

DeckTechCentral

PennWest California

CMSC 4900 – Senior Project I

Project Specifications

Dr. Chen

11/14/2023

Team Member	Major	Project Phase
Christian Messmer	Computer Science	Implementation
Paul Shriner	Computer Science	Analysis
Adir Turgeman	Computer Science	Presentation
Luke Vukovich	Computer Science	Design

Instructor Comments and Evaluation

Table of Contents

Instructor Comments and Evaluation	1
Table of Contents	2
Abstract.....	4
Description of Document.....	5
Purpose of Document	5
Intended Audience.....	5
System Description.....	6
Overview	6
Environment and Constraints	6
End User Profile	6
User Interaction	7
Hardware Constraints	7
Software Constraints.....	7
Time Constraints.....	8
Cost Constraints.....	9
Other Concerns	9
Acceptance Test Criteria	10
Testers.....	10
Criteria for User Acceptance	10
Integration of Separate Parts and Installation	11
System Modeling	12
Use Cases & Scenarios.....	12
Class Diagram	16
Dynamic Statechart	18
Main Interface Statechart.....	18
Dataflow	20
Login Dataflow	20
Search for Card Dataflow.....	22
Components / Tools Needed	24
Development Requirements	24

Back-End Server Requirements	25
Appendix.....	27
Technical Glossary	27
Team Details	32
Workflow Authentication.....	34
Report from the Writing Center	35
References.....	36

Abstract

The software engineering specifications are precisely outlined and designed for the proposed software product, DeckTechCentral, a web-based deck list management tool for “Magic: The Gathering” (MTG), a trading card game. This document will provide an in-depth description of the product specifications. After characterizing the intended audience, we provide an overall system description with discussion points consisting of development and production environments, product constraints, and test criteria. System modeling techniques are then used to describe a sample Use Case scenario, an evaluation of Entity classes, and a Dynamic State Chart of Front-End states. Finally, we provide dataflow diagrams for logging in and searching for a card, as well as list the CASE tools needed to successfully develop and deploy the product.

Description of Document

Purpose of Document

The purpose of this document is to finalize all specific software requirements for the product. The specifications include, but are not limited to, hardware requirements, cost constraints, product architecture, software design, and CASE tools. The DeckTechCentral team along with the client will utilize this document to assist in the development and implementation of the product. Once the team and client are satisfied with the specifications, this document will act as a settled agreement between the two parties.

Intended Audience

The intended audience of the document at hand is the entire DeckTechCentral team and the clients. The clients of the product are members of the development team and the professor of the senior project course. It is understood that being a client and a member of the team can introduce issues. However, it is important to note that the product will not move forward until the team fully agrees on the specifications from a client and development perspective. Christian Messmer is taking on the main role of the client. He is the final acceptor of the document and will form the final agreement between the client and development team. If the client is not satisfied, the two parties will have to work together to come to a resolution.

System Description

Overview

DeckTechCentral is a web-based deck list management tool for the game “Magic: The Gathering”. Users will create, manage, and view deck lists via the Front-End, which is a web-based user interface that is designed with ease-of-use and simplicity in mind. The Front-End can send and receive deck list information to and from the Back-End. The Back-End will host all deck list and user information, as well as process and maintain incoming deck list information. Further, the Back-End can send and receive data to and from the Front-End. This microservice design will enforce a full stack environment.

Environment and Constraints

End User Profile

The primary end users of DeckTechCentral are players of MTG who want to have a central place for deck lists to be used in MTG games. Users do not need a minimum experience level in the game in order to use DeckTechCentral, but they will be expected to have some familiarity with the game. While DeckTechCentral could be used as an educational tool (e.g., to learn about the different types of cards), it does not aim to teach one how to play MTG. Further, users will only need minimal technical experience; if they are able to go to a website in their web browser then they can use our web application. Users will be expected to be proficient in English, as support for other languages cannot be guaranteed in the initial rollout. DeckTechCentral is not intended to be used as a platform by corporate entities to market (related or unrelated) products, rather it is intended to benefit the MTG community.

User Interaction

A user's means of interacting with DeckTechCentral will depend on the device they are using to access the application. A smartphone's input includes human fingers or a stylus, while a computer's input includes a mouse and keyboard, which can be used separately or at the same time. As DeckTechCentral will be built to be responsive regardless of the platform, the actual experience, including actions and use cases, should be identical for all users.

Hardware Constraints

A user must have internet access to fully utilize and operate the DeckTechCentral website. Users will be able to access the website within a PC or mobile device environment. Any device that has the capability to install and run a modern internet browser is also viable.

Our database is only as capable as the hosting environment. Therefore, we must choose a database hosting environment that satisfies our data needs. The amount of data that needs to be stored will be small initially but can vastly grow based on the number of users. If needed, we can always adjust the hosting plan to ensure we can store all necessary data and offer our full services.

The main advantage of using a third-party server hosting platform is the ability to adapt on the go. We can always adjust our server plan to ensure we can successfully adapt to our user, data, and speed needs.

Software Constraints

The only end-user software constraint is a modern and supported internet browser. We will target Chromium browsers including Google Chrome and Microsoft Edge due to their popularity and widespread use, but we see no reason why our product wouldn't work on non-

Chromium browsers, such as Mozilla Firefox. Other than an internet browser, no other prerequisite files or software need to be installed.

