**Project Summary**

One of the well-known design techniques is the scenario-based usability engineering technique, where it focuses on creating rich and detailed user-interaction scenarios as a main representation of the software model that focus on the users’ goals, their problems, and their context to help in making the right decisions quickly and confidently. Another popular design technique is the software architecture-based technique, where it relies on an understanding of the architectural mechanisms used to achieve the software functional, quality, and business requirements at a level of abstraction by providing a series of steps for designing the conceptual software architecture.

Through this project, we will be evaluating both scenario-based usability engineering and software architecture-based designs in the context of the air traffic control case study in terms of how these two designs affect each software quality attribute, using McCall's list of software quality attributes where we will go through each of those attributes and try to analyze how both designs addressed or helped achieve that specific quality attribute.

The McCall’s list of the quality attributes that will be used to perform the comparative analysis is as follows:

* Correctness: the software meets its requirements specification e.g., the accuracy of the distance measurement varies between 5-300 miles
* Reliability: the software performs its intended functions without failure e.g., downtime for a system will not be more than 30 minutes per year
* Efficiency: the amount of hardware or software resources needed to perform a function e.g., a system is not using more than 1 GB of RAM
* Integrity: the software can protect unauthorized users from accessing the software or its data e.g., non-admin users cannot access the air traffic system
* Usability: the software is easy for users to understand and use it functions e.g. air traffic controller can easily understand how to use the navigation system
* Maintainability: it does not take a lot of effort to detect or fix an error during maintenance phase e.g., detecting a software bug is not taking more than an hour
* Flexibility: the software can be modified and improved easily e.g., improving the algorithm to compute speed, times, and distances in an air traffic control system
* Testability: the software can easily be tested to verify that it meets the specified requirements e.g., testing communication between air traffic controllers
* Portability: the software can easily be transferred from one platform to another e.g., moving the air traffic control system from Linux to Windows
* Reusability: the software’s code can easily be used in other applications e.g., air traffic detection code to be used in a military application

**Comparison**

In my opinion, nowadays, most technology tools’ infrastructure should support scalability,

**Conclusion**

In conclusion, through this project, we evaluated the 4 KWIC Index architectures and decided that Implicit Invocation architecture is the best choice to use for developing a KWIC index generation tool for an online course,