AH SDD Project: Game Assist Tool

Contents

[Analysis 6](#_Toc129430456)

[Purpose 6](#_Toc129430457)

[Pre-Development User Survey 7](#_Toc129430458)

[Considerations of Pre-Development User Survey 7](#_Toc129430459)

[Constraints 8](#_Toc129430460)

[UML Use Case Diagram 9](#_Toc129430461)

[Requirements Specification 10](#_Toc129430462)

[End-User Requirements 10](#_Toc129430463)

[Functional Requirements 11](#_Toc129430464)

[Inputs, Processes and Outputs 11](#_Toc129430465)

[Notes on Further Input Validation 12](#_Toc129430466)

[Personas 12](#_Toc129430467)

[Sean Greaves 12](#_Toc129430468)

[Winnie Manning 12](#_Toc129430469)

[James Hill 13](#_Toc129430470)

[User Stories 13](#_Toc129430471)

[User Scenarios 13](#_Toc129430472)

[Project Plan – Initial estimated version 14](#_Toc129430473)

[Project Plan – Final version 17](#_Toc129430474)

[Design 20](#_Toc129430475)

[Pseudocode Design 20](#_Toc129430476)

[Python Files 20](#_Toc129430477)

[Javascript Files 30](#_Toc129430478)

[Query Design 31](#_Toc129430479)

[Get average winrate 31](#_Toc129430480)

[Get max matchID 31](#_Toc129430481)

[Get specific winrate against an opponent 32](#_Toc129430482)

[Insert new match 32](#_Toc129430483)

[Get all records to display on page 32](#_Toc129430484)

[Data Dictionary Design 33](#_Toc129430485)

[Match table 33](#_Toc129430486)

[Arrays of Records and 2D Arrays 34](#_Toc129430487)

[UML Class Design 65](#_Toc129430488)

[mtgScraper 65](#_Toc129430489)

[matchRecord 66](#_Toc129430490)

[mtgtop8 67](#_Toc129430491)

[File Flow 67](#_Toc129430492)

[User-Interface Design 68](#_Toc129430493)

[Main Page 68](#_Toc129430494)

[Profile Page 69](#_Toc129430495)

[Entity Relationship Diagram 69](#_Toc129430496)

[Hierarchical Design of Website Structure 69](#_Toc129430497)

[Implementation 69](#_Toc129430498)

[Python Files 70](#_Toc129430499)

[sync.py 70](#_Toc129430500)

[mtgScraper.py 71](#_Toc129430501)

[matchRecord.py 74](#_Toc129430502)

[dbCMD.py 85](#_Toc129430503)

[Javascript Files 88](#_Toc129430504)

[renderer.js 88](#_Toc129430505)

[preload.js 91](#_Toc129430506)

[index.js 92](#_Toc129430507)

[UI Files 92](#_Toc129430508)

[index.html 92](#_Toc129430509)

[bottom.css 94](#_Toc129430510)

[mid.css 95](#_Toc129430511)

[top.css 97](#_Toc129430512)

[main.css 99](#_Toc129430513)

[UI Visuals 101](#_Toc129430514)

[Main Page 101](#_Toc129430515)

[Main Page with Content 102](#_Toc129430516)

[Profile 103](#_Toc129430517)

[Profile Search Opponent 104](#_Toc129430518)

[Main Page Profile Button Image 105](#_Toc129430519)

[Error Log 107](#_Toc129430520)

[Research and Development of New Skills 110](#_Toc129430521)

[Testing 111](#_Toc129430522)

[Test Table 111](#_Toc129430523)

[Component testing 113](#_Toc129430524)

[sync.py 113](#_Toc129430525)

[mtgScraper.py 113](#_Toc129430526)

[matchRecord.py 122](#_Toc129430527)

[mtgtop8.py 122](#_Toc129430528)

[dbCMD.py 123](#_Toc129430529)

[renderer.js 125](#_Toc129430530)

[index.html 125](#_Toc129430531)

[Integrative testing 125](#_Toc129430532)

[Final Testing (Requirements) 126](#_Toc129430533)

[Testing with Personas 127](#_Toc129430534)

[Evaluation 129](#_Toc129430535)

[Fitness for purpose 129](#_Toc129430536)

[Robustness 129](#_Toc129430537)

[Usability 129](#_Toc129430538)

[Maintainability 129](#_Toc129430539)

[Future Design 129](#_Toc129430540)

[Fix Current Low-Priority Issues: 129](#_Toc129430541)

[Appendix 1 – Pre-Development User Survey Results 130](#_Toc129430542)

[Q1 130](#_Toc129430543)

[Results 130](#_Toc129430544)

[Q2 130](#_Toc129430545)

[Numerical Results 130](#_Toc129430546)

[Mean 130](#_Toc129430547)

[Results 130](#_Toc129430548)

[Q3 131](#_Toc129430549)

[Results 131](#_Toc129430550)

[Q4 131](#_Toc129430551)

[Results 131](#_Toc129430552)

[Q5 132](#_Toc129430553)

[Results 132](#_Toc129430554)

[Q6 132](#_Toc129430555)

[Results 132](#_Toc129430556)

[Q7 133](#_Toc129430557)

[Results 133](#_Toc129430558)

# Analysis

## Purpose

* Currently there is no maintained game assist tool which records match data on the game: Magic: The Gathering Online (MTGO). The game is played by two players, using decks of cards, in a best of three format (called a match). This leaves users to record this match data manually on an excel document, which is very time consuming, or not record the data at all. This is a problem as the game rewards the player for winning matches, so the players are incentivised to increase their win rate (% chance of winning a game). This leads to a highly competitive environment, where players spend lots of money on the game to maximise their win rate. So, if players can get huge volume of match data, which accurately displays their overall winrate and winrate against specific opponent. This means that they can improve their gameplay by developing strategies (or a better strategy), which would hopefully increase their win rate. Which, as I’ve already outlined, they are incentivised to do.
* The end users are people who play Magic: The Gathering Online (MTGO), they are technically adept (as they play an online video game). This means they should be capable of installing and using a game assist tool. Their total age range is between 16-40, with most users being 17-24. The end users also spend lots of money on the game, as they typically want to maximise their win rate. Most end-users live in Europe and North America. They also spend a reasonable amount of time playing video games (10 to 20 hours).
* The app will be built on Electron so that an interface can be easily created using HTML, CSS and JavaScript. The app will display all currently collated data (a spreadsheet of database data of match results), which can be manipulated by the end-user, and manipulatable graphs of the data. The end-user can download the collated data and graphs. On the app, the user can press a button which runs the main processes of the data collection algorithm. This main process navigates to MTGO’s file system, where it gets the users’ match logs (stored in .dat files). The app then reads all information within the logs, which it categorises and stores in a local sqlite3 database. It may require the end-user’s input to clarify data which the match logs leave out, this is conditional on the basis that one of the players don’t concede on the final game. The app also uses the Selenium web driver to get additional information (deck names), to store in the database.
* The Advanced Higher concepts that the app meets are:
* The project will using object-orientated programming (OOP), as well as imperative programming with 2D arrays and arrays of records.
* A SQA Standard Algorithm (bubble sort algorithm).
* Other Advanced Higher Concepts:
  + Connecting to a database via a programming language.
  + SQL query design.
  + Execution of SQL operations.
  + Integrative testing (Software Design and Development and Database Design and Development).
* The app is integrated with:
  + A database (local, sqlite3) with multiple tables, whose connection will be opened (and closed) to execute SQL queries.
* New skills required for the app (which aren’t in the project specification or course specification):
* Running files based on user input, with arguments.
* Implementing regular expression (Regex) in the app.
* Sanitise user inputs which will be used as part of SQL Queries.
* Web Scraping (using the Selenium library).
* Creating and using python modules (files that store a class) to increase modularity.

## Pre-Development User Survey

* This survey was carried out to gauge potential end-user interest in the app, whether design decisions should be followed and whether assumptions about potential end-users were accurate.
* The user survey confirmed that there was an interest in an MTGO game assist tool among potential end-users (this proves the app is viable), that certain design decisions should be followed (open-source, free, etc.), and the assumptions about potential end-users were correct (age demographic, time spent playing games, etc.).
* See Appendix 1 for information on specific results and screenshots of the results.
* Details can also be found in the functional requirements under the heading: Results from Pre-Development User Survey.

### Considerations of Pre-Development User Survey

* The survey didn’t ask the potential end-user how much time they spent playing MTGO. This means that the surveyed group may not contain a wide range of MTGO players.
* The survey was untranslated to other languages, this means that only English speakers could give feedback, so huge numbers of potential end-user’s feedback may not have been collected.

## Constraints

* Technical Constraints
  + The app will run on Windows 7 or later.
  + The app itself does not take up lots of space, however, the databases have no limit on how large it can be (this is down to the user’s discretion).
  + Python’s performance is relatively slow, however, it is very sufficient for this app.
  + All the necessary resources currently exist for the project to be completed.
  + I must learn significantly more Java Script during the development of the app.
* Business Constraints
  + The scheduling and timescales must be met, otherwise the app will be incomplete.
  + There is no budget for the app.
  + There is a single person working on the app.
  + There are no running or single-time costs associated with the project as it uses free software.
  + The app can be further developed which could allow it to follow the freemium model, so the end-user could pay for more features. This would work with a website and a server, which user data will be sent to (collating all user data), so overall evaluations of cards and decks can be done. The users will pay for access levels of this collated data.
  + The app will take 80 hours to complete.
  + The app will generate no revenue and will be open source. This has the intangible benefit of increased trust in the project since anyone can check what the program does with your data.
* Legal Constraints
  + There are no legal issues with development of the app: all web scraping is within current precedent (site doesn’t have a robots.txt, so web scraping isn’t disallowed, and the web driver doesn’t login/create an account) and no personal data is stored (so GDPR isn’t in violation).
  + This project complies with the Copyright, Designs and Patents Act.
* Time Constraints
  + This project must be completed in no more time than 5 months.

## UML Use Case Diagram

A picture containing diagram

Description automatically generated

## Requirements Specification

### End-User Requirements

#### Results from Pre-Development User Survey

This information has been used to: confirm some assumptions about the end users and see if there is interest in development of the app.

See appendix 1 for more information and the results data.

* The survey found that most potential end-users are under 18 to 24, this means that the app should be designed for that age group so results should be defined clearly, try to keep a clean look.
* The survey found that most potential end-users play videogames a fair amount, this means they are likely technology literate, so design doesn’t have to be very simplified like it would have to be for older age groups.
* This survey found that most potential end-users want the app to display information stored by the app.
* The survey found that cost is a barrier of entry for most potential end-users, this confirms that the app should be free, to enable as many users to use it as possible. This is also useful for when the freemium model is applied, there are lots of end-users who may pay for the additional features.
* The survey found that potential end-users would be interest in installing the finished app, this means that there is enough interest in order to justify creation of the app.

#### Assumed Requirements

The end-user will expect the app to:

* Capture their match logs.
* Display decklists in a readable format.
* Be compatible with their device (PC).
* Be integrated with a local database.
* Try to specify the winner of a match if the match logs did not which player won the final game. This happens because the match logs may contain incomplete information on any number of games. This is a problem if a match goes to three games and the final game is undetermined, since even if no player win is in the logs, whichever player went second in the next game lost the previous game (this workaround isn’t possible if the final game is undetermined).
* Display the average winrate of the player.
* Display the winrate of the player against an opponent.

### Functional Requirements

* + - The app must be able to read .dat file data and insert that data into a local database.
    - The app must be integrated with a local database.
    - The app must be able to display the average deck for each player in each match.
    - On the app, all users will have the same interface.
    - The app will sanitise database inputs to protect against basic against SQL injection.
    - The app must be able to assign deck names and rough deck lists to each deck played, based on limited match data.
    - The app will close the database connection after queries are run to prevent any possible security breach, that could target open database connection vulnerabilities.
    - The app must show the average winrate of the player.
    - The app must show the winrate of the player against an opponent.

### Inputs, Processes and Outputs

#### Main App Page

|  |  |  |
| --- | --- | --- |
| Inputs | Processes | Outputs |
| Sync Button Click (Click input) | Runs File Reader | User: Possible error message |
| Profile Button (Click input) | Hides database content, displays profile page and gets average winrate from the database. | Average winrate. |

#### On App Load

|  |  |  |
| --- | --- | --- |
| Inputs | Processes | Outputs |
|  | Gets results of matches from the database. | Results of matches from the database. |

#### File Reader

|  |  |  |
| --- | --- | --- |
| Inputs | Processes | Outputs |
|  | Iterates over all files where match files are stored.  For each file it outputs deck lists and date to Web Scraper, to get possible deck names (from which it chooses the most probable, and if a deck name isn’t significantly probable (less than 60%), it stores all). It also stores that data in the integrated local app database. |  |

#### Web Scraper

|  |  |  |
| --- | --- | --- |
| Inputs | Processes | Outputs |
| File Data | Web scrapes data from mtgtop8.com to obtain probable deck name data using data inputted | Deck names (dictionary formatted as {“deck name”: percentage of appearance} where ‘percentage of appearance’ is a float) |

### Notes on Further Input Validation

* To validate match files, when reading the file, that file is ignored if it doesn’t meet expected formatting (all match data is stored the same way).
* Data is sanitised by parameterized queries before it’s put into the database to prevent unwanted errors.

## Personas

I have developed 3 Personas, as this allows each user to have different needs, and as must be met by the end-user requirements, each user-type can have their needs met.

### Sean Greaves

* A 23-year-old man who lives in America, on their own in a house. They’ve being playing MTGO for 4 years and are very committed to the game. They have a middling income job at Specsavers and are highly competitive as they are a high-performance swimming athlete in their spare time.

### Winnie Manning

* An 18-year-old woman who lives in the UK, in a flat (with 2 flatmates). They are in their first year of university. They’ve been playing MTGO for 2 years and are fairly committed to the game. They have a part-time job as a barista at a local indie coffee shop. They read the newspaper and always make sure to complete the sudoku, which they enjoy finishing. They also have game night with their flatmates every Saturday night and are quite competitive when it is Monopoly that night.

### James Hill

* A 16-year-old man who lives in Ireland, with their parents and a sibling. They are still in school and have only just picked up MTGO in the last 4 months. They aren’t committed to the game and aren’t very similar with it. Their only income source is pocket, birthday, and Christmas money. They play other competitive video games with their friends during weekends and after school.

## User Stories

* I, Sean Greaves, don’t have much time to analyse my game data (because I have a full-time job and am an athlete), so I want assistance in analysing my game data.
* I, Winnie Manning, struggle to understand my game data, so I want it to be displayed in an understandable format.
* I, James Hill don’t, know which cards most contribute to my success, so I want to know which cards most contribute to my success.

## User Scenarios

* Sean is busy most of the time, whether they’re swimming or working at their job. This means that they don’t have much time to play MTGO and what little time they do play, they can’t put much thought into strategy. Sean is very competitive in everything they do, so winning is tied to their enjoyment they get out of the game. This means that they want to win more and one of the best ways to do that is get better at playing the game. So, to do this, Sean wants a tool which can help him get better at playing the game, by seeing which deck he most wins with.
* Winnie Manning has trouble scanning their self-collated match data to analyse their gameplay, they also don’t know exactly what they’re looking for when trying to analyse their self-collated match data and they struggle to maintain a consistent format across their spreadsheet. They want a tool to solve these problems for them by displaying their match data in a consistent format, which would help her to understand her match data.
* James Hill doesn’t have lots of money, so they want to know exactly which cards most contribute to their success in the game, so they don’t spend time and money playing with cards which are ‘bad’ or have a lower win rate than other cards. Therefore, they want a tool to solve this problem for them as they don’t want to learn how to and collect data to analyse all by themselves. They want this the tool to do this by presenting all games, so he can look through the matches, so see which cards most contributing to his success.

## Project Plan – Initial estimated version

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | | Start Date | End date | Workflow | Resources Required |
| Analysis (27/08/22 – 19/10/22) | | | | | |
| 1. | Description of problem | 27/08/22 | 29/08/22 | Sequential |  |
| 2. | Pre-Development user survey | 31/08/22 | 05/09/22 | Sequential | Using Survey Monkey (to create a survey) and Snipping Tool (to screenshot results) |
| 3. | Constraints | 20/09/22 | 21/09/22 | Sequential |  |
| 4. | UML Use Case diagram | 22/09/22 | 27/09/22 | Sequential | Using Lucidchart |
| 5. | Create persona, user stories and scenarios | 01/10/22 | 05/10/22 | Sequential |  |
| 6. | Requirements specification: end-user req. | 07/10/22 | 08/10/22 | Sequential |  |
| 7. | Requirements specification: functional req. | 09/10/22 | 17/10/22 | Sequential |  |
| 8. | Project plan | 18/10/22 | 19/10/22 | Sequential | Using Gantt Project |
| Design (20/10/22 – 21/10/22) | | | | | |
| 9. | Pseudocode design | 20/10/22 | 22/10/22 | Sequential |  |
| 10. | UML class diagram | 24/10/22 | 27/10/22 | Sequential |  |
| 11. | Query Design | 27/10/22 | 31/10/22 | Parallel |  |
| 12. | User-Interface Design | 27/10/22 | 31/10/22 | Parallel |  |
| 13. | Arrays of Records and 2D Arrays | 27/10/22 | 31/10/22 | Parallel |  |
| 14. | Data Dictionary Design | 27/10/22 | 31/10/22 | Parallel |  |
| 15. | Hierarchical Design of App Structure | 27/10/22 | 31/10/22 | Parallel |  |
| 16. | Entity Relationship Diagram | 27/10/22 | 31/10/22 | Parallel |  |
| 17. | File Flow | 27/10/22 | 31/10/22 | Parallel |  |
| Implementation (01/11/22 – 10/12/22) | | | | | |
| 18. | Implementation (including input validation) | 01/11/22 | 10/02/22 | Parallel | Visual Studio Code, Github |
| 19. | Research and development of new skills | 01/11/22 | 10/02/22 | Parallel |  |
| 20. | Log of ongoing testing | 01/11/22 | 10/02/22 | Parallel |  |
| End Testing (12/12/22 – 23/12/22) | | | | | |
| 21. | Test Table | 10/02/22 | 16/12/22 | Sequential |  |
| 22. | Component Testing | 10/02/22 | 18/12/22 | Sequential |  |
| 23. | Integrative Testing | 10/02/22 | 20/12/22 | Sequential |  |
| 24. | Final Testing |  |  |  |  |
| 25. | Testing with personas | 10/02/22 | 24/12/22 | Sequential |  |
| Evaluation (05/01/23 – 07/01/23) | | | | | |
| 26. | Evaluation Report | 05/01/23 | 07/01/23 | Sequential |  |

## Project Plan – Final version

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | | Start Date | End date | Workflow | Resources Required |
| Analysis (27/08/22 – 19/10/22) | | | | | |
| 1. | Description of problem | 27/08/22 | 29/08/22 | Sequential |  |
| 2. | Pre-Development user survey | 31/08/22 | 05/09/22 | Sequential | Using Survey Monkey (to create a survey) and Snipping Tool (to screenshot results) |
| 3. | Constraints | 20/09/22 | 21/09/22 | Sequential |  |
| 4. | UML Use Case diagram | 22/09/22 | 27/09/22 | Sequential | Using Lucidchart |
| 5. | Create persona, user stories and scenarios | 01/10/22 | 05/10/22 | Sequential |  |
| 6. | Requirements specification: end-user req. | 07/10/22 | 08/10/22 | Sequential |  |
| 7. | Requirements specification: functional req. | 09/10/22 | 17/10/22 | Sequential |  |
| 8. | Project plan | 18/10/22 | 19/10/22 | Sequential | Using Gantt Project |
| Design (20/10/22 – 21/10/22) | | | | | |
| 9. | Pseudocode design | 20/10/22 | 22/10/22 | Sequential |  |
| 10. | UML class diagram | 24/10/22 | 27/10/22 | Sequential |  |
| 11. | Query Design | 27/10/22 | 29/10/22 | Parallel |  |
| 12. | User-Interface Design | 27/10/22 | 30/10/22 | Parallel |  |
| 13. | Arrays of Records and 2D Arrays | 29/10/22 | 31/10/22 | Parallel |  |
| 14. | Data Dictionary Design | 28/10/22 | 30/10/22 | Parallel |  |
| 15. | Hierarchical Design of App Structure | 28/10/22 | 30/10/22 | Parallel |  |
| 16. | Entity Relationship Diagram | 29/10/22 | 31/10/22 | Parallel |  |
| 17. | File Flow | 30/10/22 | 31/10/22 | Parallel |  |
| Implementation (01/11/22 – 10/12/22) | | | | | |
| 18. | Implementation (including input validation) | 01/11/22 | 10/02/22 | Parallel | Visual Studio Code, Github |
| 19. | Research and development of new skills | 01/11/22 | 10/02/22 | Parallel |  |
| 20. | Log of ongoing testing | 01/11/22 | 10/02/22 | Parallel |  |
| End Testing (12/12/22 – 23/12/22) | | | | | |
| 21. | Test Table | 10/02/22 | 16/12/22 | Sequential |  |
| 22. | Component Testing | 10/02/22 | 18/12/22 | Sequential |  |
| 23. | Integrative Testing | 10/02/22 | 20/12/22 | Sequential |  |
| 24. | Final Testing |  |  |  |  |
| 25. | Testing with personas | 10/02/22 | 24/12/22 | Sequential |  |
| Evaluation (05/01/23 – 07/01/23) | | | | | |
| 26. | Evaluation Report | 05/01/23 | 07/01/23 | Sequential |  |

# Design

## Pseudocode Design

### Python Files

#### sync.py

IMPORT mtgScraper as MODULE named Scraper

localUser = GET\_LOCAL\_USER()

path = 'C:\Users\' + localUser + '\*AppData\Local\Apps\2.0\Data\'*

for i = 0 TO 2:

    folder = GET\_DIRECTORIES(path)

    path = path + '\' + folder

mainFolders = GET\_DIRECTORIES(path)

scp = Scraper()

for folder in mainFolders:

    folderPath = path + '\' + folder + '\*' + 'Data\AppFiles'*

    for (folderPath, directory, fileNames) in WALK(folderPath):

        if length(directory) > 1:

            directory.REMOVE('.vs')

        folderPath = folderPath + '\' + directory

        exit loop

    scp.ADD\_PATH(folderPath)

scp.RUN()

#### mtgScraper.py

IMPORT mtgtop8 as MODULE named DriverController

IMPORT matchRecord as MODULE named MatchRecord

CLASS Scraper:

    CONSTRUCTOR(self):

        instansiate\_object

        self.paths = []

    METHODS(self):

        method ADD\_PATH(self, path):

            self.paths.append(path)

        method RUN(self):

            self.OPEN\_CONN()

            self.CHECK\_DB()

            top8Conn = self.CHECK\_INTERNET()

            match = MatchRecord

            dc = DriverController

            for path in self.paths:

                fileList = GET\_DIRECTORIES(path)

                self.cursor.EXECUTE("SELECT filename FROM matches;")

                filenames = self.cursor.FETCHALL()

                for filename in fileList:

                    if filename in filenames:

                        exit loop

                    decklists, extra, matchlog, players = match.getDecklists(path+'/'+filename)

                    if decklists is not None:

                        dateTime = filename.GET\_DATE

                        if top8Conn = True:

                            dictNames = dc.RETURN\_DICT\_NAMES(decklists, date)

                        else:

                            dictNames = "NA"

                        self.SQLITE\_DRIVER\_DATA(filename, date, dictNames, extra, decklists, players, matchlog)

            dc.QUIT\_DC()

            self.CLOSE\_CONN()

        method OPEN\_CONN(self):

            self.conn = CONNECT\_TO\_DATABASE

            self.cursor = CURSOR

        method CHECK\_INTERNET(self):

            url = 'https://mtgtop8.com/'

            try:

