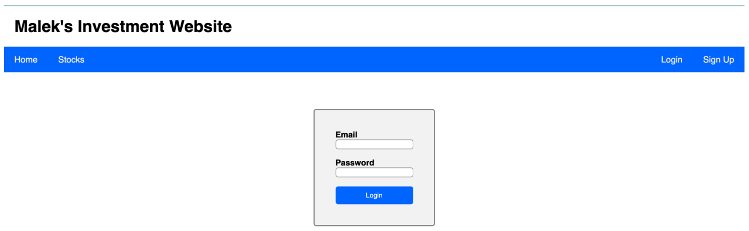
Expected Testing:

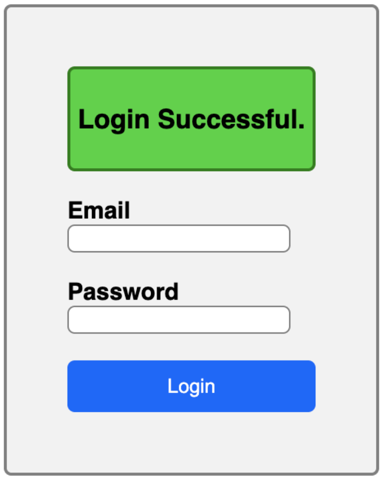
These tests are created by using the program as a normal user that isn't purposefully trying to break the code. Therefore no errors should occur.

Home Page:

When the user is first taken to the website they are taken to the default link 127.0.0.1:8080. This is my home page which gives the user a bit more information about the website. I have made the app route in my programming default a /. This is what directs the user to my homepage.

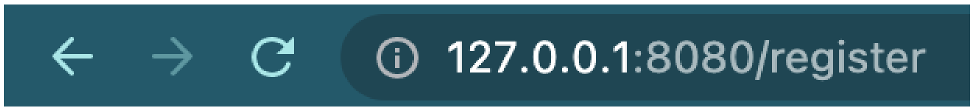
Login Page:

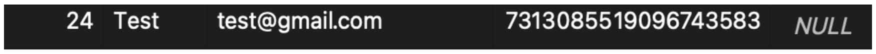


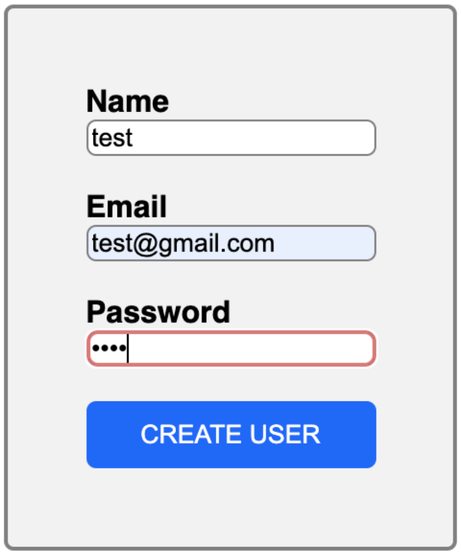


I next tested my login page which is working correctly. I tested this by using the button in the navigation bar. I also tested the link in the home page that directs the user to the login page for the full experience of the website. The message telling the user that they have successfully logged in also works as expected.

Register Page:

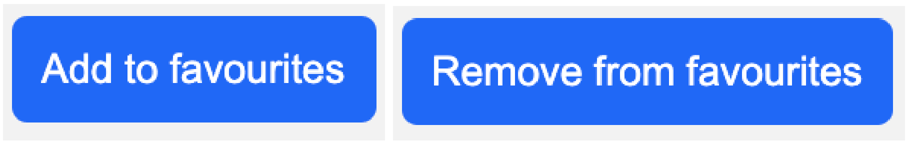


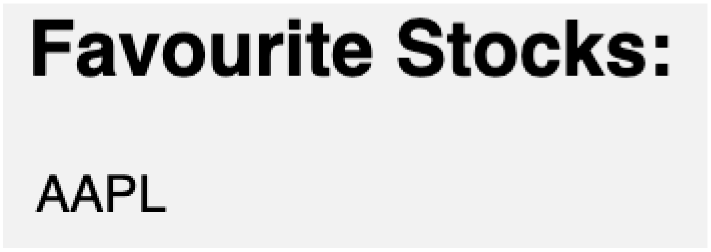

```
24  Test   test@gmail.com   7313085519096743583  NULL
```