The Back-End has several constraints that must be considered and discussed. The team intends to implement a microservice design meaning that our production environment must have Docker compatibility. Containerization is an important aspect during product development and deployment. As previously mentioned, the product also requires a database. Therefore, the team must choose a cloud service provider for containerization and deployment along with a database hosting provider. The two software solutions must interact with each other to have a complete data flow. Fortunately, there is a lot of software available to satisfy our needs.

Time Constraints

As DeckTechCentral will be made for an undergraduate senior project course, it will need to be finished by approximately the end of April 2024. However, the team members will inevitably have other responsibilities that take time away from this project, including other classes, extracurricular activities, employment, family duties, and more. Time management and planning will be very important to ensure the members are still able to participate in recurring meetings and contribute to the project. At a minimum, the features expressed in these documents must be implemented in the project. If there is a “nice to have” feature the team wants but there is not enough time, then it will have to be cut.

For the product itself, all functionalities will need to operate in a reasonable amount of time. As this is not an application critical to life, near instant response times are not required. However, functionality should respond quick enough that the user does not feel they are waiting for long periods of time. Some of the most notable time constraints in our project will be the time

to load the page, show a Magic card including details and pictures, show an entire deck list, and show ratings on a deck.

Cost Constraints

We plan on keeping costs down to a minimum as much as possible by using open-source software and free-to-use solutions. The inevitable costs will include those for purchasing a website domain and running the server. The user will have no costs at all as long as they already have a device that can access our application. Our benefits will be intangible, and its main goal will be to improve user experience. With our limited resources we won't be able to get much reliable feedback from the users, therefore we will focus on bill of cost more than cost and benefit analysis. the cost of domain name is ranging between 10-20\$ on average, while renewing it might cost up to 60\$ a year. For our project we are planning on buying a domain name for the year. At this point we're not projecting any more expenses therefore our bill of cost will be set at 20\$.

Other Concerns

An important concern for this project will be securing account information. Users should feel confident that their information will not be stolen by a bad actor. One potential option we are looking into is to use OAuth. With this, a user would sign into our application through a service that provides OAuth (such as Google), and an access token would be returned to DeckTechCentral. The benefit of this option is that we would not have to "reinvent the wheel" with account security, and the user would not have to create a separate account just for our application.

Acceptance Test Criteria

Testers

Testers for DeckTechCentral will include all members of the development team as well as the professor of the senior project course. With permission, further testing can be done by individuals chosen by the development team or the professor.

Criteria for User Acceptance

DeckTechCentral must be able to perform a number of functions, which vary depending on the access level of the user.

Guest (Unauthorized User)

- A Guest may access the DeckTechCentral website and view the public library of existing deck lists and Magic cards. The public library of deck lists is a user-based database of created deck lists that the creator grants public access to.
- Guests cannot create or manage personal deck lists.
- Guests can create a DeckTechCentral account. This requires an email, a username, and a password. The team's goal is to implement OAuth to authenticate users.
- OAuth will handle all user authentication acceptance. DeckTechCentral will pick up user handling once authentication is complete.
- Guests can export a deck list which can be used outside of DeckTechCentral.

Member (Authorized User)

- A member can log into their existing DeckTechCentral user account through OAuth.

- A Member may view the public library of existing deck lists and Magic cards. They may also create, manage, and delete their own personal deck lists that are linked to their verified user account.
- Members can like a personal or public deck list if they feel inclined to do so.

Moderator

- Moderators are able to review the website for spam or any hate-related content. They may remove content or ban users if it is required. Moderators must provide evidence as to why such actions need to be taken.

Integration of Separate Parts and Installation

Users must have internet access to view and utilize the DeckTechCentral website.

Accessing the website will initialize communication with the server.

The Server has two main parts that must be integrated: the Database and the HTTP Server. These may be configured to run in separate Docker containers. The Database and the HTTP Server must be able to communicate. Docker must be installed on the cloud servers of the hosting provider we select. The Docker containers must be configured and coordinated.

The Front End integrates with the Back End through HTTP Requests and Responses. The Front End will be configured to respond to HTTP Requests by relevant changes in the Front-End that realize the display of data requested. The Server is responsible for parsing requests and providing the relevant data.

System Modeling

Use Cases & Scenarios

Upon opening the website, the Guest will see the home page that consists of a main search bar. If the Guest wants to login, they will also be presented with a method to sign in or sign up to be a user.

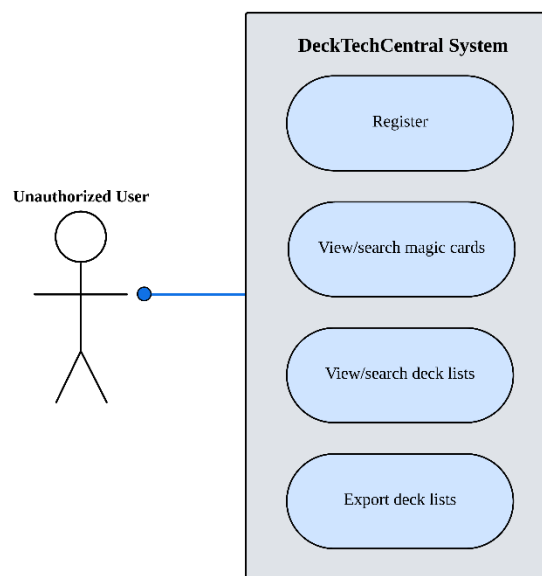


Figure 1. Average Guest (Unauthorized User) use cases

Figure 1 above depicts the average use cases of a Guest that will visit the website. The Guest can see the main search bar to search for Magic cards or deck lists. The Guest can also choose to register if they want to become a Member (Authorized User).

