User Stories based on the requirements analysis of Deckbuilder_Archive

U7: Landing Page, About Us Page, Imprint Page, Navbar & Footer

As an archive_user I want to have access to a simple navigation bar, footer and a consice explanation about all the services provided, so I can navigate through the available services, without confusion.

Recommended Approach:

1. Setup the index.js file so it includes Bootstrap, Vue Router and pinia.
2. Setup the Vue Router, so that it contains links to Landing Page (“/“), About Page (“/about“) and the Imprint Page (“/imprint“).
3. Include the Router Link and Router View components in the App.vue file.
4. Create 3 containers. One for the navbar, one for the content shown and one for the footer.
5. Create a navigation bar that includes a logo, a title name and Router Links to the Imprint and About Pages. Add the functionality for the logo to route to the Landing Page component.
6. Add contact information and support information to the footer. This should include a Router Link to the terms of service and data security aggreement on the Imprint Page component.
7. Create three components (if not allready existing) for Landing Page, About Page and Imprint Page, as well as endcaps for each of the created components.
8. Create the following contents for the Landing Page component:
    - *infos about the card archives*
    - *infos about the deck archives*
    - *infos about the deckbuilder*
    - *a call to action to get know more about us (this should route to the About Page component)*
9. Create the following components for the About Page component:
    - *infos about the creator*
    - *infos about the motivitaions behind the project*
    - *infos about the current version*
    - *infos about previous versions*
    - *infos about future versions*
10. Create the following components for the Imprint Page component:
    - *legal disclaimer*
    - *terms of service*
    - *data privacy aggreement*
    - *contact information*

Definition of Done:

*Users start on the Landing Page, when visiting the website.
*Users can see the navbar at the top of the page including a logo and the options to switch between the Landing Page, About Page and Imprint Page components.
*Users get a short introduction to all available services (current or future)
*Users get a call to action to the About Page component.
*Users can see contact information and links to legal information when scrolling to the bottom of the page.
*Navbar and footer are visible on every viewable component.
*User can read up on a detailed description in the About Page component.
*Useres can read up on detailed legal information and full contact information on the Imprint Page component.

U8: Card Archive

As an archive_user I want to be capabe of searching/filtering through card object database entries and display them in full detail, based on the following criteria: name, card type, super type, sub type, card id/set id, mana value and cost.

Recommended Approach:
1. Create a Card Archive viewing component.
2. Create a Card Archive pinia store.
3. Add the Card Archive component to the Vue Router and add an endcap.
4. Create 2 containers on the Card Archive component.
5. The first container should consist of two input fields. One for text input and the other as a radio button to select one of the 7 filter options. Use the Options Api, to create a switch case for the second input field. Add a search/submit button to make a call to the Card Archive pinia store.
6. The second container will dynamically create card objects (Javascript card items, not database card objects) based on the search result of the user.
7. Include the newly created Card Archive pinia store in the Card Archive viewing component for data transfer.
8. Setup the Card Archive pinia store, so that it properly communicates and saves the information from the deckbuilder_archive api.
9. Create a Single Card viewing component, that displays a card object in full detail and add it to the Vue Router. This component contains:
    -an image
    -a name
    -a set id
    -a super type
    -a card type
    -a sub type
    -a mana value
    -a cost
10. Add a Router Link to the card objects in 6. that links to the Single Card viewing component using the card_id of the card object.
11. Add the Card Archive viewing component to the navbar.
12. Add corresponding images of the database entries to the assets directory. These have to be named after their corresponding card_id.

Definition of Done:
*The archive_user can click on the Card Archive Router Link in the navbar to navigate to the Card Archive viewing component.
*The archive_user is presented with a searchbar as well as an option to choose between different filters.
*Adding the value of “Rag“ and choosing the filter option “name“ will display 2 database entries.
* These database entries are displayed as card objects, that contain an image cap, a name and their set id/card id.
* The archive_user can switch to a a detailed view of a database entry, by clicking the desired card object in the search results.
* When viewing a card in full detail, it should display all information of a database entry.

U9: User Account

As an archive_user I want to be able to create a user account, login to that account, as well as logout of that account and be able to change my user_name or password of my account.

