**Requirements summary: strengths**

The requirements outlined in this project are well detailed. The requirements summary lists the inputs i.e., PDF file and outputs of the system i.e., PDF web form. Functional and Non-Functional Requirements are listed as well. However, I would recommend providing specific examples of users who are allowed to use the system like software developer, lead architect, QA engineer, etc., but overall good job.

**Requirements summary: clarity**

Requirements are simply listed so that the target customer, designers, and stakeholders can read and understand them. However, When I initially read F4, the “state” term was unclear to me even after reading it multiple times, but I was able to understand what it means once I read F5 through F9, so I would definitely recommend having a better and more clear definition of what “state” means with examples provided. Overall, requirements are not too detailed or in technical terms that may be difficult for some of the audience to understand. Good job.

**Requirements summary: completeness**

Most of the necessary requirements were being considered, however, the non-functional requirements do not cover everything. For example, it was not mentioned what Unicode the system accepts for text (like ASCII, UTF-8, and UTF-16), what specification the system requires or limits (like PDF file sizes), and what accessibility the tool provides (like text-to-speech)

**Requirements summary: verifiability**

I believe each one of the requirements can be verified either through functional testing (e.g., unit testing, validation testing, and compatibility testing) for user inputs and interactions and browser compatibility, or formal reviews for legal and standards compliance.

**High-level design: strengths**

Good organization and use of the application components especially the Core PDF Backend Service part where it acts as the middleman between the user and the database including DB operations, workflows, messaging, and PDF generation services. Also, good choice for using Kafka/RabbitMQ for event handlers instead of manually handling them in the application code which improves performance because of how Kafka/RabbitMQ efficiently supports replicated queues and streams and how this would save the development time of those handlers.

**High-level design: clarity**

The use and display of multiple UI screenshots, diagrams, and database tables makes the high-level design very clear, well detailed, and understandable. Great job!

**High-level design: conceptual cohesiveness**

Each component in the application provides a separate function where it is clearly stated and outlined what each component is responsible of and how each of those components interact with other components which makes the design very cohesive. Also, I really like that there isn’t any unused or redundant implementation, that is, each implementation/component plays a critical part in the system.

**High-level design: traceability**

The provided traceability matrix clearly maps design decisions back to the stated requirements where design decisions are conveyed through test cases, user role, and expected issues. The matrix also shows that all requirements have been adequately addressed. Great job!

**High-level design: brittleness**

I believe the only parts that may resist foreseeable change in the product are the database and PDF File Generation Service parts for cases like changing the encoding type from ASCII or UTF-8 to UTF-16 to handle more input characters which will require big changes in the database infrastructure and file generation to accept those extra characters that may cause resistance for such change. Otherwise, I believe all other aspects of the design have enough details and clearly pictured with no brittleness.

**Design rationale: strengths**

The design components are very practical and logical where even non-technical people can easily understand them. I also like the comparison part between the three different architectures with the benefits and risks of each architecture included.

**Design rationale: design decisions**

The key design decisions are well explained and cover all the key requirements of the application. However, although a comparison was made between the three different architectures, it wasn’t actually indicated which of the three architectures the team decided to follow for the design process in the Design Rationale section, but it was mentioned in the Conclusion section, so I would recommend mentioning the decision choice in the Design Rationale section as well as the Conclusion’s. Lastly, as mentioned earlier, I believe the text Unicode should also be included as a key design decision as well as what language the application accepts like English, Arabic, French, etc.

**Design rationale: justification**

The design is well discussed, and each requirement aligns well with its design decision. Also, the comparison between the three architectures clearly justifies why option 2 Event-Driven would be the best option to use. However, although it was mentioned in the conclusion that a sort of hybrid solution (mix of all architectures) will be used and it was well justified why the team will do that, I believe the structure of such solution wasn’t enough detailed and was a bit generic. That is, I would recommend adding more details to the hybrid solution like how would it enhance testability or usability?

**Design rationale: evolution**

I believe both Event-Driven and suggested hybrid architectures support very well the evolution of the already existing and future requirements.

**Suggested improvements**

As mentioned in the previous sections, I would recommend providing specific examples of users who are allowed to use the system like software developer, lead architect, QA engineer, etc. In addition, I would also recommend providing more information about the text Unicode, languages, and accessibility tools supported by the system. Also, I think it would be nice if the design mentions the size limits for files uploaded to the system to avoid performance issues. Lastly, I would recommend supporting more file formats beside PDF like docx or txt as a sort of enhancement.

**Summary**

In conclusion, I believe that the design was solid, and the team did a great job in covering the key requirements of the system and well explained how each requirement is handled by the system at a high level, however, I believe implementing the suggested improvements in the previous section would make the design even better and more solid. Great job!