Register Scenario

- The Guest navigates to the product website. They can utilize the search bar to search the public library of Magic cards and decks.

- The Guest interacts with the Register function.
- Once the Register function is selected, the Guest will be presented with the OAuth authentication page.
 - Upon a successful submission, the Guest will be given a proper User ID and will now be considered a Member (Authorized User).
 - Upon an unsuccessful submission, the Guest can either try to authenticate again using OAuth or continue using the website as a Guest.

Search Scenario

- The Guest navigates to the product website. They can utilize the search bar to search the public library of Magic cards and decks.
- The Guest interacts with the search bar and decides to search “best deck list”.
- The Search function will handle the search request and display the results that are related to “best deck list”.
 - The Guest selects the first deck list that is provided from the search results.
 - All information regarding the selected deck list is displayed.
- Once finished with the current set of search results, the Guest will make another search.

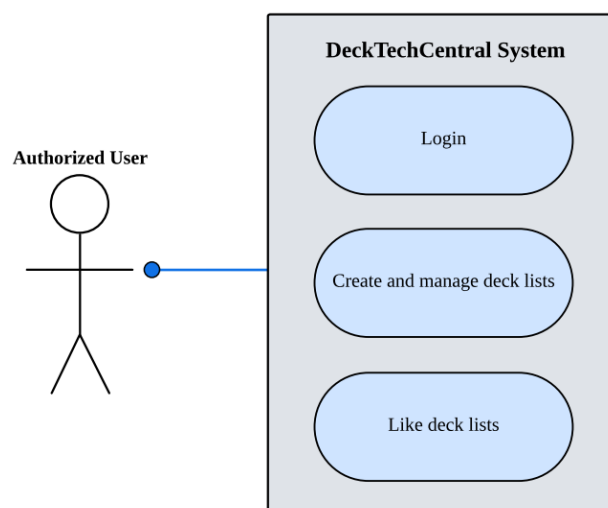


Figure 2. Average Member (Authorized User) use cases

Figure 2 above depicts the average use cases of a Member that will visit the website. Member inherits all functionalities of the Guest. The Member role adds on few distinctive functions. The Member can login to the website as well as create and manage personal deck lists.

Login Scenario

- The Member navigates to the product website.
- The Member interacts with the Login function.
- Once the Login function is selected, the Member will be presented with the OAuth authentication page.
 - Upon successful login, the Back-End will connect the existing User ID to the Member. They now have full access to the website as a Member.
 - Upon an unsuccessful submission, the Member can either try to authenticate again using OAuth or continue using the website with Guest privileges.

Deck List Scenario

- The Member navigates to the product website and successfully utilizes the Login function.
- The User now has access to the deck list menu.
 - The Member utilizes the Create Deck List function. The Create Deck List state will now display, and the Member starts to create their very own deck list.
 - The Member utilizes the Manage Deck Lists function. The Manage Deck Lists state will now display. This will provide the Member with all deck lists they have created.
 - If the Member has created no deck lists, this page will be empty.

- If the Member has created deck lists, the page will display the deck lists.
 - The Member utilizes the Edit Deck List function on their first deck list. The Member utilizes the function to edit the deck list.
 - The Member utilizes the Delete Deck List function on their first deck list. The deck list is deleted.

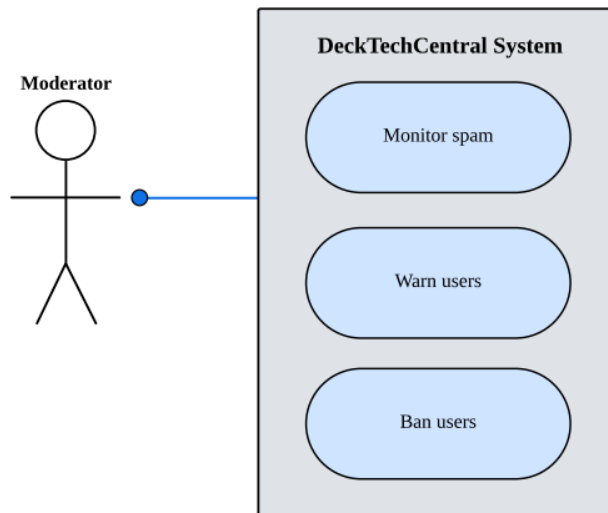


Figure 3. Average Moderator use cases

Figure 3 above depicts the average use cases of a Moderator. Moderator inherits all functionalities of the Member. The Moderator has a few extra functions such as monitoring spam and banning users.