                URLLIB.REQUEST.URLOPEN(url)

                connection\_to\_internet = True

            except:

                connection\_to\_internet = False

            return connection\_to\_internet

        method CHECK\_DB(self):

            if self.conn = FAILED:

                self.cursor.EXECUTE("""CREATE TABLE matches(

                                    matchID INTEGER PRIMARY KEY,

                                    filename TEXT,

                                    players BLOB NOT NULL,

                                    decknames BLOB,

                                    decklistP1 BLOB NOT NULL,

                                    decklistP2 BLOB NOT NULL,

                                    firstTurns BLOB NOT NULL,

                                    winLoss BLOB NOT NULL,

                                    format TEXT,

                                    type TEXT,

                                    date TEXT NOT NULL);""")

                self.conn.CONNECT()

                self.cursor.EXECUTE("""CREATE TABLE games(

                                    gamesID INTEGER NOT NULL PRIMARY KEY,

                                    gameNum INTEGER NOT NULL,

                                    startingHands BLOB NOT NULL,

                                    gameLog BLOB NOT NULL,

                                    winner BLOB,

                                    matchID INTEGER REFERENCES matches(matchID) ON UPDATE CASCADE);""")

                self.conn.COMMIT()

        method SQLITE\_DRIVER\_DATA(self, filename, dateTime, dictNames, extra, decklists, players, matchlog):

            data = (filename, players, dictNames, decklists[0], decklists[1], extra['play'], extra['winner'], 'NA', 'Constructed', dateTime)

            self.cursor.EXECUTE("INSERT INTO matches(filename, players, decknames, decklistP1, decklistP2, firstTurns, winLoss, format, type, date) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?);", data)

            self.conn.COMMIT

            self.cursor.EXECUTE("SELECT MAX(matchID) FROM matches;")

            matchID = self.cursor.FETCH\_ONE

        method CLOSE\_CONN(self):

            self.conn.COMMIT

            self.conn.CLOSE

            self.cursor.CLOSE

#### matchRecord.py

CLASS MatchRecord:

    CONSTRUCTOR(self):

        instansiate\_object

    METHODS(self):

        method GET\_DECKLISTS(self, filename):

            decklists = []

            extra = {'play':[], 'startingHands':[], 'winner':[]}

            self.matchLog = filename.DECODE()

            attempt:

                self.GET\_PLAYERS()

            if error:

                return None

            self.FORMAT\_LINES()

            if length(self.matchLog) < 2:

                return None

            for game in self.matchLog:

                if game doesn't exist:

                    exit loop

                gameDecklists = self.GET\_DECKLISTS(game)

                if gameDecklists doesn't exist:

                    exit loop

                decklists.append(gameDecklists)

                attmept:

                    extra['play'].append(self.GET\_ON\_PLAY(game))

                if error:

                    extra['play'].append('unknown')

                attempt:

                    extra['startingHands'].append(self.GET\_STARTING\_HANDS(game))

                if error:

                    extra['startingHands'].append('unknown')

                extra['winner'].append(self.getWinner(game))

        return decklists, extra, self.matchLog, self.players

    method GET\_PLAYERS(self):

        players = self.matchLog.FIND\_ALL(('@P(\S+) rolled')

        self.players = list(players)

    method FORMAT\_LINES(self):

        filteredMatch = self.matchLog.REMOVE\_NON\_ASCII

        filteredMatch = filteredMatch.REMOVE\_RANDOM\_CHARACTERS

        filteredMatch = filteredMatch.SPLIT\_BY\_GAME

    filteredMatch = filteredMatch.SPLIT\_BY\_TURN

        self.matchLog = filteredMatch

    method GET\_DECKLISTS(self, game):

        decklists = {self.players[0]: dict(), self.players[1]: dict()}

        playCardPattenMatches = self.matchLog.GET\_PLAYED\_CARDS

    playCardPattern = self.matchLog.GET\_PLAYED\_CARDS

        for turn in game:

        playCardMatches = playCardPattern.FIND\_ALL(turn)

            revealedMatches = revealedCardPattern.FIND\_ALL(turn)

            for cardPlayed in playCardMatches:

                if cardPlayed in revealedMatches:

                    revealedMatches.remove(cardPlayed)

                if cardPlayed in decklists[card]:

                    decklists[cardPlayed][cardPlayed] += 1

                else:

                    decklists[cardPlayed][cardPlayed] = 1

            for cardRevealed in revealedMatches:

                if cardRevealed in decklists[cardRevealed]:

                    decklists[cardRevealed][cardRevealed] += 1

                else:

                    decklists[cardRevealed] = 1

        return decklists

     method GET\_ON\_PLAY(self, game):

        onPlay = game.SEARCH('(\S+) chooses to play first')

        return onPlay

    method GET\_STARTING\_HANDS(self, game):

        startingHands = self.matchLog.FIND\_ALL\_STARTING\_HANDS\_IN\_GAME

        return startingHands

    method GET\_WINNER(self, game):

        concededPattern = '(\S+) has conceded'

        winsPattern = '(\S+) wins the game'

        losesPattern = '(\S+) loses the game'

        conceded = concededPattern.SEARCH(concededPattern)

        wins = winsPattern.SEARCH(winsPattern)

        loses = losesPattern.SEARCH(losesPattern)

        if a player wins:

            return wins

        else if a player conceded:

        return conceded

        else if a player loses:

            return loses

        else:

            return 'NA'

#### mtgtop8.py

CLASS DRIVER\_CONTROLLER:

    CONSTRUCTOR(self)

        self.url = "https://mtgtop8.com/search"

        self.driver = webdriver.CONNECT()

    METHODS(self)

        method RETURN\_DICT\_NAMES(self, deckLists, date):

            self.GET\_SITE()

            self.CLEAR\_COOKIE\_BANNER()

            self.INPUT\_FORM\_DATA(format, self.REFACTOR\_DECKLISTS(deckLists), date)

            deckNames = self.GET\_DECK\_NAMES()

            if deckNames are unknown:

                self.QUIT()

                return deckNames

            dictNames = self.GET\_DICT\_NAMES(deckNames)

            return dictNames

        method GET\_SITE(self):

            self.driver.get(self.URL)

        method INPUT\_FORM\_DATA(self, deckList, date):

            self.driver.find(SIDEBOARD).CLICK()

            textarea = self.driver.FIND(CARDS)

            for card in deckList:

                textarea.TYPE(card + RETURN)

            self.driver.FIND(DATE\_ELEMENT).TYPE(date)

            self.driver.FIND(SUBMIT).CLICK()

        method clearCookieBanner(self):

            self.driver.find(COOKIE\_WINDOW).CLICK()

        method getDeckNames(self):

            deckNames = []

            decks = length(self.driver.FIND\_ALL(DECKS\_ON\_PAGE))

            if decks < 1:

                return 'unknown'

            if decks > 5:

                decks = 5

            for deck = 1 To length(decks):

                deckNames.append(self.driver.FIND(DECK\_NAME).TEXT)

            return deckNames

        method QUIT\_DC(self):

            self.driver.QUIT()

        method REFACTOR\_DECKLISTS(self, deckLists):

            cards = associative\_array()

            for game in deckLists:

                for card in game:

                    if cards[card] exists:

                        cards[card] = cards[card]+1

                    else:

                        cards[card] = 1

            return cards

        method GET\_DICT\_NAMES(self, deckNames):

            dictNames = {deckName:0 for deckName in deckNames}

            for deckName in deckNames:

                dictNames[deckName] = dictNames[deckName] + 1

            for deckName in dictNames:

                dictNames[deckName] = dictNames[deckName] / length(deckNames)

        return dictNames

#### dbCMD.py

function openData():

    conn = CONNECT\_TO\_DATABASE

    cursor = CURSOR

    atempt:

        matches = cursor.EXECUTE("SELECT \* FROM matches ORDER BY matchID DESC;").FETCH\_ALL

        matches.CONVERT\_TO\_JSON

        return matches

    if error:

        pass

    finally:

        conn.CLOSE

        cursor.CLOSE

function getUserData():

    name = GET\_NAME

    conn = CONNECT\_TO\_DATABASE

    cursor = CURSOR

    matches = cursor.execute("SELECT winner FROM matches;").FETCH\_ALL

    matches.GET\_WIN\_PERCENTAGE(name)

    return matches

    conn.CLOSE

    cursor.CLOSE

function getOppWinrate(opponent):

    name = GET\_NAME

    conn = CONNECT\_TO\_DATABASE

    cursor = CURSOR

    matches = cursor.execute("SELECT opponent, winner FROM matches ORDER BY opponent;").fetchall()

    matches.REFORMAT\_DATA

    record = []

    while True:

        low = 0

high = len(results) - 1

mid = 0

while low <= high:

mid = (high + low) // 2

if results[mid][0] < opponent:

low = mid + 1

elif results[mid][0] > opponent:

high = mid - 1

else:

for win in results[mid][1]:

record.append(win)

results.remove(mid)

    return matches

    conn.CLOSE

    cursor.CLOSE

if PROGRAM\_WAS\_RUN\_BY\_ANOTHER\_FILE:

    if ARGUEMENT(1) = 'loaded':

        openData()

    elif ARGUEMENT(1) = 'profile':

        getUserData()

    elif ARGUEMENT(1) =='opponent':

        getOppWinrate(ARGUEMENT(2))

### Javascript Files

#### renderer.js

function runPythonSync(){

  PYTHON\_FILE.run('./python/sync.py'){

    if error{

      PRINT\_ERROR\_TO\_SCREEN

    }

  };

}

function loadFirstPage(){

  arguements = ['loaded']

  PYTHON\_FILE.run('./python/dbCMD.py', arguements){

    SHOW\_RESULTS\_TO\_SCREEN

    if error{

      PRINT\_ERROR\_TO\_SCREEN

    }

  }

}

function loadProfile(){

  HIDE\_MAIN\_PAGE

  SHOW\_PROFILE\_PAGE

  arguements = ['profile']

  PYTHON\_FILE.run('./python/dbCMD.py', arguements){

    SHOW\_RESULTS\_TO\_SCREEN

    if error{

      PRINT\_ERROR\_TO\_SCREEN

    }

  }

}

function closeProfile(){

  SHOW\_MAIN\_PAGE

  HIDE\_PROFILE\_PAGE

}

function getOpponentWinrate(){

  arguements = ['opponent']

  PYTHON\_FILE.run('./python/dbCMD.py', arguement){

    SHOW\_RESULTS\_TO\_SCREEN

    if error{

      PRINT\_ERROR\_TO\_SCREEN

    }

  }

}

## Design of Advanced Higher Concepts (YELLOW HIGHLIGHT)

Advanced Higher Concepts:

* OOP
* Imperative Programming (sync.py, renderer.js)
* Binary Search
* Connecting to a database via a programming language.
* Execution of SQL operations.

## Design of Beyond Advanced Higher Concepts (BLUE HIGHLIGHT)

* Web scraping (mtgtop8.py)
* Associative 1D and 2D arrays (see Arrays of Records and 2D Arrays), as dictionaries in Python.
* Utilizing the JSON format (see Arrays of Records and 2D Arrays).
* Parameterized queries.

## Query Design

### Get average winrate

|  |  |
| --- | --- |
| SELECT | winner |
| FROM | Matches |

### Get max matchID

|  |  |
| --- | --- |
| SELECT | MAXIMUM matchID |
| FROM | Matches |

### Get specific winrate against an opponent

|  |  |
| --- | --- |
| SELECT | opponent, winner |
| FROM | Matches |
| ORDER BY | opponent ASCENDING |

### Insert new match

|  |  |
| --- | --- |
| INSERT | matchID, filename, opponent, decknames, decklistP1, decklistP2, play, winner, format, type, date |
| TABLE | Matches |
| VALUES |  |

### Get all records to display on page

|  |  |
| --- | --- |
| SELECT | matchID, filename, opponent, decknames, decklistP1, decklistP2, play, winner, format, type, date |
| FROM | Matches |
| ORDER BY | matchID DESCENDING |

## Data Dictionary Design

### Match table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | PK/FK | Datatype | Validation | Formatting (if not null) | Description | Example |
| matchID | PK | Integer | Presence |  | Unique surrogate key | 1 |
| filename |  | Text | Presence |  | Name of file | Gh33-dhhd9-eiuoew-fewfw-dwa23tger.dat |
| opponent |  | Blob | Presence | (player2 name) | Player2’s username | BoringGamer |
| decknames |  | Blob |  | {“deckname”: chance, …} | Player1’s and Player2’s deck names | {“AggressiveDeck”:0.75, “ControlDeck”:0.25} |
| deckListP1 |  | Blob |  | ({“cardName”:copies, “cardName2”:copies}, etc.) | Player 1’s decklist each game | ({“Plains”:1}, {“Swamp”:2, “Plains”:3}, {“Bear”:4, “Forest”:1, “Mountain”:3}) |
| deckListP1 |  | Blob |  | ({“cardName”:copies, “cardName2”:copies}, etc.) | Player 2’s decklist each game | ({“Stun Spell”:3,}, {“Trip Spell”:2}, {“Human”:2, “Archery Spell”:3, “Swamp”:3}) |
| play |  | Blob | Presence | (player 1 name, player X name, …) | List of the player’s name which went first | (CoolGamer, CoolGamer, BoringGamer) |
| winner |  | Blob | Presence | (player X wins, player X wins, etc.) | The outcome of the match | (BoringGamer, CoolGamer, CoolGamer) |
| date |  | Text | Presence | "YYYY-MM-DD" | The date of the match | 2022-12-25 |

## Arrays of Records and 2D Arrays

Note: All 2D associative arrays fit the JSON format.

#### mtgScraper.py

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Formatting | Example |
| decklists | 2D associative array | {2:{‘player2Name’:{‘cardName1’:number, cardName2:number, etc.}, ‘player1Name’:{‘cardName1’:number, cardName2:number, etc.}, 1:{‘player2Name’:{‘cardName1’:number, cardName2:number, etc.}, ‘player1Name’:{‘cardName1’:number, cardName2:number, etc.}, 0:{‘player2Name’:{‘cardName1’:number, cardName2:number, etc.}, ‘player1Name’:{‘cardName1’:number, cardName2:number, etc.}} | {1: {'Hoarse\_Duck': {'Mountain': 1, 'Monastery Swiftspear': 1, 'Great Furnace': 1, 'Chain Lightning': 1, 'Lightning Bolt': 1}, '-Sire': {'Saprazzan Skerry': 1, 'Peat Bog': 1, 'Prologue to Phyresis': 1, 'Island': 1, 'Pentad Prism': 1, 'Contentious Plan': 1}}, 2: {'Hoarse\_Duck': {'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Kuldotha Rebirth': 2, 'Galvanic Blast': 1, 'Chromatic Star': 1, 'Chain Lightning': 1, 'Raze': 2, 'Voldaren Epicure': 2, 'Lightning Bolt': 2, 'Reckless Impulse': 1, 'Goblin Blast-Runner': 1}, '-Sire': {'Hickory Woodlot': 4, 'Saprazzan Skerry': 3, 'Everflowing Chalice': 3, 'Lotus Petal': 2, 'Contentious Plan': 3, 'Weather the Storm': 2, 'Peat Bog': 2, 'Deep Analysis': 4, 'Pentad Prism': 3, 'Experimental Augury': 2, 'Prologue to Phyresis': 2, 'Bring the Ending': 3, 'Island': 1}}} |

#### matchRecord.py

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Formatting | Example |
| matchLists | 2D associative array | {‘P2’:[{‘cardName1’:number, cardName2:number, etc.}], ‘P1’:[{‘cardName1’:number, cardName2:number, etc.}]} | {'P2': [{'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Kuldotha Rebirth': 2, 'Galvanic Blast': 1, 'Chromatic Star': 1, 'Chain Lightning': 1, 'Raze': 2, 'Voldaren Epicure': 2, 'Lightning Bolt': 2, 'Reckless Impulse': 1, 'Goblin Blast-Runner': 1}], 'P1': [{'Hickory Woodlot': 4, 'Saprazzan Skerry': 3, 'Everflowing Chalice': 3, 'Lotus Petal': 2, 'Contentious Plan': 3, 'Weather the Storm': 2, 'Peat Bog': 2, 'Deep Analysis': 4, 'Pentad Prism': 3, 'Experimental Augury': 2, 'Prologue to Phyresis': 2, 'Bring the Ending': 3, 'Island': 1}]} |
| extra | 2D associative array | {‘play’: [playerX, playerX, playerX], ‘startingHands’:[[[‘player1’, ‘number’], [‘player2’, ‘number’]], [[‘player1’, ‘number’], [‘player2’, ‘number’]], [[‘player1’, ‘number’], [‘player2’, ‘number’]]], ‘winner’: [‘playerY’, ‘playerY’, ‘playerY’]} | {'play': ['-Sire', '-Sire'], 'startingHands': [[['-Sire', 'seven'], ['Hoarse\_Duck', 'six']], [['Hoarse\_Duck', 'seven'], ['-Sire', 'five']]], 'winner': ['Hoarse\_Duck', 'Hoarse\_Duck']} |