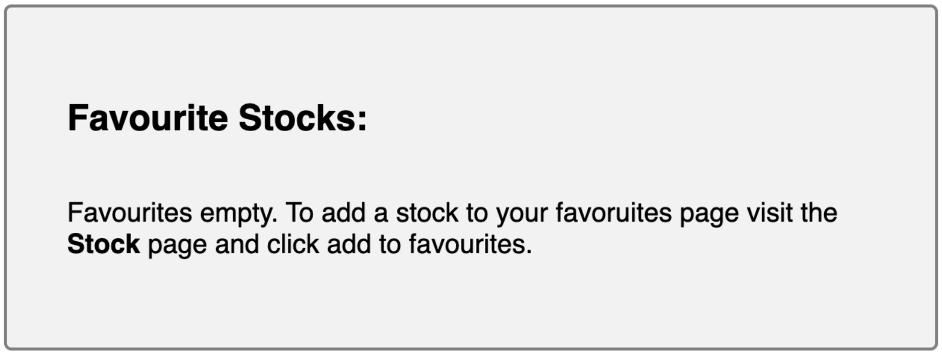
I tested that the register page is both loading and functioning correctly. As shown in the image once registering to the page the users data is correctly being stored into the database.

Favourites Page:




Favourite Stocks:

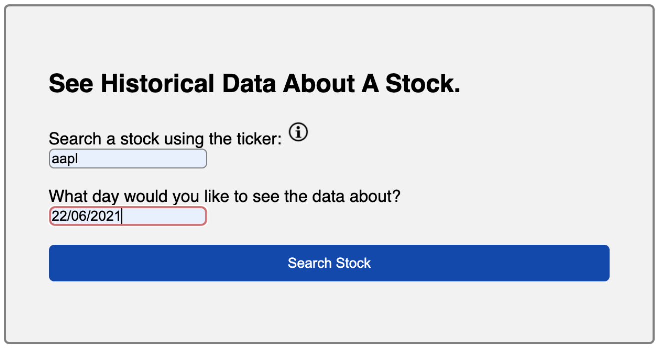
AAPL


Favourite Stocks:

Favourites empty. To add a stock to your favoruites page visit the **Stock** page and click add to favourites.

All buttons work correctly and tell users either their list of favourites or that they have none and need to add one. Link to the stock page also works correctly.

Searching for a stock/Stock Page:



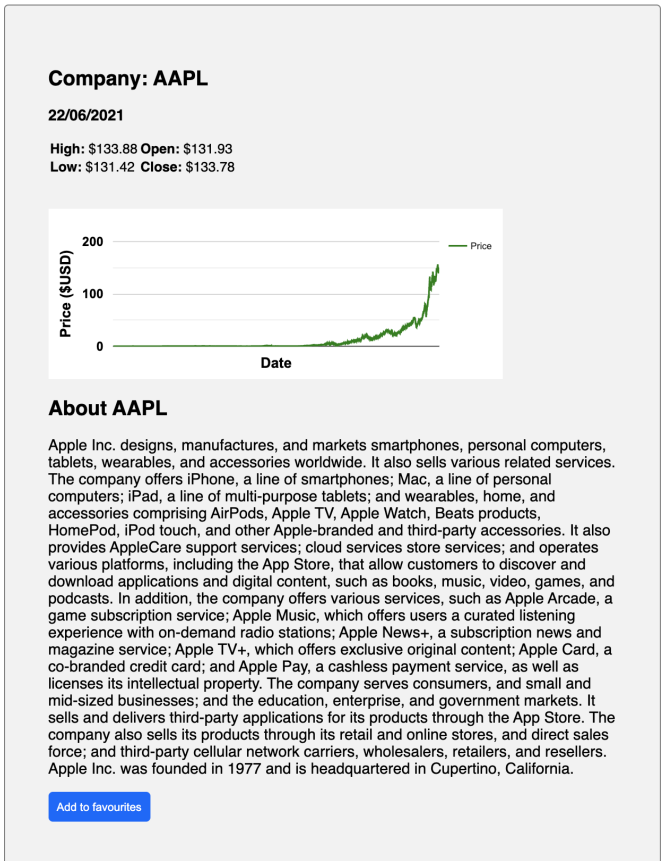
See Historical Data About A Stock.

Search a stock using the ticker:

aapl

What day would you like to see the data about?

