ATU - Department of Computing (Donegal)

Module	Commercial Programming 1	**Credits**	5
Module Coordinator	Dr. Shane Wilson	**Credit level**	6
Issue date	25/02/2025	**Submission deadline**	11/04/2025
Assignment weighting	70% of the module mark	**Type**	Team assignment
Title	Investment portfolio management system		

Assignment details:

This assignment requires you to demonstrate your ability to collaboratively design, develop, test and document a software artifact using modern software development tools, techniques and best practices.

User requirements:

You have been hired to create a stock and cryptocurrency portfolio management application. As a user I would like to be able to:

- **Add and withdraw funds to and from the application**. – Funds should be in USD and are used to buy assets (stocks and cryptocurrencies). I can only withdraw <= the total USD cash balance within the application. Assets are not automatically sold if I attempt to withdraw >= my current USD cash balance.
- **Simulate the purchase and sale of assets (stocks or cryptocurrencies) from a variety of markets** (FTSE, Dow, DASDAQ, et cetera)**.**
- **Get the total value of all assets within my portfolio(s)**.
- **Get summary information on my investments. This may be individual assets such as Microsoft stocks, all assets of a particular type or the total value of all my assets.** For example, I may have purchased 10 units of Microsoft stock each month for three months. I should see the cost of each purchase, the difference based on the current live value of Microsoft stock and my overall profit or loss.
- **Get a summary of my asset purchases or sales over a specified time range.**
- **Get a summary of my stock and cryptocurrency portfolios.**

Using a live API

- **Get a list of trending assets within a market or region** (US, GB, DE et cetera).
- **Get live information on assets (stock or cryptocurrencies).**

Accepting the assignment and joining the team

The team CA was kicked off during class on February 25th. If you missed this class. Note the team you have been assigned to (check your email). Login to your Github account and click on the link below and join the appropriate team. If you have difficulty accessing the link, try using a different browser (Chrome usually works fine).

The starter code repository is available here: https://classroom.github.com/a/P_Ji4SeJ

Additional guidance:

- You are encouraged to expand upon the functional requirements outlined on the previous page to demonstrate your technical competencies with C# .NET and associated technologies. For example, you may want to include historical data or watchlists that notify the user if the value of an asset (within the portfolio or in a live market) reaches a user specified value.

  However, at a minimum you must provide implementations for the PortfolioSystem interface and associated classes in the skeleton code provided in the starter repository.

- You can implement almost all of the required functionality without connecting to a live data service and instead use mock data provided by the MockClient class or another class that loads sample data. Carefully consider the data that should be returned by the test doubles.

- **You are not** required to implement persistent storage of data. An in-memory database using appropriate C# Collections is fine. (https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections)

- **EVERY** member of the team is expected to make meaningful contributions to the software implementation, testing suite and code review process. You will be asked about your contributions during the in-class assessment Q&A session.

- Your solution should provide a user interface (UI) to allow the user to interact with the system. This can take the form of a text based user interface.

- Your solution should start with the following fictional asset purchases.
  - 10 units of Tesla stock purchased on 01/10/25 at a price of $326.91 USD
  - 20 units of Apple Inc stock purchased on 06/03/25 at a price of $248.09 USD
  - 12 units of Nvidia Corporation stock purchased on 25/02/25 at a price of $130.28 USD
  - A $2,000 purchase of bitcoin on 25/02/25. Roughly equating to 0.022724 bitcoin. The average price for bitcoin that day was $88,010.44 USD.

- Your implementation must be written in C# and comply with the Microsoft C# style guide. You may add additional methods and attributes to your implementation(s) of the C# Portfolio interface and associated classes as required.

- **<u>It should not be possible to crash the application under normal operation.</u>**

- **<u>You should not modify the signatures of the fields, properties, methods listed in the Portfolio interface or the classes provided.</u>** If I have made a mistake let me know. You can add additional class members as you see fit.

- You are free to use any stock market API to retrieve data. I recommend that you first consider the following APIs:

  YH Finance API - https://financeapi.net/
  Rapid API Yahoo Finance API - https://rapidapi.com/apidojo/api/yahoo-finance1/

  You can view live stock data at: https://uk.finance.yahoo.com/

- Postman is an excellent service and application that you can use to construct REST API calls: https://www.postman.com/

- You should include your API key(s) with your submission for testing purposes.

- Any external libraries or 3rd party source code should be referenced appropriately within your code. Failure to do so could be considered as plagiarism. If you reuse code for 3rd party sources or online make sure that you can explain how it works. You will be asked to explain your code during the formal submission review in week 13.