#### dbCMD.py

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Formatting | Example |
| results (getUserData) | 2D associative array | {number: [{‘filename’:name, ‘players’:[playername], ‘deckname’:[{deckname:number}],‘P2’:[{‘cardName1’:number, cardName2:number, etc.}], ‘P1’:[{‘cardName1’:number, cardName2:number, etc.}], ‘play’: [playerX, playerX, playerX], ‘winner’: [‘playerY’, ‘playerY’, ‘playerY’]}, ‘date’:date}]} | {77: [{'filename': 'Match\_GameLog\_fdfcf900-646b-4913-bd59-ed03575ab6f2.dat', 'players': ['Arsmith8'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 2, 'Healer of the Glade': 1, 'Orochi Leafcaller': 1, 'Overgrown Battlement': 1}], 'P2': [{'Mountain': 2, 'Voldaren Epicure': 1, 'Experimental Synthesizer': 2, 'Lightning Bolt': 1, 'Monastery Swiftspear': 1, 'Galvanic Blast': 1, 'Kuldotha Rebirth': 1}], 'play': ['Arsmith8', '-Sire', 'Arsmith8'], 'winner': ['Arsmith8', '-Sire', 'Arsmith8'], 'date': '2023-02-13 02:38:39.241021'}], 76: [{'filename': 'Match\_GameLog\_e2d420f5-3f61-4ff0-b366-83b9085eecc4.dat', 'players': ['BamboozledBTW'], 'deckNames': [{'Grixis Affinity': 1.0}], 'P1': [{'Silverbluff Bridge': 1, 'Vault of Whispers': 1, 'Chromatic Star': 4, 'Galvanic Blast': 1, 'Mistvault Bridge': 2, 'Myr Enforcer': 1, 'Ichor Wellspring': 1, 'Drossforge Bridge': 2, 'Seat of the Synod': 1, 'Blood Fountain': 2}], 'P2': [{'Forest': 7, 'Elvish Mystic': 1, 'Saruli Caretaker': 2, 'Overgrown Battlement': 1, 'Quirion Ranger': 3, 'Crimson Acolyte': 1, 'Axebane Guardian': 3, 'Lead the Stampede': 2, 'Secret Door': 1, 'Gorilla Shaman': 1, 'Llanowar Elves': 1}], 'play': ['-Sire', 'BamboozledBTW'], 'winner': ['-Sire', '-Sire'], 'date': '2023-03-05 13:06:44.359690'}], 75: [{'filename': 'Match\_GameLog\_e2c4e9a4-3896-46ec-b179-5ad96b0b12d0.dat', 'players': ['Magic4ever\_MTGO'], 'deckNames': [{'Dimir Terror': 0.5, 'Ub Delver': 0.25, 'Dimir Control': 0.25}], 'P1': [{'Island': 4, 'Consider': 2, 'Ice Tunnel': 1, 'Snuff Out': 1, 'Brainstorm': 1, 'Preordain': 1, 'Tolarian Terror': 1, 'Gurmag Angler': 1}], 'P2': [{'Forest': 1, 'Tinder Wall': 1, 'Saruli Caretaker': 1, 'Lead the Stampede': 1, 'Orochi Leafcaller': 1, 'Shield-Wall Sentinel': 1}], 'play': ['Magic4ever\_MTGO', 'Magic4ever\_MTGO', '-Sire'], 'winner': ['-Sire', 'Magic4ever\_MTGO', 'Magic4ever\_MTGO'], 'date': '2023-03-05 13:35:20.826968'}], 74: [{'filename': 'Match\_GameLog\_d8f5fa7e-84ae-4c33-a871-31cbbef1ee32.dat', 'players': ['Beicodegeia'], 'deckNames': [{'Kuldotha Rebirth': 0.6, 'Kuldotha Burn': 0.2, 'Kuldotha': 0.2}], 'P1': [{'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Chromatic Star': 1, 'Reckless Impulse': 1, 'Great Furnace': 1, 'Kuldotha Rebirth': 1, 'Chain Lightning': 1, 'Lightning Bolt': 1, 'Unholy Heat': 1}], 'P2': [{'Forest': 3, 'Saruli Caretaker': 1, 'Tinder Wall': 2, 'Winding Way': 1, 'Quirion Ranger': 1, 'Axebane Guardian': 3, 'Reaping the Graves': 1, 'Shield-Wall Sentinel': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Beicodegeia', 'Beicodegeia'], 'date': '2023-02-13 11:05:27.820094'}], 73: [{'filename': 'Match\_GameLog\_d1fff790-8c28-4cd1-bd27-e9320bf6ea28.dat', 'players': ['FromPolonia'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.6, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Quirion Ranger': 1, 'Overgrown Battlement': 2, 'Winding Way': 2, 'Jaspera Sentinel': 1, 'Tuktuk Rubblefort': 1}], 'P2': [{'Tramway Station': 2, 'Skirk Prospector': 1, 'Ichor Wellspring': 1, 'Mountain': 2, 'Faithless Looting': 1, 'Rakdos Carnarium': 1, 'First Day of Class': 1, 'Goblin Matron': 1, 'Putrid Goblin': 1}], 'play': ['FromPolonia', 'FromPolonia'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-14 17:08:20.863595'}], 72: [{'filename': 'Match\_GameLog\_cc034428-25ee-4393-8178-46928f703cdf.dat', 'players': ['GrimLavamancer103'], 'deckNames': [{'Caw Gate': 0.2, 'Cawgates': 0.4, 'I Gatti Pazzi': 0.2, 'Azorius Cow-gate': 0.2}], 'P1': [{'Sea Gate': 1, 'Basilisk Gate': 3, 'Preordain': 1, 'Citadel Gate': 2, 'Squadron Hawk': 3, 'Brainstorm': 1, 'Heap Gate': 1, 'Counterspell': 1, 'The Modern Age': 1, 'Prismatic Strands': 1}], 'P2': [{'Contaminated Aquifer': 1, 'Island': 3, 'Thought Scour': 1, 'Brainstorm': 2, 'Mental Note': 1, 'Agony Warp': 1, 'Fallaji Archaeologist': 2, 'Tolarian Terror': 1, 'Ice Tunnel': 1}], 'play': ['-Sire', 'GrimLavamancer103', '-Sire'], 'winner': ['-Sire', 'GrimLavamancer103', '-Sire'], 'date': '2023-03-05 14:28:12.870883'}], 71: [{'filename': 'Match\_GameLog\_c3781c7a-7f86-4999-8baa-6df01399aa84.dat', 'players': ['lindoso01'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.6, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 6, 'Quirion Ranger': 3, 'Overgrown Battlement': 3, 'Axebane Guardian': 2, 'Winding Way': 1, 'Freed from the Real': 1, 'Jaspera Sentinel': 1, 'Tuktuk Rubblefort': 1}], 'P2': [{'Forest': 7, 'Elvish Mystic': 1, 'Axebane Guardian': 3, 'Quirion Ranger': 2, 'Drift of Phantasms': 3, 'Overgrown Battlement': 1, 'Reaping the Graves': 1, 'Jaspera Sentinel': 1, 'Orochi Leafcaller': 1}], 'play': ['-Sire'], 'winner': ['lindoso01'], 'date': '2023-02-13 02:14:18.974611'}], 70: [{'filename': 'Match\_GameLog\_c2fa3b21-438e-4203-b77d-7c03996b98b5.dat', 'players': ['ResponsiblyStupid'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.6, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Overgrown Battlement': 1, 'Jaspera Sentinel': 1, 'Orochi Leafcaller': 1, 'Quirion Ranger': 1, 'Tuktuk Rubblefort': 1}], 'P2': [{'Barren Moor': 5, 'Street Wraith': 6, 'Drannith Healer': 15, 'Repository Skaab': 2, 'Horror of the Broken Lands': 11, 'Ash Barrens': 4, 'Island': 2, 'Dark Ritual': 3, 'Drannith Stinger': 8, 'Cabal Ritual': 2, 'Songs of the Damned': 2, 'Reaping the Graves': 2, 'Architects of Will': 7, 'Imposing Vantasaur': 3, 'Swamp': 2}], 'play': ['-Sire', '-Sire'], 'winner': ['ResponsiblyStupid', 'ResponsiblyStupid'], 'date': '2023-02-13 01:26:52.036749'}], 69: [{'filename': 'Match\_GameLog\_c0629d7b-cdec-4ad8-9e96-cb4ae78ac72e.dat', 'players': ['bergamo83'], 'deckNames': [{'Grixis Affinity': 1.0}], 'P1': [{'Mistvault Bridge': 2, 'Seat of the Synod': 3, 'Chromatic Star': 4, 'Silverbluff Bridge': 2, 'Thoughtcast': 3, 'Metallic Rebuke': 1, 'Great Furnace': 3, 'Galvanic Blast': 4, 'Gurmag Angler': 1, 'Vault of Whispers': 1, 'Kenku Artificer': 1, 'Ichor Wellspring': 2, 'Deadly Dispute': 2, 'Makeshift Munitions': 1, 'Pyroblast': 1, 'Blood Fountain': 1, 'Drossforge Bridge': 1}], 'P2': [{'Forest': 9, 'Tinder Wall': 2, 'Axebane Guardian': 3, 'Quirion Ranger': 2, 'Winding Way': 1, 'Saruli Caretaker': 2, 'Banishing Coils': 1, 'Guardian Naga': 1, 'Gorilla Shaman': 1, 'Jaspera Sentinel': 1, 'Overgrown Battlement': 1, 'Elvish Mystic': 1, 'Drift of Phantasms': 2, 'Reaping the Graves': 1}], 'play': ['bergamo83', 'bergamo83', '-Sire'], 'winner': ['-Sire', 'bergamo83', '-Sire'], 'date': '2023-02-13 20:58:45.527291'}], 68: [{'filename': 'Match\_GameLog\_b8b02b74-c418-404e-9ee5-68a73b9a13eb.dat', 'players': ['sosfabio89'], 'deckNames': [{'Walls Combo': 0.8, 'Combo Walls': 0.2}], 'P1': [{'Forest': 5, 'Saruli Caretaker': 3, 'Quirion Ranger': 1, 'Winding Way': 2, 'Axebane Guardian': 3, 'Lead the Stampede': 1, 'Drift of Phantasms': 2, 'Shield-Wall Sentinel': 1, 'Overgrown Battlement': 1, 'Orochi Leafcaller': 1}], 'P2': [{'Scoured Barrens': 2, 'Plains': 3, 'Cast Down': 3, 'Dawnbringer Cleric': 1, 'Spirited Companion': 1, 'Ephemerate': 6, 'Swamp': 2, 'Inspiring Overseer': 1, 'Thraben Inspector': 1}], 'play': ['sosfabio89', 'sosfabio89'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-15 09:46:14.052583'}], 67: [{'filename': 'Match\_GameLog\_b89bc7ed-3a6c-47b3-aee1-4fb3470faf26.dat', 'players': ['Pazmaster'], 'deckNames': [{'Walls Combo': 0.8, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Orochi Leafcaller': 1, 'Overgrown Battlement': 1, 'Secret Door': 1, 'Saruli Caretaker': 2, 'Lead the Stampede': 1}], 'P2': [{'Snow-Covered Island': 2, 'Simic Growth Chamber': 1, 'Growth Spiral': 2, 'Frantic Inventory': 1, 'Forest': 1, 'Pieces of the Puzzle': 1}], 'play': ['-Sire', 'Pazmaster'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-13 21:28:48.189778'}], 66: [{'filename': 'Match\_GameLog\_a5067f17-a84a-4fb6-8a2c-6e2ff1514ae6.dat', 'players': ['PauperFeito'], 'deckNames': [{'Reaping the Song': 1.0}], 'P1': [{'Swamp': 3, 'Street Wraith': 4, 'Imposing Vantasaur': 2, 'Island': 1, 'Drannith Healer': 4, 'Horror of the Broken Lands': 7, 'Ash Barrens': 4, 'Mountain': 1, 'Barren Moor': 2, 'Lotus Petal': 1, 'Duress': 1, 'Cabal Ritual': 1, 'Pyroblast': 1, 'Songs of the Damned': 1}], 'P2': [{'Citadel Gate': 2, 'Basilisk Gate': 2, 'Preordain': 3, 'Relic of Progenitus': 1, 'Azorius Guildgate': 1, 'The Modern Age': 2, 'Sacred Cat': 1, 'Island': 2, 'Prismatic Strands': 1, 'Squadron Hawk': 1, 'Dawnbringer Cleric': 1, 'Counterspell': 1}], 'play': ['PauperFeito', '-Sire', 'PauperFeito'], 'winner': ['NA', 'NA', 'PauperFeito'], 'date': '2023-02-13 00:53:52.479248'}], 65: [{'filename': 'Match\_GameLog\_a4b35f94-e2b5-40b0-8da3-8bf444bb6842.dat', 'players': ['hopeof130'], 'deckNames': [{'Gruul Land Loss': 0.4, 'Gruul Ponza': 0.4, 'Temur Ponza Initiative': 0.2}], 'P1': [{'Snow-Covered Forest': 2, 'Arbor Elf': 3, 'Utopia Sprawl': 1, 'Wild Growth': 1, 'Relic of Progenitus': 1, 'Annoyed Altisaur': 1, 'Thermokarst': 1, 'Highland Forest': 1}], 'P2': [{'Forest': 2, 'Saruli Caretaker': 2, 'Overgrown Battlement': 1, 'Orochi Leafcaller': 1, 'Secret Door': 1, 'Lead the Stampede': 1}], 'play': ['hopeof130', 'hopeof130'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-15 10:00:50.193994'}], 64: [{'filename': 'Match\_GameLog\_9b672e98-40a4-47b6-aa60-1a9cc4157c17.dat', 'players': ['Heisen01'], 'deckNames': [{'1.0': 0.5, 'NA': 0.5}], 'P1': [{'Azorius Guildgate': 2, 'Sea Gate': 2, 'Preordain': 2, 'Citadel Gate': 2, 'The Modern Age': 3, 'Sacred Cat': 1, 'Counterspell': 3, 'Basilisk Gate': 3, 'Island': 1, 'Dispel': 1, 'Dawnbringer Cleric': 1, 'Journey to Nowhere': 1, 'Suffocating Fumes': 1, 'Pyroblast': 1, 'Squadron Hawk': 5, 'Behold the Multiverse': 1, 'Brainstorm': 2, 'Deep Analysis': 1}], 'P2': [{'Citadel Gate': 2, 'Plains': 2, 'Thraben Inspector': 2, 'Prismatic Strands': 4, 'Ephemerate': 4, 'Basilisk Gate': 2, 'Sacred Cat': 1, 'Negate': 1, 'Journey to Nowhere': 1, 'Cliffgate': 1, 'Coalition Honor Guard': 2, 'Heap Gate': 1, 'Sea Gate': 1, 'Ardent Elementalist': 1, 'Mulldrifter': 1, 'Preordain': 1, 'Lightning Bolt': 1, 'Pyroblast': 1, 'Smash to Dust': 1, 'Guardian of the Guildpact': 1, 'Dawnbringer Cleric': 1}], 'play': ['-Sire', 'Heisen01'], 'winner': ['-Sire', '-Sire'], 'date': '2023-03-05 18:38:02.968582'}], 63: [{'filename': 'Match\_GameLog\_94bf93f4-882c-4635-83b9-6a9f31ab0d22.dat', 'players': ['modo\_baggins'], 'deckNames': [{'Kuldotha Red': 1.0}], 'P1': [{'Mountain': 2, 'Goblin Blast-Runner': 1, 'Implement of Combustion': 1, 'Great Furnace': 2, 'Kuldotha Rebirth': 1, 'Lightning Bolt': 1, 'Goblin Grenade': 1, 'Reckless Impulse': 1}], 'P2': [{'Hickory Woodlot': 2, 'Saprazzan Skerry': 2, 'Peat Bog': 1, 'Prologue to Phyresis': 1, 'Experimental Augury': 2, 'Infectious Inquiry': 1, 'Bring the Ending': 1, 'Island': 1}], 'play': ['-Sire', 'modo\_baggins', '-Sire'], 'winner': ['-Sire', 'NA', 'NA'], 'date': '2023-02-16 14:29:08.494215'}], 62: [{'filename': 'Match\_GameLog\_86c91d39-18a8-450f-b948-98e237c684bf.dat', 'players': ['Foresterf'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 7, 'Healer of the Glade': 1, 'Overgrown Battlement': 4, 'Quirion Ranger': 2, 'Saruli Caretaker': 3, 'Lead the Stampede': 1, 'Drift of Phantasms': 2, 'Axebane Guardian': 3, 'Reaping the Graves': 2, 'Tinder Wall': 1}], 'P2': [{'Island': 6, 'Faerie Seer': 3, 'Spellstutter Sprite': 4, 'Snap': 1, 'Mutagenic Growth': 2, 'Counterspell': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Foresterf', 'Foresterf'], 'date': '2023-02-15 10:33:21.877852'}], 61: [{'filename': 'Match\_GameLog\_82dac923-6994-4a74-b2a2-33417f269bce.dat', 'players': ['L\_I\_M\_E\_\_'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.8}], 'P1': [{'Forest': 3, 'Quirion Ranger': 1, 'Overgrown Battlement': 1, 'Orochi Leafcaller': 1, 'Crimson Acolyte': 1}], 'P2': [{'Snow-Covered Island': 2, 'Brainstorm': 2, 'Ash Barrens': 2, 'Snow-Covered Mountain': 1, 'Lightning Bolt': 1, 'Augur of Bolas': 1}], 'play': ['L\_I\_M\_E\_\_', 'L\_I\_M\_E\_\_'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-13 10:36:04.943403'}], 60: [{'filename': 'Match\_GameLog\_7c1602bf-aeb1-4344-bcf6-20787f97d0c6.dat', 'players': ['CaligulaFanBoi'], 'deckNames': [{'Affinity': 1.0}], 'P1': [{'Silverbluff Bridge': 2, 'Nihil Spellbomb': 1, 'Frogmite': 1, 'Krark-Clan Shaman': 1, 'Gingerbrute': 1, 'Myr Enforcer': 1}], 'P2': [{'Ash Barrens': 3, 'Swamp': 2, 'Imposing Vantasaur': 4, 'Horror of the Broken Lands': 2, 'Drannith Healer': 2, 'Drannith Stinger': 2}], 'play': ['CaligulaFanBoi', '-Sire'], 'winner': ['CaligulaFanBoi', 'CaligulaFanBoi'], 'date': '2023-02-13 00:15:42.945056'}], 59: [{'filename': 'Match\_GameLog\_7976628e-bf55-4dd6-95e7-89722ca670ed.dat', 'players': ['Alvatar'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 4, 'Saruli Caretaker': 2, 'Axebane Guardian': 2, 'Quirion Ranger': 1, 'Tuktuk Rubblefort': 1, 'Secret Door': 1}], 'P2': [{'Heap Gate': 1, 'Basilisk Gate': 3, 'Abundant Growth': 2, 'Citadel Gate': 1, 'Kor Skyfisher': 1, 'Gatecreeper Vine': 1, 'Plains': 1, 'Journey to Nowhere': 1, 'Flame Slash': 1, 'Inspiring Overseer': 1, 'Open the Gates': 1, 'Relic of Progenitus': 1}], 'play': ['Alvatar', 'Alvatar'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-13 11:22:57.804899'}], 58: [{'filename': 'Match\_GameLog\_78d11ac6-c046-4487-a8a1-863509b45d22.dat', 'players': ['VOIDx'], 'deckNames': [{'Orzhov Aggro': 0.8, 'Orzhov Midrange': 0.2}], 'P1': [{'Plains': 2, 'Swamp': 1, 'Spirited Companion': 2, 'Ephemerate': 1}], 'P2': [{'Forest': 2, 'Tinder Wall': 1, 'Overgrown Battlement': 1, 'Axebane Guardian': 1}], 'play': ['VOIDx', 'VOIDx', '-Sire'], 'winner': ['-Sire', 'NA', '-Sire'], 'date': '2023-02-14 18:08:28.722969'}], 57: [{'filename': 'Match\_GameLog\_6d8e774f-75ab-4875-b1c8-936644ba5ff8.dat', 'players': ['Pacto\_das\_Guildas'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 2, 'Quirion Ranger': 1, 'Tinder Wall': 1, 'Healer of the Glade': 1}], 'P2': [{'Mountain': 4, 'Raze': 2, 'Great Furnace': 1, 'Voldaren Epicure': 1, 'Experimental Synthesizer': 2, 'Chromatic Star': 1, 'Reckless Impulse': 2, 'Galvanic Blast': 1, 'Goblin Blast-Runner': 1, 'Lightning Bolt': 1}], 'play': ['Pacto\_das\_Guildas', '-Sire'], 'winner': ['Pacto\_das\_Guildas', 'Pacto\_das\_Guildas'], 'date': '2023-02-13 02:07:16.757931'}], 56: [{'filename': 'Match\_GameLog\_616adf83-4899-4980-ba7a-b8da1e71e5cd.dat', 'players': ['Magic4ever\_MTGO'], 'deckNames': [{'Walls Combo': 0.8, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Winding Way': 1, 'Secret Door': 1, 'Overgrown Battlement': 1, 'Saruli Caretaker': 2, 'Axebane Guardian': 2, 'Quirion Ranger': 1, 'Pulse of Murasa': 1}], 'P2': [{'Contaminated Aquifer': 1, 'Ice Tunnel': 3, 'Brainstorm': 2, 'Snuff Out': 2, 'Island': 2, 'Counterspell': 2, 'Preordain': 1, 'Mental Note': 2, 'Fallaji Archaeologist': 1, 'Shrivel': 1, 'Tolarian Terror': 1, 'Gurmag Angler': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Magic4ever\_MTGO', 'Magic4ever\_MTGO'], 'date': '2023-02-13 11:45:54.426555'}], 55: [{'filename': 'Match\_GameLog\_542159c9-822b-441b-8b5f-c2a11a8ff71b.dat', 'players': ['ProfPidu'], 'deckNames': [{'Mono Blue Aggro': 0.8, 'Mono U Faeries': 0.2}], 'P1': [{'Snow-Covered Island': 6, 'Faerie Seer': 2, 'Of One Mind': 3, 'Faerie Miscreant': 2, 'Snap': 1, 'Echoing Truth': 1, 'Dispel': 1}], 'P2': [{'Sea Gate': 3, 'Basilisk Gate': 1, 'Dawnbringer Cleric': 2, 'Preordain': 2, 'Behold the Multiverse': 1, 'Island': 1, 'Brainstorm': 1, 'Plains': 1, 'Counterspell': 1}], 'play': ['ProfPidu', '-Sire'], 'winner': ['ProfPidu', 'ProfPidu'], 'date': '2023-03-05 17:50:52.157195'}], 54: [{'filename': 'Match\_GameLog\_45c99bc2-e534-45a5-84a6-8410a6c89d6b.dat', 'players': ['LucasACR'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 9, 'Llanowar Elves': 2, 'Lead the Stampede': 1, 'Quirion Ranger': 7, 'Saruli Caretaker': 3, 'Scattershot Archer': 2, 'Axebane Guardian': 1, 'Tinder Wall': 2, 'Orochi Leafcaller': 1, 'Freed from the Real': 1, 'Reaping the Graves': 1, 'Overgrown Battlement': 1, 'Standard Bearer': 1}], 'P2': [{'Snow-Covered Island': 8, 'Faerie Seer': 2, 'Mutagenic Growth': 2, 'Spellstutter Sprite': 6, 'Counterspell': 1, 'Bind the Monster': 2, 'Of One Mind': 3, 'Faerie Miscreant': 1, 'Blue Elemental Blast': 1, 'Snap': 2}], 'play': ['LucasACR', '-Sire'], 'winner': ['LucasACR', 'LucasACR'], 'date': '2023-02-16 15:19:03.217075'}], 53: [{'filename': 'Match\_GameLog\_43d2005e-81cc-41f2-9106-4ac793dccc49.dat', 'players': ['morvan09'], 'deckNames': [{'Caw Gate': 0.75, 'Cawgate': 0.25}], 'P1': [{'Plains': 1, 'Island': 2, 'Crimson Acolyte': 1, 'The Modern Age': 2, 'Dawnbringer Cleric': 1, 'Basilisk Gate': 1, 'Journey to Nowhere': 1, 'Counterspell': 2, 'Azorius Guildgate': 1, 'Preordain': 1, 'Sacred Cat': 1}], 'P2': [{'Mountain': 3, 'Monastery Swiftspear': 1, 'Reckless Impulse': 2, 'Experimental Synthesizer': 1, 'Implement of Combustion': 1, 'Voldaren Epicure': 1, 'Relic of Progenitus': 1, 'Kuldotha Rebirth': 1, 'Raze': 1, 'Lava Dart': 1}], 'play': ['-Sire', 'morvan09', '-Sire'], 'winner': ['NA', 'morvan09', '-Sire'], 'date': '2023-03-05 19:08:04.160233'}], 52: [{'filename': 'Match\_GameLog\_433d65ba-41c4-4878-a2b1-1bf3dcff9c8f.dat', 'players': ['Brivenix'], 'deckNames': [{'Reaping the Song': 1.0}], 'P1': [{'Barren Moor': 2, 'Street Wraith': 6, 'Ash Barrens': 2, 'Mountain': 2, 'Imposing Vantasaur': 2, 'Drannith Healer': 2, 'Horror of the Broken Lands': 2, 'Swamp': 1, 'Drannith Stinger': 2}], 'P2': [{'Citadel Gate': 2, 'Basilisk Gate': 1, 'The Modern Age': 1, 'Prismatic Strands': 1, 'Journey to Nowhere': 1, 'Azorius Chancery': 1, 'Island': 1, 'Brainstorm': 1, 'Behold the Multiverse': 1, 'Dawnbringer Cleric': 1}], 'play': ['-Sire', '-Sire', 'Brivenix'], 'winner': ['Brivenix', 'NA', 'NA'], 'date': '2023-02-12 21:43:58.666363'}], 51: [{'filename': 'Match\_GameLog\_40ab5889-f530-4d43-a938-5cff47501920.dat', 'players': ['rEPTILIUM\_'], 'deckNames': [{'Grixis Affinity': 1.0}], 'P1': [{'Great Furnace': 1, 'Krark-Clan Shaman': 1, 'Seat of the Synod': 1, 'Ichor Wellspring': 1, 'Chromatic Star': 1, 'Thoughtcast': 1, 'Silverbluff Bridge': 1, 'Swamp': 1, 'Nihil Spellbomb': 1, 'Myr Enforcer': 2, 'Gixian Infiltrator': 1}], 'P2': [{'Forest': 3, 'Quirion Ranger': 1, 'Overgrown Battlement': 2, 'Lead the Stampede': 1, 'Saruli Caretaker': 2}], 'play': ['-Sire', '-Sire'], 'winner': ['rEPTILIUM\_', 'rEPTILIUM\_'], 'date': '2023-02-20 18:27:43.933273'}], 50: [{'filename': 'Match\_GameLog\_3fb1cb36-8cac-4fd4-9261-bd49064f8afa.dat', 'players': ['KidRenan'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 6, 'Tinder Wall': 1, 'Orochi Leafcaller': 2, 'Secret Door': 1, 'Crimson Acolyte': 1, 'Quirion Ranger': 1, 'Saruli Caretaker': 1}], 'P2': [{'Drossforge Bridge': 1, 'Swamp': 1, 'Blood Fountain': 1, 'Galvanic Blast': 2, 'Mistvault Bridge': 1, 'Ichor Wellspring': 2, 'Frogmite': 1, 'Silverbluff Bridge': 1, 'Krark-Clan Shaman': 1, 'Red Elemental Blast': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['KidRenan', 'KidRenan'], 'date': '2023-02-20 18:45:18.869248'}], 49: [{'filename': 'Match\_GameLog\_3981f9a3-0739-4b01-a097-16722980ee33.dat', 'players': ['Cesta'], 'deckNames': [{'Dimir Terror': 0.6, 'Tivit, Seller Of Secrets': 0.2, 'Ub Terror': 0.2}], 'P1': [{'Contaminated Aquifer': 1, 'Snow-Covered Island': 1}], 'P2': [{'Forest': 2, 'Saruli Caretaker': 1, 'Quirion Ranger': 1, 'Standard Bearer': 1}], 'play': ['Cesta', '-Sire'], 'winner': ['Cesta', 'Cesta'], 'date': '2023-02-15 10:57:28.802106'}], 48: [{'filename': 