Documentation and Xunit testing

Documentaion Format:
https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/documentation-comments

Unit testing is an essential aspect of software development, especially for a complex application like our stock prediction system. This system, with its React frontend, C# backend, MySQL database for storage, and APIs for data interaction, presents numerous components that could fail or behave unexpectedly. By focusing on unit testing, particularly for the "get" methods in our application, we can significantly enhance the reliability, maintainability, and overall quality of our code.

Unit tests allow us to verify the correctness of individual components (functions) independently from the rest of the application. For instance, testing our `GetUsers`, `GetStocks`, or `GetStockPredictions` methods ensures that they return the expected results under various conditions. This is crucial for our stock prediction application, since accurate and reliable data retrieval is the foundation for making predictions. By implementing unit tests, especially for critical "get" methods that fetch data from the database, we can catch and fix bugs early in the development cycle. This early detection prevents the escalation of minor issues into more significant problems later on, reducing the cost and effort required for fixes.

With a well-tested codebase, we can make changes or refactor code, knowing that our tests will catch any unintended consequences of these modifications. This is particularly important in a stock prediction application, where algorithms and data retrieval mechanisms may need adjustments to improve prediction accuracy or performance. Writing unit tests also encourages us to write clean and efficient code. Functions and methods that are easy to test tend to be well-designed and follow good software engineering principles. For our application, this means a more robust, scalable, and maintainable codebase.

For our stock prediction application, focusing on unit testing for "get" methods is not just about preventing errors; it's about ensuring the integrity, reliability, and performance of our system. Given the dynamic nature of stock data and the critical importance of accurate data retrieval, unit tests provide a safety net that helps maintain and improve the quality of our application over time.