Recommended Approach:
1. Create an Account Creation viewing component.
2. Create a Login viewing component.
3. Create a User Account pinia store and set it up so that it transfers and saves JWT user tokens.
4. Include the Login as well as Account Creation viewing components in the Vue Router.
5. Create endcaps for each of these components.
6. Add four input fields, two checkboxes and a submit button to the Account Creation viewing component.
7. The first input field is for an email input named “Email“, the second for a text input and is called “Archiver Name“. The third and fourth input field are also text input fields and are named “Password“ and “Confirm Password“ respectively.
8. The first checkbox is named “Terms of Service“, the second one is called “Data Privacy Aggreement“. Both checkboxes contain a Router Link to the Imprint Page component, ideally jumping to their respetive paragraphs (use container ID’s for that). These are placed below the input fields.
9. The submit button is placed at the bottom of the viewing component is only available if both checkboxes have been flagged as true (when they are both checked). Pressing the submit button will create a new account.
10. Setup the pinia store in a way where it automatically saves the newly created user token.
11. Setup the Login viewing component with two input fields, one for “Email“ and the other for “Password“, as well as a login button. Clicking the logged in button with proper credentials, will use a Router Link to navigate to the Landing Page.
12. Setup the App.vue component like this:
add a dropdown menu to the navbar
create a boolean on whether a user token is available or not, use the Archiver Name inside the user token to replace the label of the element
create a boolean that switches between account creation and account settings based on whether a user token is available
add a login, logout, account creation and account settings option to the dropdown menu
13. Create an Account Settings viewing component, that contains a sidebar with two option called “Settings“ and “Decks“. Use the Options Api for this.
14. Create an endcap for the “Decks“ option of the sidebar.
15. Add a text element to the “Settings“ option, that displays the current Archiver Name based on the currently available user token, then create a text input field below that, labeled “Change Name?“, with an added submit button below that. The submit button should be locked if the text input field above is empty.
16. Create a second text input field below that, labeled “Change Password?“. This should also have a submit button, that follows the same rules as in 15.

Definition of Done:
***The archive_user is presented with a dropdown menu on the navbar, called “Account“.**
*Clicking the dropdown menu will reveal two options. Either Login and Create Account or Logout and Account Settings, depending on the logged-in-status.
*Clicking Login, Create Account or Account Settings will route to their respective viewing components. Clicking a submit button and successfully communicating through the pinia store will route back to the Landing Page component.
*Clicking logout deletes the user token of the pinia store and routes back to the Landing Page.

U10: Deck Creator

As an archive_user I want to create my own decklists, by naming them and chosing a format, or delete my own decklists.

Recommended Approach:
1. Create a Deck Creator viewing component.
2. Create a Deckbuilder pinia store.
3. Create a Sandbox viewing component, with an endcap.
4. Include both viewing components in the Vue Router.
5. Include the Deck Creator component in the Account dropdown menu. Make sure it is only displayed if an archive_user is logged in.
6. Create a text input field, a radio box and a submit button for the Deck Creator viewing component. Use the options api for the radio buttons.
7. The submit button can only be unlocked once an input has been put into the text box.
8. Pressing the submit button, will use the text input fields value as the deck_name, and the chosen option as the format value. This will then call the Deckbuilder pinia store, which will call the User Accounts pinia store, for the currently logged in user token (in particular the user_id value). This is followed by a database post, that will create the new decklist object.
9. If the creation of a new database entry was successfull the Deckbuilder pinia store, will save the returned deck_id value as the current value. The archive_user will then be linked to the Sandbox viewing component gratulating them displaying the current deck_id, format and deck_name values.
10. If the deck creation was not succcessfull, an error message should pop up on the Deck Creator viewing component, without changing pages.
11. Setup the Deckbuilder pinia store with a function that deletes the value for the current id, before it tries to run through any other function.

Definition of Done:
*The archive_user can see a Create Deck Option in the Account dropdown, while they are logged in. Clicking that button, will send them to the Deck Creator viewing component.
*The archive_user can create a deck by entering a name and choosing a format with the radio buttons, before clicking the submit button.
*If a archive_user can’t create a new database object, they should be held at the Deck Creator viewing component, displaying an error message.
*If a archive_user successfully creates a decklist, they will be routed to the Sandbox viewing component, that will display the data of the currently created decklist object.

U11: Building a Deck

As an archive_user I want to be able to add and remove cards from a decklist of my choice, that I have previously created.