'Match\_GameLog\_342b6d42-f5c2-4c63-a292-d39ce7f5850c.dat', 'players': ['A\_AdeptoTerra'], 'deckNames': [{'Walls Combo': 0.6666666666666666, 'Walls': 0.3333333333333333}], 'P1': [{'Forest': 6, 'Llanowar Elves': 1, 'Winding Way': 2, 'Quirion Ranger': 1, 'Axebane Guardian': 3, 'Shield-Wall Sentinel': 1, 'Orochi Leafcaller': 1, 'Secret Door': 1, 'Overgrown Battlement': 2}], 'P2': [{'Peat Bog': 1, 'Island': 1, 'Everflowing Chalice': 1, 'Prologue to Phyresis': 3, 'Hickory Woodlot': 2, 'Bring the Ending': 1, 'Contentious Plan': 1, 'Pentad Prism': 1, 'Anoint with Affliction': 2, 'Deep Analysis': 2, 'Saprazzan Skerry': 1, 'Nihil Spellbomb': 1, 'Cast Down': 1, 'Forest': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['A\_AdeptoTerra', 'A\_AdeptoTerra'], 'date': '2023-02-16 13:10:50.724319'}], 47: [{'filename': 'Match\_GameLog\_2c9f23ad-5f24-41f8-8e4a-5c1cc7f6305d.dat', 'players': ['FranMtg'], 'deckNames': [{'Caw Gate': 0.6, 'Cawgates': 0.4}], 'P1': [{'Azorius Guildgate': 2, 'Preordain': 1, 'Sea Gate': 1, 'Squadron Hawk': 2, 'Basilisk Gate': 1, 'The Modern Age': 1, 'Destroy Evil': 1}], 'P2': [{'Great Furnace': 2, 'Monastery Swiftspear': 2, 'Mountain': 1, 'Chain Lightning': 1, 'Reckless Impulse': 1, 'Experimental Synthesizer': 1}], 'play': ['FranMtg', '-Sire'], 'winner': ['FranMtg', 'FranMtg'], 'date': '2023-03-05 19:21:29.372175'}], 46: [{'filename': 'Match\_GameLog\_2998aec5-aa88-48ea-99e3-cae065203d30.dat', 'players': ['PieSmug'], 'deckNames': [{'UR Skred': 1.0}], 'P1': [{'Snow-Covered Island': 2, 'Ash Barrens': 2, 'Snow-Covered Mountain': 2, 'Accumulated Knowledge': 1, 'Faerie Seer': 2, 'Preordain': 1, 'Volatile Fjord': 2, 'Spellstutter Sprite': 1, 'Red Elemental Blast': 1, 'Augur of Bolas': 1, 'Lightning Bolt': 1}], 'P2': [{'Sea Gate': 3, 'Azorius Guildgate': 2, 'Brainstorm': 1, 'Preordain': 2, 'Basilisk Gate': 2, 'Deep Analysis': 1, 'Guardian of the Guildpact': 1, 'Sacred Cat': 1, 'Counterspell': 1, 'Squadron Hawk': 2}], 'play': ['-Sire', 'PieSmug'], 'winner': ['-Sire', '-Sire'], 'date': '2023-03-05 15:00:12.185103'}], 45: [{'filename': 'Match\_GameLog\_25ae01b1-46be-4ab7-95eb-095f71526e52.dat', 'players': ['callowaysmith1'], 'deckNames': [{'Walls Combo': 0.75, 'Walls': 0.25}], 'P1': [{'Forest': 4, 'Jaspera Sentinel': 1, 'Winding Way': 1, 'Overgrown Battlement': 1, 'Obsidian Acolyte': 1, 'Orochi Leafcaller': 1, 'Lead the Stampede': 1, 'Saruli Caretaker': 1, 'Axebane Guardian': 1, 'Shield-Wall Sentinel': 1, 'Secret Door': 1}], 'P2': [{'Island': 2, 'Preordain': 2, 'Brainstorm': 1, 'Thought Scour': 1, 'Contaminated Aquifer': 1, 'Tolarian Terror': 3, 'Shrivel': 1, 'Ice Tunnel': 1}], 'play': ['-Sire', 'callowaysmith1', '-Sire'], 'winner': ['-Sire', 'callowaysmith1', '-Sire'], 'date': '2023-02-13 01:58:35.849606'}], 44: [{'filename': 'Match\_GameLog\_25048b2b-2557-4ce7-86a1-183b432b0d05.dat', 'players': ['lindoso01'], 'deckNames': [{'Walls Combo': 0.6, 'Combo Walls': 0.2, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 5, 'Quirion Ranger': 2, 'Tinder Wall': 2, 'Axebane Guardian': 1, 'Lead the Stampede': 1, 'Saruli Caretaker': 2, 'Drift of Phantasms': 2, 'Reaping the Graves': 1}], 'P2': [{'Silverbluff Bridge': 2, 'Seat of the Synod': 1, 'Chromatic Star': 1, 'Krark-Clan Shaman': 2, 'Vault of Whispers': 1, 'Deadly Dispute': 1, 'Gixian Infiltrator': 1, 'Great Furnace': 1, 'Ichor Wellspring': 1, 'Blood Fountain': 1, 'Myr Enforcer': 1}], 'play': ['lindoso01', '-Sire'], 'winner': ['lindoso01', 'lindoso01'], 'date': '2023-02-14 17:42:40.747859'}], 43: [{'filename': 'Match\_GameLog\_2225113e-b639-49df-827b-657bb75e2ca5.dat', 'players': ['Uchoa'], 'deckNames': [{'1.0': 0.5, 'NA': 0.5}], 'P1': [{'Peat Bog': 1, 'Everflowing Chalice': 1, 'Nihil Spellbomb': 1, 'Energy Refractor': 2}], 'P2': [{'Saprazzan Skerry': 2, 'Everflowing Chalice': 2, 'Hickory Woodlot': 1, 'Nihil Spellbomb': 1, 'Island': 1, 'Pentad Prism': 2, 'Experimental Augury': 1, 'Infectious Inquiry': 1, 'Prologue to Phyresis': 1}], 'play': ['Uchoa', 'Uchoa'], 'winner': ['U', '-Sire'], 'date': '2023-02-16 13:23:27.644685'}], 42: [{'filename': 'Match\_GameLog\_20f5a869-20ea-4045-9173-0666a5a6ec6e.dat', 'players': ['anduyn'], 'deckNames': [{'Mono U Faeries': 1.0}], 'P1': [{'Island': 4, 'Faerie Seer': 4, 'Bind the Monster': 1, 'Preordain': 1, 'Snap': 3, 'Counterspell': 1, 'Of One Mind': 1, 'Faerie Miscreant': 1, 'Brinebarrow Intruder': 1}], 'P2': [{'Forest': 1, 'Scattershot Archer': 1, 'Llanowar Elves': 2, 'Tinder Wall': 1, 'Jaspera Sentinel': 1, 'Winding Way': 1}], 'play': ['-Sire', '-Sire', 'anduyn'], 'winner': ['anduyn', '-Sire', 'anduyn'], 'date': '2023-02-20 19:17:40.475841'}], 41: [{'filename': 'Match\_GameLog\_19bb2287-d011-4ad4-a9ad-90fa2496a956.dat', 'players': ['cbouz'], 'deckNames': [{'Grixis Affinity': 0.4, 'Affinity': 0.6}], 'P1': [{'Drossforge Bridge': 3, 'Seat of the Synod': 2, 'Silverbluff Bridge': 1, 'Electrickery': 1, 'Frogmite': 1, 'Myr Enforcer': 1, 'Krark-Clan Shaman': 2, 'Great Furnace': 2, 'Mistvault Bridge': 1, 'Ichor Wellspring': 1, 'Nihil Spellbomb': 1, 'Galvanic Blast': 1, 'Chromatic Star': 1}], 'P2': [{'Forest': 3, 'Tinder Wall': 1, 'Winding Way': 2, 'Quirion Ranger': 1, 'Orochi Leafcaller': 2, 'Axebane Guardian': 3, 'Drift of Phantasms': 1, 'Banishing Coils': 1, 'Crimson Acolyte': 1, 'Jaspera Sentinel': 1, 'Saruli Caretaker': 1, 'Tuktuk Rubblefort': 1}], 'play': ['-Sire', 'cbouz', '-Sire'], 'winner': ['-Sire', 'cbouz', 'NA'], 'date': '2023-02-13 21:16:15.435274'}], 40: [{'filename': 'Match\_GameLog\_17d73bc1-0a51-4a17-ac8d-5f99e455521f.dat', 'players': ['Beppone'], 'deckNames': [{'UR Skred': 1.0}], 'P1': [{'Volatile Fjord': 1, 'Snow-Covered Island': 5, 'Spellstutter Sprite': 2, 'Ash Barrens': 4, 'Augur of Bolas': 1, 'Preordain': 3, 'Maestros Theater': 1, 'Counterspell': 3, 'Faerie Seer': 3, 'Lightning Bolt': 2, 'Red Elemental Blast': 2}], 'P2': [{'Saprazzan Skerry': 1, 'Island': 2, 'Bring the Ending': 2, 'Everflowing Chalice': 4, 'Lotus Petal': 3, 'Infectious Inquiry': 2, 'Experimental Augury': 1, 'Prologue to Phyresis': 1, 'Weather the Storm': 3, 'Hickory Woodlot': 2, 'Deep Analysis': 1}], 'play': ['Beppone', '-Sire', 'Beppone'], 'winner': ['Beppone', '-Sire', 'NA'], 'date': '2023-02-16 14:09:16.474615'}], 39: [{'filename': 'Match\_GameLog\_0b0b1ca4-1b9f-4dac-97e5-f40ff000a961.dat', 'players': ['Piergi'], 'deckNames': [{'Walls Combo': 0.6, 'Green Gates Initiative': 0.2, 'Combo Walls': 0.2}], 'P1': [{'Forest': 2, 'Quirion Ranger': 1, 'Overgrown Battlement': 1, 'Tinder Wall': 1, 'Axebane Guardian': 1, 'Saruli Caretaker': 1}], 'P2': [{'Great Furnace': 1, 'Galvanic Blast': 1, 'Experimental Synthesizer': 2, 'Mountain': 2, 'Ancient Den': 1, 'Smash to Dust': 1, 'Lightning Bolt': 2, 'Ichor Wellspring': 1, 'Makeshift Munitions': 1}], 'play': ['-Sire', '-Sire', 'Piergi'], 'winner': ['Piergi', '-Sire', 'Piergi'], 'date': '2023-02-16 14:51:37.650101'}], 38: [{'filename': 'Match\_GameLog\_052679ad-d68b-4bb3-a40f-a7ff030dc218.dat', 'players': ['sc00nes'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 5, 'Overgrown Battlement': 1, 'Axebane Guardian': 2, 'Winding Way': 1, 'Tinder Wall': 1, 'Lead the Stampede': 2, 'Quirion Ranger': 1, 'Banishing Coils': 1, 'Secret Door': 1, 'Shield-Wall Sentinel': 1, 'Drift of Phantasms': 1, 'Valakut Invoker': 1, 'Guardian Naga': 1}], 'P2': [{'Silverbluff Bridge': 1, 'Preordain': 1, 'Vault of Whispers': 1, 'Terrarion': 1, 'Mistvault Bridge': 2, 'Seat of the Synod': 2, 'Ichor Wellspring': 2, 'Thoughtcast': 3, 'Pyroblast': 1, 'Hydroblast': 1, 'Deadly Dispute': 1, 'Chromatic Star': 1, 'Krark-Clan Shaman': 1, 'Myr Enforcer': 1}], 'play': ['-Sire', '-Sire', 'sc00nes'], 'winner': ['sc00nes', '-Sire', 'sc00nes'], 'date': '2023-02-14 17:27:37.852952'}], 37: [{'filename': 'Match\_GameLog\_04c55b86-8728-411c-a31a-dcd1d57f7f7e.dat', 'players': ['ProfPidu'], 'deckNames': [{'Mono Blue Aggro': 0.6, 'Ub Faeries': 0.2, 'Mono U Faeries': 0.2}], 'P1': [{'Snow-Covered Island': 6, 'Spellstutter Sprite': 3, 'Gut Shot': 1, 'Snap': 1, 'Faerie Seer': 1, 'Of One Mind': 1, 'Counterspell': 1}], 'P2': [{'Sea Gate': 1, 'Plains': 1, 'The Modern Age': 3, 'Prismatic Strands': 3, 'Brainstorm': 2, 'Azorius Guildgate': 2, 'Squadron Hawk': 4, 'Citadel Gate': 2, 'Preordain': 1, 'Island': 1, 'Sacred Cat': 1, 'Suffocating Fumes': 1}], 'play': ['-Sire', 'ProfPidu'], 'winner': ['-Sire', '-Sire'], 'date': '2023-03-05 21:10:46.227567'}], 36: [{'filename': 'Match\_GameLog\_04608bd8-1192-43a7-93b5-91b253171d1c.dat', 'players': ['Corzalan'], 'deckNames': [{'Walls Combo': 0.8, 'Combo Walls': 0.2}], 'P1': [{'Forest': 4, 'Overgrown Battlement': 2, 'Axebane Guardian': 2, 'Winding Way': 2, 'Quirion Ranger': 2, 'Orochi Leafcaller': 2, 'Valakut Invoker': 1}], 'P2': [{'Mountain': 2, 'Great Furnace': 1, 'Dwarven Forge-Chanter': 1, 'Chain Lightning': 2, 'Lava Dart': 2, 'Reckless Impulse': 1, 'Lightning Bolt': 1, 'Experimental Synthesizer': 1, 'Implement of Combustion': 1}], 'play': ['Corzalan', '-Sire'], 'winner': ['Corzalan', 'Corzalan'], 'date': '2023-02-15 10:14:41.441359'}], 35: [{'filename': 'Match\_GameLog\_02c28beb-adca-4ff2-8bcc-8ae594d3fc74.dat', 'players': ['Hoarse\_Duck'], 'deckNames': [{'1.0': 0.5, 'NA': 0.5}], 'P1': [{'Hickory Woodlot': 4, 'Saprazzan Skerry': 3, 'Everflowing Chalice': 3, 'Lotus Petal': 2, 'Contentious Plan': 3, 'Weather the Storm': 2, 'Peat Bog': 2, 'Deep Analysis': 4, 'Pentad Prism': 3, 'Experimental Augury': 2, 'Prologue to Phyresis': 2, 'Bring the Ending': 3, 'Island': 1}], 'P2': [{'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Kuldotha Rebirth': 2, 'Galvanic Blast': 1, 'Chromatic Star': 1, 'Chain Lightning': 1, 'Raze': 2, 'Voldaren Epicure': 2, 'Lightning Bolt': 2, 'Reckless Impulse': 1, 'Goblin Blast-Runner': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Hoarse\_Duck', 'Hoarse\_Duck'], 'date': '2023-02-15 21:21:59.164663'}], 34: [{'filename': 'Match\_GameLog\_fdfcf900-646b-4913-bd59-ed03575ab6f2.dat', 'players': ['Arsmith8'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 2, 'Healer of the Glade': 1, 'Orochi Leafcaller': 1, 'Overgrown Battlement': 1}], 'P2': [{'Mountain': 2, 'Voldaren Epicure': 1, 'Experimental Synthesizer': 2, 'Lightning Bolt': 1, 'Monastery Swiftspear': 1, 'Galvanic Blast': 1, 'Kuldotha Rebirth': 1}], 'play': ['Arsmith8', '-Sire', 'Arsmith8'], 'winner': ['Arsmith8', '-Sire', 'Arsmith8'], 'date': '2023-02-13 02:38:39.241021'}], 33: [{'filename': 'Match\_GameLog\_d8f5fa7e-84ae-4c33-a871-31cbbef1ee32.dat', 'players': ['Beicodegeia'], 'deckNames': [{'Kuldotha Rebirth': 0.6, 'Kuldotha Burn': 0.2, 'Kuldotha': 0.2}], 'P1': [{'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Chromatic Star': 1, 'Reckless Impulse': 1, 'Great Furnace': 1, 'Kuldotha Rebirth': 1, 'Chain Lightning': 1, 'Lightning Bolt': 1, 'Unholy Heat': 1}], 'P2': [{'Forest': 3, 'Saruli Caretaker': 1, 'Tinder Wall': 2, 'Winding Way': 1, 'Quirion Ranger': 1, 'Axebane Guardian': 3, 'Reaping the Graves': 1, 'Shield-Wall Sentinel': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Beicodegeia', 'Beicodegeia'], 'date': '2023-02-13 11:05:27.820094'}], 32: [{'filename': 'Match\_GameLog\_d1fff790-8c28-4cd1-bd27-e9320bf6ea28.dat', 'players': ['FromPolonia'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.6, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Quirion Ranger': 1, 'Overgrown Battlement': 2, 'Winding Way': 2, 'Jaspera Sentinel': 1, 'Tuktuk Rubblefort': 1}], 'P2': [{'Tramway Station': 2, 'Skirk Prospector': 1, 'Ichor Wellspring': 1, 'Mountain': 2, 'Faithless Looting': 1, 'Rakdos Carnarium': 1, 'First Day of Class': 1, 'Goblin Matron': 1, 'Putrid Goblin': 1}], 'play': ['FromPolonia', 'FromPolonia'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-14 17:08:20.863595'}], 31: [{'filename': 'Match\_GameLog\_c3781c7a-7f86-4999-8baa-6df01399aa84.dat', 'players': ['lindoso01'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.6, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 6, 'Quirion Ranger': 3, 'Overgrown Battlement': 3, 'Axebane Guardian': 2, 'Winding Way': 1, 'Freed from the Real': 1, 'Jaspera Sentinel': 1, 'Tuktuk Rubblefort': 1}], 'P2': [{'Forest': 7, 'Elvish Mystic': 1, 'Axebane Guardian': 3, 'Quirion Ranger': 2, 'Drift of Phantasms': 3, 'Overgrown Battlement': 1, 'Reaping the Graves': 1, 'Jaspera Sentinel': 1, 'Orochi Leafcaller': 1}], 'play': ['-Sire'], 'winner': ['lindoso01'], 'date': '2023-02-13 02:14:18.974611'}], 30: [{'filename': 'Match\_GameLog\_c2fa3b21-438e-4203-b77d-7c03996b98b5.dat', 'players': ['ResponsiblyStupid'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.6, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Overgrown Battlement': 1, 'Jaspera Sentinel': 1, 'Orochi Leafcaller': 1, 'Quirion Ranger': 1, 'Tuktuk Rubblefort': 1}], 'P2': [{'Barren Moor': 5, 'Street Wraith': 6, 'Drannith Healer': 15, 'Repository Skaab': 2, 'Horror of the Broken Lands': 11, 'Ash Barrens': 4, 'Island': 2, 'Dark Ritual': 3, 'Drannith Stinger': 8, 'Cabal Ritual': 2, 'Songs of the Damned': 2, 'Reaping the Graves': 2, 'Architects of Will': 7, 'Imposing Vantasaur': 3, 'Swamp': 2}], 'play': ['-Sire', '-Sire'], 'winner': ['ResponsiblyStupid', 'ResponsiblyStupid'], 'date': '2023-02-13 01:26:52.036749'}], 29: [{'filename': 'Match\_GameLog\_c0629d7b-cdec-4ad8-9e96-cb4ae78ac72e.dat', 'players': ['bergamo83'], 'deckNames': [{'Grixis Affinity': 1.0}], 'P1': [{'Mistvault Bridge': 2, 'Seat of the Synod': 3, 'Chromatic Star': 4, 'Silverbluff Bridge': 2, 'Thoughtcast': 3, 'Metallic Rebuke': 1, 'Great Furnace': 3, 'Galvanic Blast': 4, 'Gurmag Angler': 1, 'Vault of Whispers': 1, 'Kenku Artificer': 1, 'Ichor Wellspring': 2, 'Deadly Dispute': 2, 'Makeshift Munitions': 1, 'Pyroblast': 1, 'Blood Fountain': 1, 'Drossforge Bridge': 1}], 'P2': [{'Forest': 9, 'Tinder Wall': 2, 'Axebane Guardian': 3, 'Quirion Ranger': 2, 'Winding Way': 1, 'Saruli Caretaker': 2, 'Banishing Coils': 1, 'Guardian Naga': 1, 'Gorilla Shaman': 1, 'Jaspera Sentinel': 1, 'Overgrown Battlement': 1, 'Elvish Mystic': 1, 'Drift of Phantasms': 2, 'Reaping the Graves': 1}], 'play': ['bergamo83', 'bergamo83', '-Sire'], 'winner': ['-Sire', 'bergamo83', '-Sire'], 'date': '2023-02-13 20:58:45.527291'}], 28: [{'filename': 'Match\_GameLog\_b8b02b74-c418-404e-9ee5-68a73b9a13eb.dat', 'players': ['sosfabio89'], 'deckNames': [{'Walls Combo': 0.8, 'Combo Walls': 0.2}], 'P1': [{'Forest': 5, 'Saruli Caretaker': 3, 'Quirion Ranger': 1, 'Winding Way': 2, 'Axebane Guardian': 3, 'Lead the Stampede': 1, 'Drift of Phantasms': 2, 'Shield-Wall Sentinel': 1, 'Overgrown Battlement': 1, 'Orochi Leafcaller': 1}], 'P2': [{'Scoured Barrens': 2, 'Plains': 3, 'Cast Down': 3, 'Dawnbringer Cleric': 1, 'Spirited Companion': 1, 'Ephemerate': 6, 'Swamp': 2, 'Inspiring Overseer': 1, 'Thraben Inspector': 1}], 'play': ['sosfabio89', 'sosfabio89'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-15 09:46:14.052583'}], 27: [{'filename': 'Match\_GameLog\_b89bc7ed-3a6c-47b3-aee1-4fb3470faf26.dat', 'players': ['Pazmaster'], 'deckNames': [{'Walls Combo': 0.8, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Orochi Leafcaller': 1, 'Overgrown Battlement': 1, 'Secret Door': 1, 'Saruli Caretaker': 2, 'Lead the Stampede': 1}], 'P2': [{'Snow-Covered Island': 2, 'Simic Growth Chamber': 1, 'Growth Spiral': 2, 'Frantic Inventory': 1, 'Forest': 1, 'Pieces of the Puzzle': 1}], 'play': ['-Sire', 'Pazmaster'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-13 21:28:48.189778'}], 26: [{'filename': 'Match\_GameLog\_a5067f17-a84a-4fb6-8a2c-6e2ff1514ae6.dat', 'players': ['PauperFeito'], 'deckNames': [{'Reaping the Song': 1.0}], 'P1': [{'Swamp': 3, 'Street Wraith': 4, 'Imposing Vantasaur': 2, 'Island': 1, 'Drannith Healer': 4, 'Horror of the Broken Lands': 7, 'Ash Barrens': 4, 'Mountain': 1, 'Barren Moor': 2, 'Lotus Petal': 1, 'Duress': 1, 'Cabal Ritual': 1, 'Pyroblast': 1, 'Songs of the Damned': 1}], 'P2': [{'Citadel Gate': 2, 'Basilisk Gate': 2, 'Preordain': 3, 'Relic of Progenitus': 1, 'Azorius Guildgate': 1, 'The Modern Age': 2, 'Sacred Cat': 1, 'Island': 2, 'Prismatic Strands': 1, 'Squadron Hawk': 1, 'Dawnbringer Cleric': 1, 'Counterspell': 1}], 'play': ['PauperFeito', '-Sire', 'PauperFeito'], 'winner': ['NA', 'NA', 'PauperFeito'], 'date': '2023-02-13 00:53:52.479248'}], 25: [{'filename': 'Match\_GameLog\_a4b35f94-e2b5-40b0-8da3-8bf444bb6842.dat', 'players': ['hopeof130'], 'deckNames': [{'Gruul Land Loss': 0.4, 'Gruul Ponza': 0.4, 'Temur Ponza Initiative': 0.2}], 'P1': [{'Snow-Covered Forest': 2, 'Arbor Elf': 3, 'Utopia Sprawl': 1, 'Wild Growth': 1, 'Relic of Progenitus': 1, 'Annoyed Altisaur': 1, 'Thermokarst': 1, 'Highland Forest': 1}], 'P2': [{'Forest': 2, 'Saruli Caretaker': 2, 'Overgrown Battlement': 1, 'Orochi Leafcaller': 1, 'Secret Door': 1, 'Lead the Stampede': 1}], 'play': ['hopeof130', 'hopeof130'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-15 10:00:50.193994'}], 24: [{'filename': 'Match\_GameLog\_94bf93f4-882c-4635-83b9-6a9f31ab0d22.dat', 'players': ['modo\_baggins'], 'deckNames': [{'Kuldotha Red': 1.0}], 'P1': [{'Mountain': 2, 'Goblin Blast-Runner': 1, 'Implement of Combustion': 1, 'Great Furnace': 2, 'Kuldotha Rebirth': 1, 'Lightning Bolt': 1, 'Goblin Grenade': 1, 'Reckless Impulse': 1}], 'P2': [{'Hickory Woodlot': 2, 'Saprazzan Skerry': 2, 'Peat Bog': 1, 'Prologue to Phyresis': 1, 'Experimental Augury': 2, 'Infectious Inquiry': 1, 'Bring the Ending': 1, 'Island': 1}], 'play': ['-Sire', 'modo\_baggins', '-Sire'], 'winner': ['-Sire', 'NA', 'NA'], 'date': '2023-02-16 14:29:08.494215'}], 23: [{'filename': 'Match\_GameLog\_86c91d39-18a8-450f-b948-98e237c684bf.dat', 'players': ['Foresterf'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 7, 'Healer of the Glade': 1, 'Overgrown Battlement': 4, 'Quirion Ranger': 2, 'Saruli Caretaker': 3, 'Lead the Stampede': 1, 'Drift of Phantasms': 2, 'Axebane Guardian': 3, 'Reaping the Graves': 2, 'Tinder Wall': 1}], 'P2': [{'Island': 6, 'Faerie Seer': 3, 'Spellstutter Sprite': 4, 'Snap': 1, 'Mutagenic Growth': 2, 'Counterspell': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Foresterf', 'Foresterf'], 'date': '2023-02-15 10:33:21.877852'}], 22: [{'filename': 'Match\_GameLog\_82dac923-6994-4a74-b2a2-33417f269bce.dat', 'players': ['L\_I\_M\_E\_\_'], 'deckNames': [{'Combo Walls': 0.2, 'Walls Combo': 0.8}], 'P1': [{'Forest': 3, 'Quirion Ranger': 1, 'Overgrown Battlement': 1, 'Orochi Leafcaller': 1, 'Crimson Acolyte': 1}], 'P2': [{'Snow-Covered Island': 2, 'Brainstorm': 2, 'Ash Barrens': 2, 'Snow-Covered Mountain': 1, 'Lightning Bolt': 1, 'Augur of Bolas': 1}], 'play': ['L\_I\_M\_E\_\_', 'L\_I\_M\_E\_\_'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-13 10:36:04.943403'}], 21: [{'filename': 'Match\_GameLog\_7c1602bf-aeb1-4344-bcf6-20787f97d0c6.dat', 'players': ['CaligulaFanBoi'], 'deckNames': [{'Affinity': 1.0}], 'P1': [{'Silverbluff Bridge': 2, 'Nihil Spellbomb': 1, 'Frogmite': 1, 'Krark-Clan Shaman': 1, 'Gingerbrute': 1, 'Myr Enforcer': 1}], 'P2': [{'Ash Barrens': 3, 'Swamp': 2, 'Imposing Vantasaur': 4, 'Horror of the Broken Lands': 2, 'Drannith Healer': 2, 'Drannith Stinger': 2}], 'play': ['CaligulaFanBoi', '-Sire'], 'winner': ['CaligulaFanBoi', 'CaligulaFanBoi'], 'date': '2023-02-13 00:15:42.945056'}], 20: [{'filename': 'Match\_GameLog\_7976628e-bf55-4dd6-95e7-89722ca670ed.dat', 'players': ['Alvatar'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 4, 'Saruli Caretaker': 2, 'Axebane Guardian': 2, 'Quirion Ranger': 1, 'Tuktuk Rubblefort': 1, 'Secret Door': 1}], 'P2': [{'Heap Gate': 1, 'Basilisk Gate': 3, 'Abundant Growth': 2, 'Citadel Gate': 1, 'Kor Skyfisher': 1, 'Gatecreeper Vine': 1, 'Plains': 1, 'Journey to Nowhere': 1, 'Flame Slash': 1, 'Inspiring Overseer': 1, 'Open the Gates': 1, 'Relic of Progenitus': 1}], 'play': ['Alvatar', 'Alvatar'], 'winner': ['-Sire', '-Sire'], 'date': '2023-02-13 11:22:57.804899'}], 19: [{'filename': 