- Remember to double check your commit user name and email. If these are not set correctly on your development PC, commits to the repository may not be correctly associated to you. Use a github no reply email address for your git user.email setting.

Assignment deliverables:

1. Relevant system design documentation and rationale for design choices made. Highlight any functional elements that go beyond the functional requirements listed on page 1.
2. Visual Studio C# project that includes:
   - Software implementation.
   - Test suite.

- Relevant source code documentation (code comments, C# documentation comments).

3. A short report (5 pages maximum) outlining how the team approached the delivery of the assignment and lessons learned. Screenshots of relevant MS Teams conversations, project board, issues or pull requests should be included in the main body of the report or appendix. Material within the appendices is considered supplementary and as a result may or may not be read by the marker. The appendices do **<u>not contribute</u>** to the overall report page count.
4. Individual team member contribution sheets (see last page of the assignment brief).

Submission details:

- The team lead should submit the team's solution to blackboard by 11/04/2025 at 22:00.
- Blackboard submission should contain:
    1. A zip file containing the source code.
    2. Relevant design documentation.
    3. The team's report.
    4. Team academic integrity form signed by all team members.
    5. The team leaders individual team member contribution sheet.
- **Each member** of the team is also required to submit an individual team member contribution sheet to BBL. **<u>Failure to do so will result in a 5% reduction in your individual assignment mark.</u>**

Submission demo and Q&A:

Each team will be required to demonstrate their submission during class on the 28th of April. **<u>All team members are required to attend this formal assessment.</u>** Failure to attend will lead to a mark of zero. A schedule for the demos will be posted on advance of these dates. You will only be required to attend your teams scheduled assessment session.

Late penalties:

In accordance with ATU Marks and Standards policy, a 5% daily penalty will be applied to late submissions.

Plagiarism:

At ATU Donegal plagiarism is defined as: The act of presenting as your own, the words or ideas of someone else, whether published or not, without proper acknowledgement, within one's own work is called plagiarism.

Plagiarism is a form of cheating and is dishonest. Suspected incidents of plagiarism will be dealt with through the University's disciplinary procedures. More Info:

https://www.atu.ie/sites/default/files/2024-02/aqae022-academic-integrity-policy-1.pdf

The use of generative artificial intelligence (AI) tools:

The use of generative AI as a tool to assist in the process of creating your submission is not permitted.

Assessment criteria:

Classification	Software implementation (40%)	Software testing (20%)	Code readability and documentation (20%)	Project management, team collaboration, code reviews (20%)
80% - 100% Outstanding work	An error free, fully functional software implementation demonstrating outstanding knowledge and application of software development skills using C# .NET. The implementation exceeds the specified functional requirements with additional functional elements demonstrating retrieval and processing of data from a live API. The submission demonstrates an outstanding indepth understanding and application of C# .NET.	Exceeds excellent criteria by demonstrating a highly competent and professional use of the testing framework to create a robust and comprehensive suite of integration and unit tests. Outstanding application of testing best practices and tooling.	All code organised clearly and logically structured. Provided documentation (code and standalone) is excellent in terms of presentation, clarity consistency and conciseness. Code complies fully with Microsoft style guide. Outstanding and consistent use of industry standard code analysis tools to enforce code correctness and comprehension.	Outstanding use of technology to deliver the project. Extensive and advanced use of communication and collaboration tools throughout the duration of the project. Extensive evidence of detailed pre-integration code reviews / checks. Clear evidence of several types of code refactoring taking place as a result of code reviews. All team meetings documented clearly and concisely with all relevant information captured. Outstanding use of project issues and PRs to capture and document project progress.
70% - 79% Excellent work	An error free, fully functional software implementation demonstrating excellent knowledge and application of software development skills with C# .NET. The implementation has implemented all of the specified functional requirements using a mock data	TDD clearly informing and driving development throughout the project. Excellent application of testing best practices and tooling. Comprehensive suite of integration and unit tests.	All code organised clearly and logically. Excellent documentation provided. Code complies fully with Microsoft style guide. Excellent and consistent use of industry standard code analysis tools to	Excellent use of technology to deliver the project. Excellent use of communication and collaboration tools throughout the duration of the project. Clear evidence of comprehensive code reviews as pre-commit checks. Clear evidence of several types of

