**Code review:**

**Requirement 1**

Overall, the code looks great. It implements all the required functions and can handle various error cases. Also, it is easy to read and understand.

Implementation: The implementation of the code is straightforward and easy to follow.

Readability: The code is well-structured, with each function performing a single task. The variable names are descriptive, and the code has comments to explain what certain sections are doing.

Dependencies: The code has a few dependencies such as Flask and SQLite, but they are well-documented and widely used libraries, making the code more reliable and maintainable.

However, there are a few areas that could be improved:

Error Handling:

According to sign up test cases 2. Submitting with black value, it reported error message: Invalid Email Address. However, it will make more sense if it reports ‘Please fill in all required fields.’

According to sign up test cases 7. Capitalized email address, it is case-sensitive to think ‘zhangsan@example.com’ and ‘Zhangsan@example.com’ are two different mailboxes, yet they should be the same mailbox and should report ‘This email has been registered already.’

Code Duplication:

There is some duplication in the code. In the signup and login routes, they both connect to the database and execute queries. It is better to create a separate function to handle database interactions and reuse it in both routes.

**Requirement 2**

Add comments: It's better to add comments to improve readability and maintainability.

Error handling: Including all kinds of error handling mechanisms that we can think of so far.

Code optimization: The code can be optimized by reducing repetitive code. For example, we can combine the if statements that check if tradeType is "Buy" or "Sell" into a single statement.

**Requirement 3**

Add comments: It's better to add comments to improve readability and maintainability.

Handle exceptions: Currently, the code catches all exceptions but only prints an error message. It would be better to handle the exceptions more specifically and perhaps return an error code or message to the caller.

Data security: When working with a database, it's recommended to use a context manager (with statement) to handle opening and closing the connection automatically. This can prevent resource leaks and ensure that the database is properly closed even if an exception occurs.

Consistency: The code uses f-strings in some places but not in others. It would be good to be consistent and use f-strings throughout the code.

**Requirement 4**

The code is well-structured and easy to follow. The functions are modular, have a clear purpose, and are named appropriately.

The code uses descriptive variable names, which makes it easy to understand what the code is doing.

The code handles exceptions and prints out informative error messages.

It would be good to include more comments in the code to explain what is happening, especially for complex functions like ‘restrict’.

The ‘get\_portfolio\_statistics’ function is doing a lot of different things and could be refactored into smaller, more modular functions.

**Requirement 5**

Code duplication: There are some instances of code duplication, such as the repeated use of database connections and cursors. Consider creating a function that handles database connections and cursors, and use that function throughout the code.

Modularity: The code is relatively modular, but it might benefit from further abstraction into smaller functions that perform specific tasks.

Testability: Breaking the code into small functions could make it easier to test individual functions.

**Requirement 6**

We can use more descriptive variable names, particularly for variables that are used throughout the code. For example, instead of using generic names like "trace1" or "returns", consider using more descriptive names like "priceTrace" or "stockReturns".

Additionally, it would be helpful to include more detailed comments within each section of the code, particularly for more complex algorithms like the linear regression function or the MACD calculation. This would make it easier for others to understand the code and make modifications if necessary.

**Requirement 9**

The code fulfills the requirements and specifications.

In general all pages were designed to maintain a consistent style.

Codes for welcome, register, log in and admin pages are readable and well-organized.

However, too much information in finance home and transaction page. It is very important to conceptualize the page layout well in advance before writing the code.

**Requirement 11**

The ‘formatPricePerShare()’ function iterates through each row in the table and formats the price per share to two decimal places. However, it assumes that the price per share is stored in a hidden ‘span’ element with class ‘hidden-price-per-share’. It would be helpful to include a comment in the code to clarify this assumption.

The ‘showDialog()’ function simply displays an alert when called. However, the function name does not accurately reflect its purpose, and it is not clear when this function would be called.

There is a mix of single and double quotes used throughout the code. It would be good to be consistent in the choice of quote style.