22/06/2021



Company: AAPL

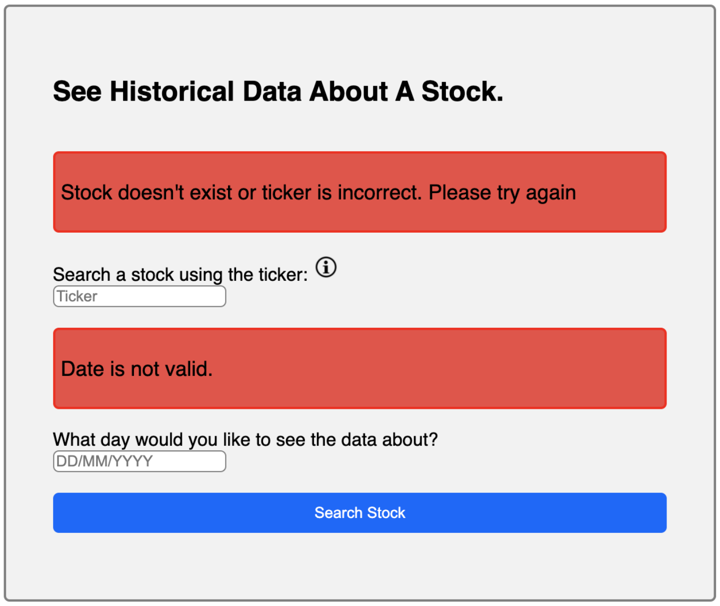
22/06/2021

High: $133.88 **Open:** $131.93
Low: $131.42 **Close:** $133.78

About AAPL

Apple Inc. designs, manufactures, and markets smartphones, personal computers, tablets, wearables, and accessories worldwide. It also sells various related services. The company offers iPhone, a line of smartphones; Mac, a line of personal computers; iPad, a line of multi-purpose tablets; and wearables, home, and accessories comprising AirPods, Apple TV, Apple Watch, Beats products, HomePod, iPod touch, and other Apple-branded and third-party accessories. It also provides AppleCare support services; cloud services store services; and operates various platforms, including the App Store, that allow customers to discover and download applications and digital content, such as books, music, video, games, and podcasts. In addition, the company offers various services, such as Apple Arcade, a game subscription service; Apple Music, which offers users a curated listening experience with on-demand radio stations; Apple News+, a subscription news and magazine service; Apple TV+, which offers exclusive original content; Apple Card, a co-branded credit card; and Apple Pay, a cashless payment service, as well as licenses its intellectual property. The company serves consumers, and small and mid-sized businesses; and the education, enterprise, and government markets. It sells and delivers third-party applications for its products through the App Store. The company also sells its products through its retail and online stores, and direct sales force; and third-party cellular network carriers, wholesalers, retailers, and resellers. Apple Inc. was founded in 1977 and is headquartered in Cupertino, California.

Data is correct and compared to yahoo finance (proved in data integrity) also displays correct stock name, date and description.



See Historical Data About A Stock.

Stock doesn't exist or ticker is incorrect. Please try again

Search a stock using the ticker:

Ticker

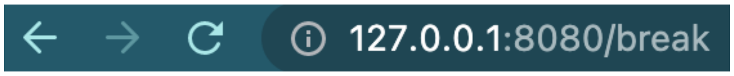
Date is not valid.

What day would you like to see the data about?

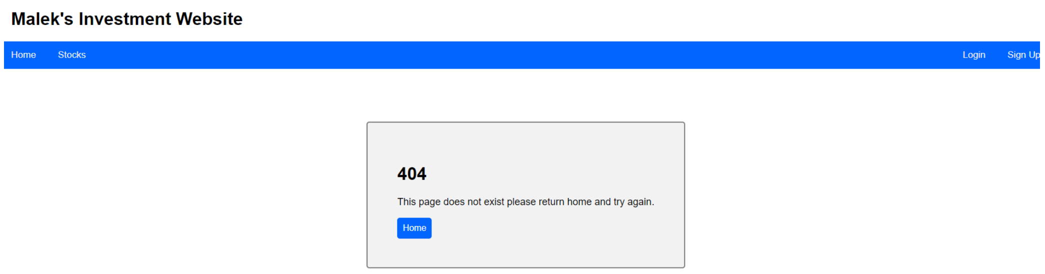
DD/MM/YYYY

Errors show up correctly when the input does not match fields given in the form/search.

