Application name: Cards Playground

Problem area:

Cards Playground is a program designed for enthusiasts and players of trading card games. It serves the field of deckbuilding and custom cardmaking communities of games such as Magic The Gathering, Yu-Gi-Oh, World of Warcraft TCG, Pokemon TCG etc. The system aims to enable the creation, management and analysis of custom cards and decks.

Project goals:

The primary goal of the application is to ease the creation and management of cards for card game fans. It supports the users in designing cards, managing them, organizing their collections and building decks by bringing features and tools at the user's disposal, all in one place.

System responsibilities:

- Allow users to create and customize their own cards
- Enable users to view, search and filter cards from their collection
- Provide functionality to change, update or delete existing cards
- Implement an account system, so each account has it's own collection
- Enable users to build and save decks using cards from their collections
- Provide analytical tools for deck creation
- Provide support for uploading your own images as card's art
- Offer the ability to generate semi-random attributes and values for each of the custom card's field in card creation

System users:

-Guest
-Registered user
-Admin

Functionalities:

Card creation:

- Visual card creator tool
- Upload custom images for cards
- Generate semi-random attributes for each of the card's field

Collection management:

- Browse your cards and decks collection
- Use filters to search for specific cards
- Edit already existing cards, by loading them into card creator

- Edit already existing decks, by loading them into deck creator
- Delete cards and decks from your collection

User management:

- Registering an account
- Logging in

Deckbuilding:

- Create decks by adding cards from the user's collection
- Give user analytical data: mana distribution curve, toughness and attack distribution curves

User requirements:

Part one:

Entities:

- User: id, username, password
- Card: id, name, cost, type, attack, toughness, text, image
- Deck: name

Relationships:

- User can own multiple cards
- User can own multiple decks
- Deck can contain multiple cards
- Each deck has exactly 1 owner
- Each card has exactly 1 owner

Part two:

Card creation:

- Visual Card Creator tool: Users can visually create custom cards by specifying attributes such as name, cost, type, attack, toughness, text, and art image. This tool will allow the user to see a preview of the card as they are designing it.
- Upload custom images: Users can upload their own images to be used as the card's art. This feature will support common image formats such as JPEG, PNG etc.
- Generate semi-random attributes: For each listed field of the card the user can choose to assign a random value for it. Note that for fields such as name, type, text the randomly generated value will be a value randomly assigned from a determined list of possible attributes.

Card management:

- Browse cards collection: Users can view all the cards in their respective collection. The

system will provide an organized visual interface to facilitate browsing.
- Search tool for cards: Users will be able to search for specific cards in their collection, by searching for their names or and apply additional filters to the search. Examples of such filters would be only showing cards above a certain mana cost, or displaying cards with specific names
- Edit existing cards: Users will be able to take a card from their collection and load it into the card creator tool.
- Delete cards: Users can delete cards from their own respective collections.

User management:
- Register: Guests can create their own accounts by providing a name and a password. This will allow for accessing the majority of the functionalities of the program.
- Log-in: The users can log in into their accounts to gain access to the collection of cards and decks assigned to that account. This ensures that each user can have their own private and secure experience.

Deckbuilding:
- Create decks: Users can create new decks by adding to them cards from their own collections. This feature provides a visual interface where the user can click to add cards from their collections.
- Save decks: Users can save their created decks under any custom name, allowing the user to manage multiple decks. Each saved deck is stored and binded to the account.
- Edit decks: Just like with custom cards users can edit and update their existing decks by loading them into the deck builder tool.
- Delete decks: The decks can be deleted as they are in spirit a bundle of cards. This will not affect the cards in any way. The deletion of a deck cannot impact anything apart from itself. This allows the user to keep an organized collection.
- Analytical data: The system provides analytical tools that give the users insights into their decks. Those insight are: showing the mana distribution curve to help the user build a balanced and fun deck; distribution of card statistics (toughness and attack) to show the user, which card stats might be under or overdeveloped for the deck.

