|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | Contemporary Software Development | **Credits** | 10 |
| **Module Coordinator** | Dr. Shane Wilson | **Credit level** | 8 |
| **Issue date** | 23/01/2023 | **Submission  deadline** | 27/02/2023 at 22:00 |
| **Assignment weighting** | 70% of the module mark | **Type** | Team assignment |
| **Title** | Aviation CO2 emission application | | |

# 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.

## Problem:

Your team has been hired by a climate change NGO to develop an application to track, analyse and provide insights regarding CO2 emissions by the aviation industry. The client would like the application to provide answers to the following questions:

* At any given moment, what are the estimated total global CO2 emissions of all of the scheduled live flights currently in the air?
  + Consider only those flights with specified departure and destination airports.
* What are the estimated global CO2 emissions over the last five years (2017-2022)?
  + What are the emissions for each year/month within that timeframe?
* What are the top twenty most polluting routes globally, regionally (USA, Europe) and by country.
  + Within each region and country, differentiate by domestic and international flights.
* What are the total CO2 emissions by each Airline?
  + Within this, filter by routes and short-haul and long-haul.
* What are the estimated CO2 emissions by airport?
  + Show a breakdown of both arrivals and departures and then short-haul and long-haul.
* Top twenty countries responsible for aviation CO2 emissions. Filter by:
  + By domestic flights only.
  + By international flights only.
  + Combined CO2 emissions from domestic and international flights.
  + By country of aircraft registration.
* Top twenty aircraft types responsible for the most CO2 emissions.

## Additional guidance:

* You should aim to go beyond answering the questions listed on the previous page and provide additional insights and analysis using available data sources. Discuss your ideas with the module coordinator.
* When calculating total emissions, carefully consider how you can use available live and historical data to calculate these values. Also consider the possible timeframes:
  + Live flights.
  + Each day, week, month, year.
* The application **must** be created using one or more of the following programming languages.
  + C#
  + Java
  + Python
  + LINQ/SQL
* 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.
* All source code should comply with the Google style guide for the languages that you use to implement the application.
* All source code should be appropriate commented and documented.
* **You are not** required to implement persistent storage of data. An in-memory database using appropriate collections is fine.   
    
  Java collections: <https://docs.oracle.com/javase/tutorial/collections/index.html>)   
  C# collections: <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections>   
  Python collections: <https://docs.python.org/3/library/collections.html>

However, if you need to supplement data obtained from the live aviation data sources with persistent datasets, you may want to consider using JSON files. If you are considering using a database solution, please discuss this with the module coordinator before implementing a persistent storage solution.

* **EVERY** member of the team is expected to make meaningful contributions to the software implementation, testing and code review and documentation processes.
* 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 or graphical user interface.
* **It should not be possible to crash the application under normal operation.**

## Data sources:

You are free to use any free data sources to retrieve aviation information and calculate CO2 emissions. You should justify the selection of data sources within your assignment submission. I highly recommend that you first consider the APIs and data sources:

**Realtime flight information**

AirLabs: <https://airlabs.co/>

Aviationstack: <https://aviationstack.com>

**Additional sources on aviation emissions**:

myClimate flight emission calculator: <https://co2.myclimate.org/en/flight_calculators/new>

myClimate flight emission calculator algorithm:

<https://www.myclimate.org/fileadmin/user_upload/myclimate_-_home/01_Information/01_About_myclimate/09_Calculation_principles/Documents/myclimate-flight-calculator-documentation_EN.pdf>

C Level Flight carbon calculator (see FAQ for algorithm used): <https://www.clevel.co.uk/flight-carbon-calculator/>

Eurocontrol: <https://www.eurocontrol.int/>

ICAO: <https://www.icao.int/Pages/default.aspx>

OCED Air Transport CO2 emissions database: <https://stats.oecd.org/Index.aspx?DataSetCode=AIRTRANS_CO2>

Verifavia algorithm for aircraft CO2 emissions: <https://www.verifavia.com/greenhouse-gas-verification/fq-how-are-aircraft-co2-emissions-calculated-11.php>

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

## Additional notes:

* Short-haul flights are any flight <1,500 km.
* Long-haul flights are any flight >=1,500 km.
* Domestic flights are any flight where the departure and destination airports are within the same country.
* International flights are any flight where the departure and destination airports are in different countries.

## Assignment deliverables:

1. System documentation:
   1. Selection and justification of data sources used by your application. (1/2 page)
   2. Highlight any functional elements that go beyond the functional requirements listed on page 1. (1 page bullet points)
   3. Your methodology and rationale for calculating the CO2 emissions in each of the questions presented on page one. This should include an explanation the algorithm and input data used to calculate CO2 emissions for airlines, aircraft, flights, regions, countries and airports. (2 pages)
   4. System design and rationale for design choices made. This should include a high level diagram describing you’re the architecture of your system.
   5. Relevant system design documentation. (2-3 pages)
2. Assignment repository containing:

* IDE (Visual Studio, VS Code or IntelliJ) project and all necessary files to rebuild your solution.
* Fully documented source code.
* Test suites.
* Relevant source code documentation (XML code comments, Javadocs)
* Any persistent data files (CSVs, JSON, XML files)
* Any SQL database backups if relevant.