'Match\_GameLog\_78d11ac6-c046-4487-a8a1-863509b45d22.dat', 'players': ['VOIDx'], 'deckNames': [{'Orzhov Aggro': 0.8, 'Orzhov Midrange': 0.2}], 'P1': [{'Plains': 2, 'Swamp': 1, 'Spirited Companion': 2, 'Ephemerate': 1}], 'P2': [{'Forest': 2, 'Tinder Wall': 1, 'Overgrown Battlement': 1, 'Axebane Guardian': 1}], 'play': ['VOIDx', 'VOIDx', '-Sire'], 'winner': ['-Sire', 'NA', '-Sire'], 'date': '2023-02-14 18:08:28.722969'}], 18: [{'filename': 'Match\_GameLog\_6d8e774f-75ab-4875-b1c8-936644ba5ff8.dat', 'players': ['Pacto\_das\_Guildas'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 2, 'Quirion Ranger': 1, 'Tinder Wall': 1, 'Healer of the Glade': 1}], 'P2': [{'Mountain': 4, 'Raze': 2, 'Great Furnace': 1, 'Voldaren Epicure': 1, 'Experimental Synthesizer': 2, 'Chromatic Star': 1, 'Reckless Impulse': 2, 'Galvanic Blast': 1, 'Goblin Blast-Runner': 1, 'Lightning Bolt': 1}], 'play': ['Pacto\_das\_Guildas', '-Sire'], 'winner': ['Pacto\_das\_Guildas', 'Pacto\_das\_Guildas'], 'date': '2023-02-13 02:07:16.757931'}], 17: [{'filename': 'Match\_GameLog\_616adf83-4899-4980-ba7a-b8da1e71e5cd.dat', 'players': ['Magic4ever\_MTGO'], 'deckNames': [{'Walls Combo': 0.8, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 3, 'Winding Way': 1, 'Secret Door': 1, 'Overgrown Battlement': 1, 'Saruli Caretaker': 2, 'Axebane Guardian': 2, 'Quirion Ranger': 1, 'Pulse of Murasa': 1}], 'P2': [{'Contaminated Aquifer': 1, 'Ice Tunnel': 3, 'Brainstorm': 2, 'Snuff Out': 2, 'Island': 2, 'Counterspell': 2, 'Preordain': 1, 'Mental Note': 2, 'Fallaji Archaeologist': 1, 'Shrivel': 1, 'Tolarian Terror': 1, 'Gurmag Angler': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Magic4ever\_MTGO', 'Magic4ever\_MTGO'], 'date': '2023-02-13 11:45:54.426555'}], 16: [{'filename': 'Match\_GameLog\_45c99bc2-e534-45a5-84a6-8410a6c89d6b.dat', 'players': ['LucasACR'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 9, 'Llanowar Elves': 2, 'Lead the Stampede': 1, 'Quirion Ranger': 7, 'Saruli Caretaker': 3, 'Scattershot Archer': 2, 'Axebane Guardian': 1, 'Tinder Wall': 2, 'Orochi Leafcaller': 1, 'Freed from the Real': 1, 'Reaping the Graves': 1, 'Overgrown Battlement': 1, 'Standard Bearer': 1}], 'P2': [{'Snow-Covered Island': 8, 'Faerie Seer': 2, 'Mutagenic Growth': 2, 'Spellstutter Sprite': 6, 'Counterspell': 1, 'Bind the Monster': 2, 'Of One Mind': 3, 'Faerie Miscreant': 1, 'Blue Elemental Blast': 1, 'Snap': 2}], 'play': ['LucasACR', '-Sire'], 'winner': ['LucasACR', 'LucasACR'], 'date': '2023-02-16 15:19:03.217075'}], 15: [{'filename': 'Match\_GameLog\_433d65ba-41c4-4878-a2b1-1bf3dcff9c8f.dat', 'players': ['Brivenix'], 'deckNames': [{'Reaping the Song': 1.0}], 'P1': [{'Barren Moor': 2, 'Street Wraith': 6, 'Ash Barrens': 2, 'Mountain': 2, 'Imposing Vantasaur': 2, 'Drannith Healer': 2, 'Horror of the Broken Lands': 2, 'Swamp': 1, 'Drannith Stinger': 2}], 'P2': [{'Citadel Gate': 2, 'Basilisk Gate': 1, 'The Modern Age': 1, 'Prismatic Strands': 1, 'Journey to Nowhere': 1, 'Azorius Chancery': 1, 'Island': 1, 'Brainstorm': 1, 'Behold the Multiverse': 1, 'Dawnbringer Cleric': 1}], 'play': ['-Sire', '-Sire', 'Brivenix'], 'winner': ['Brivenix', 'NA', 'NA'], 'date': '2023-02-12 21:43:58.666363'}], 14: [{'filename': 'Match\_GameLog\_40ab5889-f530-4d43-a938-5cff47501920.dat', 'players': ['rEPTILIUM\_'], 'deckNames': [{'Grixis Affinity': 1.0}], 'P1': [{'Great Furnace': 1, 'Krark-Clan Shaman': 1, 'Seat of the Synod': 1, 'Ichor Wellspring': 1, 'Chromatic Star': 1, 'Thoughtcast': 1, 'Silverbluff Bridge': 1, 'Swamp': 1, 'Nihil Spellbomb': 1, 'Myr Enforcer': 2, 'Gixian Infiltrator': 1}], 'P2': [{'Forest': 3, 'Quirion Ranger': 1, 'Overgrown Battlement': 2, 'Lead the Stampede': 1, 'Saruli Caretaker': 2}], 'play': ['-Sire', '-Sire'], 'winner': ['rEPTILIUM\_', 'rEPTILIUM\_'], 'date': '2023-02-20 18:27:43.933273'}], 13: [{'filename': 'Match\_GameLog\_3fb1cb36-8cac-4fd4-9261-bd49064f8afa.dat', 'players': ['KidRenan'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 6, 'Tinder Wall': 1, 'Orochi Leafcaller': 2, 'Secret Door': 1, 'Crimson Acolyte': 1, 'Quirion Ranger': 1, 'Saruli Caretaker': 1}], 'P2': [{'Drossforge Bridge': 1, 'Swamp': 1, 'Blood Fountain': 1, 'Galvanic Blast': 2, 'Mistvault Bridge': 1, 'Ichor Wellspring': 2, 'Frogmite': 1, 'Silverbluff Bridge': 1, 'Krark-Clan Shaman': 1, 'Red Elemental Blast': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['KidRenan', 'KidRenan'], 'date': '2023-02-20 18:45:18.869248'}], 12: [{'filename': 'Match\_GameLog\_3981f9a3-0739-4b01-a097-16722980ee33.dat', 'players': ['Cesta'], 'deckNames': [{'Dimir Terror': 0.6, 'Tivit, Seller Of Secrets': 0.2, 'Ub Terror': 0.2}], 'P1': [{'Contaminated Aquifer': 1, 'Snow-Covered Island': 1}], 'P2': [{'Forest': 2, 'Saruli Caretaker': 1, 'Quirion Ranger': 1, 'Standard Bearer': 1}], 'play': ['Cesta', '-Sire'], 'winner': ['Cesta', 'Cesta'], 'date': '2023-02-15 10:57:28.802106'}], 11: [{'filename': 'Match\_GameLog\_342b6d42-f5c2-4c63-a292-d39ce7f5850c.dat', 'players': ['A\_AdeptoTerra'], 'deckNames': [{'Walls Combo': 0.6666666666666666, 'Walls': 0.3333333333333333}], 'P1': [{'Forest': 6, 'Llanowar Elves': 1, 'Winding Way': 2, 'Quirion Ranger': 1, 'Axebane Guardian': 3, 'Shield-Wall Sentinel': 1, 'Orochi Leafcaller': 1, 'Secret Door': 1, 'Overgrown Battlement': 2}], 'P2': [{'Peat Bog': 1, 'Island': 1, 'Everflowing Chalice': 1, 'Prologue to Phyresis': 3, 'Hickory Woodlot': 2, 'Bring the Ending': 1, 'Contentious Plan': 1, 'Pentad Prism': 1, 'Anoint with Affliction': 2, 'Deep Analysis': 2, 'Saprazzan Skerry': 1, 'Nihil Spellbomb': 1, 'Cast Down': 1, 'Forest': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['A\_AdeptoTerra', 'A\_AdeptoTerra'], 'date': '2023-02-16 13:10:50.724319'}], 10: [{'filename': 'Match\_GameLog\_25ae01b1-46be-4ab7-95eb-095f71526e52.dat', 'players': ['callowaysmith1'], 'deckNames': [{'Walls Combo': 0.75, 'Walls': 0.25}], 'P1': [{'Forest': 4, 'Jaspera Sentinel': 1, 'Winding Way': 1, 'Overgrown Battlement': 1, 'Obsidian Acolyte': 1, 'Orochi Leafcaller': 1, 'Lead the Stampede': 1, 'Saruli Caretaker': 1, 'Axebane Guardian': 1, 'Shield-Wall Sentinel': 1, 'Secret Door': 1}], 'P2': [{'Island': 2, 'Preordain': 2, 'Brainstorm': 1, 'Thought Scour': 1, 'Contaminated Aquifer': 1, 'Tolarian Terror': 3, 'Shrivel': 1, 'Ice Tunnel': 1}], 'play': ['-Sire', 'callowaysmith1', '-Sire'], 'winner': ['-Sire', 'callowaysmith1', '-Sire'], 'date': '2023-02-13 01:58:35.849606'}], 9: [{'filename': 'Match\_GameLog\_25048b2b-2557-4ce7-86a1-183b432b0d05.dat', 'players': ['lindoso01'], 'deckNames': [{'Walls Combo': 0.6, 'Combo Walls': 0.2, 'Go-getting Walls': 0.2}], 'P1': [{'Forest': 5, 'Quirion Ranger': 2, 'Tinder Wall': 2, 'Axebane Guardian': 1, 'Lead the Stampede': 1, 'Saruli Caretaker': 2, 'Drift of Phantasms': 2, 'Reaping the Graves': 1}], 'P2': [{'Silverbluff Bridge': 2, 'Seat of the Synod': 1, 'Chromatic Star': 1, 'Krark-Clan Shaman': 2, 'Vault of Whispers': 1, 'Deadly Dispute': 1, 'Gixian Infiltrator': 1, 'Great Furnace': 1, 'Ichor Wellspring': 1, 'Blood Fountain': 1, 'Myr Enforcer': 1}], 'play': ['lindoso01', '-Sire'], 'winner': ['lindoso01', 'lindoso01'], 'date': '2023-02-14 17:42:40.747859'}], 8: [{'filename': 'Match\_GameLog\_2225113e-b639-49df-827b-657bb75e2ca5.dat', 'players': ['Uchoa'], 'deckNames': [{'1.0': 0.5, 'NA': 0.5}], 'P1': [{'Peat Bog': 1, 'Everflowing Chalice': 1, 'Nihil Spellbomb': 1, 'Energy Refractor': 2}], 'P2': [{'Saprazzan Skerry': 2, 'Everflowing Chalice': 2, 'Hickory Woodlot': 1, 'Nihil Spellbomb': 1, 'Island': 1, 'Pentad Prism': 2, 'Experimental Augury': 1, 'Infectious Inquiry': 1, 'Prologue to Phyresis': 1}], 'play': ['Uchoa', 'Uchoa'], 'winner': ['U', '-Sire'], 'date': '2023-02-16 13:23:27.644685'}], 7: [{'filename': 'Match\_GameLog\_20f5a869-20ea-4045-9173-0666a5a6ec6e.dat', 'players': ['anduyn'], 'deckNames': [{'Mono U Faeries': 1.0}], 'P1': [{'Island': 4, 'Faerie Seer': 4, 'Bind the Monster': 1, 'Preordain': 1, 'Snap': 3, 'Counterspell': 1, 'Of One Mind': 1, 'Faerie Miscreant': 1, 'Brinebarrow Intruder': 1}], 'P2': [{'Forest': 1, 'Scattershot Archer': 1, 'Llanowar Elves': 2, 'Tinder Wall': 1, 'Jaspera Sentinel': 1, 'Winding Way': 1}], 'play': ['-Sire', '-Sire', 'anduyn'], 'winner': ['anduyn', '-Sire', 'anduyn'], 'date': '2023-02-20 19:17:40.475841'}], 6: [{'filename': 'Match\_GameLog\_19bb2287-d011-4ad4-a9ad-90fa2496a956.dat', 'players': ['cbouz'], 'deckNames': [{'Grixis Affinity': 0.4, 'Affinity': 0.6}], 'P1': [{'Drossforge Bridge': 3, 'Seat of the Synod': 2, 'Silverbluff Bridge': 1, 'Electrickery': 1, 'Frogmite': 1, 'Myr Enforcer': 1, 'Krark-Clan Shaman': 2, 'Great Furnace': 2, 'Mistvault Bridge': 1, 'Ichor Wellspring': 1, 'Nihil Spellbomb': 1, 'Galvanic Blast': 1, 'Chromatic Star': 1}], 'P2': [{'Forest': 3, 'Tinder Wall': 1, 'Winding Way': 2, 'Quirion Ranger': 1, 'Orochi Leafcaller': 2, 'Axebane Guardian': 3, 'Drift of Phantasms': 1, 'Banishing Coils': 1, 'Crimson Acolyte': 1, 'Jaspera Sentinel': 1, 'Saruli Caretaker': 1, 'Tuktuk Rubblefort': 1}], 'play': ['-Sire', 'cbouz', '-Sire'], 'winner': ['-Sire', 'cbouz', 'NA'], 'date': '2023-02-13 21:16:15.435274'}], 5: [{'filename': 'Match\_GameLog\_17d73bc1-0a51-4a17-ac8d-5f99e455521f.dat', 'players': ['Beppone'], 'deckNames': [{'UR Skred': 1.0}], 'P1': [{'Volatile Fjord': 1, 'Snow-Covered Island': 5, 'Spellstutter Sprite': 2, 'Ash Barrens': 4, 'Augur of Bolas': 1, 'Preordain': 3, 'Maestros Theater': 1, 'Counterspell': 3, 'Faerie Seer': 3, 'Lightning Bolt': 2, 'Red Elemental Blast': 2}], 'P2': [{'Saprazzan Skerry': 1, 'Island': 2, 'Bring the Ending': 2, 'Everflowing Chalice': 4, 'Lotus Petal': 3, 'Infectious Inquiry': 2, 'Experimental Augury': 1, 'Prologue to Phyresis': 1, 'Weather the Storm': 3, 'Hickory Woodlot': 2, 'Deep Analysis': 1}], 'play': ['Beppone', '-Sire', 'Beppone'], 'winner': ['Beppone', '-Sire', 'NA'], 'date': '2023-02-16 14:09:16.474615'}], 4: [{'filename': 'Match\_GameLog\_0b0b1ca4-1b9f-4dac-97e5-f40ff000a961.dat', 'players': ['Piergi'], 'deckNames': [{'Walls Combo': 0.6, 'Green Gates Initiative': 0.2, 'Combo Walls': 0.2}], 'P1': [{'Forest': 2, 'Quirion Ranger': 1, 'Overgrown Battlement': 1, 'Tinder Wall': 1, 'Axebane Guardian': 1, 'Saruli Caretaker': 1}], 'P2': [{'Great Furnace': 1, 'Galvanic Blast': 1, 'Experimental Synthesizer': 2, 'Mountain': 2, 'Ancient Den': 1, 'Smash to Dust': 1, 'Lightning Bolt': 2, 'Ichor Wellspring': 1, 'Makeshift Munitions': 1}], 'play': ['-Sire', '-Sire', 'Piergi'], 'winner': ['Piergi', '-Sire', 'Piergi'], 'date': '2023-02-16 14:51:37.650101'}], 3: [{'filename': 'Match\_GameLog\_052679ad-d68b-4bb3-a40f-a7ff030dc218.dat', 'players': ['sc00nes'], 'deckNames': [{'Walls Combo': 1.0}], 'P1': [{'Forest': 5, 'Overgrown Battlement': 1, 'Axebane Guardian': 2, 'Winding Way': 1, 'Tinder Wall': 1, 'Lead the Stampede': 2, 'Quirion Ranger': 1, 'Banishing Coils': 1, 'Secret Door': 1, 'Shield-Wall Sentinel': 1, 'Drift of Phantasms': 1, 'Valakut Invoker': 1, 'Guardian Naga': 1}], 'P2': [{'Silverbluff Bridge': 1, 'Preordain': 1, 'Vault of Whispers': 1, 'Terrarion': 1, 'Mistvault Bridge': 2, 'Seat of the Synod': 2, 'Ichor Wellspring': 2, 'Thoughtcast': 3, 'Pyroblast': 1, 'Hydroblast': 1, 'Deadly Dispute': 1, 'Chromatic Star': 1, 'Krark-Clan Shaman': 1, 'Myr Enforcer': 1}], 'play': ['-Sire', '-Sire', 'sc00nes'], 'winner': ['sc00nes', '-Sire', 'sc00nes'], 'date': '2023-02-14 17:27:37.852952'}], 2: [{'filename': 'Match\_GameLog\_04608bd8-1192-43a7-93b5-91b253171d1c.dat', 'players': ['Corzalan'], 'deckNames': [{'Walls Combo': 0.8, 'Combo Walls': 0.2}], 'P1': [{'Forest': 4, 'Overgrown Battlement': 2, 'Axebane Guardian': 2, 'Winding Way': 2, 'Quirion Ranger': 2, 'Orochi Leafcaller': 2, 'Valakut Invoker': 1}], 'P2': [{'Mountain': 2, 'Great Furnace': 1, 'Dwarven Forge-Chanter': 1, 'Chain Lightning': 2, 'Lava Dart': 2, 'Reckless Impulse': 1, 'Lightning Bolt': 1, 'Experimental Synthesizer': 1, 'Implement of Combustion': 1}], 'play': ['Corzalan', '-Sire'], 'winner': ['Corzalan', 'Corzalan'], 'date': '2023-02-15 10:14:41.441359'}], 1: [{'filename': 'Match\_GameLog\_02c28beb-adca-4ff2-8bcc-8ae594d3fc74.dat', 'players': ['Hoarse\_Duck'], 'deckNames': [{'1.0': 0.5, 'NA': 0.5}], 'P1': [{'Hickory Woodlot': 4, 'Saprazzan Skerry': 3, 'Everflowing Chalice': 3, 'Lotus Petal': 2, 'Contentious Plan': 3, 'Weather the Storm': 2, 'Peat Bog': 2, 'Deep Analysis': 4, 'Pentad Prism': 3, 'Experimental Augury': 2, 'Prologue to Phyresis': 2, 'Bring the Ending': 3, 'Island': 1}], 'P2': [{'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Kuldotha Rebirth': 2, 'Galvanic Blast': 1, 'Chromatic Star': 1, 'Chain Lightning': 1, 'Raze': 2, 'Voldaren Epicure': 2, 'Lightning Bolt': 2, 'Reckless Impulse': 1, 'Goblin Blast-Runner': 1}], 'play': ['-Sire', '-Sire'], 'winner': ['Hoarse\_Duck', 'Hoarse\_Duck'], 'date': '2023-02-15 21:21:59.164663'}]} |
| results (getOppWinrate) | 2D associative array | {0:{‘winner’:[‘player’, ‘player’]}} | {0: {'winner': ['Hoarse\_Duck', 'Hoarse\_Duck']}, 1: {'winner': ['Corzalan', 'Corzalan']}, 2: {'winner': ['sc00nes', '-Sire', 'sc00nes']}, 3: {'winner': ['Piergi', '-Sire', 'Piergi']}, 4: {'winner': ['Beppone', '-Sire', 'NA']}, 5: {'winner': ['-Sire', 'cbouz', 'NA']}, 6: {'winner': ['anduyn', '-Sire', 'anduyn']}, 7: {'winner': ['U', '-Sire']}, 8: {'winner': ['lindoso01', 'lindoso01']}, 9: {'winner': ['-Sire', 'callowaysmith1', '-Sire']}, 10: {'winner': ['A\_AdeptoTerra', 'A\_AdeptoTerra']}, 11: {'winner': ['Cesta', 'Cesta']}, 12: {'winner': ['KidRenan', 'KidRenan']}, 13: {'winner': ['rEPTILIUM\_', 'rEPTILIUM\_']}, 14: {'winner': ['Brivenix', 'NA', 'NA']}, 15: {'winner': ['LucasACR', 'LucasACR']}, 16: {'winner': ['Magic4ever\_MTGO', 'Magic4ever\_MTGO']}, 17: {'winner': ['Pacto\_das\_Guildas', 'Pacto\_das\_Guildas']}, 18: {'winner': ['-Sire', 'NA', '-Sire']}, 19: {'winner': ['-Sire', '-Sire']}, 20: {'winner': ['CaligulaFanBoi', 'CaligulaFanBoi']}, 21: {'winner': ['-Sire', '-Sire']}, 22: {'winner': ['Foresterf', 'Foresterf']}, 23: {'winner': ['-Sire', 'NA', 'NA']}, 24: {'winner': ['-Sire', '-Sire']}, 25: {'winner': ['NA', 'NA', 'PauperFeito']}, 26: {'winner': ['-Sire', '-Sire']}, 27: {'winner': ['-Sire', '-Sire']}, 28: {'winner': ['-Sire', 'bergamo83', '-Sire']}, 29: {'winner': ['ResponsiblyStupid', 'ResponsiblyStupid']}, 30: {'winner': ['lindoso01']}, 31: {'winner': ['-Sire', '-Sire']}, 32: {'winner': ['Beicodegeia', 'Beicodegeia']}, 33: {'winner': ['Arsmith8', '-Sire', 'Arsmith8']}, 34: {'winner': ['Hoarse\_Duck', 'Hoarse\_Duck']}, 35: {'winner': ['Corzalan', 'Corzalan']}, 36: {'winner': ['-Sire', '-Sire']}, 37: {'winner': ['sc00nes', '-Sire', 'sc00nes']}, 38: {'winner': ['Piergi', '-Sire', 'Piergi']}, 39: {'winner': ['Beppone', '-Sire', 'NA']}, 40: {'winner': ['-Sire', 'cbouz', 'NA']}, 41: {'winner': ['anduyn', '-Sire', 'anduyn']}, 42: {'winner': ['U', '-Sire']}, 43: {'winner': ['lindoso01', 'lindoso01']}, 44: {'winner': ['-Sire', 'callowaysmith1', '-Sire']}, 45: {'winner': ['-Sire', '-Sire']}, 46: {'winner': ['FranMtg', 'FranMtg']}, 47: {'winner': ['A\_AdeptoTerra', 'A\_AdeptoTerra']}, 48: {'winner': ['Cesta', 'Cesta']}, 49: {'winner': ['KidRenan', 'KidRenan']}, 50: {'winner': ['rEPTILIUM\_', 'rEPTILIUM\_']}, 51: {'winner': ['Brivenix', 'NA', 'NA']}, 52: {'winner': ['NA', 'morvan09', '-Sire']}, 53: {'winner': ['LucasACR', 'LucasACR']}, 54: {'winner': ['ProfPidu', 'ProfPidu']}, 55: {'winner': ['Magic4ever\_MTGO', 'Magic4ever\_MTGO']}, 56: {'winner': ['Pacto\_das\_Guildas', 'Pacto\_das\_Guildas']}, 57: {'winner': ['-Sire', 'NA', '-Sire']}, 58: {'winner': ['-Sire', '-Sire']}, 59: {'winner': ['CaligulaFanBoi', 'CaligulaFanBoi']}, 60: {'winner': ['-Sire', '-Sire']}, 61: {'winner': ['Foresterf', 'Foresterf']}, 62: {'winner': ['-Sire', 'NA', 'NA']}, 63: {'winner': ['-Sire', '-Sire']}, 64: {'winner': ['-Sire', '-Sire']}, 65: {'winner': ['NA', 'NA', 'PauperFeito']}, 66: {'winner': ['-Sire', '-Sire']}, 67: {'winner': ['-Sire', '-Sire']}, 68: {'winner': ['-Sire', 'bergamo83', '-Sire']}, 69: {'winner': ['ResponsiblyStupid', 'ResponsiblyStupid']}, 70: {'winner': ['lindoso01']}, 71: {'winner': ['-Sire', 'GrimLavamancer103', '-Sire']}, 72: {'winner': ['-Sire', '-Sire']}, 73: {'winner': ['Beicodegeia', 'Beicodegeia']}, 74: {'winner': ['-Sire', 'Magic4ever\_MTGO', 'Magic4ever\_MTGO']}, 75: {'winner': ['-Sire', '-Sire']}, 76: {'winner': ['Arsmith8', '-Sire', 'Arsmith8']}} |
| results (openData) | 2D associative array | [[‘player’, [‘player’, ‘player’], [‘player’, [‘player’, ‘player’]] | [['A\_AdeptoTerra', ['A\_AdeptoTerra', 'A\_AdeptoTerra']], ['A\_AdeptoTerra', ['A\_AdeptoTerra', 'A\_AdeptoTerra']], ['Alvatar', ['-Sire', '-Sire']], ['Alvatar', ['-Sire', '-Sire']], ['Arsmith8', ['Arsmith8', '-Sire', 'Arsmith8']], ['Arsmith8', ['Arsmith8', '-Sire', 'Arsmith8']], ['BamboozledBTW', ['-Sire', '-Sire']], ['Beicodegeia', ['Beicodegeia', 'Beicodegeia']], ['Beicodegeia', ['Beicodegeia', 'Beicodegeia']], ['Beppone', ['Beppone', '-Sire', 'NA']], ['Beppone', ['Beppone', '-Sire', 'NA']], ['Brivenix', ['Brivenix', 'NA', 'NA']], ['Brivenix', ['Brivenix', 'NA', 'NA']], ['CaligulaFanBoi', ['CaligulaFanBoi', 'CaligulaFanBoi']], ['CaligulaFanBoi', ['CaligulaFanBoi', 'CaligulaFanBoi']], ['Cesta', ['Cesta', 'Cesta']], ['Cesta', ['Cesta', 'Cesta']], ['Corzalan', ['Corzalan', 'Corzalan']], ['Corzalan', ['Corzalan', 'Corzalan']], ['Foresterf', ['Foresterf', 'Foresterf']], ['Foresterf', ['Foresterf', 'Foresterf']], ['FranMtg', ['FranMtg', 'FranMtg']], ['FromPolonia', ['-Sire', '-Sire']], ['FromPolonia', ['-Sire', '-Sire']], ['GrimLavamancer103', ['-Sire', 'GrimLavamancer103', '-Sire']], ['Heisen01', ['-Sire', '-Sire']], ['Hoarse\_Duck', ['Hoarse\_Duck', 'Hoarse\_Duck']], ['Hoarse\_Duck', ['Hoarse\_Duck', 'Hoarse\_Duck']], ['KidRenan', ['KidRenan', 'KidRenan']], ['KidRenan', ['KidRenan', 'KidRenan']], ['L\_I\_M\_E\_\_', ['-Sire', '-Sire']], ['L\_I\_M\_E\_\_', ['-Sire', '-Sire']], ['LucasACR', ['LucasACR', 'LucasACR']], ['LucasACR', ['LucasACR', 'LucasACR']], ['Magic4ever\_MTGO', ['Magic4ever\_MTGO', 'Magic4ever\_MTGO']], ['Magic4ever\_MTGO', ['Magic4ever\_MTGO', 'Magic4ever\_MTGO']], ['Magic4ever\_MTGO', ['-Sire', 'Magic4ever\_MTGO', 'Magic4ever\_MTGO']], ['Pacto\_das\_Guildas', ['Pacto\_das\_Guildas', 'Pacto\_das\_Guildas']], ['Pacto\_das\_Guildas', ['Pacto\_das\_Guildas', 'Pacto\_das\_Guildas']], ['PauperFeito', ['NA', 'NA', 'PauperFeito']], ['PauperFeito', ['NA', 'NA', 'PauperFeito']], ['Pazmaster', ['-Sire', '-Sire']], ['Pazmaster', ['-Sire', '-Sire']], ['PieSmug', ['-Sire', '-Sire']], ['Piergi', ['Piergi', '-Sire', 'Piergi']], ['Piergi', ['Piergi', '-Sire', 'Piergi']], ['ProfPidu', ['-Sire', '-Sire']], ['ProfPidu', ['ProfPidu', 'ProfPidu']], ['ResponsiblyStupid', ['ResponsiblyStupid', 'ResponsiblyStupid']], ['ResponsiblyStupid', ['ResponsiblyStupid', 'ResponsiblyStupid']], ['Uchoa', ['U', '-Sire']], ['Uchoa', ['U', '-Sire']], ['VOIDx', ['-Sire', 'NA', '-Sire']], ['VOIDx', ['-Sire', 'NA', '-Sire']], ['anduyn', ['anduyn', '-Sire', 'anduyn']], ['anduyn', ['anduyn', '-Sire', 'anduyn']], ['bergamo83', ['-Sire', 'bergamo83', '-Sire']], ['bergamo83', ['-Sire', 'bergamo83', '-Sire']], ['callowaysmith1', ['-Sire', 'callowaysmith1', '-Sire']], ['callowaysmith1', ['-Sire', 'callowaysmith1', '-Sire']], ['cbouz', ['-Sire', 'cbouz', 'NA']], ['cbouz', ['-Sire', 'cbouz', 'NA']], ['hopeof130', ['-Sire', '-Sire']], ['hopeof130', ['-Sire', '-Sire']], ['lindoso01', ['lindoso01', 'lindoso01']], ['lindoso01', ['lindoso01']], ['lindoso01', ['lindoso01', 'lindoso01']], ['lindoso01', ['lindoso01']], ['modo\_baggins', ['-Sire', 'NA', 'NA']], ['modo\_baggins', ['-Sire', 'NA', 'NA']], ['morvan09', ['NA', 'morvan09', '-Sire']], ['rEPTILIUM\_', ['rEPTILIUM\_', 'rEPTILIUM\_']], ['rEPTILIUM\_', ['rEPTILIUM\_', 'rEPTILIUM\_']], ['sc00nes', ['sc00nes', '-Sire', 'sc00nes']], ['sc00nes', ['sc00nes', '-Sire', 'sc00nes']], ['sosfabio89', ['-Sire', '-Sire']], ['sosfabio89', ['-Sire', '-Sire']]] |