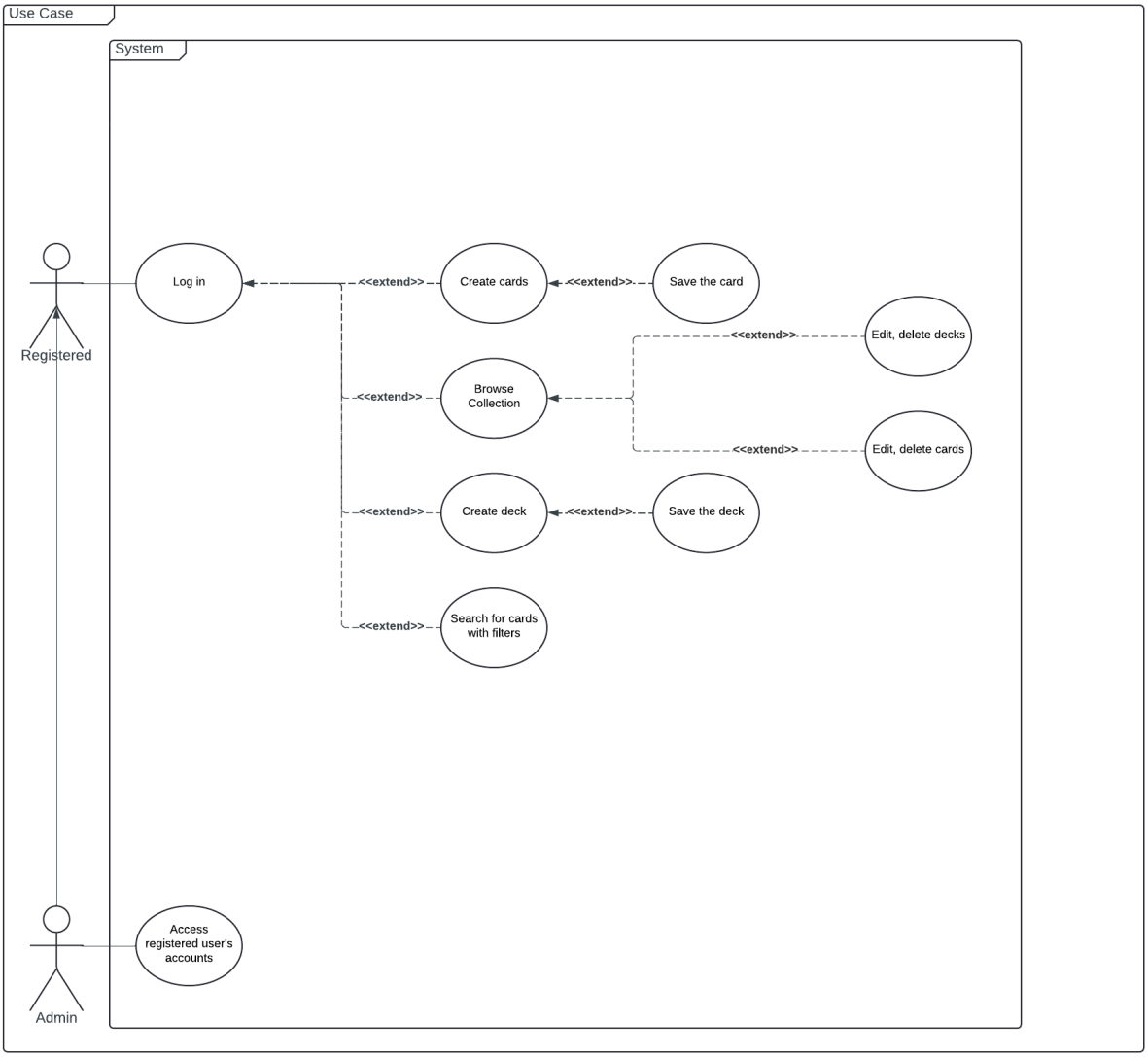
Inheritance:
- Registered user: the registered account has its own assigned collection, therefore is able to save the cards, view the collection, create decks etc.
-Admin: inherits from Registered user; has additional privileges and access to collections and accounts of registered users.

