Final Exercise in Workshop in Modern Application Development

Name of the app: Contentify
Link to the Android App repo: https://github.cs.huji.ac.il/yar-gav/final_project_pair_11
Link to the Backend Service repo: https://github.cs.huji.ac.il/shorot/final_project_pair_12
Contact Info:
Itamar Shechter, itamar.shechter1@mail.huji.ac.il, https://github.cs.huji.ac.il/itamar-sh
Merav Diamant, merav.diamant@mail.huji.ac.il, https://github.cs.huji.ac.il/diamant-merav

Short technical design description of the app: The app has five Fragments, 9 aws lambdas and five S3 buckets. The first fragment is a welcome screen with recycle view of pdf's. The recycle works with the concept of MVVM while the data is taken and upload to the DynamoDB on aws. The last 10 pdf's are taken from the DB and shown to the user using get_pdfs aws lambda . The uploading is to a S3 Bucket which we call pdfBucket and performed with url which we get via pdf_uploader aws lambda.

After Pressing on one pdf, the Pdf management screen. In case that this is a new pdf file we want to process his data so the summarize_processing lambda is triggered.
In this screen there are 3 buttons to navigate to the three other fragments, while navigating to this screen the chosen pdf is already on processed on aws and 3 buckets are getting the info for the required tasks.
When the summarized text is inserted to the appropriate bucket the Studies processing is triggered and when the studies text is put into his bucket the Q&A processing lambda is triggered. All those stages work closely with the statuses of the item in the DB.

The remaining 3 Fragments are much simpler, a simple scroll view that get's his text with a matched getter lambda specific bucket using url. The Summarized Fragment use the get_summary lambda that retrieve text from SummerizedBucket. In the same way we have Questions Fragment with get_question lambda and question bucket, I remind that also the answers to the questions are generated but we omit them from the names. At last we have the studies Fragment with get_studies and studies bucket.

One challenge: Our first idea was to take NN models and do a fine tuning, after short consultation and searching we understood it's not easy task, especially the Question and Answers part. Then we tried To use trained models, it took a lot of time to find the correct ones and make the connections and adaptions to our goal. In the end, after we found a solution, we encounter a problem of deploying those big models to aws. The fact that aws lambda runs on Linux was the final straw, the models and libraries were problematic to import. In the end we chose to use only Chat GPT through Openai API.

Something we would done differently: We waited with the tests to be our final stage, which forced us to change many things in the implementations. Next time we will build a test for any working component right away.