**Section 1: Theory supported by code samples (50%, 1400 words plus code samples)**

Evidence for learning outcome: *Demonstrate critical understanding of the theory and application of advanced programming technique*s; *Design and implement programs for real world problems*.

1. **[20 marks] Identify one part of your program design that could be redesigned to use Python Threads. Discuss what specific issues need to be considered given the overall design of your program, clearly demonstrate how data and/or communications will be passed between threads, and which Python constructs would support this. You should provide an overview diagram of your existing program flow and clearly identify where concurrency could be of potential benefit. (500 words maximum, plus appendices containing up 1 page maximum of supporting code examples, and 2 page diagrams)**

…….

1. **[20 marks] With specific refence to GUI interface constructs and best practice regarding interface layouts, discuss how your GUI supports the required user interactions, given the client’s requirements. You should clearly identify the different interactions required by the user, the GUI and code construct/s that support it, and your reasoning for selecting these over other viable options. (500 words maximum, plus appendices containing up to 1 page maximum of supporting code examples, and 2 pages of wire frame/state designs)**

……

1. **Java and Python are both high-level programming languages, each providing similar functionality but via different mechanisms and syntax. Having experienced both languages, discuss which you believe to be the most effective in terms of ‘manipulation of data containers’ (constructs used for storing, structuring, and manipulating data within the program such as a List), and give specific Python examples (only) from your program that support your argument. (400 words maximum, plus appendices containing up to 1 page maximum of supporting code examples)**

**Java OOP only so have to create objects for everything**

**Python – functional paradigm**

According to usage data from GitHub’s 73 million users, Python and Java were the second and third most used languages in 2021 respectively [1]. It is very similar when it comes to working with data; Towards Data Science surveyed more than two thousand data scientists and machine learning developers, finding that Python is the most popular language, used by 57% of recipients. They found that Java was third, used by 41% of recipients [2].

Both Java and Python are versatile, efficient, high-level programming languages that have their own advantages and disadvantages. One of Python’s most cited advantages is its extensibility thanks to its ecosystem of high-quality libraries, such as Pandas and Numpy, both of which can be used to manipulate data structures. Numpy integrates C, C++, and Fortran code to allow for very fast work with vectors and matrices whereas Pandas, which is built on top of the NumPy, allows for vectorised operations that transform data structures without having to use slow for-loops.

An example of significant efficiency savings from the assignment application through using Pandas can be seen in Appendix A. Using core Python to merge two datasets would require a nested for-loop with an asymptotic runtime of O(n2), whereas using Pandas merge function negated the need for any loops.

However, Java also has extensive libraries, with Tablesaw as an equivalent to Pandas and ND4J as an equivalent to Numpy for data structure manipulation. These are inherently fast as unlike Python, Java is a compiled language. In addition, both Python and Java have good memory management capability, especially garbage collection, making them a great choice for managing and manipulating large data structures [3].   
  
Finally, Python code is very simple with common manipulations for data containers requiring just one or two lines of code. This is partly due to Python being dynamically typed. Conversely, Java is a verbose language that is statically typed and requires a lot more code.

As both languages can yield the same outcomes using very similar tools and libraries, the effectiveness of the languages in terms of manipulating data containers depends on how effectiveness is defined for a specific project. If runtime speed is the priority, then Java is the best option, however, if ease and speed of writing the code is the priority then Python is the best option.

**Section 2: Design decisions supported by code samples (40% 1200 words plus code samples)**

1. ***[10 marks]* With specific reference to the client’s data manipulation requirements, discuss your reasoning for your selected data format (JSON, XML, entity relationship structure), and what advantages/disadvantages it has demonstrated in this context? (300 words maximum, and 1 page maximum of supporting code examples, 1 page data format diagram)**

As the application is a prototype and there is no client requirement for a database, it makes sense

JSON and XML fulfil a similar purpose of organizing complex data in an understandable and readable format to various APIs (Application Programming Interfaces) and programming languages, such as Python,  
  
  
In fact, similarly to XML, JSON also receives data from a web server and transmits it to a web page. However, it needs less coding and the size is smaller, thus contributing to faster processes and data transportation.  
  
First, as previously mentioned, while XML is a markup language, JSON, on the other hand, is a data format. One of the most significant advantages of using JSON is that the file size is smaller; thus, transferring data is faster than XML.  
  
  
XML might be "old" and complex, but its complexity is what enables this language to not only transfer data but also to process and format objects and documents.

Unlike JSON, a document in XML is normally self-describing. Usually, an XML document has a link to its schema on the header (schemas are also written in XML, and defined in the XML specification by W3C). Because the schema of a document describes what can or cannot be on a document, it has two advantages:  
  
2. Document can be validated against the schema. In other words, the app that loads the document can check if it is correct, without missing tags or other errors.

