So, the creation of RAG powered chatbot was the easy part (had built similar thing earlier as well, and Langchain docs are always present for help).

Followed simple steps,

Loaded the pdf using Applauder,

Then embedding and storing in database part was implemented,

Simple rag chain implemented to retrieve and generate answers.  
  
The complex part was to create the dataset.  
Created the dataset, focusing on the below points

1. Making sure that almost every part of the pdf was covered.
2. Some questions were based on values or information mentioned in the tables of pdf, to make sure that the retriever is able to extract them as well (although I believe this would be more kind of testing for pdf loader)
3. Chose some easy questions, to make sure that basic accuracy is there.
4. Then chose some questions with complex answer, basically whose answers are available on different pages (as in one-part mentions, to read other part as well, which is mentioned in some other page)
5. Chose some questions, whose answer may contain good amount of context, but the answer would be relatively very small w.r.t to its context
6. Chose some terms which had semantically similar terms mentioned in the pdf (ex – windscreen damage, windscreen repair)
7. Tried to make the language of some questions unusual , so as to test similarity searching
8. Chose some general questions as well, whose answer is present in the pdf , but they are so easy that more context can be easily provided by the llm , this was done in order to check whether rag stick to the pdf , or provided more content.

Choosing the metrics for evaluation ,again a logical part to be implemented

I had decided to go with four metrics:

1. Checking hallucinations (Answer correctness metrics)
2. Checking context relevancy with respect to the question and ground truth answer (Context relevancy metrics)
3. Checking the relevancy between question and generated answer (Answer Relevance metrics)
4. Checking semantic similarity between answer and ground truth (Answer similarity metrics)

Reasons for choosing the metrics:

1. Hallucination checking is obviously an important part to make sure that the chatbot was not adding any inconsistent or additional information.
2. Context relevancy is obviously important, since for generating a relevant answer, first you need to have a relevant context, it is basically a game changer.
3. Then checking how good is the answer with respect to the prompt is also necessary, because even though if you have a good context, it is possible that it may not lead to a relevant answer.
4. Also checking, semantic similarity between answer and ground truth as language and its structure matters a lot, that’s why prompt engineering also exists as a field.  
   Also it is possible that answer may not similar due to the language and the hallucination checker may count it as inaccurate, however semantically they might be similar.

Did not know much steps to improve accuracy, but below is what I tried

What I did to improve the accuracy was to try different models with different embeddings as well.

Tried different types of query retriever techniques.

Tried hybrid searching.

Also tried to make the prompts clearer and more definitive to make sure that it worked well.

Tried to improve my dataset as well.

Also, the streamlit design has been kept very simple, like just storing the chat history in session variable and displaying it. Acknowledging that major advancements can be done in this part.