Software architecture acts as the blueprint for the development of software, allowing the complexity of a system to be modelled, while establishing communication between components and connectors (Jaiswal, 2019; Qureshi *et al.,* 2011). The blueprint can be represented in various styles; this application has been demonstrated through both the ‘layered’ and ‘repository’ style of software architecture. Ideally, the system should perform well, be secure, be maintainable, allow iteration and scalability, be feasible and efficient; however actually implementing these qualities require trade-offs to be considered.

A similarity both types of architecture exhibit is the capacity for scalability and iteration, due to both styles having the capability for loose coupling of components. However the repository (AKA blackboard) style is more accommodating for scalability as components and knowledge sources can be added and removed from the system without having to restructure the model; Whereas, this may be untrue for the layered style, as restructuring of layers and scaling of the entire application to accommodate changes at one specific layer is often required, thus scaling a layered model may be more resource intensive and less feasible (Akmel *et al.,* 2017). Due to the nature of the application, updates (more playable sports) and new features will be implemented so scalability, and in turn economic feasibility, is a priority.

The layered architecture style provides a higher level of security due to the concept of onion skinning, by compounding layers of protection on each layer of the system, creating almost an onion skin affect. On the other hand, the flexibility that the repository architecture provides for allowing the addition of new components to the main repository may reduce the overall confidence in this model’s system security. However, due to how independent components in a repository system are, the system may be able to continue to function even if one component breaks or the function becomes degraded; this could be useful in a multi-functioning app such as the one proposed, for example if the co-operative feature of the app is currently under maintenance, the user can still access other services such as AI coaching and exchanging messages with friends.

Overall, for this project there is preference towards the repository architecture style due to feasibility and flexibility the style allows for the application, however there is also a use-case for the layered architecture style as it also accommodates for features the repository style does not do as well.

**References:**

Akmel, F., Birihanu, E., Siraj, B. and Shifaw, S. (2017) A Comparative Analysis on Software Architecture Styles. International Journal in Foundations of Computer Science & Technology [online]. 7, pp. 11-22. [Accessed 24 Novemver 2022].

Jaiswal, M. (2019) Software Architecture and Software Design. Social Science Research Network [online]. 6 (11) [Accessed 22 November 2022].

Qureshi, N., Ikram, N., Bano, M. and Usman, M. (2011) Empirical Evidence in Software Architecture: A Systematic Literature Review Protocol. The Sixth International Conference on Software Engineering Advances [online]. [Accessed 22 November 2022].