User Stories based on the requirements analysis of Deckbuilder_Archive

U1: Navigation Bar

As an archive_user, I want to be able to navigate between display options and editing options by entering specific parameters.

Recommended Approach:
1. Create a switch-case controller (connected to the index.php) that routes between display options and editing options.
2. Connect a sub-branch that forwards instructions in the form of card archive options.
3. Connect another navigation branch for account options.
4. Connect a third controller that forwards commands to a deck archive controller.
5. Connect a fourth controller that forwards commands to a deckbuilder controller.
5. Create a „JSON View“ file that displays content in JSON format.
6. Create Endcaps for each controller class.

Definition of Done:
* There is a main navigation branch that routes input parameters.
* There is a sub-navigation branch responsible for handling card archive options.
* There is a sub-navigation branch responsible for handling deck archive options.
* There is a sub-navigation branch responsible for handling account options.
* There is a sub-navigation branch responsible for handling deckbuilder options.
* Endcaps with corresponding individual output in JSON view have been created to check if each navigation branch has been reached.

U2: Card Display

As an archive_user, I want to be able to display various and individual cards by entering the card_id, card_name, card_type, sub-type, super-type, mana_value or cmc.

Recommended Approach:
1. Create a config file that holds the neccessary database connection.
2. Add CORS authentication headers for frontend support.
3. Create a model class, that has the same properties as objects pulled from the databse.
4. Create a gateway classes for the following filter options:
 * Searching by card_name
 * Searching by card_id
 * Searching by card_type
 * Searching by card_sub_type
 * Searching by card_super_type
 * Searching by card_cost
 * Searching by card_cmc
5. Create a service class that communicates between the gateway class and its controller.
6. Create a DTO that displays the card model objeccts properly with a URL for a detailed look at an object.
7. Add several different functions to the controller to initiate the service class and display functions to view the the database responses.

Definition of Done:
* The archive_user can display one or more cards by entering search parameters once (e.g., deckbuilder_archive/api/index.php/search/card_type = Enchantment).
* Each search option returns one or more results in JSON format.

U3: User Accounts

As an archive_user, I want to be able to create one or more archive_users to better identify my own decks and also be able to delete an archive_user if I no longer want to keep my personal data or find no use for that archive_user. (backend ONLY)

Recommended Approach:
1. Create a gateway class for account handling. Connect it to the controller class and add a function that creates a user and displays the archive_name and corresponding user_id in JSON format.
2. Create a session variable that stores a user_id. This session variable must contain the following limitations (backend ONLY):
   * Only one user_id can exist at a time
   * A function must be provided to end the session
   * A function must be created to start a new session with a specific user_id
3. Create a login function, that checks if the archive_user exists and verifies their password, before initiating a user session (backend ONLY).
4. Create a logout function that empties the current user session and terminates it (backend ONLY).
5. Extend the gateway class with a function that properly deletes the database entry with the desired user_id or gives an error message if the user_id still has deck_id’s assigned to it.

Definition of Done:
* Archive_users can create an archive_user containing an archive_name, email address, password, and an automatically generated user_id.
* The archive_user will receive both the chosen archive_name and the associated user_id in JSON format upon successfully creating an archive_user.
* The archive_user can start or end a session with a specific archive_user.
* The archive_user can delete all entries in the database with the corresponding user_id.

U4: Deck Builder

As an archive_user, I want to be able to create and delete a decklist in the archive.

Recommended Approach:
1. Create a deck model class, that represents objects pulled from the database.
2. Create a DTO for the deck model class, to properly display object data.
3. Create a gateway class that creates a new deck and returns the corresponding deck_id as a response.
4. Create a second gateway class, that checks for user created database content.
5. This deck_id should be stored in a separate session variable.
6. This session should contain the following limitations:
   * Only one deck_id per session
   * If a new deck_id is chosen, the old one should be removed from the session
7. Create an option to switch between deck lists.
8. Create a function to delete all card entries from the cards_decklists table whose user_id and deck_id match the specified parameters.
9. Create a function that deletes the entry in the decklists table with the corresponding deck_id.

Definition of Done:
* Archive_users can create a new deck list.
* Archive_users can navigate between this deck list and another deck list with their archive_user_id.
* Archive_users can delete one of their chosen decklists.

U5: Update Decklists

As an archive_user, I want to be able to add and remove single cards from my decklist.

Recommended Approach:
1. Expand the deckbuilder gateway class with the following functions:
    * Create a function that pushes into the cards_decklists table whith side_board & maybe_board have the values „No“.
    * Create a function that pushes into the cards_decklists table with side_board has the value „Yes“.
    * Create a function that accesses the cards_decklists table with maybe_board has the value „Yes“.
2. Expand the deckarchive gateway class with similar functions to the deckbuilder gateway class, that select the corresponding database entries.
2. Extend the service classes, that communicate between the controllers and the gateway classes.
3. Create three functions that innitiate separate sessions for „main“, „side“ and „maybe“ (associatve arrays) based on their database contents.
4. Create three seperate functions that adds a single card to one of the sessions. This function should also track the amount of same objects (cards) that have been added to one of the session arrays.
5. Create three seperate functions that remove a single card from their respective session. This should also reduce the quantity of objects (cards) with the same ID by one.
6. Create a function that can be called upon by the remove functions, that removes an object from one of the session arrays, once its quantity reaches 0.
7. Create a function that uses the gateway class to first delete all preknown entries of the database with their corresponding deck ID and then fills them with the current contents of each session.
8. Extend the select function from U4 with another function that automatically empties all three sessions and then fills them with the corresponding database entries of the newly selected deck ID.

Definition of Done:
* Archive_users should be able to add a card to each of their sessions.
* Archive_users should be able to remove a card from each of their sessions.
* If a cards quantity reaches 0 it should be removed from the session array.
* The archive_users should be able to save their decklist contents in the database.
* The archive_users should be able to have their deckbuilder automatically initiated/filled when selecting a legal deck ID.

U6: Decklist Display

As an archive_user, I want to be able to view created deck lists.

Recommended Approach:
1. Create a deck contents DTO, that properly displays card objects as part of deck objects.
2. Extend the archive gateway class so that it displays all deck lists from the decklists table in the database.
3. Create object links that leads to a detailed look of a list, for each search result.
4. Add a function to the controller class that calls upon the card archive service class to display all entries from the cards table corresponding to the card_id’s stored in the cards_dekclists table of the selected deck_id.

Definition of Done:
* Archive_users should be able to display all corresponding entries in the database in JSON format by passing a name, deck_id, or user_id.
* Archive_users should be able to view a deck list by selecting it.