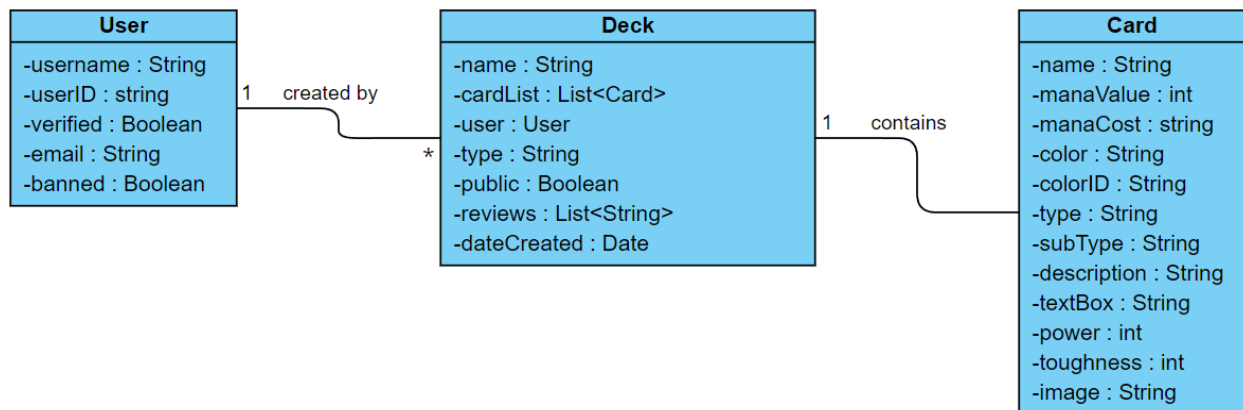
Ban Scenario

- The Moderator accesses the website. It is assumed that the Moderator is logged in and already has been given proper Moderator privileges.
- The Moderator will scour the website and look for any type of ill intent. To do this, they view the public library of deck lists from the home page and filter the results to display the most recently created decks.

- The Moderator sees a massive number of deck lists created in a row by the same user.
Each deck list was created seconds apart, which is unrealistic for a legitimate user. The Moderator assumes this is some sort of attack.
- The Moderator navigates to the user from the deck list information. From the user interface, they have the option to ban a user:
 - Automatically ban the user as the actions are extremely harmful. The Moderator can do this by using the Ban function.

Class Diagram

The figure below represents a class diagram. Each class and its relationship will be outlined and explained.



User Class

The User class stores the username, email, and userID for deck querying. There are also two associated Boolean values that store verification and banning information.

Deck Class

The Deck class will store a list of Card objects which is the main composition of the decks. A User object will allow for easy user association. The public attribute is responsible for

making a deck public or private. The reviews attribute is a list of strings that contain all the associated reviews of a deck. The Deck name and type are standard string data types.

Card Class

The Card class contains all necessary information to perfectly describe and allocate an individual playing card. All data associated with a card will be pulled from the Scryfall API.

User-Deck Relation

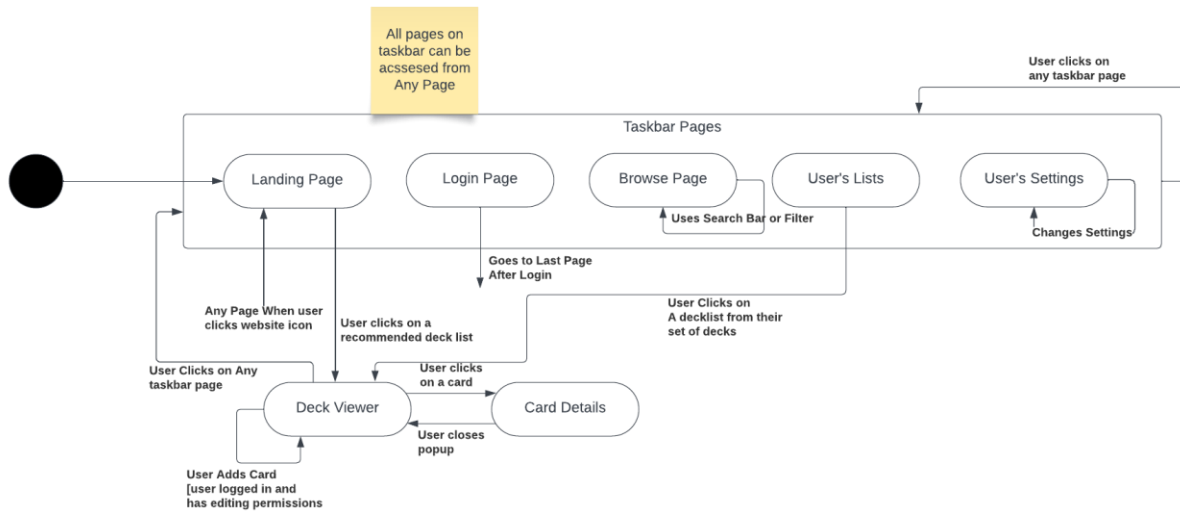
A User can create multiple Deck objects. The '1' near the User entity indicates that each User can be associated with multiple Deck entities. Each Deck stores only one User object. The '*' near the Deck entity indicates that each Deck is associated with exactly one User. This User object is used to associate the User that created the Deck.

Deck-Card Relation

A Deck contains and is made up of multiple Card entities. The '1' near the Deck entity indicates that each Deck can be associated with multiple Card entities. Decks are highly dependent on Cards. A Card is a standalone class and does not depend on a Deck or any other data.

Dynamic Statechart

Main Interface Statechart



States

- Landing Page
 - The default state when any user navigates to DeckTechCentral.
 - If the user is logged out, they can login or use the search bar to view public decks.
 - If the user is logged in, they can use the search bar or create a deck.
- Login Page
 - Allows the user to become authenticated.
 - From login, the user can return to the Landing Page, go to the Deck Search Page, or go to the Deck Viewer.
- Browse Page
 - If the user is logged out, they can go to the Login Page from here.
 - All users can use the search function or view a deck.

