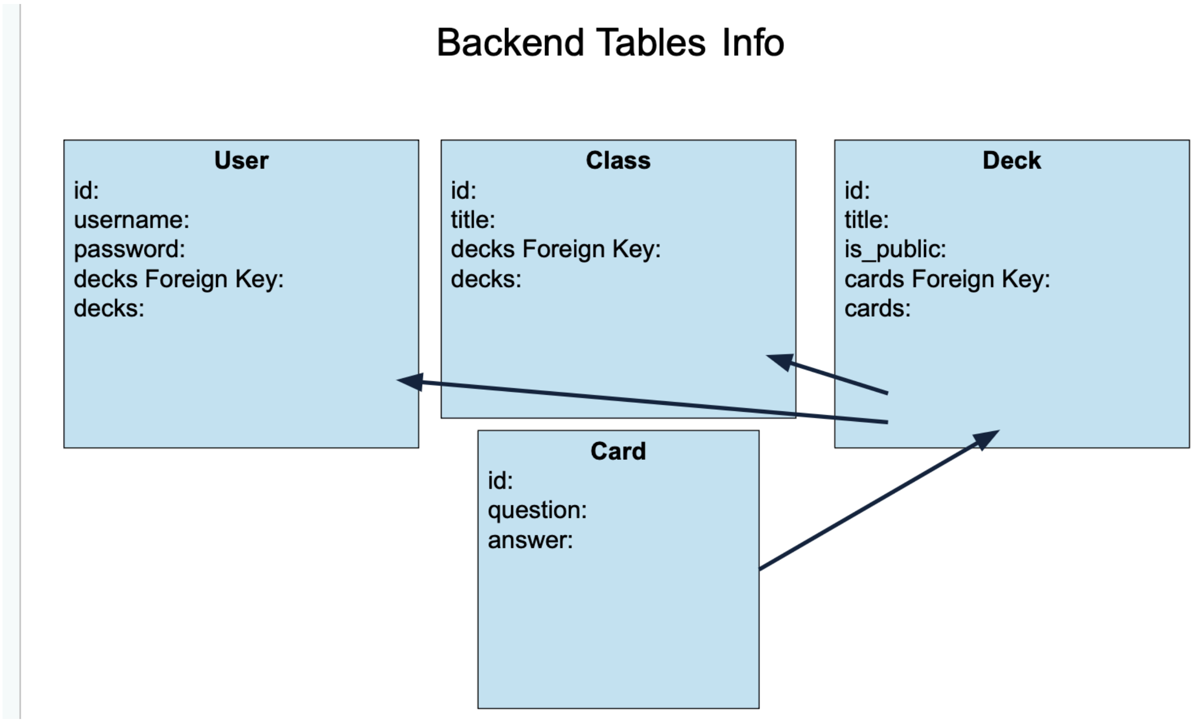
Rishi, Fadi, Elmo, Sebastian

Hack Challenge 4/28 Milestone

Explanation of database & API design (Google Doc/drawing)

Briefly, our project is a Quizlet-style flashcard application. On the backend, as such, there are 4 tables in our database.



The tables and fields are pretty self-explanatory with the image above, as for the relationships in between them – users have a one-to-many relationship with decks that they have made. Decks are effectively units in the classes that they are in (e.g., a Data Structures unit or a time complexity unit for CS2110). As such, there is a one-to-many relationship between classes and decks. We decided not to put a relationship between users and classes for ease – so users can access all public decks in every class, and if they set their own decks to public, other users will also be able to access them. Finally, decks are composed of (flash)cards, which is again a one-to-many relationship.

The CRUD operations for the API are detailed below

Users:

- Create: Allow users to sign up and create a new account.

- Read: Retrieve user information (e.g., username, email, profile details).
- Update: Allow users to update their profile information (e.g., change username, update email). This might be more of a stretch goal.
- Delete: Allow users to delete their account. Likewise, might be more of a stretch goal.

Classes:

- Create: Allow users to create new classes. We will decide later on if we want to hardcode some classes or implement this.
- Read: Retrieve information about classes, including their title and description.

Decks:

- Create: Allow users to create new decks, associating them with their account, and choose their visibility.
- Read: Retrieve information about decks, including their title, description, and visibility (public or private). Might have the option to query within or across decks from a class to provide a “randomized quiz” like Quizlet, but again this is a stretch goal.
- Update: Allow users to edit decks they own (e.g., change title, update description).
- Delete: Allow users to delete decks they own.

Cards:

- Create: Allow users to add new flashcards to their decks. Note that Cards will have the same visibility as the deck they are within.
- Read: Retrieve the question and answer of individual flashcards.
- Update: Allow users to edit flashcard details (e.g., modify question or answer).
- Delete: Allow users to delete flashcards from their decks.