Part three:
- The application is meant for desktop devices

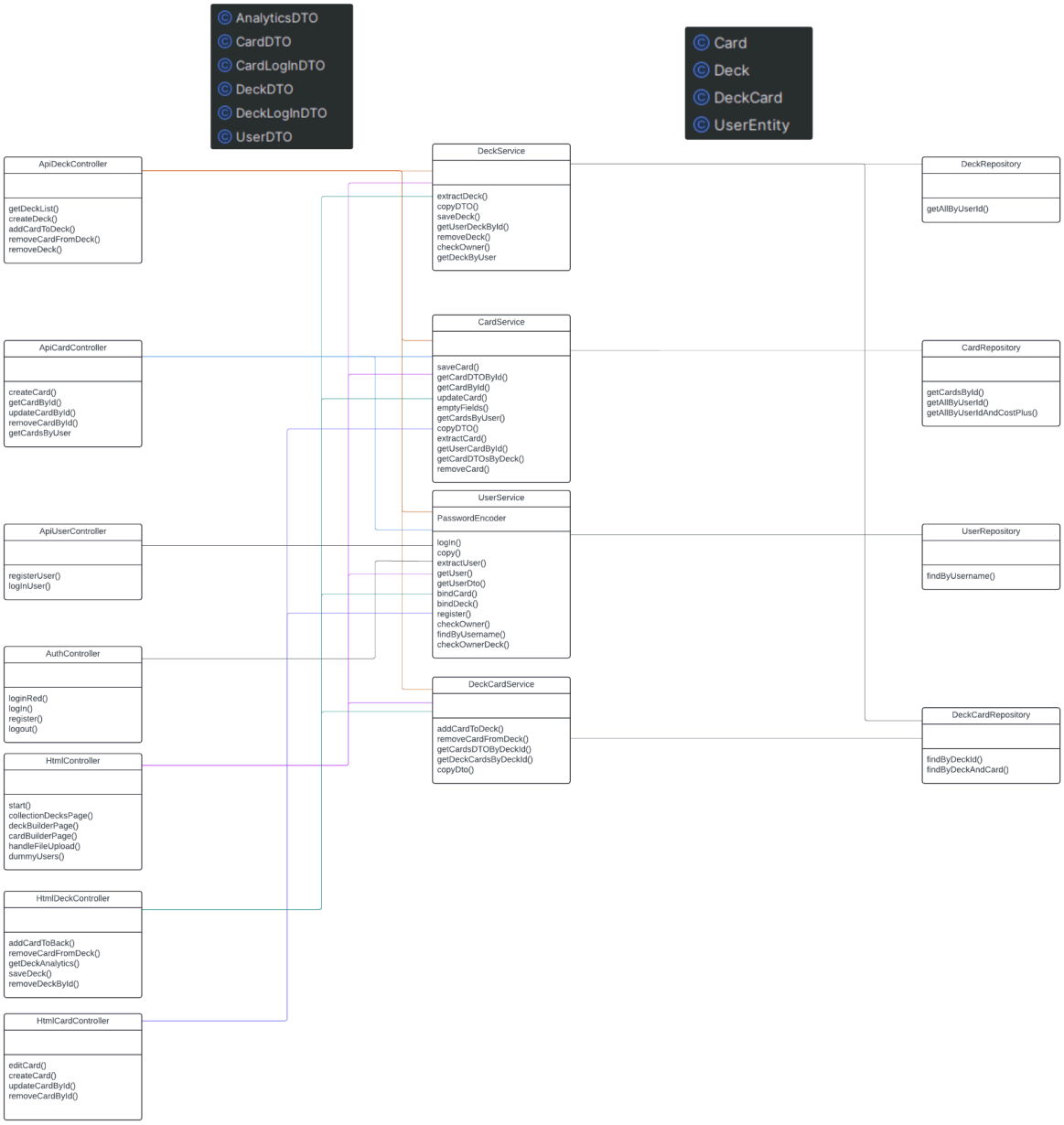
- The system is meant to be compatible with modern web browsers
- The application should be able to load pages and operate smoothly, without significantly long loadings
- The user passwords must be securely saved
- The application should have implemented measures to prevent common security vulnerability and exploits such as code injections
- The user interface should be intuitive and clean for the user to have the best experience
- The application would allow for a visual setting to be changed [light mode and dark mode] for better user experience and readability