	service (no live data). The submission demonstrates an excellent understanding and application of C# .NET.		enforce code correctness and comprehension.	code refactoring taking place as a result of code reviews. All team meetings documented clearly and concisely with all relevant information captured. Excellent use of project issues and PRs to capture and document project progress.
60% - 69% **Good quality**	An error free software implementation demonstrating competent software development skills with C# .NET. Almost all of the functional requirements have been successfully implemented and act as required. System used data from a mock data service. The submission demonstrates a good understanding and application of C# .NET.	Clear evidence of TDD being used and derived from the requirements. Good code coverage and robust use of core testing features.	All code organised clearly and logically. Documentation is clear, consistent, readable. Code doc comments provided. Code complies with Microsoft style guide. Code analysis tools to enforce code correctness and comprehension used throughout the project.	Good use of technology to deliver the project. Pull requests incorporating code reviews examining correctness, comprehension and consistency with some subsequent refactoring taking place. Most team meetings documented but inconsistently. Some relevant information missing. Project issues and PRs used to capture and document project progress.
50% - 59% **Acceptable**	Software implementation contains several errors or bugs. Some of the core functional requirements are not implemented fully or correctly. The submission demonstrates a correct but limited understanding and application of C# .NET.	Some evidence of TDD being adopted but test suites are poorly implemented and do not provide adequate code coverage or range of tests. Poorly named tests.	Most code organised clearly and logically. Comments mostly present but inappropriate level of detail. Limited used of code documentation comments. Code mostly complies with Microsoft style guide. Some tooling for code analysis.	Acceptable use of technology to deliver the project. Limited evidence of code reviews or insufficient examination of code correctness, comprehension and consistency. No subsequent refactoring. Issues and pull requests lack detail. Team meetings poorly documented. Project issues and PRs used inconsistently across the team.

40% - 49% Adequate work	Software implementation contains a significant number of serious bugs or errors. Few of the specified functional requirements have been successfully implemented. Draft implementation of several functional elements are incomplete. The submission demonstrates a weak/incorrect understanding and application of C# .NET.	No evidence of TDD driving development. Plain old unit testing has been adopted (POUT). Tests poorly written, insufficient code coverage.	Limited attention to code organisation. Lack of code commentary. Consistency and readability are an issue. Some appropriate rules and conventions followed to maximise readability (tabbing, whitespace, naming). No effective used of code analysis tools.	Limited or inappropriate use of technology to deliver and manage the project. No evidence of code reviews and or subsequent refactoring taking place. Issues and pull requests are poorly documented. Meetings not documented or significant elements missing. Issues and PRs not used correctly or rarely.
Marginal Fail 35% - 39%	Software implementation is functional but contains several serious bugs. Very little of the desired functional requirements have been implemented or do not function as required. The submission demonstrates little or incorrect understanding and application of C# .NET.	No evidence of TDD being adopted. Testing is sporadic or ad hoc in nature.	Code shows little attention to organisation. Some code commentary. Inconsistent application of code style. No effective used of code analysis tools.	Collaboration and communication tools are not used effectively. No evidence of code reviews/refactoring. Issues and pull requests if present are poorly documented. Meetings not documented or significant elements missing. Issues and PRs not used or used incorrectly.
Fail 0 – 34%	Software implementation does not work or compile. Few of the specified functional requirements appear to be implemented. The submission demonstrates a little understanding and incorrect application of C# .NET.	No formal testing apparent.	Code is largely incoherent. No documentation. No attention to coding rules or conventions. No effective used of code analysis tools.	Little or no collaboration or communication across the team. Issues and pull requests poorly documented or not present. No evidence of code reviews/refactoring. No recording of team meetings.

Team assessment

Use the following table to evaluate the relative contributions of your team members toward the entire project effort. Things you should consider include:

- The quality and quantity of contributions.
- Team-player attitude.
- Each members engagement over the entire duration of the project.
- Any other aspects that you feel are relevant.

The method is as follows:

- Take 100 points, and divide them among the ***N*** team members, including yourself.
- Give points based on your opinion of what proportion of the credit each member deserves.
- Remember to allocate yourself points.

Each team member's total project grade will be the team grade ***T***, multiplied by a peer-evaluated adjustment factor ***A***. If each team member is allocated the same share of points, ***A*** = 100% and each member of the team will be assigned the team grade ***T***. The adjustment factor ***A*** for member ***X*** will be averaged from all members except ***X***.

Members failing to submit peer evaluations will receive (A-5) * T.

In the following table allocate each member of the team (including yourself) their share of the 100 points available. List the major contributions of each team member. Where possible list exemplary repository commits, pull requests or issues to evidence team member contributions.

Team member (Print student ID)	**Description of the work undertaken by the student in completion of the assignment**	**Points allocation**

The mark assigned to each member of the team will be based on the module coordinators evaluation of their contributions to the project. The peer assessment form will be used to inform the module coordinators decision in this regard.