404 Page:



127.0.0.1:8080/break



Malek's Investment Website

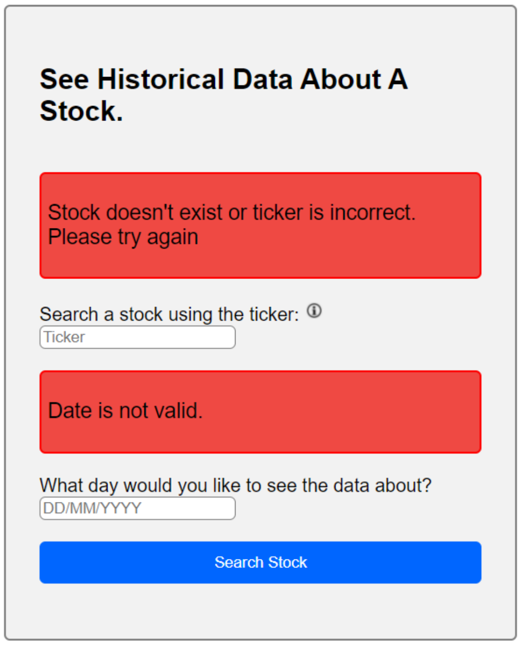
404

This page does not exist please return home and try again.

When searching a page that does not exist, the 404 page works correctly as well as links back to the home page.

Unexpected case testing:
These tests would be very unlikely to happen by a normal user but the point is to make sure that in the one off scenario that it did no errors would occur.

What is being tested?	Expected Result	Actual Result (Pass yes or no). (all tests that do not pass are later addressed in the fixing table.)
Home Page:		
That user is directed to the home page one first visiting website.	To be taken to the homepage on visit.	Yes
That all links to favourites page works.	On clicking the stock page the user should be taken to http://127.0.0.1:8080/stocks. On clicking the favorites page the user should be taken to http://127.0.0.1:8080/favourites.	Yes
Register/Login works using different inputs:		
Testing that using special characters in password still works.	To be added to the database as expected.	**Fix 1.** No. Although I was testing special characters it was given an error. When trying to use more than 5 special characters an error is given.
Testing that using numbers in password still works.	To be added to the database as expected.	No. This problem also occurs when trying to create a password with more than 49 numbers. This is to do with the hashing it is being unable to deal with this many numbers. This is the same reason the special characters do not work.
Testing that using normal characters in password still works.	To be added to the database as expected.	AS the previous 2 tests when trying to make a password longer than 89 characters long an error will be given.
		All of these issues are being caused by the same problem and will be addressed with the same fix.

Testing unable to leave password blank	Failed let user register	**Fix 2. No**
Login check:		
Testing that when putting things like special characters and long numbers into the login box nothing crashes.	For the user to be given the green box telling them they have successfully logged in.	**Fix 3.** No. as previously tested in register box when using long characters and numbers it crashes due to int being too long. However this will likely be fixed with the same code that fixes the problems when registering.
Testing searching a stock form:		
To test these forms I have used the same things that broke my register and login form	The user should be shown to 2 red boxes that tell user stock is invalid as well as date is also invalid.	Yes. Works as expected user is given 2 errors.  **See Historical Data About A Stock.** Stock doesn't exist or ticker is incorrect. Please try again. Search a stock using the ticker: Ticker. Date is not valid. What day would you like to see the data about? DD/MM/YYYY. Search Stock
404 page/ different lengths into url.		
Adding long numbers + special characters + letters into url to make sure nothing can give an error.	Should always direct the user to 404 pages.	Yes. Even using huge strings with different characters always takes the user to the 404 page.
When registering a	This should tell the user	No. The user was able to add a second

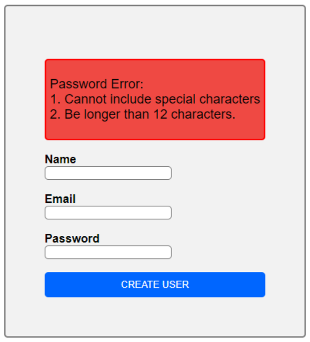
user cannot enter email already in the database.	there is an error and that they should log in as already having an account.	input with the same email.

Fixing unexpected case issues:

Fix 1. After testing special character when using more than 5 as well as numbers longer than 49 and letters sequences that are longer than 89 characters an error given a jinja error:
Before:

```
OverflowError: Python int too large to convert to SQLite INTEGER
```

After:

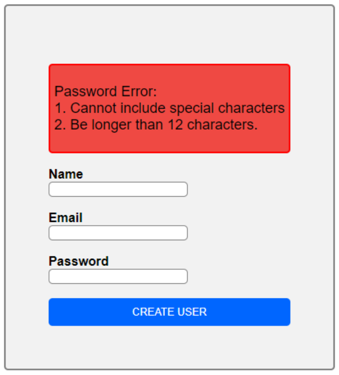


Password Error:
1. Cannot include special characters
2. Be longer than 12 characters.