Functional requirements:

Use Case diagram:


Use Case
System
Registered
Log in
<<extend>>
Create cards
<<extend>>
Save the card
<<extend>>
Edit, delete decks
<<extend>>
Browse Collection
<<extend>>
Edit, delete cards
<<extend>>
Create deck
<<extend>>
Save the deck
<<extend>>
Search for cards with filters
Admin
Access registered user's accounts

Description of the system structure:



Non-functional requirements:

- Compatibility with modern web browsers such as Chrome, Edge etc.
- Metric: the application will be tested on different browsers to ensure compatibility and that the all of the core functionalities work without issues

- Smooth operation and fast load times, the application should load pages and operate smoothly, without long loading times

- Metric: To test the loading and operation of the application the program will be tested on different devices

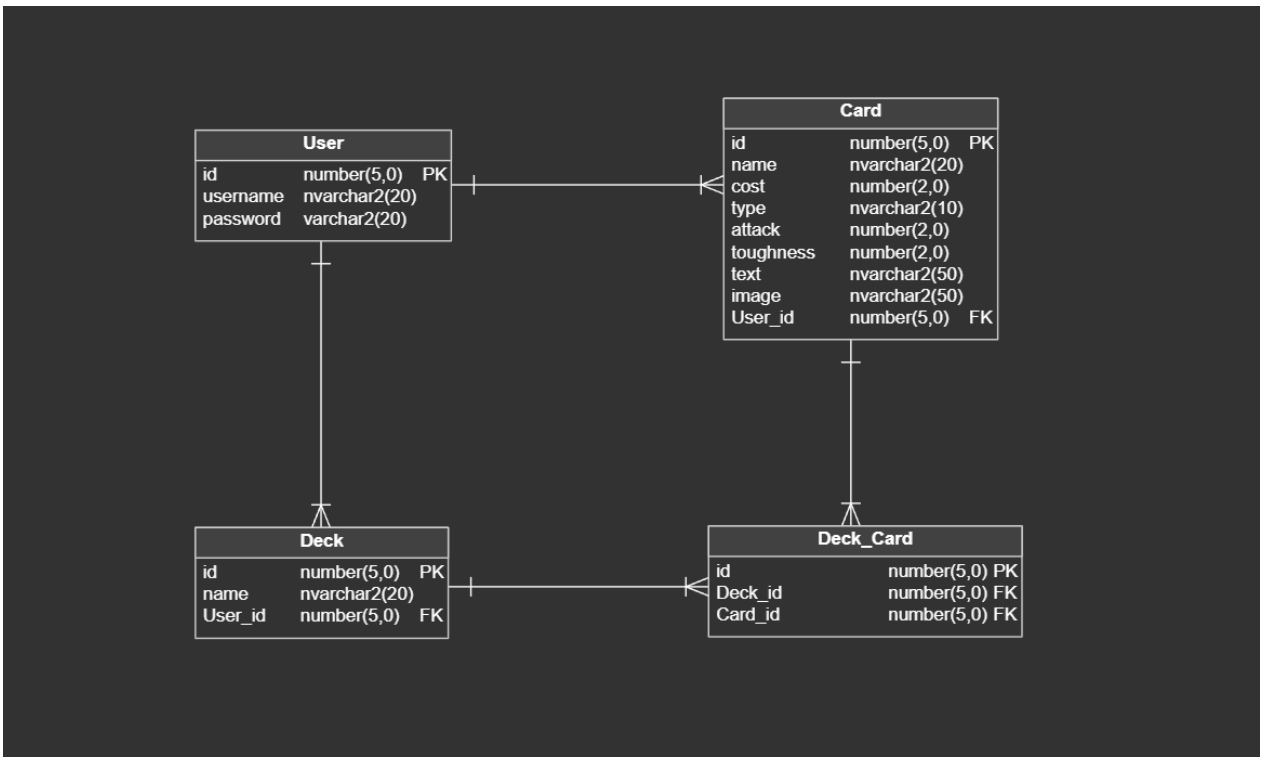
- Passwords of the users must be stored securely
- Metric: the password will be hashed to ensure security