Another great advantage of using XML is that it handles comments, metadata, and namespaces.  
  
  
Despite the many differences between JSON and XML, what mainly distinguishes them is data parsing.  
  
  
But, that’s all JSON can do. It doesn’t have any of the powerful validation and schema related features that XML has.

Bulky and slow in parsing, leading to slower data transmission

Very fast as the size of file is considerably small, faster parsing by the JavaScript engine and hence faster transfer of data

Document size is bulky and with big files, the tag structure makes it huge and complex to read.

Compact and easy to read, no redundant or empty tags or data, making the file look simple.

Supports many complex data types including charts, images, and other non-primitive data types.

JSON supports only strings, numbers, arrays Boolean, and object. Even objects can only contain primitive types.

One thing is to compare both technologies considering their purpose according to the developer's goals. In that case, JSON is faster and easier to use. However, another thing would be to compare them considering the features each technology offers. In this regard, even though XML is slower and more complex, it also provides additional features that, to these days, JSON has not yet developed.

All in all, to perform data exchanges that do not require many concerns regarding validation and syntax, JSON is most likely the best option.

As we can observe, JSON and XML differ in various aspects, from applicability to coding representation, data structure, and even security. After weighing both XML and JSON in the same balance, one concludes that JSON is the fastest and easiest way to fulfill the data structuring and exchanging mechanism. In this regard, JSON's performance surpasses XML. However, XML keeps playing a significant role in data storage, and its document formats are still very used by developers and set as default in numerous tools.

Choosing one format over the other depends on your use case.

If you only want to store primitive types as supported by JSON in a simple, human-readable format, JSON is the way to go. If you need all the power and complexity of a markup language, use XML. You probably don't want to invent a document format based on JSON.

The bottleneck with parsing JSON and XML usually is not the parsing itself, but the interpretation/representation of the data. An event-based XML parser usually is very fast, but building a complex DOM tree of thousands of small objects is not. If you need to parse XML to nested native data structures such as lists and dictionaries, the slow part will be the interpretation of the parsing results, not the actual string analysis. Since JSON parses right to those primitive types rather than a complex object tree, it will likely be faster.

1. **[10 marks] With support of appropriate code sample/s discuss how you implemented both parts client's 3rd requirement as stated below. You should use appropriate terminology and clearly identify the code constructs/functions/APIs you have selected and why you selected them over alternatives. You should make reference to any data cleaning needed to get to this stage in the application. (300 words maximum, plus appendices containing up to 1 page maximum of supporting code examples, and 1 page maximum of screen shots)**

Text

Description automatically generated

…..

1. **[10 marks] With support of appropriate code sample/s discuss how you implemented appropriate visualisations to meet the clients 4th requirement as stated below. You should clearly identify the APIs you selected and why you choose them over other viable options, ensuring you make clear reference to the advantages/disadvantages in this context. You should provide screenshots to demonstrate the visualisation/s you selected. You should make reference to any data cleaning need to get to this stage in the application. (300 words maximum, plus appendices containing up to 1 page maximum of supporting code examples, and 1 page maximum of screen shots)**

Text

Description automatically generated  
  
…..

1. **[10 marks] Using code and screen shots of your output demonstrate how you have determined if there is a significant correlation between the communication frequencies used by the 3 different airports. Clearly state If you consider what you have identified to be a significant correlation and why you think this is the case. You should make reference to any data cleaning need to get to this stage in the application. (300 words maximum, plus appendices containing up to 1 page maximum of supporting code examples, and 1 page maximum of screen shots)**

Text

Description automatically generated

…

**Section 3: Reflection on the ethics, moral and legal aspects**

1. **Reflect on the ethics, moral and legal aspects of computing as discussed in the module and respond to the following statement:**

***“Software engineers should not be subject to regulation by a central body as this would have a detrimental impact on innovation. It would also add needless bureaucracy around the development of essential security updates and patches thus putting organisations and their data at risk.”***

Software has become an integral and unavoidable part of our lives, running critical systems across society, such as in government, finance, healthcare, infrastructure and in our homes. Software innovations have changed the world, such as the use of AI in medicine [4], the potential of autonomous cars to save lives [5], and the prediction of wind for clean energy production [6].

Due to software’s now ubiquitous nature, the consequences of errors and misuses can be far reaching and can impact environmental, social, economic, and educational contexts. A prominent example of software misuse is the 2015 Volkswagen emissions scandal [7], which is estimated to have caused at least 59 early deaths in the US between 2008-2015 and to have had a social cost of $450m during that time [8]. As noted by Patel [9], this was not the work of a small number of rogue software engineers but instead involved a significant portion of the organisation.   
  