Fix 2. After testing special characters when using more than 5 as well as numbers longer than 49 and letters sequences longer than 89 characters an error given a jinja error. To fix this issue I created a series of checks for the length as well as the characters the user has inputted. If the password is inputted they are given an error.
Before:

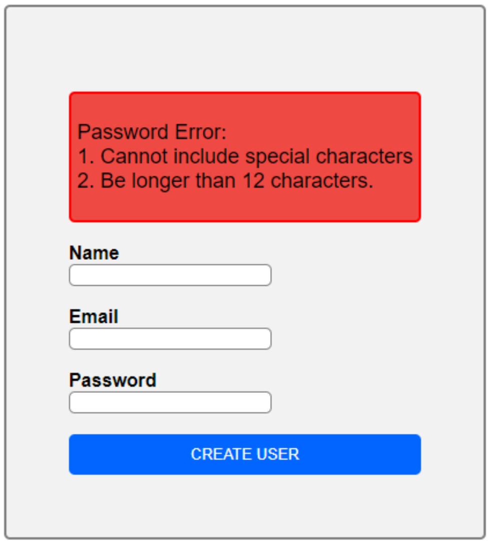
```
OverflowError: Python int too large to convert to SQLite INTEGER
```

After:



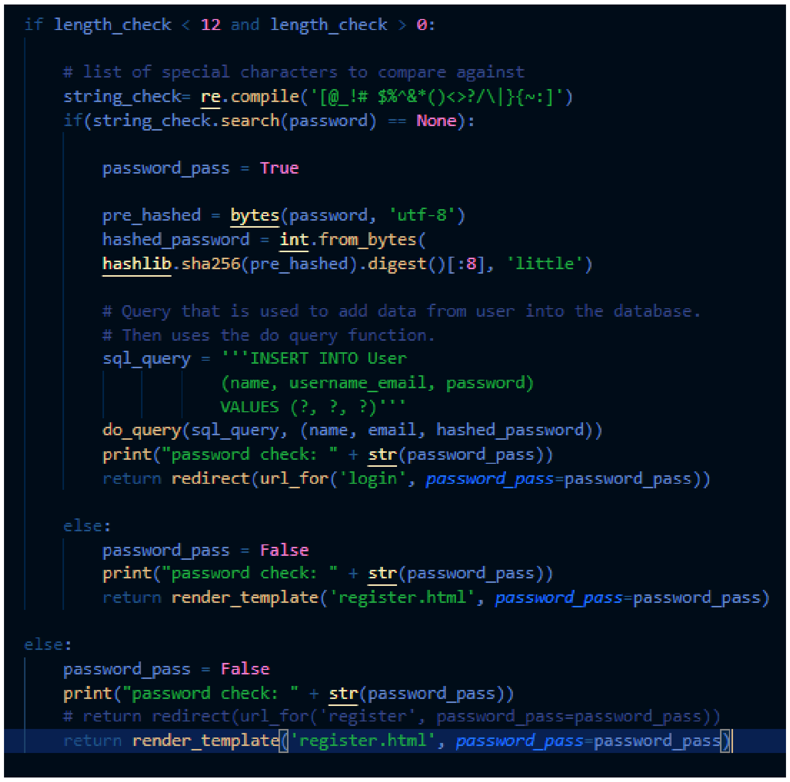
Password Error:
1. Cannot include special characters
2. Be longer than 12 characters.

Fix 3. User is able to leave the password blank and still register. To fix this issue I created a range in which the password must continue to register. Otherwise error is given:



Password Error:
1. Cannot include special characters
2. Be longer than 12 characters.