#### renderer.js

All arrays of records and 2D arrays in renderer.js are from dbCMD.py, so they are shared.

## UML Class Design

### mtgScraper

Text

Description automatically generated

### matchRecord

Text

Description automatically generated

### mtgtop8

Graphical user interface, text

Description automatically generated

## File Flow

Diagram

Description automatically generated

## User-Interface Design

### Main Page

Table

Description automatically generated

* Top right hand corner image is the profile image (see Main Page Profile Button Image, UI Visuals), and when clicked on, take the user to the profile page.
* When the Sync button is clicked, renderer.js, runs sync.py
* Table will have results from renderer.js calling dbCMD.py, to get database results

### Profile Page

Graphical user interface, text, application

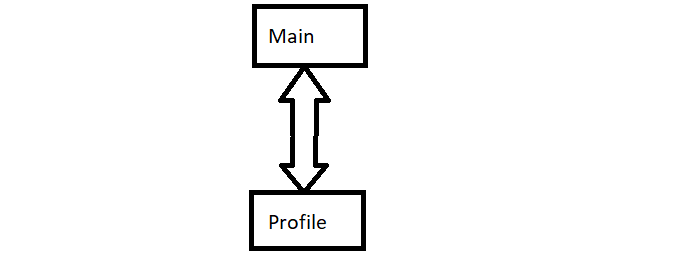
Description automatically generated

* Close Button returns page to Main Page
* Submit button sends the Get specific winrate against opponent query (see query design), where opponent is equal to the contents of the textbox.

## Entity Relationship Diagram

Since there is only one database table, there are no entity relationships.

## Hierarchical Design of App Structure



# Implementation

Some of the implementation does not match the pseudocode design due to complications being discovered late in the development.

## Python Files

### sync.py

import os

from os import walk

from mtgScraper import Scraper

#gets path of mtgo

localUser = os.getlogin()

path = f'C:\\Users\\{localUser}\\AppData\\Local\\Apps\\2.0\\Data\\'

#gets next two folders and adds them to path

for i in range(0,2):

    folder = os.listdir(path)

    path = path + '\\' + folder[0]

#gets the main mtgo folders, they are prefixed by 'mtgo..tion\_'

mainFolders = os.listdir(path)

scp = Scraper()

for folder in mainFolders:

    folderPath = f'{path}\\{folder}\\Data\\AppFiles'

    #gets name of the user

    scp.getPlayer(f'{folderPath}\\application\_settings')

    #gets the directories of the appfiles folder

    for (folderPath, directory, fileNames) in walk(folderPath):

        #.vs file may be in one of the directories, and is unnecessary

        if len(directory) > 1:

            directory.remove('.vs')

        #gets path of match files

        folderPath = folderPath + '\\' + directory[0]

        #don't look inside any subdirectory

        break

    #adds folder to list of folders to check

    scp.addPath(folderPath)

scp.run()

### mtgScraper.py

from matchRecord import MatchRecord

from mtgtop8 import DriverController

import os.path

from datetime import datetime

import urllib.request

import sqlite3

import json

import re

class Scraper():

    def \_\_init\_\_(self):

        self.paths = []

        self.player = ''

    def getPlayer(self, filename):

        try:

            with open('./playerName.txt', 'rb') as f:

                self.player = f.read().decode(encoding='utf-8', errors='replace')

        except:

            #reads file contents

            with open(filename, 'rb') as f:

                file = f.read().decode(encoding='utf-8', errors='replace').replace(' ', '').splitlines()

            #gets player name from file contents

            self.player = re.findall('"(.\*)"', file[file.index('Setting="LastLoginName"')+1])[0]

    def addPath(self, path):

        #adds path to object

        self.paths.append(path)

    def run(self):

        #opens connection

        self.openConn()

        #checks if database exists

        self.checkDB()

        top8Conn = self.checkInternet()

        #initilises modules

        match = MatchRecord()

        if top8Conn == True:

            dc = DriverController()

        #gets list of files in database

        self.cursor.execute("SELECT filename FROM matches;")

        filenames = self.cursor.fetchall()

        filenames = [json.loads(i[0])['filename'] for i in filenames]

        for path in self.paths:

            #gets list of files in path

            fileList = [f for f in os.listdir(path) if f.endswith('.dat')]

            #loops through file list

            for filename in fileList:

                #if file already scraped

                if filename in filenames:

                    break

                #gets decklists from MatchRecord

                #other stored data is implemented into database in MatchRecord

                decklists, extra, players = match.getDecklists(f'{path}/{filename}')

                #if the file is valid

                if decklists != {} and decklists is not None:

                    players.remove(self.player)

                    #gets and reformats date

                    dateTime = str(datetime.fromtimestamp(os.path.getmtime(path + '\\' + filename)))

                    x, y, z = dateTime.split(' ')[0].split('-')

                    date = {'date':f'{z}/{y}/{x}'}

                    if top8Conn == True:

                        #gets the possible deck names from DriverController

                        deckName, matchLists = dc.returnDeckName(decklists, date)

                    else:

                        deckName = {'NA': 1.0}

                    #sends info to sqliite db

                    print(extra)

                    print(matchLists)

                    self.sqlliteDriverData({'filename':filename}, dateTime, deckName, extra, {'players':players}, matchLists)

        #closes webdriver

        if top8Conn == True:

            dc.quitDc()

        #close connection

        self.closeConn()

    def openConn(self):

        self.userConnection = sqlite3.connect("./database/mtgoAssist.db")

        self.cursor = self.userConnection.cursor()

    def checkInternet(self):

        #tries to open mtgtop8.com, if it can't then program will not open later

        url = 'https://mtgtop8.com/'

        try:

            urllib.request.urlopen(url)

            conInt = True

        except:

            conInt = False

        return conInt

    def checkDB(self):

        try:

            self.cursor.execute("SELECT MAX(matchID) FROM matches;")

        except:

            #creates the matches table

            self.cursor.execute("""CREATE TABLE matches(

                                matchID INTEGER PRIMARY KEY,

                                filename BLOB NOT NULL,

                                opponent BLOB NOT NULL,

                                decknames BLOB,

                                decklistP1 BLOB,

                                decklistP2 BLOB,

                                play BLOB NOT NULL,

                                winner BLOB NOT NULL,

                                date BLOB NOT NULL);""")

            self.userConnection.commit()

    def sqlliteDriverData(self, filename, dateTime, deckName, extra, players, matchLists):

        #inserts match into database

        data = (json.dumps(filename), json.dumps(players), json.dumps(deckName), json.dumps({'P1':matchLists['P1']}), json.dumps({'P2':matchLists['P2']}), json.dumps({'play':extra['play']}), json.dumps({'winner':extra['winner']}), json.dumps({'date':dateTime}))

        self.cursor.execute("INSERT INTO matches(filename, opponent, decknames, decklistP1, decklistP2, play, winner, date) VALUES(?,?,?,?,?,?,?,?);", data)

        self.userConnection.commit()

    def closeConn(self):

        self.userConnection.commit()

        self.cursor.close()

        self.userConnection.close()

### matchRecord.py

import re

class MatchRecord:

    def \_\_init\_\_(self):

        pass

    def getDecklists(self, filename):

        decklists = {}

        extra = {'play':[], 'startingHands':[], 'winner':[]}

        #gets matchLog

        with open(filename, 'rb') as f:

            #unknown characters are replaced by \ufffd (those question marks)

            self.matchLog = f.read().decode(encoding='utf-8', errors='replace')

        #tries to create self.players

        try:

            self.getPlayers()

            if self.players == []:

                return None, None, None

        except (IndexError, ValueError):

            #if there is a problem reading the player names or the match is not 1v1 (fie invalid)

            return None, None, None

        #formats self.matchLog

        self.formatLines()

        #loops through each game

        for gameNo in range(1,len(self.matchLog)):

            #gets decklists from game

            gameDecklists = self.getDeckLists(self.matchLog[gameNo])

            if gameDecklists is None:

                break

            decklists[gameNo] = gameDecklists

            #gets player on play

            try:

                extra['play'].append(self.getOnPlay(self.matchLog[gameNo]))

            except:

                extra['play'].append('NA')

            #get number of cards in each player's hand

            try:

                extra['startingHands'].append(self.getStartingHands(self.matchLog[gameNo]))

            except:

                extra['startingHands'].append('NA')

            #gets winner of game

            extra['winner'].append(self.getWinner(self.matchLog[gameNo]))

        #if only two matches were played, then the person who was on the draw game 2, won the match

        if 'NA' in extra['winner'] and len(extra['play']) == 2:

            #gets player on draw in game 2

            player = [i for i in self.players if extra['play'][1] != i]

            #loops through each 'NA' result, and replaces it with the winner

            for indice in [i for i, elem in enumerate(extra['winner']) if elem == 'NA']:

                extra['winner'][indice] = player[0]

        #if 1 or fewer winners were recorded, and only two games were played, the player in the draw in game 2 won the match

        if len(extra['winner']) < 2 and len(extra['play']) == 2:

            #gets player on draw in game 2

            player = [i for i in self.players if extra['play'][1] != i]

            #while there are less than 2 recorded winners

            while len(extra['winner']) < 2:

                extra['winner'].append(player[0])

        return decklists, extra, self.players

    def getPlayers(self):

        #finds all players

        #the first player in the game log is always the user, or the second player in this list

        self.players = list(re.compile('@P(\S+) rolled').findall(self.matchLog))

        #if more than two players are detected, then the dice rolls where equal at least once in the game opening

        if len(self.players) > 2:

            while len(self.players) > 2:

                self.players.pop()

    def formatLines(self):

        #formats self.matchLog in [['random text', 'random text', 'player rolled', 'player rolled'], [game 1], [game 2...]] and removes noise

        #if there is a full stop in a player's name, then it must obtain to prefix, to prevent it being removed

        altPlayers = {}

        for player in self.players:

            if '.' in player:

                altPlayers[player] = []

                prefix = player.split('.')

                for count, pre in enumerate(prefix):

                    if count % 2 == 0:

                        altPlayers[player].append(pre)

        #removes all non-ascii characters and some non-random characters

        filteredMatch = re.split(r'[**\x00**-**\x08\x0b\x0c\x0e**-**\x1f\x7f**-**\xff**\ufffd**\{\}\|\\**=#**\^**><$]', self.matchLog)

        filteredMatch = [f'{line}.' for line in filteredMatch if len(line) > 4]

        #removes all full stops

        if len(altPlayers.keys())> 0:

            for player in altPlayers.keys():

                for pre in altPlayers[player]:

                    filteredMatch = [i for i in ''.join(filteredMatch).replace(' ',' {').split('{')]

                    filteredMatch = [i.replace('.', '}') if pre not in i and '.' in i else i for i in filteredMatch]

                    filteredMatch = ''.join(filteredMatch).split('}')

                    filteredMatch = [re.sub('^.\*@P', '@P', line) for line in filteredMatch if len(line) > 4]

        else:

            filteredMatch = ''.join(filteredMatch).split('.')

            filteredMatch = [re.sub('^.\*@P', '@P', line) for line in filteredMatch if len(line) > 4]

        #splits by game

        txt = ' '.join(filteredMatch)

        regex = f'@P{re.escape(self.players[1])} joined the game @P{re.escape(self.players[0])} joined the game|@P{re.escape(self.players[0])} joined the game @P{re.escape(self.players[1])} joined the game'

        filteredMatch = re.compile(regex).split(txt)

        #splits by turn

        regex = f'Turn [1-9]: {re.escape(self.players[0])}|Turn [1-5][0-9]: {re.escape(self.players[0])}|Turn [1-5][0-9]: {re.escape(self.players[1])}|Turn [1-9]: {re.escape(self.players[1])}'

        filteredMatch = [re.compile(regex).split(turn) for turn in filteredMatch]

        #removes last random characters

        filteredMatch = [[re.split('(@P)', turn) for turn in game] for game in filteredMatch]

        filteredMatch = {gameNo:{turnNo:[re.sub(r'@P|[^**\x00**-**\x7F**]', '', line) for line in turn if len(line) > 4] for turnNo, turn in enumerate(game)} for gameNo, game in enumerate(filteredMatch)}

        #deletes random characters at start of match

        del filteredMatch[0][0]

        self.matchLog = filteredMatch

    def getDeckLists(self, game):

        decklists = {self.players[0]: dict(), self.players[1]: dict()}

        #stores cards each player has played, revealed, discarded, cycled

        #game actions are formatted as @P(player\_name) (casts|plays|discards|cycles|reveals)

        #card names are formatted as @[Card Name@:numbers,numbers:@]

        playCardPattern = re.compile(f'({re.escape(self.players[0])}|{re.escape(self.players[1])}) (casts|plays|discards|cycles) (@\[([a-zA-Z\s,-]+)@:[0-9,]+:@\])')

        revealedCardPattern = re.compile(f'({re.escape(self.players[0])}|{re.escape(self.players[1])}) (reveals) (@\[([a-zA-Z\s,-]+)@:[0-9,]+:@\])')

        for turn in range(0, len(game.keys())-1):

            #finds matched patterns

            playCardMatches = playCardPattern.findall(' '.join(game[turn]))

            revealedMatches = revealedCardPattern.findall(' '.join(game[turn]))

            for actions in playCardMatches:

                #if a card has been revealed, and has interacted with the game, remove it from revealedMatches

                for match in revealedMatches:

                    if actions[3] in match:

                        revealedMatches.remove(match)

                #adds card to decklists

                if actions[3] in decklists[actions[0]]:

                    decklists[actions[0]][actions[3]] += 1

                else:

                    decklists[actions[0]].update({actions[3]:1})

            for revealed in revealedMatches:

                #adds card to decklists

                if revealed[3] in decklists[revealed[0]]:

                    decklists[revealed[0]][revealed[3]] += 1

                else:

                    decklists[revealed[0]].update({revealed[3]:1})

        return decklists

    def getOnPlay(self, game):

        # Who is on the play in this game?

        # Returns 'player' or 'opponent'

        text = ' '.join(game[0])

        onPlay = re.compile(f'({re.escape(self.players[0])}|{re.escape(self.players[1])})\ chooses\ to\ play\ first\ ').search(text).group(1)

        return onPlay

    def getStartingHands(self, game):

        #format [(player1, cards), (player2, cards)]

        regex = re.compile(f'({re.escape(self.players[0])}|{re.escape(self.players[1])}) begins the game with (no|a|two|three|four|five|six|seven) (?:card|cards) in hand | ({re.escape(self.players[0])}|{re.escape(self.players[1])}) puts (?:a|two|three|four|five|six|seven) (?:card|cards) on the bottom of their library and begins the game with (no|a|two|three|four|five|six|seven) cards in hand')

        startingHands = re.findall(regex, ' '.join(game[0]))

        startingHands = [[b for b in i if len(b) > 1] for i in startingHands]

        result = {i[0]:i[1] for i in startingHands}

        return startingHands

    def getWinner(self, game):

        #determines the winner or loser

        concededPattern = re.compile(f'({re.escape(self.players[0])}|{re.escape(self.players[1])}) has conceded')

        winsPattern = re.compile(f'({re.escape(self.players[0])}|{re.escape(self.players[1])}) wins the game')

        losesPattern = re.compile(f'({re.escape(self.players[0])}|{re.escape(self.players[1])}) loses the game')

        conceded = concededPattern.search(' '.join(game[len(game)-1]))

        wins = winsPattern.search(' '.join(game[len(game)-1]))

        loses = losesPattern.search(' '.join(game[len(game)-1]))

        if wins:

            return self.players[self.players.index(wins.group(1))][0]

        elif conceded:

            return [i for i in self.players if i not in self.players[self.players.index(conceded.group(1))]][0]

        elif loses:

            return self.players[self.players.index(loses.group(1))][0]

        return 'NA'

    def getWinner(self, game):

        #determines the winner or loser

        concededPattern = re.compile('(\S+) has conceded')

        winsPattern = re.compile('(\S+) wins the game')

        losesPattern = re.compile('(\S+) loses the game')

        conceded = concededPattern.search(' '.join([' '.join(i) for i in game]))

        wins = winsPattern.search(' '.join([' '.join(i) for i in game]))

        loses = losesPattern.search(' '.join([' '.join(i) for i in game]))

        if wins:

            return self.players[self.players.index(wins.group(1))][0]

        elif conceded:

            return [i for i in self.players if i not in self.players[self.players.index(conceded.group(1))]][0]

        elif loses:

            return self.players[self.players.index(loses.group(1))][0]

        else:

            return 'NA'

mtgTop8.py

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

from selenium.webdriver.support import expected\_conditions as EC

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.support.select import Select

from selenium.webdriver.common.keys import Keys

class DriverController():

    def \_\_init\_\_(self):

        self.url = "https://mtgtop8.com/search"

        #adds options to the webdriver, in this case, to let webpage load, and bypass rate limiting

        driverOptions = webdriver.ChromeOptions()

        driverOptions.add\_argument('--ignore-certificate-errors')

        driverOptions.add\_argument('--ignore-ssl-errors')

        driverOptions.add\_argument('--start-maximized')

        driverOptions.headless = True

        #check validity of comment

        #downloads and runs a webdriver, that stops and is uninstalled after the program exits, using the options declared above

        chrome\_path = ChromeDriverManager().install()

        chrome\_service = Service(chrome\_path)

        self.driver = webdriver.Chrome(options=driverOptions, service=chrome\_service)

    def returnDeckName(self, deckLists, date):

        #calls the 'getSite()' method to open the url through the driver

        self.getSite()

        #calls the 'cookieBanner()' method to try and clear the cookie banner

        self.clearCookieBanner()

        deckLists = self.refactorDecklist(deckLists)

        #calls the 'inputFormData()' method to get all decks to be scraped

        self.inputFormData(deckLists['P1'][0], date)

        #gets the deck urls and names from the 'getDeckUrls()' method

        deckNames = self.getDeckNames()

        #if there are no decks found, then return deckNames (will have value of 'unknown')

        if deckNames == 'unknown':

            deckNames = {'NA', 1.0}

        #calls getDeckName() to create a dictionary of deckNames and %

        dictNames = self.getDictNames(deckNames)

        return dictNames, deckLists

    def getSite(self):

        self.driver.get(self.url)

    def inputFormData(self, deckList, date):

        #includes cards found in games played so far, as the what cards were sideboarded or not is impossible to find out

        self.driver.find\_element(By.XPATH, '//input[@name="SB\_check"]').click()