- Deck Viewer
 - Allows the user to view a deck.
 - Logged in users can also create a deck.
 - Logged in users can edit the deck given they have editing permissions for it.
- User's Lists
 - All deck lists that the user has created or has editing permissions for.
 - Users can create a new deck from this page.
- User's Settings
 - Users can edit their settings such as display name from this menu.
- Card Details
 - A popup that will display more details about a card, such as mana cost, name, text, legality, and others along with the image of the card.

Events

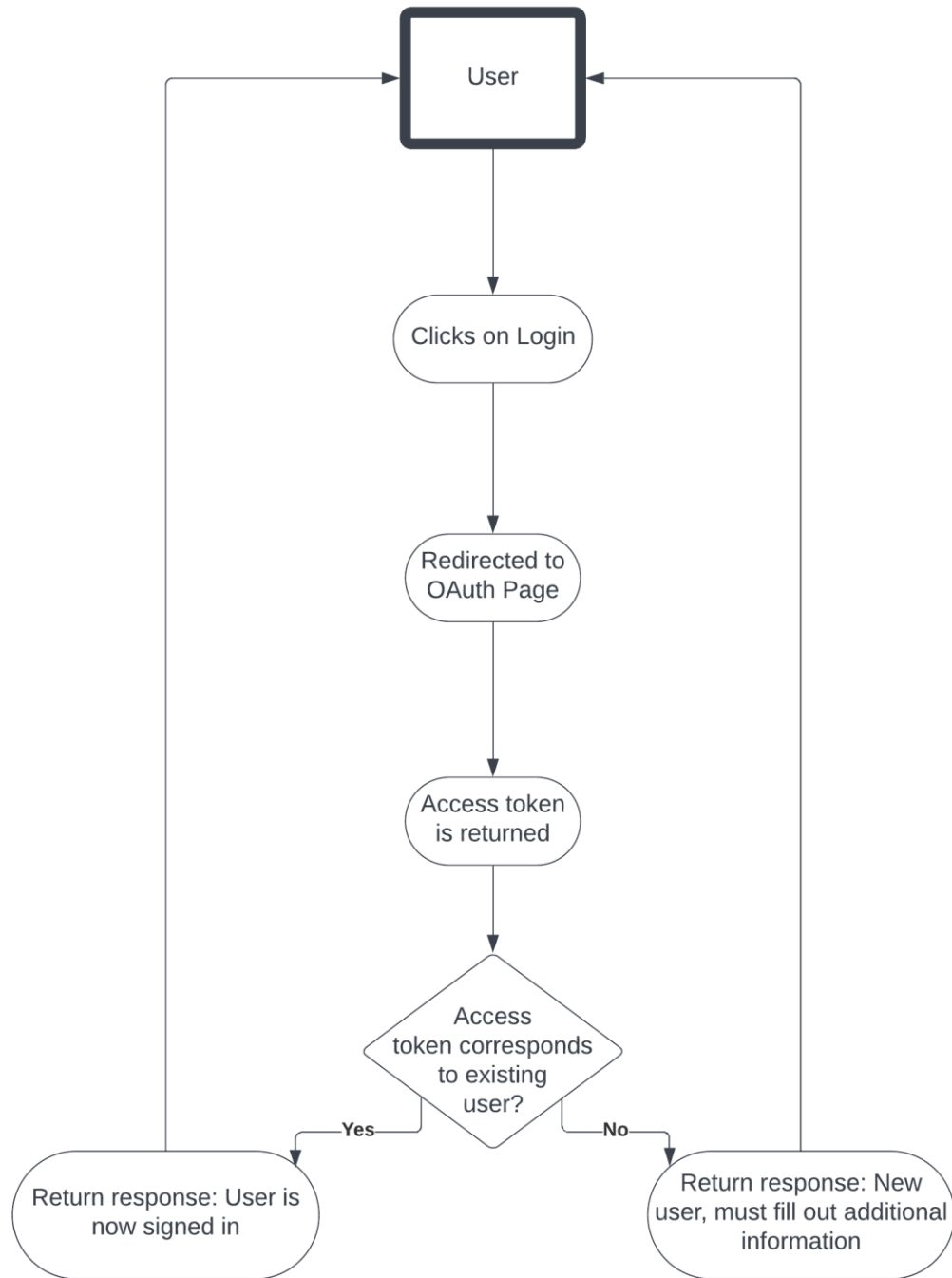
Events will occur when a user does any action on the page. For example, pressing a button will require checking different conditions and possibly moving the user into a different state. Once this has occurred and all data has loaded in, then no further events should occur.

Transitions

Transitions will occur due to a user doing an action on the page. The user will start on the Landing Page, and from there potentially go to different pages on the application. At any points where the user enters data, error-checking will be needed to ensure the user does not intentionally enter invalid data (for example to cause an overflow or exploit a vulnerability).