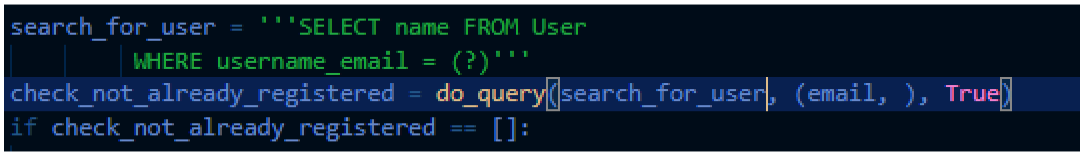
Solution to these 3 issues:



```python
if length_check < 12 and length_check > 0:

    # list of special characters to compare against
    string_check= re.compile('[@_!# $%^&*()<>?/\|}{~:]')
    if(string_check.search(password) == None):

        password_pass = True

        pre_hashed = bytes(password, 'utf-8')
        hashed_password = int.from_bytes(
        hashlib.sha256(pre_hashed).digest()[:8], 'little')

        # Query that is used to add data from user into the database.
        # Then uses the do query function.
        sql_query = '''INSERT INTO User
                    (name, username_email, password)
                    VALUES (?, ?, ?)'''
        do_query(sql_query, (name, email, hashed_password))
        print("password check: " + str(password_pass))
        return redirect(url_for('login', password_pass=password_pass))

    else:
        password_pass = False
        print("password check: " + str(password_pass))
        return render_template('register.html', password_pass=password_pass)

else:
    password_pass = False
    print("password check: " + str(password_pass))
    # return redirect(url_for('register', password_pass=password_pass))
    return render_template('register.html', password_pass=password_pass)
```

Fix 4.

This means before being registered it will now check to make sure that the user isn't already in the database.


```
search_for_user = '''SELECT name FROM User
        WHERE username_email = (?)'''
check_not_already_registered = do_query(search_for_user, (email, ), True)
if check_not_already_registered == []:
```

Edge/boundary cases:

I have managed to find one way that I am able to experience problems within my website. This edge case is when trying to search a stock some smaller/lower market cap stocks although on the US stock market will not show up. As I am limited to the data from my API I have no control on what data is available. This means that although most stocks will work there will be some cases in which no data is available.

Website standard:

My goal for the media standard was to have a good looking website that had both functionality while still being easy to navigate but at the same time looking aesthetically pleasing.

Purpose of each page:

1. Home page, to tell the user a brief piece of information about the use of the website and what the point of it is.
2. Stock Page - The main part of the website that the user is able to search and find out historic information and data about the stock they want to know about. It also allows the user to add it to their favourites.

3. Login and sign up page - Lets users create an account which then gives them more funcuaility while using the website e.g. allows them to create a favorites page that they can add all stocks they want quick access to.
4. Favourites page - Allows them to see all stocks added from the stock page. Only able to be accessed once logged in.

Relevant implications

1. Usability - the degree to which something is able or fit to be used. Usability is a very important part of all websites. It means that the website is both easy to navigate and everything works as expected. Another aspect is that anyone should be able to use the website as expected without any prior knowledge.

   How I achieved usability in my website is through a few different ways. The first By trying to keep the layout as simple as possible by doing this it doesn't overwhelm the user and allows them to clearly understand what is being shown to them. I have also added titles and paragraphs to explain to the user what the use of something is which helps the user to not be confused.

2. Aesthetics - a set of principles concerned with the nature and appreciation of beauty. Aesthetics are a key thing in all websites, it is the looks, style and layout of the website. This is because it is what the user will be constantly looking at so making something that is very easy to read and look at.

To achieve making the best visually appealing website I could I took many things into consideration. These included formatting, colours used, fonts and sizing. Some problems that users could occur are things like poor eyesight and also being colour blind. While taking these things into account while designing my website I chose to go with colours that are not affected by colour blind people (blue, white and black).

3. Sustainability and future proofing - These concepts mean that the project will be able to cope both with scale and future developments with regard to subject matter.