- Protection against vulnerabilities, the application should be able to prevent some common vulnerabilities such as code injection into text fields
- Metric: all information passed from the user will be done in a secure manner to ensure no injections happening

- The user interface should be intuitive and clean to ensure the best user experience
- Metric: the application will be shown to people with no background in card games to look for ways to make the application more intuitive to new users

- The application should implement visual settings for better user interface clarity and accessibility: light and dark mode
- Metric: Users should be able to switch between light and dark mode with just a few clicks, and the option to do so should be visible in every moment of using the application

Database schema:



API endpoints list with description:

POST /api/users/register
- allows to register an account

POST /api/users/login
- allows to login to an account

GET /api/cards
- retrieve all cards from a user’s collection

GET /api/cards/{id}
- retrieve a card by id

POST /api/cards/create
- create a new card in a users collection

DELETE /api/cards/remove/{id}
- remove a card

PATCH /api/cards/update/{id}
- update a card

GET /api/decks/{id}
- retrieve a deck by id from a user

POST /api/decks/create
-create a new deck in a users collection

POST /api/decks/add/{deckId}/{cardId}
-add a card to a deck

DELETE /api/decks/remove/{deckId}/{cardId}
-remove a card from deck

DELETE /api/decks/remove/{id}
-deletes a deck from a users collection

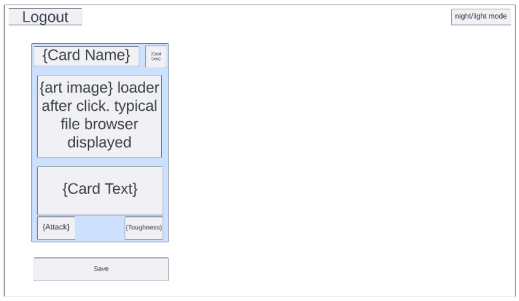
List of used technologies:

- Spring Data JPA: used for interaction with database
- Hibernate: database operations
- H2 Database: used for databases
- Spring web: used for handling web requests and responses
- Thymeleaf: used for rendering web pages
- Bootstrap: also used for frontend

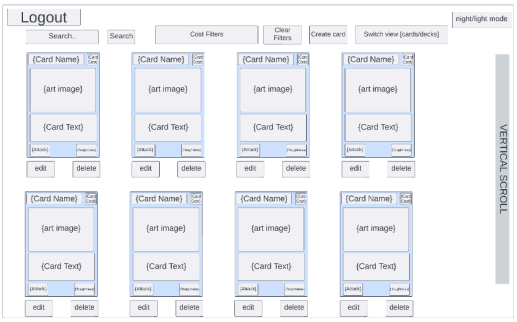
- Postman: for api testing
- Spring security: used for security
- JSDelivr: for creating charts in deck creation