Dataflow

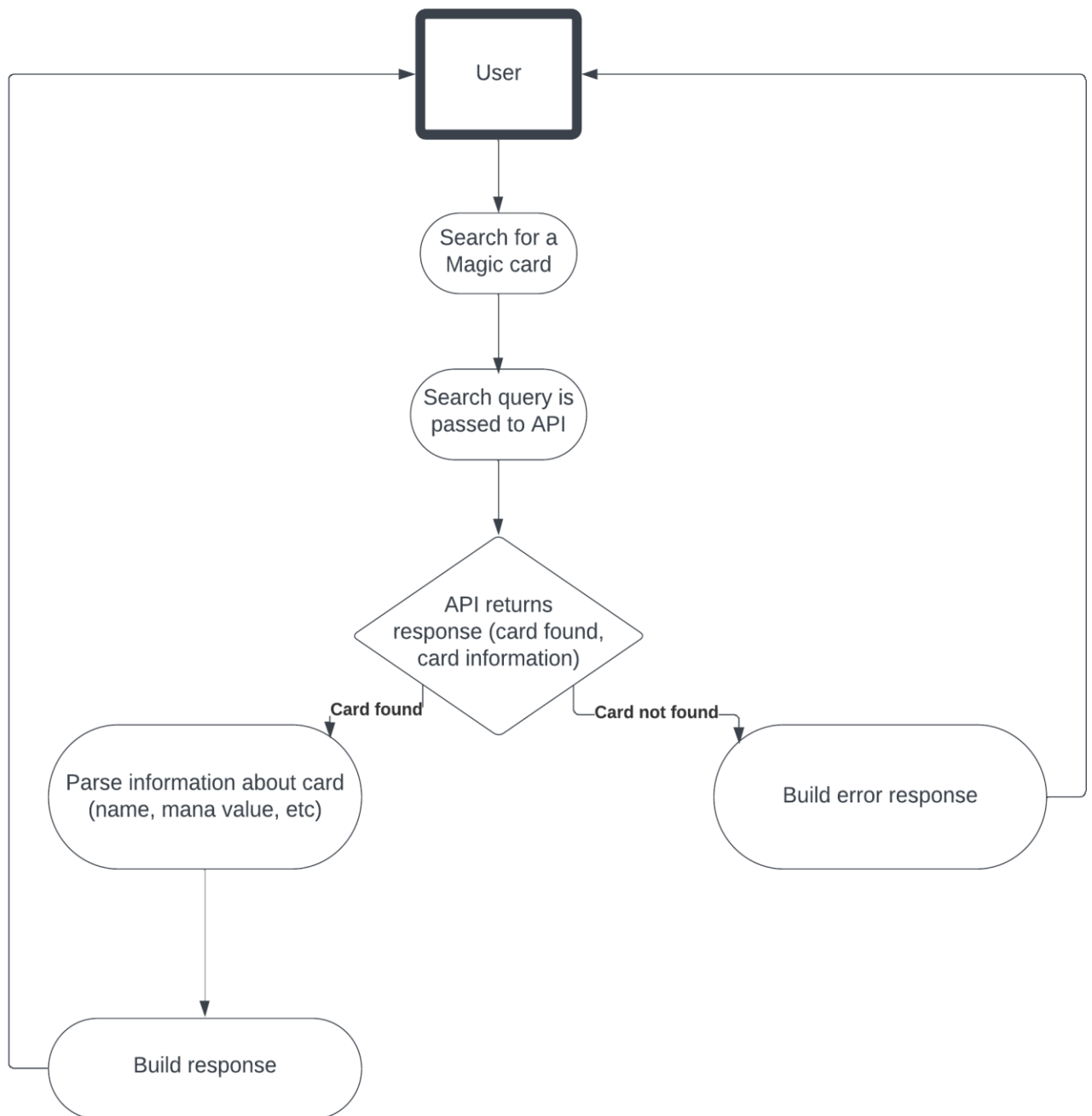
Login Dataflow



This dataflow diagram shows how a login request to DeckTechCentral is handled.

1. A user makes a login request by clicking on the “Login” button.
2. They are redirected to an OAuth page for signing in. The user may be able to sign in immediately, or they may need to create an account with the OAuth provider we use, but it does not matter to our application as it is handled separately. What is important to our application is when the user successfully signs in, which leads into the next step.
3. The OAuth provider returns an access token to our application.
4. The token will be cross-checked with our account database. The token is either already associated with a DeckTechCentral account or it is a new account.
5. If an associated account already exists, a response is returned to the user, and the user is now signed in.
6. If this is a new user, then more information will be needed from the user to create an account. This would be represented by a separate dataflow diagram.

Search for Card Dataflow



This dataflow diagram shows how searching for a Magic card is handled.

1. A user enters a search term into the search box.c
2. Their query is passed to an API.
3. The API returns whether a match is found for the card, and information about the card.
4. If no card was found, then an error response is returned to be shown to the user.
5. If a card was found, information about it will need to be parsed from the API response.
6. A response will be built with the parsed information to be returned and shown to the user.
7. This process would be repeated if multiple cards are a match for the search query given.

Components / Tools Needed

Development Requirements

Visual Studio Code

Visual Studio Code (<https://code.visualstudio.com/>) is a streamlined code editor with support for development operations like debugging, task running, and version control. Visual Studio Code utilizes third-party extensions to enhance development; this product requires the usage of the following extensions: Go, Typescript, Live Share.

Git

Git (<https://git-scm.com/>) is a distributed version control system designed to handle projects with speed and efficiency. Git is used to facilitate version control of product files.

Docker

Docker (<https://www.docker.com/>) is a platform-as-a-service product that uses Virtualization to enable developers to quickly deliver products by utilizing containerization. Docker is used to deploy and test the application's components.