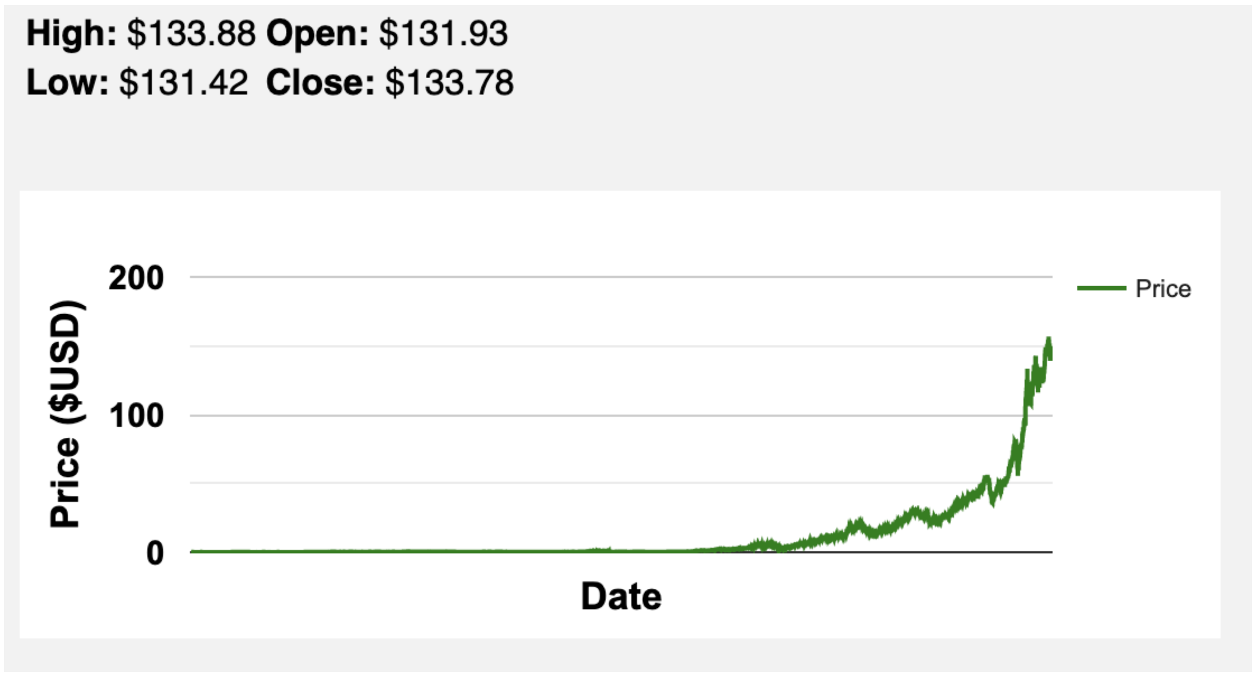
   My project has been future proofed mainly due to the implementation of the API. This is because by using this API I do not personally have to update any data as it is all done for me. As a result of this I am able to leave my project without checking it and can be fairly confident that it is functioning as expected as long as the API is still running. Examples of subject matter development would be the bringing on of cryptocurrency markets once an API is available. This would be relatively simple by replicating the current code I have for Yahoo finance. In terms of scalability it would seem unlikely the current database will be overwhelmed but options would include migrating to the cloud.

Data Integrity:

Data integrity is the accuracy and completeness of the data on my website. I have used the YFinance API to get my data for stocks. As shown in the images below the top image is the graph located on my website. The data graphed through the website is the same as the data as shown on Yahoo Finance, proving the data gathered from my API and graphed is correct.

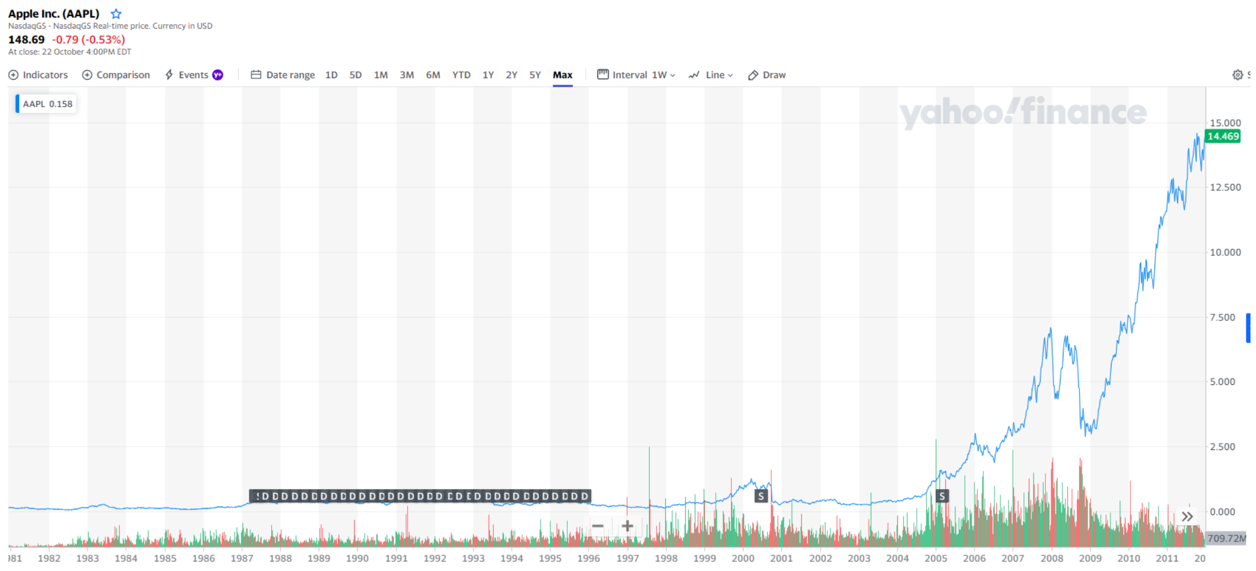
To ensure all data was reliable I chose to use yahoo finance. This is because yahoo finance is a huge company that has constantly provided users data for years. Therefore I was confident that by using their tool I would be providing accurate and reliable data.