        #loops through the list of cards, writing each card into the <textarea>

        textarea = self.driver.find\_element(By.XPATH, '//textarea[@name="cards"]')

        #writes the cards to the textbox

        for card in deckList:

            textarea.send\_keys(card + Keys.RETURN)

        #sets date, so deck possibilites from after that game aren't considered (as they don't exist at the time of playing)

        self.driver.find\_element(By.XPATH, '//input[@name="date\_end"]').send\_keys(date['date'])

        #clicks the submit button of the form

        self.driver.find\_element(By.XPATH, '//td[@colspan="2"]/input[@type="submit"]').click()

    def clearCookieBanner(self):

        #if cookie banner present, click it, as it obstructs webdriver's view

        try:

            self.driver.find\_element(By.XPATH, '//\*[@id="cookie\_window"]/div[2]/button').click()

        except:

            pass

    def getDeckNames(self):

        deckNames = []

        #gets number of decks on page

        decks = len(self.driver.find\_elements(By.XPATH, '//td[@class="S12"]'))/3

        #if there are no decks on page, then return unknown as deckname, and quit DriverController()

        if decks < 1:

            return 'unknown'

        #sets number of decks to 5, if not under 5, so as to reduce time taken

        if decks > 5:

            decks = 5

        #loops through all decks on the page and gets their and name

        for deck in range(2, int(decks)+2):

            deckNames.append(self.driver.find\_element(By.XPATH, f'//body/div/div/table/tbody/tr/td[2]/form/table/tbody/tr[{deck}]/td[2]/a').text)

        return deckNames

    def quitDc(self):

        #stops the driver

        self.driver.quit()

    def refactorDecklist(self, deckLists):

        #reformats decklists

        cards = {}

        for game in deckLists:

            for i, player in enumerate(deckLists[game]):

                if i == 0:

                    tempPlayer = 'P2'

                else:

                    tempPlayer = 'P1'

                cards[tempPlayer] = [{}]

                for card in deckLists[game][player]:

                    cards[tempPlayer][0].update({f'{card}':deckLists[game][player][card]})

        return cards

    def getDictNames(self, deckNames):

        #creates a dictionary of deckNames

        dictNames = {'deckNames':[{deckName:0 for deckName in deckNames}]}

        #gets number of decks with same name

        for deckName in deckNames:

            dictNames['deckNames'][0][deckName] = dictNames['deckNames'][0][deckName] + 1

        #gets percentage chance of said deck

        for deckName in dictNames['deckNames'][0]:

            dictNames['deckNames'][0][deckName] = dictNames['deckNames'][0][deckName] / len(deckNames)

        return dictNames

### dbCMD.py

import sqlite3

import sys

import json

#gets data from database to be displayed on table

def openData():

    userConnection = sqlite3.connect("./database/mtgoAssist.db")

    cursor = userConnection.cursor()

    try:

        matches = cursor.execute("SELECT \* FROM matches ORDER BY matchID DESC;").fetchall()

        results = {}

        for matchTuple in matches:

            results[matchTuple[0]] = [{}]

            for data in matchTuple:

                if isinstance(data, int) == False:

                    data = json.loads(data)

                    results[matchTuple[0]][0][list(data.keys())[0]] = data[list(data.keys())[0]]

        print(json.dumps(results))

    except:

        pass

    finally:

        cursor.close()

        userConnection.close()

#gets user averages

def getUserData():

    try:

        with open('python\playerName.txt', 'rb') as f:

            player = f.read().decode(encoding='utf-8', errors='replace')

    except:

        print('NA')

        return

    userConnection = sqlite3.connect("./database/mtgoAssist.db")

    cursor = userConnection.cursor()

    matches = cursor.execute("SELECT winner FROM matches;").fetchall()

    results = {}

    for num, match in enumerate(matches):

        results[num] = json.loads(match[0])

    winList = []

    count = 0

    for match in results:

        for res in results[match]['winner']:

            if res != 'NA':

                if res == player:

                    count += 1

                winList.append(res)

    print(count / len(winList))

    cursor.close()

    userConnection.close()

def getOppWinrate(opponent):

    try:

        with open('python\playerName.txt', 'rb') as f:

            player = f.read().decode(encoding='utf-8', errors='replace')

    except:

        print('NA')

        return

    userConnection = sqlite3.connect("./database/mtgoAssist.db")

    cursor = userConnection.cursor()

    matches = cursor.execute("SELECT opponent, winner FROM matches ORDER BY opponent;").fetchall()

    #[[opponent, [name, name...]],[]]

    results = []

    for num, match in enumerate(matches):

        results.append([])

        for nom, item in enumerate(match):

            if nom == 0:

                results[num].append(json.loads(item)['players'][0])

            else:

                results[num].append(json.loads(item)['winner'])

            #results[num] = match[0]

    record = []

    while True:

        low = 0

        high = len(results) - 1

        mid = 0

        while low <= high:

            mid = (high + low) // 2

            if results[mid][0] < opponent:

                low = mid + 1

            elif results[mid][0] > opponent:

                high = mid - 1

            else:

                for win in results[mid][1]:

                    record.append(win)

                results.pop(mid)

        #element not present

        break

    count = 0

    for play in record:

        if play == player:

            count += 1

    if count != 0:

        print(count / len(record))

    else:

        print(count)

if \_\_name\_\_ == "main":

    pass

else:

    if sys.argv[1] == 'loaded':

        openData()

    elif sys.argv[1] == 'profile':

        getUserData()

    elif sys.argv[1] =='opponent':

        getOppWinrate(sys.argv[2])

## Javascript Files

### renderer.js

function runPythonSync(){

  var pyshell =  require('python-shell');

  var pjson = require('./package.json');

  let options = {

    mode: 'text',

    pythonOptions: ['-u'],

    args: [pjson.version]

  }

  pyshell.PythonShell.run('./python/sync.py', options, function  (err, results)  {

    if (err)  throw err;

    console.log('sync.py finished.');

  });

}

function loadFirstPage(){

  var pyshell =  require('python-shell');

  var pjson = require('./package.json');

  let options = {

    mode: 'json',

    pythonOptions: ['-u'],

    args: ["loaded", pjson.version]

  }

  pyshell.PythonShell.run('./python/dbCMD.py', options, function  (err, results)  {

    console.log(results);

    let keys = Object.keys(results[0])

    if (results){

      let leng = Object.keys(results[0]).length;

      for (let i = 0; i < leng; i++){

        let append = `<tr>

                <th class="matchID" id="row${i}">${Object.keys(results[0])[i]}</th>

                <th class="filename"><p class="hidden">${results[0][keys[leng-i-1]][0]['filename']}</p><button class="replaceImageFilename">...</button></th>

                <th class="players"><p class="replaceTextOpponent">${results[0][keys[leng-i-1]][0]['players'][0]}</p><button class="replaceImageOpponent">...</button></th>

                <th class="decknames"><p class="replaceText">${JSON.stringify(results[0][keys[leng-i-1]][0]['deckNames'][0]).replace(/[[**\]**]|"|{|}/g, '').replace(/:/g, ': ').replace(/,/g, '<br>')}</p><button class="replaceImage">...</button></th>

                <th class="p1Deck"><p class="replaceTextDeck">${structureDecklist(results, keys, leng, i, 'P1')}</p><button class="replaceImageDeck">...</button></th>

                <th class="p2Deck"><p class="replaceTextDeck">${structureDecklist(results, keys, leng, i, 'P2')}</p><button class="replaceImageDeck">...</button></th>

                <th class="turnOrder"><p class="replaceTextExtra">${JSON.stringify(results[0][keys[leng-i-1]][0]['play']).replace(',','<br>').replace(/[[**\]**]|"/g, '')}</p><button class="replaceImageExtra">...</button></th>

                <th class="winList"><p class="replaceTextExtra">${JSON.stringify(results[0][keys[leng-i-1]][0]['winner']).replace(',','<br>').replace(/[[**\]**]|"/g, '')}</p><button class="replaceImageExtra">...</button></th>

                <th class="date">${JSON.stringify(results[0][keys[leng-i-1]][0]['date']).replace('"','').split(' ')[0]}</th>

              </tr>`

        document.getElementById('databaseContent').insertAdjacentHTML('beforeend', append);

      }

    }

    if(err) throw err;

  });

}

function structureDecklist(results, keys, leng, i, player){

  let deck = JSON.stringify(results[0][keys[leng-i-1]][0][player][0]).replace(/[[**\]**]|"|{|}/g, '').split(/:|,/g)

  let res = ''

  for (let b = 0; b < deck.length; b++){

    if (b % 2 == 0){

      res += deck[b+1] + ' ' + deck[b] + '<br>'

    }

  }

  return res

}

function loadProfile(){

  document.getElementById('mid').style.display = 'none';

  document.getElementById('bottom').style.display = 'none';

  document.getElementById('profile').style.display = 'block';

  var pyshell =  require('python-shell');

  var pjson = require('./package.json');

  let options = {

    mode: 'text',

    pythonOptions: ['-u'],

    args: ["profile", pjson.version]

  }

  pyshell.PythonShell.run('./python/dbCMD.py', options, function  (err, results)  {

    console.log(results);

    document.getElementById('winrate').innerHTML = results;

    if(err) throw err;

  });

}

function getOpponentWinrate(){

  let name = document.getElementById('oppName').value;

  var pyshell =  require('python-shell');

  var pjson = require('./package.json');

  let options = {

    mode: 'text',

    pythonOptions: ['-u'],

    args: ["opponent", name, pjson.version]

  }

  pyshell.PythonShell.run('./python/dbCMD.py', options, function  (err, results)  {

    console.log(results);

    document.getElementById('opponentWinrate').innerHTML = results;

    if(err) throw err;

  });

}

function closeProfile(){

  document.getElementById('mid').style.display = 'block';

  document.getElementById('bottom').style.display = 'block';

  document.getElementById('profile').style.display = 'none';

}

document.querySelector('#pyBtnProfile').addEventListener('click', () => {

  loadProfile();

})

document.querySelector('#close').addEventListener('click', () => {

  closeProfile();

})

document.querySelector('#pyBtnSync').addEventListener('click', () => {

  runPythonSync();

})

document.addEventListener('DOMContentLoaded', () =>{

  loadFirstPage();

})

document.querySelector('#submitUser').addEventListener('click', () => {

  getOpponentWinrate();

})

### preload.js

//loads content before app can be seen, so user doesn’t watch the content loading

window.addEventListener('DOMContentLoaded', () => {

    const replaceText = (selector, text) => {

      const element = document.getElementById(selector)

      if (element) element.innerText = text

    }

    for (const type of ['chrome', 'node', 'electron']) {

      replaceText(`${type}-version`, process.versions[type])

    }

})

### index.js

//creates window and hosts webpage from there

const { app, BrowserWindow } = require('electron')

const path = require('path')

const createWindow = () => {

    const win = new BrowserWindow({

      width: 800,

      height: 600,

      webPreferences: {

        nodeIntegration: true,

        contextIsolation: false,

        preload: path.join(\_\_dirname, 'preload.js'),

      }

    })

    win.loadFile('index.html')

  }

app.whenReady().then(() => {

    createWindow()

})

app.on('window-all-closed', () => {

    if (process.platform !== 'darwin') app.quit()

})

## UI Files

### index.html

<html>

  <head>

    <title>MTGO Assist</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="stylesheet" href="./css/index/bottom.css">

    <link rel="stylesheet" href="./css/index/top.css">

    <link rel="stylesheet" href="./css/index/mid.css">

    <link rel="stylesheet" href="./css/index/main.css">

  </head>

  <body>

    <section id="top">

        <section id="bar">

            <section id="topbar">

                <ul class="listBar">

                    <li class="listBarItem">Hello</li>

                </ul>

            </section>

            <section id="navbar">

                <ul class="listBar">

                    <li class="listBarItemNav" id="profileListItem"><button id="pyBtnProfile"><img src="./profile.png" id="profileImage"></button></li>

                    <li class="listBarItemNav"><button id="pyBtnSync">Sync</button></li>

                </ul>

            </section>

        </section>

    </section>

    <section id="mid">

        <section id="databaseSection">

                <table>

                    <tbody id="databaseContent">

                        <tr id="colNames">

                            <th class="tableHeader"><label id="matchID"><label id="matchText">Match </label>ID</label></th>

                            <th class="tableHeader"><label>File<label id="fileText">name</label></label></th>

                            <th class="tableHeader"><label>Opponent</label></th>

                            <th class="tableHeader"><label>Deck<label class="extraHeaderText">name</label>s</label></th>

                            <th class="tableHeader"><label class="altHeaderText">P1</label><label class="extraHeaderText">Player 1 Decklist</label></th>

                            <th class="tableHeader"><label class="altHeaderText">P2</label><label class="extraHeaderText">Player 2 Decklist</label></th>

                            <th class="tableHeader"><label class="extraHeaderTextExtra">Turn </label><label>Order</label></th>

                            <th class="tableHeader"><label id="altHeaderTextExtra">Wins</label><label class="extraHeaderTextExtra">Win List</label></th>

                            <th class="tableHeader"><label>Date</label></th>

                        </tr>

                    </tbody>

                </table>

        </section>

    </section>

    <section id="bottom">

        <div id="nextPage">

                <button id="pyBtnNextPage">Next Page</button>

        </section>

    </section>

    <section id="profile" style="display:none;">

        <div id="results">

            <p>Winrate: <label id="winrate"></label>%</p><br>

            <p>Opponent Username:

                <input type="text" id="oppName">

                <button id="submitUser">Submit</button>

            </p>

            <p>Winrate against opponent: <label id="opponentWinrate"></label>%</p><br><br>

        </div>

        <button id="close">Close</button>

    </section>

    <script>

      require('./renderer.js')

      require('./preload.js')

    </script>

  </body>

</html>

### bottom.css

#nextPage {

    display: inline-block;

    position: absolute;

    width: 99%;

    text-align: center;}

#pyBtnNextPage {

    color:black;}

### mid.css

table,td,th{

    border:1px solid black;}

table{

    border-collapse: collapse;}

#databaseSection {

    color:black;}

.tableHeader {

    padding-left:1.3vw;

    padding-right:1.3vw;

    white-space: nowrap;

    max-width:200px;}

.matchID, .filename, .players, .decknames, .p1Deck, .p2Deck, .turnOrder, .winList, .format, .type, .date{

    max-width:200px;

    word-wrap:break-word;}

.filename{

    max-width:100%;

    white-space:nowrap;}

.replaceImage, .altHeaderText, #altHeaderTextExtra, .replaceImageOpponent, .replaceImageDeck, .replaceImageExtra, .hidden{

    display: none;}

.decknames{

    white-space: nowrap;}

#matchText, #fileText, .extraHeaderText, .replaceText, .replaceTextDeck{

    display:inline-block;}

.winRecord{

    background-color: #59cf78;}

@media only screen and (max-width:1600px){

    #matchtext, #fileText{

        display:none;}}

@media only screen and (max-width:1480px){

    .extraHeaderTextExtra, .replaceTextExtra{

        display:none;}

    #altHeaderTextExtra, .replaceImageExtra{

        display:inline-block;}}

@media only screen and (max-width:560px){

    #matchID, .matchID, tr th:nth-child(1){

        display:none;}}

@media only screen and (max-width:1280px){

    .replaceTextOpponent{

        display:none;}

    .replaceImageOpponent{

        display:inline-block;}}

@media only screen and (max-width:1240px){

    .replaceTextDeck{

        display:none;}

    .replaceImageDeck{

        display:inline-block;}}

@media only screen and (max-width:1090px){

    .tableHeader, .matchID, .players, .decknames, .p1Deck, .p2Deck, .turnOrder, .winList, .format, .type, .date{

        font-size:14px;}}

@media only screen and (max-width:960px){

    .tableHeader, .matchID, .players, .decknames, .p1Deck, .p2Deck, .turnOrder, .winList, .format, .type, .date{

        font-size:12px;}}

@media only screen and (max-width:830px){

    #type, .type, tr th:nth-child(10){

        display:none;}}

@media only screen and (max-width:750px){

    .tableHeader, .matchID, .players, .decknames, .p1Deck, .p2Deck, .turnOrder, .winList, .format, .type, .date{

        font-size:12px;}

    .tableHeader {

        padding-left:1.3vw;

        padding-right:1.3vw;

        white-space: nowrap;

        max-width:200px;}

    .matchID, .players, .decknames, .p1Deck, .p2Deck, .turnOrder, .winList, .format, .type, .date {

        word-wrap:break-word;}

    .extraHeaderText, .replaceText{

        display:none;}

    .altHeaderText, .replaceImage{

        display: inline-block;}}

### top.css

/\* #top, body{width:1000%} \*/

#bar{

    height: 60px;

    width:auto;

    background-color: rgb(134, 134, 134);}

#topbar, #navbar {

    height: 60px;}

#topbar {

    width:30%;

    float:left;}

.listBar {

    list-style-type: none;

    padding: 0px;

    margin: auto;

    height:100%;}

.listBarItem, .listBarItemNav {

    text-align: center;

    vertical-align: middle;

    padding:20px 30px;}

.listBarItem {

    float: left;

    font-size:1.4vw}

.listBarItem a {

    display: block;

    color: rgb(0, 0, 0);

    text-align: center;

    padding: 14px 16px;

    text-decoration: none;}

.listBarItemNav {

    float: right;}

#navbar {

    width:70%;

    float:right;

    background-color: rgb(134, 134, 134);}

#pyBtnSync {

    color:black;}

#profileImage {

    width: 40px;

    height: auto;}

#profileListItem {

    padding-top:7px;

    padding-left: 30px;}

#colNames {

    width:100%;}

#lowHeader {

    color:black;}

@media only screen and (max-width:1250px){

    .listBarItem {

        float: left;

        font-size:16px;}

}

### main.css

h1 {

    color:black};

## UI Visuals

### Main Page

Graphical user interface, application

Description automatically generated

### Main Page with Content

Graphical user interface

Description automatically generated with medium confidence

### Profile

Graphical user interface, application, Word

Description automatically generated

### Profile Search Opponent

Graphical user interface, application, Word

Description automatically generated

### Main Page Profile Button Image

Icon

Description automatically generated

## Error Log

|  |  |  |
| --- | --- | --- |
| Problem | Why | Fix |
| Didn’t know how to run a python file from electron. | NA | var pyshell = require('python-shell');  var pjson = require('./package.json');  let options = {  mode: 'text',  pythonOptions: ['-u'],  args: [pjson.version]  }  pyshell.PythonShell.run('./python/sync.py', options, function (err, results) {INSERT CODE HERE}); |
| mtgtop8.py inefficiency | Started and stopped mtgtop8.py each time it was called | Instead opened it at the start of mtgScraper and close it at the end of mtgScraper.py |
| The Driver Controller (mtgtop8.py) broke when trying to scrape too many times | Emulating the webpage (showing it on screen) causes lag, which disrupts the process. | Added argument:  driverOptions.headless = True |
| <https://www.mtgtop8.com/> would not load in the Driver Controller | It had an invalid security ticket. | Added arguments:  driverOptions.add\_argument('--ignore-certificate-errors')  driverOptions.add\_argument('--ignore-ssl-errors') |
| Driver failed to click on certain elements on the page | mtgtop8.com’s cookie banner obstructed the driver from clicking certain elements on the page | self.driver.find\_element(By.XPATH, '//\*[@id="cookie\_window"]/div[2]/button').click() |
| Didn’t know how to pass arguments from python to pythonShell | NA | print(Arguments here) |
| Complications inputting data into database | Data types are inconsistent across app | Formatted all arrays into the json format. |
| Database datatype insert error | Data had to be turned into a json object via json.dumps() | json.dumps(filename), json.dumps(players), json.dumps(deckName), json.dumps({'P1':matchLists['P1']}), json.dumps({'P2':matchLists['P2']}), json.dumps({'play':extra['play']}), json.dumps({'winner':extra['winner']}), json.dumps({'date':dateTime}) |
| Database insert not working | Quotes in the json data caused problems when trying to be put into the database | Implemented parameterized queries.  Before: self.cursor.execute("INSERT INTO matches(filename, opponent, decknames, decklistP1, decklistP2, play, winner, date) VALUES(json.dumps(filename), json.dumps(players), json.dumps(deckName), json.dumps({'P1':matchLists['P1']}), json.dumps({'P2':matchLists['P2']}), json.dumps({'play':extra['play']}), json.dumps({'winner':extra['winner']}), json.dumps({'date':dateTime}));")  After: self.cursor.execute("INSERT INTO matches(filename, opponent, decknames, decklistP1, decklistP2, play, winner, date) VALUES(?,?,?,?,?,?,?,?,?,?);", data) |
| Duplicate database entries occurred. | The app stored file that already existed in the database | Gets existing filenames of the files in the database  self.cursor.execute("SELECT filename FROM matches;")  filenames = self.cursor.fetchall()  filenames = [json.loads(i[0])['filename'] for i in filenames]  Check implemented in the loop, to move onto the next file, if it’s already been stored.  if filename in filenames:  break |
| App lacked necessary data to insert a record despite passing the null checks | decklists could be empty, and not null, passing the null checks | Checked if decklists was empty  if decklists != {} and |
| App would fail to access database at seemingly random points | If the latest changes weren’t committed to the database and the connection (and cursor) wasn’t closed, the database would become locked | After each time the database needed to be saved. When appending all the stored data, this only needs to be done each time, since the app in unable to read the last matchID (since the change hasn’t been commited), so each record would have tried to be committed with the same matchID.  self.userConnection.commit()  Also, at the end of the sync.py process:  self.cursor.close()  self.userConnection.close() |

## Research and Development of New Skills

* Web Scraping, using the Selenium Library
  + This skill allowed the developer to estimate the name of the deck played by the user, depending on the contents of their deck, without pre-stored data.
  + The developer also learnt about the XPATH in order to navigate the html structure of a webpage.
  + The Selenium library is not only relevant to Python, but also Java, CSharp, Ruby, JavaScript and Kotlin.
  + The Selenium library is the most prominent driver-based (ability to be automated, or scrape JavaScript-based web pages) web scraping library.
* Electron Framework
  + This skill allowed the developer to create a web-based (HTML, CSS and JavaScript) interface for the app.
  + This framework is use in many real-world instances (Microsoft Teams and Github Desktop), so may be very useful to have a grasp of in the future.
* Greater Knowledge of JavaScript
  + The developer learnt more about JavaScript, including the ways it operates with the Electron Framework (nodeJS) and listener events.
* Python
  + Learnt the programming language Python, which is was very useful in order to program the main functionality of the app (data collection and web scraping).
  + This language is one of the most versatile languages and has applications of project mockups to data-analysis.

# Testing

## Test Table

This is the test table of data used in the component and integrative testing.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| matchID | | filename | | opponent | | decknames | | decklistP1 | decklistP2 | play | winner | date |
| 3 | Match\_GameLog\_04608bd8-1192-43a7-93b5-91b253171d1c.dat | | ['Corzalan'] | | [{'Walls Combo': 0.8, 'Combo Walls': 0.2}] | | [{'Forest': 4, 'Overgrown Battlement': 2, 'Axebane Guardian': 2, 'Winding Way': 2, 'Quirion Ranger': 2, 'Orochi Leafcaller': 2, 'Valakut Invoker': 1}] | | [{'Mountain': 2, 'Great Furnace': 1, 'Dwarven Forge-Chanter': 1, 'Chain Lightning': 2, 'Lava Dart': 2, 'Reckless Impulse': 1, 'Lightning Bolt': 1, 'Experimental Synthesizer': 1, 'Implement of Combustion': 1}] | ['Corzalan', '-Sire'] | ['Corzalan', 'Corzalan'] | 2023-02-15 10:14:41.441359 |
| 2 | Match\_GameLog\_02c28beb-adca-4ff2-8bcc-8ae594d3fc74.dat | | ['Hoarse\_Duck'] | | [{'1.0': 0.5, 'NA': 0.5}] | | [{'Hickory Woodlot': 4, 'Saprazzan Skerry': 3, 'Everflowing Chalice': 3, 'Lotus Petal': 2, 'Contentious Plan': 3, 'Weather the Storm': 2, 'Peat Bog': 2, 'Deep Analysis': 4, 'Pentad Prism': 3, 'Experimental Augury': 2, 'Prologue to Phyresis': 2, 'Bring the Ending': 3, 'Island': 1}] | | [{'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Kuldotha Rebirth': 2, 'Galvanic Blast': 1, 'Chromatic Star': 1, 'Chain Lightning': 1, 'Raze': 2, 'Voldaren Epicure': 2, 'Lightning Bolt': 2, 'Reckless Impulse': 1, 'Goblin Blast-Runner': 1}] | ['-Sire', '-Sire'] | ['Hoarse\_Duck', 'Hoarse\_Duck'] | 2023-02-15 21:21:59.164663 |
| 1 | Match\_GameLog\_052679ad-d68b-4bb3-a40f-a7ff030dc218.dat | | ['sc00nes'] | | [{'Walls Combo': 1.0}] | | [{'Forest': 5, 'Overgrown Battlement': 1, 'Axebane Guardian': 2, 'Winding Way': 1, 'Tinder Wall': 1, 'Lead the Stampede': 2, 'Quirion Ranger': 1, 'Banishing Coils': 1, 'Secret Door': 1, 'Shield-Wall Sentinel': 1, 'Drift of Phantasms': 1, 'Valakut Invoker': 1, 'Guardian Naga': 1}] | | [{'Silverbluff Bridge': 1, 'Preordain': 1, 'Vault of Whispers': 1, 'Terrarion': 1, 'Mistvault Bridge': 2, 'Seat of the Synod': 2, 'Ichor Wellspring': 2, 'Thoughtcast': 3, 'Pyroblast': 1, 'Hydroblast': 1, 'Deadly Dispute': 1, 'Chromatic Star': 1, 'Krark-Clan Shaman': 1, 'Myr Enforcer': 1}] | ['-Sire', '-Sire', 'sc00nes'] | ['sc00nes', '-Sire', 'sc00nes'] | 2023-02-14 17:27:37.852952 |

## Component testing

Component testing is the post-implementation testing log of all methods, functions, listener events and functionless files.