Recomended Approach:
1. Include an option for Decklists in the Account Settings option of the Account Services viewing component.
2. Create a Deck Archive pinia store, that searches decklists based on the user tokens user_id (Account Services pinia store).
3. Include the Deck Archive pinia store in the Decklists option. Have it dynamically display all entries by that user_id. Add a function that routes to the Sandbox viewing component and calls upon the Deckbuilder pinia store, to save the deck_id of the chosen deck object.
4. Extend the Deckbuilder pinia store, with arrays for ‘main‘, ‘side‘ and ‘maybe‘. Have these emptied automatically, before filling them with database contents.
5. Extend the Sandbox viewing component with a sidebar container that uses the Card Archive pinia store to access the database for database objects of the cards table via card_name. The search results should be placed below the searchbar within the sidebar container and should create card objects. These card objects consist out of an image cap and a button that opens a small dropdown menu, with three options: Add to Main, Add to Side as well as Add to Maybe.
6. Add three containers to the Sandbox viewing components that have the ID ‘main‘, ‘side‘ and ‘maybe‘. Make sure all three containers are dynamically placed on the page. The ‘main‘ container should be placed left of the other two containers, while the ‘side‘ container should be placed above the ‘maybe‘ container.
7. Connect the containers to the created card objects ‘Add to …‘ options of the dropdown menu of the sidebar. Now clicking the one of these options should add the object with the chosen id to the respective container.
8. Objects inside the 3 containers should be displayed as a list with the following attributes:
name
mana value
quantity
add-function-button
remove-function-button
9. The add-function-button of the container objects should increase the quantity of the respective object by one.
10. The remove-function-button of the container objects should decrease the quantity of their respective object by one.
11. Objects that reach a quantity of 0 should be removed from the container.
12. Create a submit button, that calls on the Sandbox pinia store to post the decklist into the database.
13. Create a final container, that holds the information of the decklist on top of the web page, below the navbar.
14. Include the Counter store for tracking the quantity within every container. Let the total quantity of each container be displayed next to the container name.
15. Create a message that displays success or failure upon clicking the submit button, depending on whether the data has been properly saved in the database.

Definition of Done:
*The archive_user can now navigate through the Account Services viewing component to the Sandbox Viewing component.
* By clicking the sidebar, a searchbar with a text input can now be used to search for databasse entries of the cards table, using card_name as a parameter.

*Search result card objects include a dropdown menu with three options that can add the respective object to the desired container.
*Objects inside one of the containers can be incremented by one or decreased by one, which is signaled by the quantity value next to their object name.
*Decreasing the quantity of an object to 0 removes the object from the container.
*Saving will display a message of success or failure.
* Navigating through the Account Services viewing component and switching decklists for alteration will display the most current state of the database entry.

U12: Deck Archives

As an archive_user I want to be able to search for decklist database entries based on their format or possible deck names.

Recomended Approach:
1. Create a Deck Archive viewing component.
2. Include the new viewing component in the Vue Router and the navbar.
3. Extend the Deck Archive pinia store with functions that access the database based on deck_name and format.
4. Build the Deck Archive viewing component identical to the Card Archive viewing component, but have it include the Deck Archive pinia store instead of the Card Archive viewing component and unlock the submit button. Change the options to simply filter for all entries of the chosen format.
5. Create a Deck Contents Display viewing component.
6. Include the Deck Contents Display viewing component in the Vue Router.
7. Create a function that links to the Deck Contents Display viewing component, using the deck_id, by clicking on the created card object of the search results.
8. When recieving a deck_id the Deck Contents Display viewing component should call upon the Deck Archive pinia store to fill three arrays: ‘main‘, ‘side‘ and ‘maybe‘. Have it dynamically create a container for each of those arrays, containing different objects, with their respective values of name, mana value and quantity. Have the containers placed like in U11.
9. Make sure that the arrays ‘main‘, ‘side‘ and ‘maybe‘ are allways emptied before filling them with new data.
10. Create a display message for all containers with no contents.

Definition of Done:
*An archive_user can now click on the Deck Archive option of the navbar, regardless of the logged-in status.
*Clicking the Deck Archive option links the archive_user to the Deck Archive viewing component.
*By choosing between the radio-buttons and entering a value into the text input field (not rquried) the container beneath the searchbar displays all database entries of the selected format and (if given) names like the input field values.
* Clicking on one of the search results, will display alle the decklist contents of the databse entries of the cards_decklists table, based on the chosen deck_id, by linking to the Deck Contents Display viewing component.