

**Catalogue ya deck**

**Card scanner and cataloguing system**

**Eoin Gallagher**

**BEng(H) in Software & Electronic Engineering**

**Project Engineering**

Atlantic Technological University

2023/2024

# Declaration:

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used, or incorporated during this project, this is acknowledged and referenced.

# Acknowledgements:

I would like to thank my supervisor Brian for his help throughout the year as well as the other lecturers, they all gave me some advice over the course of the year and helped me reach where I am now.

# Summary:

The project is a database and card scanning website for the trading card game Magic the Gathering. It uses many of the skills and knowledge I’ve gained over the 4 years of study. The project is python code for the card recognition using YOLO which was gotten from their own GitHub and edited for my own purposes, this is along side react code for the front end which was developed in Visual Studio IDE. Some of the features of the app include:

* Card logging.
* Sorting of card data.
* Extended card use and rulings.
* Mongoose SQL database.  Card searching.

The user runs a local database that will hold the images of the cards that they upload to the app, the app then uses those images to gather the necessary data such as name of card, type etc.

Contents

[Declaration: 2](#_Toc176172632)

[Acknowledgements: 3](#_Toc176172633)

[Summary: 4](#_Toc176172634)

[1. INTRODUCTION: 6](#_Toc176172635)

[1.1. Project Goals 6](#_Toc176172636)

[1.2. Scope 6](#_Toc176172637)

[1.3. Report Overview: 6](#_Toc176172638)

[2. BLOCK DIAGRAM: 7](#_Toc176172639)

[3. IMAGE RECOGINITION: 8](#_Toc176172640)

[3.1. Usage 8](#_Toc176172641)

[3.2. YOLO: 8](#_Toc176172642)

[3.3. Python: 8](#_Toc176172643)

[3.4. Dataset: 8](#_Toc176172644)

[4. MONGODB DATABASE: 9](#_Toc176172645)

[4.1. MongoDB: 9](#_Toc176172646)

[4.2. Schema: 9](#_Toc176172647)

[5. WEBSITE: 11](#_Toc176172648)

[5.1. User Interface and Experience: 11](#_Toc176172649)

[6. PROBLEM SOLVING: 12](#_Toc176172650)

[6.1. YOLO: 12](#_Toc176172651)

[6.2. ScryFall API: 12](#_Toc176172652)

[6.3. Time Management: 12](#_Toc176172653)

[7. CONCLUSION: 13](#_Toc176172654)

[8. FUTURE WORK: 14](#_Toc176172655)

[9. REFERENCES: 15](#_Toc176172656)

# INTRODUCTION:

# Project Goals

The primary goals of the project were to create a comprehensive tool for cataloguing and managing Magic the Gathering cards, using image recognition technology to automate the process as much as possible. The specific goals included:

* Developing a card scanning system that utilizes YOLO (You Only Look Once) for object detection, allowing users to automatically recognize and log cards into a database.
* Storing the recognized card data into a MongoDB database, enabling efficient retrieval and management of large collections.
* Creating a responsive website with React that allows users to view, edit, and manage their card collections seamlessly across different devices.
* Ensuring the system is easy to use, with minimal manual data entry required from users, focusing on automation and user-friendly interfaces.

# Scope

The project is designed to serve as a tool for Magic the Gathering players and collectors who want a streamlined method of cataloguing their card collections. The application was developed with scalability in mind, so it can potentially be expanded to support other trading card games in the future.

# Report Overview:

Where ever there is code shown or explained it’ll be shown using screenshots and snippets like the example below.



# BLOCK DIAGRAM:

The block diagram below illustrates the architecture of the card scanning and cataloguing system. It highlights the interaction between the different components, including the YOLO-based image recognition, MongoDB database, and the React-based front end.

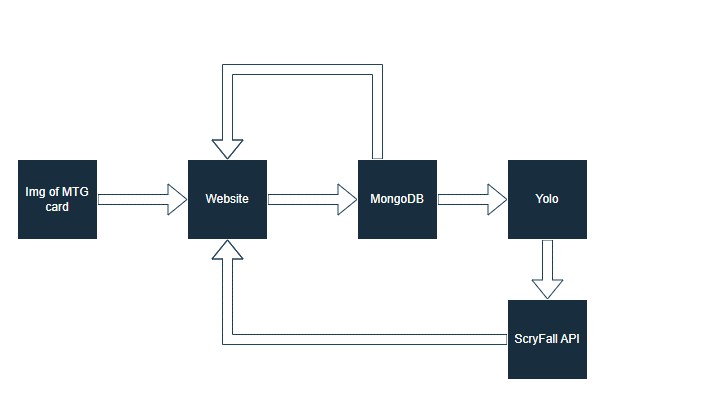


Figure 1: System Architecture Block Diagram

# IMAGE RECOGINITION:

# Usage

The image recognition system plays a central role in the project by automating the data entry process. The decision to use YOLO for this task was based on its high accuracy and speed, which are critical for providing a smooth user experience. By allowing users to upload images of their cards, the system extracts relevant information such as the card name, type, and colour, significantly reducing the time and effort required to catalogue a large collection.

# YOLO:

YOLO’s ability to perform object detection in a single pass over an image makes it an ideal choice for real-time applications like this one. The specific version used in this project is YOLOv5, which offers several improvements over its predecessors, including better accuracy and faster processing times. Training the model involved creating a custom dataset that includes a wide range of card images to ensure the system could accurately recognize different cards, even those with varying degrees of wear and tear or different lighting conditions.

# Python:

Python was chosen for the image recognition component due to its extensive library support and ease of use in machine learning applications. Libraries such as OpenCV and TensorFlow were instrumental in processing images and building the YOLO model. The modularity of Python also facilitated the integration of the YOLO model with other parts of the system, such as the database and front end.