### sync.py

|  |  |  |  |
| --- | --- | --- | --- |
| Function/ Method/File | Test data/input | Expected Result | Actual Result |
| File | Filenames: | Adds player name to Scraper (scp) and relevant files to module, then runs the scp module | Text  Description automatically generated with medium confidence |

### mtgScraper.py

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function/ Method/ File | Test data/input | | | | Expected Result | | | Actual Result | | | | | | Comment | | | |
| addPath | path: C:\Users\harri\AppData\Local\Apps\2.0\Data\\JQDHOZ0O.KYM\V578TL38.DLE\mtgo..tion\_80b7d5223ec4322a\_0003.0004\_676bddafd60cc0fe\Data\AppFiles\E8BC386C00E942D40363482907EEDEEA | | | | Path added to self.paths array | | |  | | | | | |  | | | |
| getPlayer | filename: C:\Users\harri\AppData\Local\Apps\2.0\Data\\JQDHOZ0O.KYM\V578TL38.DLE\mtgo..tion\_80b7d5223ec4322a\_0003.0004\_676bddafd60cc0fe\Data\AppFiles\application\_settings | | | | self.player is set to the user’s playername, if playerName.txt file exists, else read in the file (filename) and set self.player to the last login name from the file | | |  | | Error: quotes were included in the result.  Code: self.player = re.findall('".\*"', file[file.index('Setting="LastLoginName"')+1])[0]  Fix: self.player = re.findall('"(.\*)"', file[file.index('Setting="LastLoginName"')+1])[0]  Added brackets () to regex  Result: | | | | | | |
| openConn | None | | | | Code Modified: def openConn(self):          self.userConnection = sqlite3.connect("./database/mtgoAssist.db")          self.cursor = self.userConnection.cursor()          try:              self.cursor.execute("SELECT MAX(matchID) FROM matches;")              print("connected")          except:              print("unconnected")  Connects to database. Should print connected is connected, else, print unconnected. | | | | | | | When connected:  When fails to connect: | | | |  |
| checkDB | Test Table | | | | To test if a table exists, the app tries to get the highest matchID, if the try fails, then the Matches table is created. | | | | | | | When table exists, MAX(matchID) = | | | |  |
| checkInternet | | None | | Returns true or false whether the computer could connect to mtgtop8.com | | |  | | Error: conInt wasn’t returned  Code: def checkInternet(self):          url = 'https://mtgtop8.com/'          try:              urllib.request.urlopen(url)              conInt = True          except:              conInt = False            return  Fix: def checkInternet(self):          url = 'https://mtgtop8.com/'          try:              urllib.request.urlopen(url)              conInt = True          except:              conInt = False            return conInt | | | | | | | |
| run | | Test Table | | Runs main process:  gets, and reformats data from mtgo files into the database | | | NULL | | Error: Missed self.  Code: def run(self):          #opens connection          self.openConn()          #checks if database exists          self.checkDB()          top8Conn = self.checkInternet()            #initilises modules          match = MatchRecord()          dc = DriverController()            #gets list of files in database          self.cursor.execute("SELECT filename FROM matches;")          filenames = self.cursor.fetchall()          filenames = [json.loads(i[0])['filename'] for i in filenames]          for path in paths:                #gets list of files in path              fileList = [f for f in os.listdir(path) if f.endswith('.dat')]              #loops through file list              for filename in fileList:                  #if file already scraped                  if filename in filenames:                      break                    #gets decklists from MatchRecord                  #other stored data is implemented into database in MatchRecord                  decklists, extra, players = match.getDecklists(f'{path}/{filename}')                  #if the file is valid                  if decklists != {} and decklists is not None:                      players.remove(self.player)                      #gets and reformats date                      dateTime = str(datetime.fromtimestamp(os.path.getmtime(path + '\\' + filename)))                      x, y, z = dateTime.split(' ')[0].split('-')                      date = {'date':f'{z}/{y}/{x}'}                        if top8Conn == True:                          #gets the possible deck names from DriverController                          deckName, matchLists = dc.returnDeckName(decklists, date)                      else:                          deckName = {'NA': 1.0}                        #sends info to sqliite db                      self.sqlliteDriverData({'filename':filename}, dateTime, deckName, extra, {'players':players}, matchLists)          #closes webdriver          dc.quitDc()            #close connection          self.closeConn()  Fix: def run(self):          #opens connection          self.openConn()          #checks if database exists          self.checkDB()          top8Conn = self.checkInternet()            #initilises modules          match = MatchRecord()          dc = DriverController()            #gets list of files in database          self.cursor.execute("SELECT filename FROM matches;")          filenames = self.cursor.fetchall()          filenames = [json.loads(i[0])['filename'] for i in filenames]          for path in self.paths:                #gets list of files in path              fileList = [f for f in os.listdir(path) if f.endswith('.dat')]              #loops through file list              for filename in fileList:                  #if file already scraped                  if filename in filenames:                      break                    #gets decklists from MatchRecord                  #other stored data is implemented into database in MatchRecord                  decklists, extra, players = match.getDecklists(f'{path}/{filename}')                  #if the file is valid                  if decklists != {} and decklists is not None:                      players.remove(self.player)                      #gets and reformats date                      dateTime = str(datetime.fromtimestamp(os.path.getmtime(path + '\\' + filename)))                      x, y, z = dateTime.split(' ')[0].split('-')                      date = {'date':f'{z}/{y}/{x}'}                        if top8Conn == True:                          #gets the possible deck names from DriverController                          deckName, matchLists = dc.returnDeckName(decklists, date)                      else:                          deckName = {'NA': 1.0}                        #sends info to sqliite db                      self.sqlliteDriverData({'filename':filename}, dateTime, deckName, extra, {'players':players}, matchLists)          #closes webdriver          dc.quitDc()            #close connection          self.closeConn() | | | | | | | |
| sqlliteDriverData | | | filename: 'Match\_GameLog\_02c28beb-adca-4ff2-8bcc-8ae594d3fc74.dat'  dateTime: '2023-02-15 21:21:59.164663'  deckName: {'NA': 1.0}  extra: {'play': ['-Sire', '-Sire'], 'startingHands': [[['-Sire', 'seven'], ['Hoarse\_Duck', 'six']], [['Hoarse\_Duck', 'seven'], ['-Sire', 'five']]], 'winner': ['Hoarse\_Duck', 'Hoarse\_Duck']}  players: ['Hoarse\_Duck']  matchLists: {'P2': [{'Mountain': 4, 'Monastery Swiftspear': 2, 'Experimental Synthesizer': 1, 'Kuldotha Rebirth': 2, 'Galvanic Blast': 1, 'Chromatic Star': 1, 'Chain Lightning': 1, 'Raze': 2, 'Voldaren Epicure': 2, 'Lightning Bolt': 2, 'Reckless Impulse': 1, 'Goblin Blast-Runner': 1}], 'P1': [{'Hickory Woodlot': 4, 'Saprazzan Skerry': 3, 'Everflowing Chalice': 3, 'Lotus Petal': 2, 'Contentious Plan': 3, 'Weather the Storm': 2, 'Peat Bog': 2, 'Deep Analysis': 4, 'Pentad Prism': 3, 'Experimental Augury': 2, 'Prologue to Phyresis': 2, 'Bring the Ending': 3, 'Island': 1}]} | | | A record in the database, with that information | | | A screenshot of a computer  Description automatically generated with medium confidence  When testing to see if record exists. | | Error: When tested for if the record exists, the record wasn’t commited  Code: def sqlliteDriverData(self, filename, dateTime, deckName, extra, players, matchLists):          data = (json.dumps(filename), json.dumps(players), json.dumps(deckName), json.dumps({'P1':matchLists['P1']}), json.dumps({'P2':matchLists['P2']}), json.dumps({'play':extra['play']}), json.dumps({'winner':extra['winner']}), json.dumps({'date':dateTime}))            self.cursor.execute("INSERT INTO matches(filename, opponent, decknames, decklistP1, decklistP2, play, winner, date) VALUES(?,?,?,?,?,?,?,?);", data)  Fix: def sqlliteDriverData(self, filename, dateTime, deckName, extra, players, matchLists):          data = (json.dumps(filename), json.dumps(players), json.dumps(deckName), json.dumps({'P1':matchLists['P1']}), json.dumps({'P2':matchLists['P2']}), json.dumps({'play':extra['play']}), json.dumps({'winner':extra['winner']}), json.dumps({'date':dateTime}))            self.cursor.execute("INSERT INTO matches(filename, opponent, decknames, decklistP1, decklistP2, play, winner, date) VALUES(?,?,?,?,?,?,?,?);", data)          self.userConnection.commit() | | | |
| closeConn | | None | | Closes connection. | | Connection is closed. | | | | | | |  | | | |

### matchRecord.py

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function/ Method/File | | Test data/input | | Expected Result | | | Actual Result | Comment | |
| getDecklists | filename: | | | | Returns decklists, extra and players. |  | | |  | |
| getPlayers |  | | | Gets each player in the match | | |  |  | |
| formatLines | | |  | Reformates the matchLog | | |  |  | |
| getDeckLists | | | game: | Gets the decklists in a game. | | |  |  | |
| getOnPlay | | | game: | Gets the player on the play in a game. | | |  |  | |
| getStartingHands | | | game: | Gets the startingHands of each player in a game. | | |  |  | |
| getWinner | | | game: | Attempts to get the winner in a game. | | |  |  | |

### mtgtop8.py

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Function/ Method/ File | Test data/input | Expected Result | Actual Result | | Comment |
| returnDeckName | deckLists: date: | Returns dictNames and deckLists |  | |  |
| getSite | Self.url = https://www.mtgtop8.com | Loads the mtgtop8.com site | Doesn’t load the mtgtop8.com site | Error: Used a depreciated function url(), rather than the get() function.  Code: def getSite(self):  self.driver.url(self.url)  Fix: def getSite(self):  self.driver.get(self.url) | |
| inputFormData | deckList:  date: | Inputs the form data on the page, in order to get results. |  | |  |
| clearCookieBanner | None | Clears the cookie banner on the website. |  | |  |
| quitDc | None | Stops the Driver Controller. |  | |  |
| refactorDecklist | deckLists: | Changes the formatting of the decklist variable. |  | |  |
| getDictNames | deckNames: | Creates and returns decknames. |  | |  |
| getDeckNames | None | Gets the first five deck names. |  | |  |

### dbCMD.py

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function/ Method/ File | Test data/input | | Expected Result | Actual Result | | | Comment | |
| openData | Test Table | | Prints all data from the database. | NA | | Error: Didn’t .fetchall() matches, which turns an sql object to a set  Code: def openData():      userConnection = sqlite3.connect("./database/mtgoAssist.db")      cursor = userConnection.cursor()      try:          matches = cursor.execute("SELECT \* FROM matches ORDER BY matchID DESC;")            results = {}          for matchTuple in matches:              results[matchTuple[0]] = [{}]              for data in matchTuple:                  if isinstance(data, int) == False:                      data = json.loads(data)                      results[matchTuple[0]][0][list(data.keys())[0]] = data[list(data.keys())[0]]          print(json.dumps(results))      except:          pass      finally:          cursor.close()          userConnection.close()  Fix: def openData():      userConnection = sqlite3.connect("./database/mtgoAssist.db")      cursor = userConnection.cursor()      try:          matches = cursor.execute("SELECT \* FROM matches ORDER BY matchID DESC;").fetchall()            results = {}          for matchTuple in matches:              results[matchTuple[0]] = [{}]              for data in matchTuple:                  if isinstance(data, int) == False:                      data = json.loads(data)                      results[matchTuple[0]][0][list(data.keys())[0]] = data[list(data.keys())[0]]          print(json.dumps(results))      except:          pass      finally:          cursor.close()          userConnection.close() | | |
| getOppWinrate | | Test Table  opponent: VOIDx | Prints the winrate% an opponent has against the user | |  | | |  | |
| getUserData | Test Table | | Prints the overall winrate% of the user |  | | |  | |

### renderer.js

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function/ Method/ File | | | | | Test data/input | | | | | | Expected Result | | | | Actual Result | | | Comment |
| DOM Loaded Event Listener | | | Test Table | Calls loadFirstPage function | | | | Results are retrieved and displayed.  Graphical user interface  Description automatically generated with medium confidence | | | | | | | | | | |
| Sync Button Clicked Listener | | | | | Click on id=#pyBtnSync element  Test Table | | | | | | Calls runPythonSync function | | | | Empty database before storing data (SELECT \* FROM matches;):  Number of records after storing data (SELECT COUNT(matchID) FROM matches;): | | |  |
| Close Button Clicked Listener | | Click on id=#close element  Test Table | | | Calls closeProfile function | | | |  | | | | | | | | Error: called profile() function, which does not exist  Code: profile();  Fix: closeProfile(); | |
| Profile Button Clicked Listener | Click on id=#pyBtnProfile  Element  Test Table | | | | | | Calls loadProfile function | | | No response | | | | Error: Event listener incorrectly named  Code: document.querySelector('#pyBtnProfile').addEventListener('clicked', () => {    loadProfile();  })  Fix: document.querySelector('#pyBtnProfile').addEventListener('click', () => {    loadProfile();  }) | | | | |
| Submit Button Clicked Listener | Click on id=#submitUser  Element  Test Table | | | | | Calls getOpponentWinrate function | | | | | | | No response | | | Error: Wrong id referenced.  Code: document.querySelector('#close').addEventListener('click', () => {    getOpponentWinrate();  })  Fix: document.querySelector('#submitUser').addEventListener('click', () => {    getOpponentWinrate();  }) | | |
| structureDecklists | | | | | Results = Test Table records  Keys = [‘1’, ‘2’, ‘3’]  Leng = 3  i = 0  Player = P1 | | | | | | | Reformatted decklists, with line breaks, and the number of copies of the card played in front of the card, as a string. | | |  | | | |
| loadProfile | | | | | Test Table | | | | | | Shows the profile elements and hides the main page elements, then runs dbCMD.py with “profile” as an argument. | | | |  | | |  |
| loadFirstPage | | | | | Test Table | | | | | | Runs dbCMD.py with “loaded” as an argument. | | | |  | | |  |
| runPythonSync | | | | | Test Table | | | | | | Runs sync.py. | | | |  | | |  |
| getOpponentWinrate | | | | | Test Table | | | | | | Runs dbCMD.py with “opponent”, and the ‘name’ variable as arguments. | | | |  | | |  |
| closeProfile | | | | | None | | | | | | Hides the profile elements and shows the main page elements. | | | |  | | |  |

## Integrative testing

Integrative testing is post-component, post-implementation testing where the functionality is tested.

|  |  |  |
| --- | --- | --- |
| Process | Test Data/Input | Output |
| Sync Button Clicked: To store all game entries and get estimated deck names for the user’s deck in each match | Click input on Sync Button, when database has entries |  |
| Sync Button Clicked: To store all game entries and get estimated deck names for the user’s deck in each match | Click input on Sync Button, when connection to mtgtop8.com can’t be established | Error:  Fails to connect to chrome driver, as the Driver Controller is instansiated even if top8Conn is false.  Code: top8Conn = self.checkInternet()            #initilises modules          match = MatchRecord()          dc = DriverController()  Fix: top8Conn = self.checkInternet()            #initilises modules          match = MatchRecord()          if top8Conn == True:              dc = DriverController()  . . .  if top8Conn == True:              dc.quitDc()  End: |
| Profile Button Clicked: To load | Click input on Profile Button, when database has entries | Text  Description automatically generated with medium confidence |
| Profile Button Clicked: | Click input on Profile Button, when database has no entries | Text  Description automatically generated with medium confidence |
| Submit Button Clicked: To get | Click input on Submit Button, after username entered  Opponent username: VOIDx | Text  Description automatically generated with medium confidence |
| Submit Button Clicked: | Click input on Submit Button, when no username entered | A picture containing graphical user interface  Description automatically generated |
| Submit Button Clicked: | Click input on Submit Button, when the entered username doesn’t exist | A picture containing graphical user interface  Description automatically generated |
| Close Button Clicked: | Click input on Close Button | Text  Description automatically generated |
| App Opened | None |  |

## Final Testing (Requirements)

Final testing is the list of requirements, and evidence of their implementation.

|  |  |
| --- | --- |
| Requirement | Evidence |
| Capture their match logs. | Graphical user interface  Description automatically generated with medium confidence |
| Display database data in a readable format. | Table  Description automatically generated |
| Display the winrate of the player against an opponent. | Graphical user interface, text  Description automatically generated with medium confidence |
| Display the average winrate of the player. | Logo  Description automatically generated |
| Attempt to estimate who won each game if the game log doesn’t specify the winner of that game. | #if only two matches were played, then the person who was on the draw game 2, won the match          if 'NA' in extra['winner'] and len(extra['play']) == 2:                #gets player on draw in game 2              player = [i for i in self.players if extra['play'][1] != i]              #loops through each 'NA' result, and replaces it with the winner              for indice in [i for i, elem in enumerate(extra['winner']) if elem == 'NA']:                  extra['winner'][indice] = player[0]          #if 1 or fewer winners were recorded, and only two games were played, the player in the draw in game 2 won the match          if len(extra['winner']) < 2 and len(extra['play']) == 2:                #gets player on draw in game 2              player = [i for i in self.players if extra['play'][1] != i]              #while there are less than 2 recorded winners              while len(extra['winner']) < 2:                  extra['winner'].append(player[0]) |
| Be integrated with a local database. | As testing from openConn concludes: |
| Be compatible with their device (PC). | App is able to run on a PC. |
| The app should try to keep a clean look. | Simple design, with very little ‘pop’.    Scatter chart  Description automatically generated with medium confidence |

## Testing with Personas

|  |  |  |
| --- | --- | --- |
| Persona | Requirements | Evidence of Fulfilment |
| Sean Greaves | See which deck most gives him an advantage, by browsing the records to see which deck overperforms, if any deck does. | Graphical user interface  Description automatically generated with medium confidence  Sean is able to see here that they won with Grixis Affinity, 2-0 in a best of 3 match, compared to losing with Walls Combo, Dimir Terror and Kuldotha Rebirth. |
| Winnie Manning | A consistent format for match data. | A picture containing table  Description automatically generated  As seen with this image, and previous images of the app, the table format is totally consistent, so is fairly easy to read. |
| James Hill | Presentation of all decklists, so he can look through them, so see which cards most give him an advantage. | A picture containing calendar  Description automatically generated  Here James can see that they played similar cards in the matches they won, Forest and Quirion Ranger. This is a different strategy to the middle game where they played Silverbluff bridge and Myr Enforcer. So this small dat set indicates they should play more Quirion Ranger and Forest more, since they won those matches. |

# Evaluation

## Fitness for purpose

The app is fit for purpose as it:

* Captures the end-user’s match logs.
* Displays the stored database data in a readable format.
* Displays the end-user’s win rate against a specified opponent.
* Displays the average win rate of the end-user.
* Attempts to estimate who won each game if the game log doesn’t specify the winner of that game.
* Is integrated with a local database.
* Is compatible with modern PCs.
* Keeps a clean look in the interface.

This means that the app is fit for purpose as it meets the design requirements.

## Robustness

The app is fairly robust, as it validates each match file, and uses parameterized queries to avoid issues with certain characters. However, there is no way to validate the authenticity of a match file (whether it’s actually malicious code, that happens to look like a match file). Also, the end-user’s textbox input (when getting the win percentage against a specific opponent) isn’t validated, so is vulnerable to SQL injection.

## Efficiency

The project is somewhat efficient as it quicky stores the end-user’s match data, however, the web-scraping process is fairly slow, due to the limitations of Selenium. Every time the driver is instantiated, the driver is temporarily installed, this takes about 2 seconds, then loading the site (getSite) takes another 2 seconds, inputting the form data (inputFormData) takes roughly 4 seconds, and quitting takes 2 seconds. This means that each loop takes about 6 seconds + getting and storing data (less than a second to 2 seconds). A potential future solution is storing each deck in either a JSON file or database, and each time the app tries to get the deck name, the app first searches each file/database to get the deck name, and if this fails, then the app attempts to web-scrape.

## Usability

Due to the simple design, the app doesn’t appear complex, and there are not many actions the end-user can take, so the app is streamlined. This means that the end-user should have an easy time to interact with the app, and there is unlikely to be any difficulties. The text and images are also high contrast, so the app is very readable. However, the text doesn’t scale, so that may be a problem for some people. The app also assumes that people know how to operate this app (which shouldn’t be a problem for its target demographic).

## Maintainability

The maintainability of the app is undemanding, as it’s dependencies (mtgtop8.com and MTGO) have existed for almost 20 years, so aren’t likely to go down. Equally, there is little chance the file formatting will change since it has been similar for decades. This means that it is unlikely for any new edge cases that require action to occur, and if they do occur, they will likely be minor (e.g. a new keyword to search for in regex). The code is also white-spaced and commented, so past work can be understood.

## Future Design

* Display individual games, when a match is clicked.
* Display card win% (% of times you played a card and won).
* Search functionality.
* Store average deck data locally, so the app doesn’t have to web scrape each time a deck name is required.
* Allow the user to modify unknown decknames.

### Fix Current Low-Priority Issues:

* When no deckname is found (when connected to the internet), it is given the deckname NA and Unknown, resulting in {NA:0.5, Unknown:0.5}, and NA: 0.5 Unknown: 0.5 being displayed. Declaration of deckname when none is found should be updated.
* Sync button may be clicked multiple times, resulting in errors, and or duplicate records in the database, so a limiting function should be implemented.
* Move refactor decklists method to mtgScraper, so error doesn’t occur when the app cannot connect to the internet.
* The app never creates a text file containing the user’s name, so the try/except check will always except, so it’s useless, unless a text file is created.
* Have the profile page as a separate page, rather than hiding, and showing different page content

# Appendix 1 – Pre-Development User Survey Results

## Chart, bar chart Description automatically generatedQ1

Results: Most potential end-users are under 18 to 24, so app design doesn’t have to account for many elderly people.

## Graphical user interface, text, application, email, website Description automatically generatedQ2

Numerical Results: 5, 50+, 34, 1, 23, 1, 4, 6, 14, 2, 0, 0, 10, 10, 12, 10, 10, 31, 17, 18, 24, 16

Mean: 14 (2 s.f.)

Results: Most potential end-users play videogames a fair amount.

## Chart Description automatically generatedQ3

Results: Most potential end-users would want the app to display feedback on their game.

## Q4

Chart

Description automatically generated

Results: Most potential end-users would want the app to try and help them improve.

## Chart, bar chart Description automatically generatedQ5

Results: Most potential end-users want a combination of ways to display information stored by the app.

## Chart, bar chart Description automatically generatedQ6

Results: Cost is a barrier of entry for most potential end-users, this confirms that the app should be free, to enable as many users to use it as possible.

## Chart, bar chart Description automatically generatedQ7

Results: Most potential end-users would be interest in installing the finished app.