Magic the Gathering Deck Builder

Software Requirements Documentation

Player 4

April 30, 2021

Authors:

Joshua Millikan

Alex Eckert

Dominic Turmenne

Franque Gonzalez

We have abided by the UNCG Academic Integrity Policy on this Assignment

Contents

Introduction

1.1 Purpose

This application is intended to assist the user with managing decks of cards for the Magic the Gathering card game.

1.2 Document Conventions

This document was created as a group effort by all members of our group, and is formatted using LaTeX. all text is in Computer Modern 12pt font.

1.3 Intended Audience

This document is intended for our development team to provide guidance in developing this application, and our professor, to grade as part of this project.

1.4 Definitions

Glossary

HTTPS an extension of Hyper Text Transfer Protocol(HTTP) that is encrypted using transport layer security. 10

Magic the Gathering A card game first released by Wizards of the Coast LLC in 1993. 3, 5, 6, 10

REST Respresentation state transfer, a HTTP based web service protocol. 10

SQLite A embedded SQL database engine that does not run as a seprate server process and writes directly to an ordinary file. 7

user the person using this software for it's intended purpose. 1, 3, 5, 6

Acronyms

API application programming interface. 5, 6, 7, 8, 9, 10, 12, 13

JVM Java virtual machine. 9

MTG Magic the Gathering. 6, 12, 13

TLS transport layer security. 10

1.5 Project Scope

This application will run on locally on the user's computer and will allow the user to create lists of cards representing a deck using up to date information accessed from the internet. The user will be able to save the deck to a local database as well as load and modify previously saved decks. This application will provide relevant information to the user and enforce game rules for what can be put in a deck, including rules such as no more than 4 cards with the same name in a deck, and not allowing cards that are officially banned in the chosen format. The user will have a variety of options including the ability to select what format of Magic the Gathering to use the rules for, as well as searching and filtering cards based on attributes such as name, mana color, release set, power, toughness, and card type (creature, land, etc..).

1.6 Technical Challenges

For most of the developers, this is their first time working on a large programming project with multiple people, which may make organization and coordination a challenge.

1.7 References

API used: https://scryfall.com/docs/api

Overall Description

2.1 Product Features

The proposed project is a deck building application for the collectible card game “Magic the Gathering”. MTG is an exciting game where players build a variety of decks from thousands of card choices. While most players collect and play the game using physical cards, the sheer amount of distinct cards in existence precludes them from being able view every card, and a better system is needed for identifying cards relevant to their current deck. Our project seeks to eliminate this problem by creating a program that allows users to search for cards using an existing API database. User will be able to search using a variety of parameters and restrictions. As they find cards of interest, they will then be able to add them to one or multiple decks. Options to remove cards from existing decks will also be included. Users will be able to save their work, and return at any time.

2.2 User Characteristics

Our target audience is MTG players of all ages. Although this encompasses a wide range of ages, users will skew in age towards teenagers, and young adults. Despite having no specific training, users are assumed to have a general knowledge of computer usage. They are likely to use our product at home for recreational purposes.

2.3 Operating Environment

The proposed project will be deployed as a desktop application; user input will be received via a combination of key, and mouse strokes. Should be able to run on any system with Java version 11 or greater installed. System requirements should be kept low to improve accessibility; the final product should require no more than RAM of 2GB and 100 MB of disk space.

2.4 Design and Implementation Constraints

Implementation will be completed using NetBeans/Intellij to build a suitable JFrame architecture. Ease of use is paramount, as many users are not technologically advanced, and will use the program recreationally. All card data needs to be pulled from an existing external database using API. All Deck data will be stored using a localized database. Excluding a card name, No other data on cards will be stored locally. Users need to be able browse and delete from their deck, as well as search and add new cards. Both of these functions will require comparing the card name in the local database, against the matching card name in the API database. There also should be a feature that checks the validity of a card for a chosen rule set. Adapters should be implemented to support future deployments.

2.5 Assumptions and Dependencies

The application is highly dependent on an external API. It is assumed this database will continue to be available and accessible, without major changes in structure. Searching, and adding cards is entirely dependent on the external database, and will not work should that connection be broken. Displaying decks will likewise be unable to display the full details; however, in the interests of usability, users should still be able to retrieve cards in their existing decks using the local database. The deletion of cards from decks is completely independent of the API. The application's database is planned to be implemented using SQLite, however it should not be strongly dependent on any particular database and should be designed to make it easy to change what kind of database is used if necessary.