React

React (<https://react.dev/>) is a JavaScript library created by Facebook designed for creating user interfaces (Codecademy, 2021). It is intended to make development easier through reusability and allow responsive design across platforms.

C#

C# (<https://learn.microsoft.com/en-us/dotnet/csharp/>) is an open-source programming language created by Microsoft (Microsoft, n.d.).

NPM

NPM (<https://docs.npmjs.com/about-npm>) is a source of various JavaScript software. It is required to use React.

Discord

Discord (<https://discord.com/>) is an online platform for group communication (Discord, n.d.).

One can make a server with channels for separating related conversations, allowing for improved organization. Further, Discord has voice chat and screen sharing capabilities, allowing group members of this project to effectively communicate their current progress.

Back-End Server Requirements

Docker

Docker (<https://www.docker.com/>) is a platform-as-a-service product that uses Virtualization to enable developers to quickly deliver products by utilizing containerization. Docker is used to deploy and test the application's components.

Scryfall API

Scryfall (<https://scryfall.com/>) provides a RESTful API for ingesting Magic card data programmatically. The API exposes information available on the regular site in easy-to-consume formats.

Postman

Postman (<https://www.postman.com/>) is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration to create better APIs.

MongoDB

MongoDB (<https://www.mongodb.com/>) is a document database with the scalability and flexibility for querying and indexing. MongoDB will be the primary means of storage for the product's data.

Appendix

Technical Glossary

API

Abbreviation for Application Programming Interface. A set of rules that allows programmers to develop software for a particular operating system without having to be completely familiar with that operating system (Merriam-Webster, 2023).

Authorized User

Has the same capabilities as an Unauthorized User but can also create and review deck lists.

Back-End

Handles data and requests from the user. (freeCodeCamp, 2022).

Bad Actor

A person with malicious intent.

Browser Engine

The underlying software that turns HTML pages into the Web page the user sees (PCMag, n.d.).

Examples include Chromium and Gecko.

CASE

Abbreviation for “Computer Aided Software Engineering”; the implementation of computer facilitated tools in software development (GeekForGeeks, 2023).

Class Diagram

A visual representation of the objects in the application, as well as their attributes and methods.

Containerization

The concept of keeping services separate from each other, making them easier to manage.

Database

Organizes and stores an application's resources in a way that is easily accessible and maintainable (freeCodeCamp, 2022).

Dataflow Diagram

Shows the flow of data for a given series of actions.

Deck list

Set of Magic cards the player utilizes throughout the game. In a real-life game, this would be a physical deck. In this project, it will be a list. Decks have a minimum of 40 cards and have different types of cards including lands and creatures (DiceBreaker, 2019).

Development Environment

A setup of tools and configurations suited for development.

Dynamic Statechart

A visual representation of the possible states a user could enter into, as well as the various paths to said states.

Front-End

What the user sees and interacts with (freeCodeCamp, 2022).

Git

A version control system (GitHub, n.d.).

GitHub

A platform for hosting and managing Git repositories (GitHub, n.d.).

Hardware

Physical components of a computer or related device. Examples of hardware include the CPU (Central Processing Unit), Memory, Storage, and Input Devices (keyboard, mouse, stylus, touch, etc.)

Hosting Environment

Can refer to where the data for a web application is stored and accessed, but it can also refer to a *Development Environment* (The European Financial Review, 2022).

HTTP Request

An action that specifies the transfer of data between servers. Some common request methods include GET for receiving data and POST for sending data (MDN Web Docs, 2023).

Magic

See *Magic: The Gathering*

Magic: The Gathering

Trading card game created by Richard Garfield in 1993 and published by Wizards of the Coast (MTG Fandom, n.d.).

Microservice Design

An architectural approach in which a single application is composed of many loosely coupled and independently deployable smaller components, or services (IBM, n.d.).

Moderator

Has the same capabilities as an Authorized User but can also take action against malicious activity. This includes, but is not limited to, spam or toxicity.

OAuth

Open Authorization, an open standard protocol for token-based authorization on the internet (TechTarget, 2020). Some platforms that provide OAuth functionality include Facebook and Google.

Parsing

The act of extracting information from a response for some purpose, likely to display it to the user or store it for later use.

Production Environment

The expected environment a user would be running the application on.

Software

What runs on top of hardware. An example is a web browser.

Unauthorized User

A user who has not registered with the service. These users can view publicly available deck lists but can't create a deck list or review someone else's deck list.

User Interface

See *Front-End*

Version control

Allows tracking history of changes to a repository and recovery of an older version (GitHub, n.d.).

Virtualization

An abstraction layer that allows the components of a computer to be divided across several “virtual” or non-physical computers (IBM, 2023).

Web Application

See *Web-based*

Web-based

Refers to an application that runs within a web browser. Web-based applications are easier to update and are inherently cross platform but can’t take advantage of capabilities specific to a hardware platform or operating system.

Web Browser

A computer program used for accessing sites or information on a network (Merriam-Webster, 2019). Examples include Google Chrome, Mozilla Firefox, Microsoft Edge, Brave, and Opera.

Team Details

The current revision of the specifications document was prepared by the team. The team takes on the responsibility of revising the document and preparing all necessary material for the project at hand. The team utilized Discord and other public meeting areas to successfully meet online and in person. Contributions to the document at hand are outlined below.