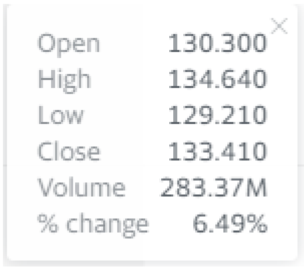
Figure 1: The website data for AAPL as at 22/06/21



High: $133.88 **Open:** $131.93
Low: $131.42 **Close:** $133.78

Figure 2: The data for AAPL, taken from Yahoo Finance to check the data integrity.





Open	130.300
High	134.640
Low	129.210
Close	133.410
Volume	283.37M
% change	6.49%

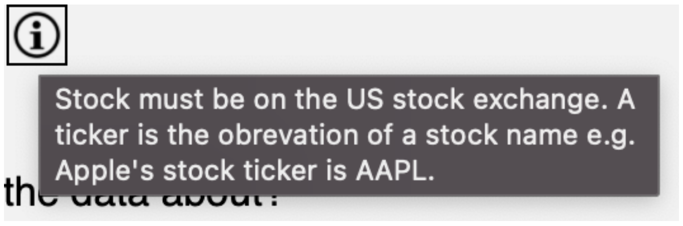
The high low data on my website differs with yahoo finance. Through analysis, Yahoo Finance took high and low data from the last 24 hours rather than the beginning of the date measured. Therefore the data being gathered from the API is correct, but has a different philosophy than Yahoo Finance.

Data integrity is shown again when the user is both logging in and also registering. As seen in the website and programming testing, the user's input is tested by checking that at all data is matched from input from the user to when it is being stored in the database as seen in the testing page.

Feedback:

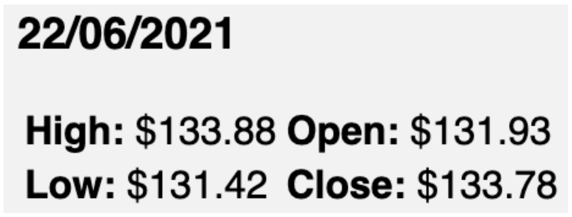
After asking people to try out my website I received a few comments:

1. "Font size is quite small and hard to read." After taking this piece into consideration I then decided to try and scale up my text sizes to make it as easy as possible for the user to read while still being functional and looking good.
2. "Clarify that it is a US stock." This was a good suggestion and I added it into the stock page.



Stock must be on the US stock exchange. A ticker is the obrevation of a stock name e.g. Apple's stock ticker is AAPL.

3. "Default to today's date in the search." Although this may be better in some cases the point of the website is to see historical data therefore I will not make that change.
4. "Input boxes need to be bigger." This was a suggestion that also related to the sizing of the font so I have tried to enlarge everything to make it better for usability.
5. "Print date once searched." This is a good suggestion that I have decided to make.



22/06/2021

High: $133.88 **Open:** $131.93
Low: $131.42 **Close:** $133.78

6. “Layout of search.” Although this could be done for better use in some cases I find that overall it is the best structured how it currently looks.
7. “Loading button.” Although this may be useful for some less tech savvy users search engines display when pages are loading therefore by adding another seems pointless.

Sources of information:

These are some websites that I have used for information and some which I have based my own code off.

1. https://jsfiddle.net/API/post/library/pure/. This was used to create the google chart for the stock page.
2. https://pypi.org/project/yfinance/. This is where I learnt how to use the new API.
3. https://testdriven.io/blog/flask-sessions/. What I based my use of sessions off.
4. https://flask.palletsprojects.com/en/2.0.x/. I used examples from the flask docs throughout my project.
5. https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673. This was used when converting data from my API into the google chart.
6. My 404 page was based on code from others in class. (Aden and Riley).

7. Making sure the password can be stored into database https://www.codespeedy.com/check-if-a-string-contains-a-special-character-or-not-python/.
8. https://nitratine.net/blog/post/how-to-hash-passwords-in-python/. Used to learn how to hash users passwords.
9. https://www.w3schools.com/. Used a lot to find details on css and how to use html and python.