1. A short report (10 pages maximum) outlining how the team approached the delivery of the assignment and lessons learned. Each team member should include an individual reflection section (1/2 page each). Screenshots of relevant MS Teams conversations, GitHub 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 **not contribute** to the overall report page count. **Remember to include a link to your repository**.

## Submission details:

* The team lead should submit the team’s solution to blackboard by 27/02/2023 at 22:00.
* Blackboard submission should contain:
  1. System documentation.
  2. A zip file containing the source code and all necessary files required to rebuild the solution.
  3. The team’s report.
  4. Assignment coversheet signed by all members of the group.
  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. **Failure to do so will result in a 5% reduction in your individual assignment mark.**

## Late penalties:

In accordance with ATU Donegal policy on late submissions the following penalties will apply. Up to one week late, a 15% reduction in the total mark would be applied: e.g. a mark of 50% would become 43%. Up to two weeks late, a 30% reduction in the total mark would be applied: e.g. a mark of 50% would become 35%. Work received more than two weeks late will receive a mark of zero.

## Plagiarism:

**Assignment submissions will be reviewed for plagiarism**. All external libraries and 3rd party source code should be appropriately referenced. Plagiarism is a form of cheating and is dishonest. Suspected incidents of plagiarism will be dealt with through the Institute’s disciplinary procedures. More Info: [www.lyit.ie/About/Policies-Publications/Quality-Assurance](http://www.lyit.ie/About/Policies-Publications/Quality-Assurance)

## Assessment criteria:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification** | **Software design (10%)** | **Software implementation (30%)** | **Software testing (20%)** | **Code readability and documentation (20%)** | **Project management, team collaboration, code reviews (20%)** |
| **80% - 100%**  **Outstanding work** | Detailed and innovative software design and accompanying rationale, that clearly demonstrates application of advanced, relevant and perceptive software design, informed by current research and best practices. | An error free, highly complex and elegant software implementation demonstrating outstanding software development skills. The implementation greatly exceeds the specified functional requirements demonstrating outstanding levels of creativity and innovation. | Exceeds excellent criteria by demonstrating a highly competent and innovative use of testing framework to create a robust and comprehensive suite of tests. | All code organised clearly and logically. Provided documentation (code and standalone) is excellent in terms of presentation, clarity consistency and conciseness. Code complies fully with Google style guide. | 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. |
| **70% - 79%**  **Excellent work** | Comprehensive software design and rationale that demonstrates application of software design best practices. | An error free, complex and elegant software implementation demonstrating excellent software development skills. The implementation greatly exceeds the specified functional requirements demonstrating excellent levels of creativity and innovation. | TDD or BDD clearly informing and driving development throughout the project. Excellent application of testing best practices and tooling. | All code organised clearly and logically. Excellent documentation provided. Code complies fully with Google style guide. | 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 code refactoring taking place as a result of code reviews. |
| **60% - 69%**  **Good quality** | Clear, logical and informed software design and rationale which embodies key software design principles | An error free software implementation demonstrating competent software development skills. All of the functional requirements implemented. | Clear evidence of TDD or BDD 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. Javadocs provided. Code complies with Google style guide. | Good use of technology to deliver the project. Pull requests incorporating code reviews examining correctness, comprehension and consistency with some subsequent refactoring taking place. |
| **50% - 59%**  **Acceptable** | Coherent software design embodying core principles of software design with rationale for design decisions taken. Rationale provided is acceptable but more appropriate design solutions evident. | Software implementation functions but contains some minor errors or bugs. Some of the core functional requirements are not implemented fully or correctly. | Some evidence of TDD or BDD 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 javadocs. Code mostly complies with Google style guide. | 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. |
| **40% - 49% Adequate work** | Inappropriate software design. Little or no consideration of core principles of software design. Rationale may be weak in nature or design decisions inappropriate. | Software implementation functions but contains a significant number of serious bugs or errors. Several key functional requirements are not implemented or do not function as required. | No evidence of TDD or BDD 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). | 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. |
| **Marginal Fail**  **35% - 39%** | Incomplete or incoherent software design presented. Design does not embody principles of good software design (high cohesion, low coupling, abstraction, flexibility, extensibility, reusability, testability). No rationale. | Software implementation is functional but contains several serious bugs and few of the specified functional requirements. | No evidence of TDD or BDD being adopted. Testing is sporadic or ad hoc in nature. | Code shows little attention to organisation. Some code commentary. Inconsistent application of code style. | Collaboration and communication tools are not used effectively. No evidence of code reviews/refactoring. Issues and pull requests if present are poorly documented. |
| **Fail**  **0 – 34%** | No conceptual software design presented or poorly documented, no rationale. | Software implementation does not work or compile. Few of the specified functional requirements appear to be implemented. | No formal testing apparent. | Code is largely incoherent. No documentation. No attention to coding rules or conventions. | Little or no collaboration or communication across the team. Issues and pull requests poorly documented or not present. No evidence of code reviews/refactoring. |

## 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.