Christian Messmer created the diagram for the Dynamic Statechart as well as its associated descriptions. Christian utilized his knowledge and experience with Magic: The Gathering to make several additions and improvements to other sections of the document, including the Use Cases, Class Diagram, and System Description.

Paul Shriner acted as the workflow leader of this document. Paul set up the layout and formatting of the document, including elements such as the Table of Contents and Workflow Authentication pages. Paul worked on the System Description, the Dataflow Diagrams, and did the initial notes of what would become the Class Diagram. Finally, Paul set up the Glossary and References.


Luke Vukovich worked on the Use Case Diagrams and Class Diagram. Luke designed the diagrams using UML software, then wrote the descriptions in order to ensure the reader understands the meaning behind these diagrams, and their place in this document. Luke also worked on the Abstract and added to the Components section.

Adir Turgeman added details throughout the document to ensure consistency and completeness. He kept track of several terms that could be confusing to users unfamiliar with the subject matter of the document, which were later added to the Glossary. Finally, he reviewed the

document several times and acted as another approval layer to catch potential mistakes and oversights.

Workflow Authentication

I, Christian Messmer, attest that I executed the functions listed within the team details section of the document. Also, I agree with all the information stated within the specifications document.

Christian Messmer		November 14, 2023
Printed Name	Signature	Date

I, Paul Shriner, attest that I executed the functions listed within the team details section of the document. Also, I agree with all the information stated within the specifications document.

Paul Shriner		November 14, 2023
Printed Name	Signature	Date

I, Adir Turgeman, attest that I executed the functions listed within the team details section of the document. Also, I agree with all the information stated within the specifications document.

Adir Turgeman		November 14, 2023
Printed Name	Signature	Date

I, Luke Vukovich, attest that I executed the functions listed within the team details section of the document. Also, I agree with all the information stated within the specifications document.

Luke Vukovich		November 14, 2023
Printed Name	Signature	Date

Report from the Writing Center

Students and Writing Consultant reviewed their written project and checked for issues of grammar, layout, readability, and how easy it was for the Writing Consultant to understand their program without being familiar with their topic or programming. Writing Consultant brought up only a small amount of potential layout and readability suggestions with the students, which they discussed more in depth to explain and dissect why they made their original choices. After discussing, Writing Consultant made sure they had no other questions or topics to discuss before their appointment ended.

References

About Postman. (n.d.). Postman. <https://www.postman.com/company/about-postman/>

Application programming interface. (2023, November 11). Merriam-Webster.

<https://www.merriam-webster.com/dictionary/application%20programming%20interface>

C# | Modern, open-source programming language for .NET. (n.d.). Microsoft.

<https://dotnet.microsoft.com/en-us/languages/csharp>

Cobb, M. (2020, February). *What is OAuth and How Does it Work?* TechTarget.

<https://www.techtarget.com/searcharchitecture/definition/OAuth>

Definition of BROWSER. (2019). Merriam-Webster. <https://www.merriam->

[webster.com/dictionary/browser](https://www.merriam-webster.com/dictionary/browser)

Definition of browser engine. (n.d.). PCMAG. Retrieved November 13, 2023, from

<https://www.pcmag.com/encyclopedia/term/browser-engine>

Discord. (2022). *Discord — A New Way to Chat with Friends & Communities.* Discord.

<https://discord.com/>

Docker. (2020, April 9). *Docker overview.* Docker Documentation. [https://docs.docker.com/get-](https://docs.docker.com/get-started/overview/)

[started/overview/](https://docs.docker.com/get-started/overview/)

GeeksforGeeks. (2023, January 3). *Computer Aided Software Engineering (CASE).*

GeeksforGeeks. [https://www.geeksforgeeks.org/computer-aided-software-engineering-](https://www.geeksforgeeks.org/computer-aided-software-engineering-case/)
[case/](https://www.geeksforgeeks.org/computer-aided-software-engineering-case/)

Get started with GitHub documentation. GitHub Docs. (n.d.). <https://docs.github.com/en/get-started/>

IBM. (2023). *What is Virtualization?* IBM. <https://www.ibm.com/topics/virtualization>

Lemonaki, D. (2022, March 18). *Frontend VS Backend – What’s the Difference?*

FreeCodeCamp.org. <https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference/>

Magic: The Gathering. (n.d.). MTG Wiki; Fandom.

https://mtg.fandom.com/wiki/Magic:_The_Gathering/

Microsoft. (2019). *Visual Studio.* Visual Studio; Microsoft. <https://visualstudio.microsoft.com/>

Miller, S. (2021, September 13). *What is React?* Codecademy News.

<https://www.codecademy.com/resources/blog/what-is-react/>

MongoDB. (2019). *What Is MongoDB?* MongoDB. <https://www.mongodb.com/what-is-mongodb>

Mozilla. (2019, March 23). *HTTP request methods.* MDN Web Docs.

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

REST API Documentation. (n.d.). Scryfall Magic the Gathering Search.

<https://scryfall.com/docs/api>

What are Microservices? (n.d.). IBM. <https://www.ibm.com/topics/microservices>

What Does a Hosting Environment Mean in a Web Server? (2022, July 11). The European

Financial Review. <https://www.europeanfinancialreview.com/what-hosting-environment-mean/>

What is Git? (n.d.). Git-Scm. <https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>