# Dataset:

The dataset was a crucial part of the project. Initially, the lack of a comprehensive pre-existing dataset for Magic the Gathering cards posed a challenge. As a result, a custom dataset was created using RoboFlow, where images of personal card collections were annotated and prepared for training. The dataset included various card conditions and lighting environments to improve the robustness of the model. The annotated dataset was then fed into YOLO, which required several iterations of training and validation to achieve the desired accuracy.A screen shot of a computer program

Description automatically generated below.

# MONGODB DATABASE:

# MongoDB:

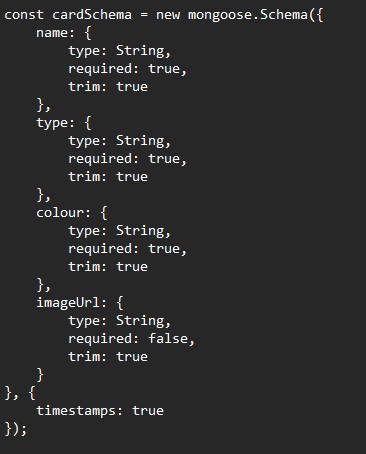
MongoDB was chosen for its flexibility and scalability, which are essential for handling the potentially vast amounts of data involved in managing large card collections. Unlike traditional relational databases, MongoDB’s document-based structure allowed for more efficient storage and retrieval of the diverse data types associated with each card, such as text, images, and metadata.

# Schema:

The database schema was designed to be both comprehensive and flexible. Each card entry includes the following fields:

* **Name**: A string field that stores the card name, as recognized by the YOLO model.
* **Type**: A string field that categorizes the card (e.g., Creature, Sorcery, etc.).
* **Colour**: A string field indicating the card’s color (e.g., Red, Blue, etc.).
* **ImageURL**: A string field that stores the URL of the card image, enabling easy display on the website.

To optimize performance, indexes were created on key fields such as Name and Type, allowing for faster query execution and improved user experience when searching and sorting through large collections.



# WEBSITE:

A website we developed to allow users to access the information and to add more cards to the database. Other features of the site are to allow the users to manually delete entries of the database and to view and sort through what is already saved to the database.

A screenshot of a video game

Description automatically generatedBelow is a sample of how the home page looks with some data already on the site.

As well as the main parts of the site I mentioned above, I also made a bottom tab bar for navigation between the add card page and the home page, this was so if later I wanted to port the site to mobile it would easily work on phone as well as PC.

# User Interface and Experience:

The website’s user interface was designed with simplicity and functionality in mind. A minimalistic approach was taken to avoid overwhelming users with too many options or complex navigation paths. The homepage features a clean layout where users can view their card collections, sort them based on different attributes, and access detailed information about each card with a single click. The use of React provided a responsive and dynamic user experience, with real-time updates to the card data as changes are made.

* 1. Mobile Optimization:

Although the initial design focused on desktop users, significant consideration was given to mobile optimization. Responsive design techniques were employed to ensure that the website functions well on devices of various sizes. Future updates could include further enhancements such as touch-friendly controls and improved performance on mobile devices to cater to the growing number of users who prefer to manage their collections on the go.

# PROBLEM SOLVING:

# YOLO:

The integration of YOLO into the system posed several challenges, particularly regarding the model’s accuracy in recognizing cards under varying conditions. Adjustments to the model’s hyperparameters, such as learning rate and batch size, were necessary to achieve better performance. Additionally, the need for a well-annotated dataset was crucial, and several rounds of data collection and training were required to fine-tune the model. Future improvements might include exploring transfer learning with a pre-trained model to reduce the time required to achieve high accuracy.

# ScryFall API:

The initial plan to use the ScryFall API for additional card information was reconsidered due to access restrictions. This decision led to a pivot towards developing a more self-contained system, relying on the data captured directly from the cards. However, integrating a community-maintained API like ScryFall could still be a valuable enhancement in future versions, offering users a richer set of data and features..

# Time Management:

Time management was one of the key challenges faced during this project. Balancing the demands of multiple academic responsibilities alongside the development of this project proved to be difficult. In retrospect, adopting a more rigorous project management approach, perhaps utilizing Agile methodologies, could have allowed for more consistent progress tracking and better handling of unforeseen challenges.

# CONCLUSION:

The project successfully demonstrates the feasibility of using modern machine learning techniques like YOLO for practical applications such as card cataloguing. While some goals were not fully realized, particularly in the area of image recognition, the project nonetheless provides a functional and user-friendly tool for Magic the Gathering enthusiasts. The lessons learned, especially in terms of time management and dealing with technical challenges, will be invaluable in future endeavours. There is ample room for expansion and improvement, particularly in enhancing the accuracy of the OCR system, integrating third-party APIs, and optimizing the website for mobile use. With these enhancements, the project has the potential to evolve into a comprehensive solution for managing trading card collections.

# FUTURE WORK:

In the future, several improvements and extensions could be considered:

* **Expanded Database Features**: Introduce more sophisticated search and filtering options, such as filtering by card rarity or set, to enhance user experience.
* **Mobile Application**: Develop a native mobile application to complement the website, offering users more flexibility in managing their collections.
* **Integration with Other APIs**: Revisit the integration with the ScryFall API or similar services to enrich the data available within the system, such as card pricing and in-game statistics.
* **Community Features**: Implement features that allow users to share their collections or trade cards with others, transforming the system into a community-driven platform.

# REFERENCES:

YOLOv5: <https://github.com/ultralytics/yolov5/tree/master/data>

Scryfall API: https://github.com/ultralytics/yolov5/tree/master/data