Due to their role in developing these systems, software engineers have significant opportunity to do good and to cause harm. It is, therefore, easy to see why regulatory oversight of software engineers may be suggested. However, the profession is incredibly broad and as such any regulations from a central body would likely be wrought with ambiguities resulting in a more convoluted, expensive, and slower software development process.   
  
Reducing the speed of software development could have particularly negative consequences when it comes the ability of organisations to release software security patches. Persistent vulnerabilities represent serious cybersecurity threats, with most cyberattacks exploiting known vulnerabilities [10]. However, this doesn’t necessarily impact innovation.

Anecdotally, there is likely to be an assumption that regulatory oversight has a detrimental impact on innovation, however, research is mixed. [11] conclude that regulations result in decreased incremental innovation but increase more radical breakthroughs. [12] finds that regulation negatively impacts short-term innovation but often has positive longer-term implications. Therefore, it is difficult to predict the impact of centralised regulation of software engineers on innovation.

From a practical standpoint, regulating such a diverse profession is likely to be very difficult. Such regulation would be in addition to existing regulations such as GDPR, and industry specific regulations such as those specific to banking or healthcare. A more practical approach is to extend those regulations to consider the uses of software, such as how algorithmic systems are monitored [13] and to improve ethics education for software engineers [14].

**References**

|  |  |
| --- | --- |
| [1] | GitHub Inc., “The 2021 State of the Octoverse,” [Online]. Available: https://octoverse.github.com/. [Accessed 13 April 2022]. |
| [2] | Towards Data Science, “What is the best programming language for Machine Learning?,” 2017. [Online]. Available: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7. [Accessed 13 April 2022]. |
| [3] | M. Ismail and G. E. Suh, “Quantitative Overhead Analysis for Python,” *IEEE International Symposium on Workload Characterization (IISWC),* pp. 36-47, 2018. |
| [4] | G. Briganti and O. Le Moine, “Artificial intelligence in medicine: today and tomorrow,” *Frontiers in medicine,* vol. 7, no. 27, 2020. |
| [5] | M. V. Rajasekhar and A. K. Jaswal, “Autonomous vehicles: The future of automobiles,” *IEEE International Transportation Electrification Conference (ITEC),* pp. 1-6, 2015. |
| [6] | G. Giebel, L. Landberg, G. Kariniotakis, et al., “State-of-the-art Methods and software tools for short-term prediction of wind energy production,” *EWEC (European Wind Energy Conference and exhibition),* 2003. |
| [7] | Q. Schiermeier, “The science behind the Volkswagen emissions scandal,” Nature News, 2015. [Online]. Available: https://www.nature.com/articles/nature.2015.18426. [Accessed 14 April 2022]. |
| [8] | S. R. Barrett, R. L. Speth, S. D. Eastham, et al., “Impact of the Volkswagen emissions control defeat device on US public health,” *Environmental Research Letters,* vol. 10, no. 11, 2015. |
| [9] | P. Patel, “Engineers, Ethics, and the VW Scandal,” IEEE Spectrum, 2015. [Online]. Available: https://spectrum.ieee.org/vw-scandal-shocking-but-not-surprising-ethicists-say. [Accessed 14 April 2022]. |
| [10] | A. Arora, R. Telang and H. Xu, “Optimal policy for software vulnerability disclosure,” *Management Science,* vol. 54, no. 4, pp. 642-656, 2008. |
| [11] | P. Aghion, A. Bergeaud and J. Van Reenen, “The impact of regulation on innovation,” *National Bureau of Economic Research,* 2021. |
| [12] | K. Blind, “The impact of regulation on innovation,” in *Handbook of innovation policy impact*, Cheltenham, Edward Elgar Publishing, 2016, p. 450–482. |
| [13] | K. Yeung, “Algorithmic regulation: A critical interrogation,” *Regulation & Governance,* vol. 12, no. 4, pp. 505-523, 2018. |
| [14] | Y. Lurie and S. Mark, “Professional ethics of software engineers: An ethical framework,” *Science and engineering ethics,* vol. 22, no. 2, pp. 417-434, 2016. |

**Appendix**

**Section 1 Q2**

**Diagram

Description automatically generated**

**Appendix A**Using core Python to merge two datasets would require a nested for-loop with an asymptotic runtime of O(n2). Using Pandas does not require any for-loops.   
  
The code snippet below is from the assignment application and shows a dictionary containing airport references as the keys and nested lists of communications frequencies as the values being merged with a dataframe of the airport data.   
  
**Text

Description automatically generated**