Silencer Task

Challenges Faced:

- Which type of Schema & Database to use for storing the data?
  - I was considering MongoDB, AWS DynamoDB, Firebase RealtimeDB. I picked MongoDB as it is cost efficient and works well with the used backend technologies i.e. NodeJS and expressJS
  - I mainly considered 2 choices for the schemas, 1st to store every question-answer pair with its corresponding box as a property, and 2nd to make an object having boxes as keys, and storing the corresponding question-answer pairs in the object as separate arrays.
  - I went with 1st approach as it made the code a lot easier to write and understand, and at the same time did not cause any performance throttle.
  - I discarded the 2nd option mainly due to the complexity of the backend Schemas.
  - 1st approach also provided me a better way to implement APIs which can work with minimal broadband and can provide offline access to the app with uploading the data once online, though this feature has not been fully implemented yet but it can be easily and efficiently done.
  - I've also added a userId field with the questions for further support for user accounts.
- How to make the Card Animation efficient and dynamic?
  - Since I did not know how many Questions are to be displayed, I had to think how to animate cards which is memory efficient as I cannot render each card at once beyond the screen.
  - I choose to work with only 2 Card components and made them animate in a way such that they alternatively come from left of the screen and exit from right, making it seem like a scrolling list of cards.
  - Card flip animations were also implemented to show the answer to the user when the card arrives.

References and resources

- I read the Wikepedia and several articles on Leitner System to understand its principles and to implement it.
- I used the official React-Native typescript template for initialising the frontend part of the application.
- I took help from official documentations of various libraries used (e.g. redux, react-redux, axios, etc) and StackOverflow from time-to-time.