Visualization

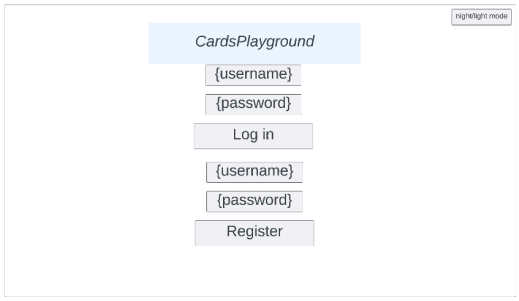
Visual card creator



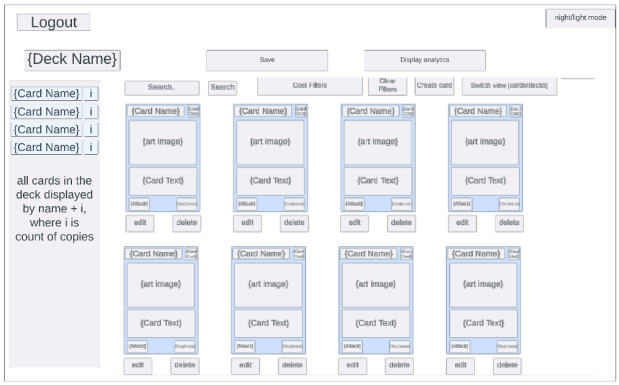
Collection



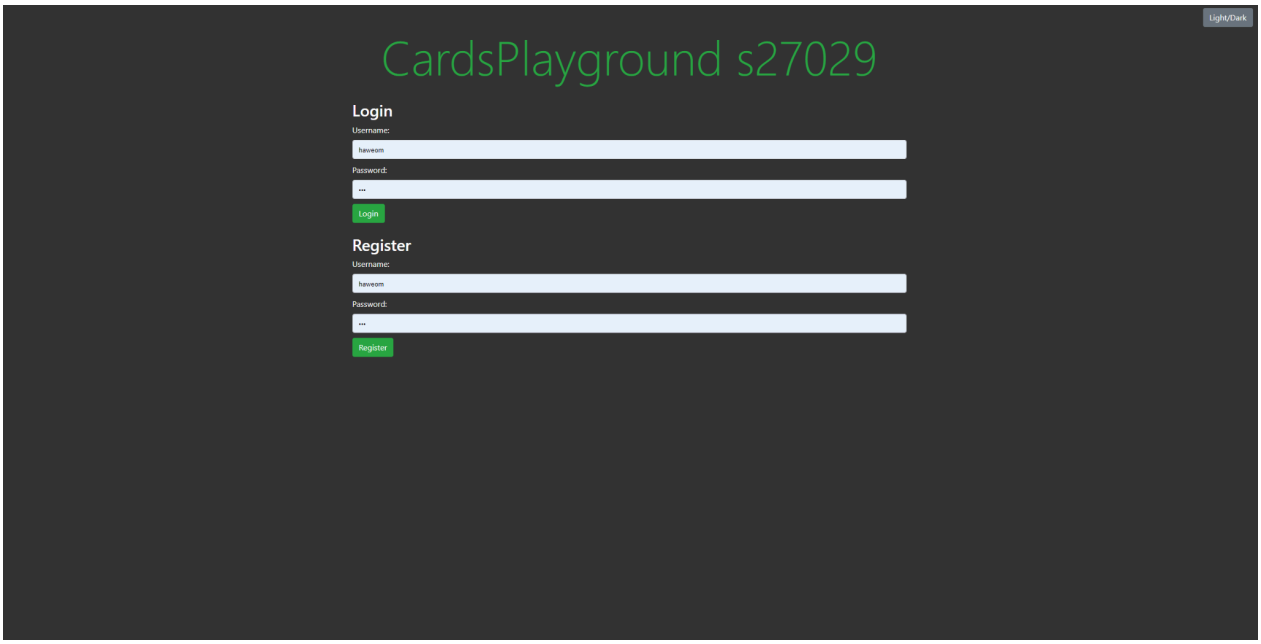
LogIn screen

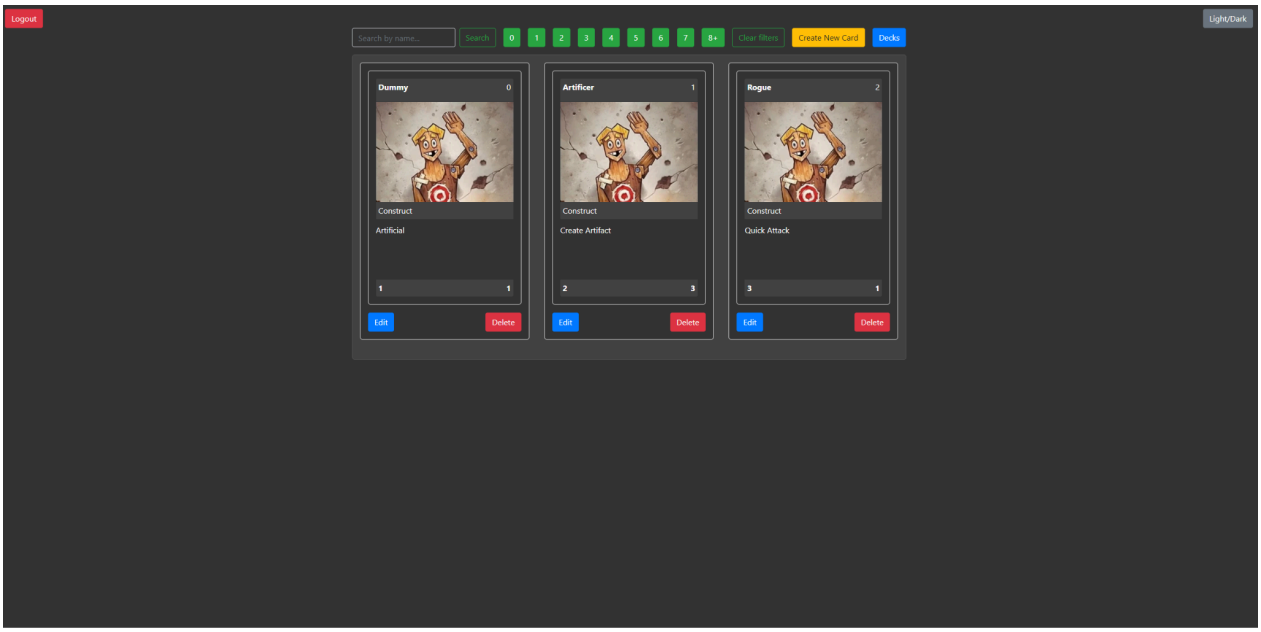


Deckbuilder

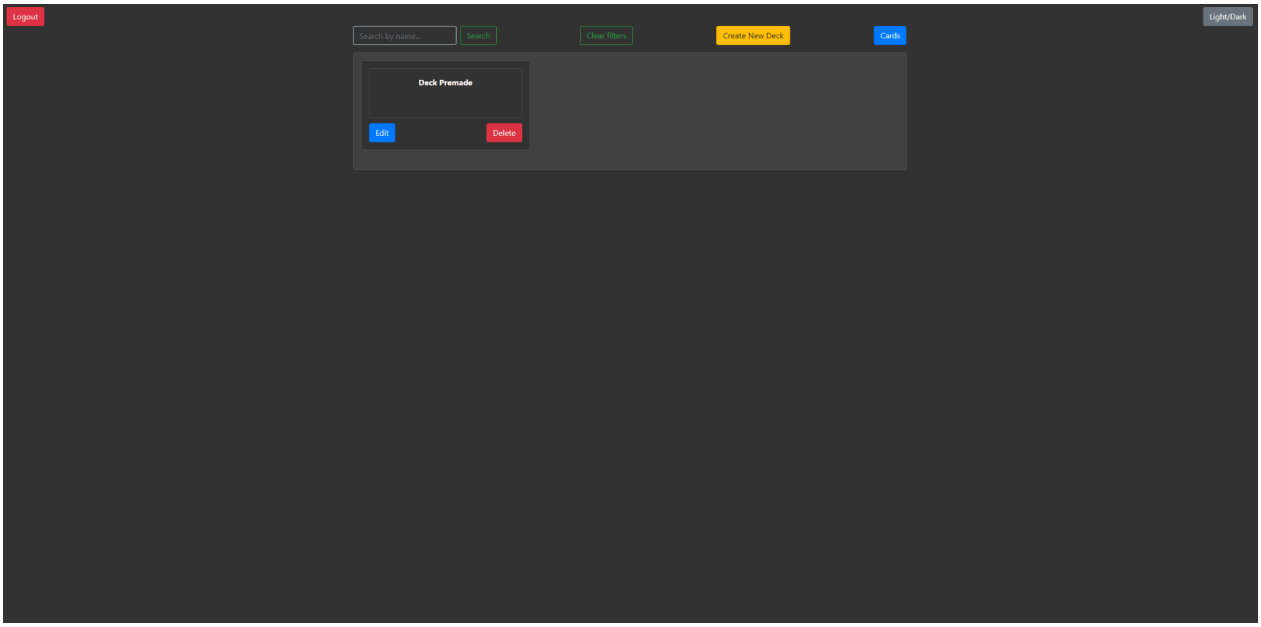