Functional Requirements

3.1 Primary

Permanent data storage is required for saving deck information. Users need to be able to add and remove cards, as well as decks. There should also be checks in place to test the tournament validity of each card as it is added to a deck. The application must be able gather information from the user, then search the API database, and return relevant card results based on the given input. The software needs to present both deck results, and search results in an easy to browse format. On one page, a listing of all relevant card names should be viewable, as well as an active card display showing the complete details of that card.

3.2 Secondary

Ideally, the app should include features that allow easy retrieval of card names in the event the API connection is severed. Back end design should be conducive towards future expansion. A built in printer-formatting feature would also improve user experience.

Technical Requirements

4.1 Operating Systems/Compatibility

The Magic of The Gathering Deck Builder application will run on the JVM and as such should be compatible with any system with Java installed. This application will be built using JDK version 11 and as such with require a Java runtime version of 11 or greater to run.

4.2 Interface Requirements

4.2.1 User Interface

The application includes many user interface tools. Those include a look up new cards feature, to see which cards are available to be added to the deck. This feature will call upon the API to see all the possible cards and look them up based on their name. Also, a part of the user interface includes an add card button which will allow the user to add any card that they look up to the deck of their choice. A remove card feature will be implemented to remove any card from whichever deck is chosen. A feature to display the deck will also be available to see all the cards that are present within a deck. There will be a feature to start a new deck, this will allow multiple decks to be made with different names and attributes. Finally, a remove deck feature will be implemented to allow the user to remove a deck that the user no longer wishes to keep.

4.2.2 Hardware Interface

The application will not implement many uses of hardware. The only hardware prevalent will be the inputs of the keyboard and mouse.

4.2.3 Software Interface

The application will be utilizing an API in order to retrieve data from the Magic the Gathering game in order to find cards and figure out what they do so that they may be used in the deck building application. The API being used is provided by Scryfall. It contains a REST-like API for ingesting card data programmatically. The API exposes information available on the regular site in easy-to-consume formats. The API uses UTF-8 character encoding for all responses. Also, API requests are only served over HTTPS, using TLS 1.0, 1.1, and 1.2.

4.2.4 Communications Interface

The application will not utilize any communications interface.

Nonfunctional Requirements

5.1 Performance Requirements

This application is planned to lack intensity on the user's system. This application should be capable of running on any computer capable of running Java 11 or newer.

5.2 Safety/Recovery Requirements

This application will periodically save the user's data (created decks, cards, etc.) to a localized database during use. In the event that the application experiences an unexpected crash or shutdown, the user's most recently saved data will be reloaded from the local database upon starting the application. If the user's storage system becomes entirely corrupted or compromised, there will be no cloud-based saved data to pull from, making their data effectively unrecoverable. For this reason, it is important that the program maintains its high integrity.

5.3 Security Requirements

While using this application, the user will not have to log into or create an account to access the application's features of searching cards and creating/-managing decks.

5.4 Policy Requirements

As per our policy requirements, this application will adhere to the policies of the API in regards to the handling of user data and to the validity of cards in the rule-sets as specified by MTG.

5.5 Software Quality Attributes

5.5.1 Availability

The service this application provides is available on demand by user request.

5.5.2 Correctness

This application will aim to accurately display card information with their most recently updated values and their tournament legality given by the implemented API.

5.5.3 Maintainability

This application will not be updated regularly outside of the necessary adjustments if and when the MTG API framework is updated to support/change data fields.

5.5.4 Reusability

All of the application's code elements will not be used as a part of another application with the exception of the API connection code possibly being used at a later date.

5.5.5 Portability

This application is strictly designed for desktop systems that can run java applications. As it is implemented in Java, the application should be able to run on any system with Java 11 or greater installed

5.6 Process Requirements

5.6.1 Development Process Used

During this application's development, a waterfall implementation process was used. The project team was divided into four roles in charge of a specific task:

1. Github administrator: Manages Github pull requests and branch merging.
2. UI/Frontend Developer: Develops the user interface from which the user interacts with our application.
3. Persistent Data Storage: Manages the storage of the user's data in an effective manner.
4. API Connection: Connects the application to the MTG API and parses necessary information.

5.6.2 Time Constraints

The project team was given four months (January - April) to build this application in its entirety.

5.6.3 Cost and Delivery Date

As is standard with project-integrated-courses, the team will build this application during the Spring 2021 UNCG semester without pay and is to be delivered by the end of the semester on April 30th, 2021.