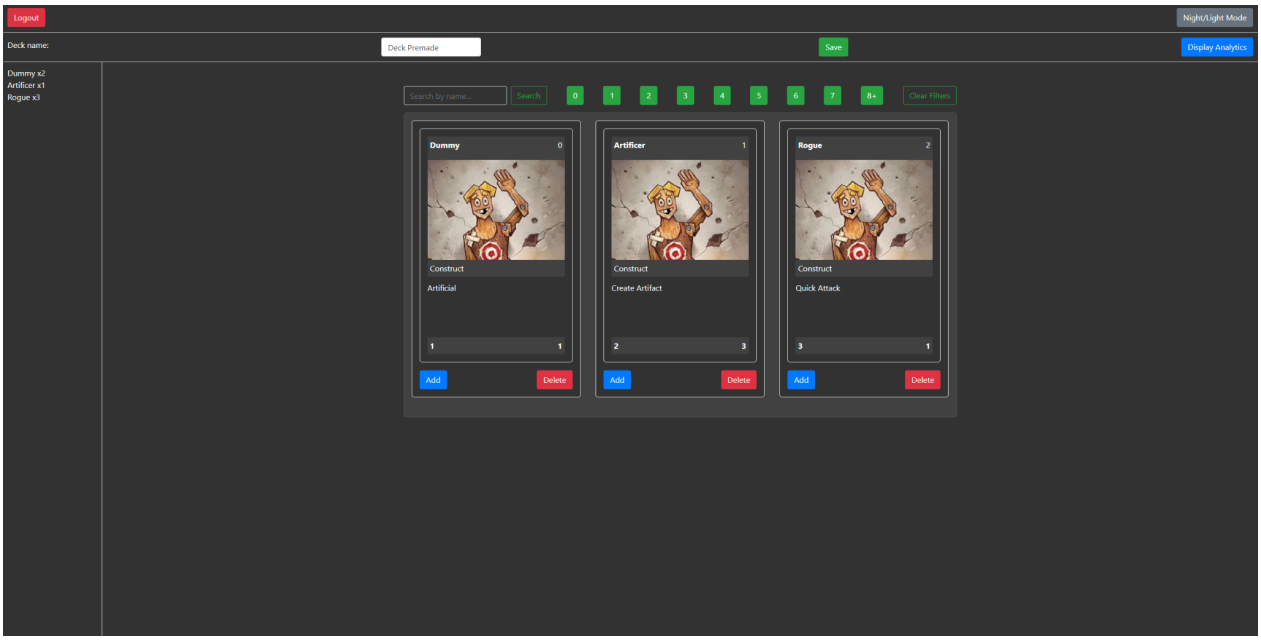
Actual looks:






Back
Card Builder
Light/Dark
Name:
Cost:
0
Randomize
Randomize
Image:
Choose File
No file chosen
Type:
Randomize
Text:
Randomize
Attack:
Toughness:
0
0
Randomize
Randomize
Save Card


Logout
Light/Dark
Search by name...
Search
Clear filters
Create New Deck
Cards
Deck Premade
Edit
Delete



Changes to documentation:
-added more technologies

-redefined APIs to work with the actual program

-changes to diagrams to better reflect project structure

-added changes visualizations