AH SDD Project: Game Assist Tool

Contents

[Analysis 1](#_Toc114748791)

[Description of problem 1](#_Toc114748792)

[User Survey 2](#_Toc114748793)

[Constraints 2](#_Toc114748794)

[UML Use Case Diagram 3](#_Toc114748795)

[Requirements Specification 3](#_Toc114748796)

[End-User Requirements 3](#_Toc114748797)

[Functional Requirements 3](#_Toc114748798)

[User Stories 3](#_Toc114748799)

[User Scenarios 4](#_Toc114748800)

[Project Plan 4](#_Toc114748801)

# Analysis

## Description of problem

* No maintained tool exists which records match data on the game Magic: The Gathering Online (MTGO). This leaves users to record data manually on an excel document, if at all, which is very time consuming.
* The end users are people who play Magic: The Gathering Online (MTGO), they are fairly technically adept (as they play an online game), their age range is roughly 16-45. The end users also spend a decent to a lot of money on the program. This is relevant as they typically want to maximise their win percentage, and are capable of installing, and using, an assist tool.
* The app is built on Electron. On the app, the user can press a button which runs the main processes of the data collection. This main process navigates to MTGO’s file system, where it gets the users’ match logs (.dat files). The app then reads all information within the logs, which it categorises and stores in a local database (in sqlite3), potentially using more user input to clarify potential inconsistencies. The app also uses the Selenium web driver to get additional information (deck names), to store in the database. All information stored in the local database is easily readable in the Electron app. All stored information is also displayed on graphs on the app.
* The project meets these requirements:
  + Is an SDD project integrated with a local database.
  + The project will use a sort algorithm, object-oriented and procedural programming, an array of records (specifically dictionaries and lists, as python doesn’t have arrays), create a local database (if it doesn’t already exist) and update, select, delete records (based on what the end-user selects) from that database.

## User Survey

## Constraints

* Technical Constraints
  + The program will run on Windows 7 or later.
  + The program 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 program.
* Business Constraints
  + The scheduling and timescales must be met, otherwise the program will be incomplete.
  + There is no budget for the program.
  + There is a single person working on the program.
  + There are no licensing issues with the program.
* Feasibility Study
  + Economic
    - There are no costs associated with the project as it uses free software. It also generates no revenue. However, it can be further built upon and in such a way that can create revenue (e.g. membership program for more features, not yet implemented).
  + Time
    - The program will take 80 hours to complete.
  + Legal
    - There are no legal issues that are currently present: 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 highly personal data is stored (other than the optional username and password for the game and the app itself), and that data is stored locally, inaccessible by the internet.
  + Technical
    - All the necessary resources currently exist for the project to be completed.

## UML Use Case Diagram

Actors

Use case

Relationships

## Requirements Specification

### End-User Requirements

### Functional Requirements

* Inputs
* Processes
* Outputs

AS <user role> I want <what?>

## Personas

Sean Greaves is 23 years old man that lives in America, on their own in a flat. 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.

## User Stories

* As a (user type) I want to

## User Scenarios

## Project Plan – Initial estimated version

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | | Start Date | End date | Resources Required |
| Analysis (27/08/22 – 19/10/22) | | | | |
| 1. | Description of problem | 27/08/22 | 29/08/22 |  |
| 2. | Create user survey | 31/08/22 | 05/09/22 | Using Survey Monkey |
| 3. | Constraints | 20/09/22 | 21/09/22 |  |
| 4. | UML Use Case diagram | 22/09/22 | 27/09/22 | Using Lucidchart |
| 5. | Create persona and user stories | 01/10/22 | 05/9/22 |  |
| 6. | Requirements specification: end-user req. | 07/10/22 | 08/10/22 |  |
| 7. | Requirements specification: functional req. | 09/10/22 | 17/10/22 |  |
| 8. | Project plan | 17/10/22 | 19/10/22 | Using Gantt Project |
| Design (20/10/22 – 21/10/22) | | | | |
| 9. | Pseudocode design | 20/10/22 | 22/10/22 |  |
| 10. | UML class diagram | 24/10/22 | 27/10/22 |  |
| 11. | Project Design | 27/10/22 | 30/10/22 |  |
| 12. | User-Interface Design | 27/10/22 | 31/10/22 |  |
| Implementation (01/11/22 – 10/12/22) | | | | |
| 13. | Implementation | 01/11/22 | 10/12/22 |  |
| 14. | Research and development of new skills | 01/11/22 | 05/12/22 |  |
| 15. | Log of ongoing testing | 01/11/22 | 10/12/22 |  |
| End Testing (12/12/22 – 23/12/22) | | | | |
| 16. | Final Test Plan | 12/12/22 | 16/12/22 | Using pytest |
| 17. | Requirements Testing | 17/12/22 | 18/12/22 |  |
| 18. | Testing with personas and test cases | 20/12/22 | 23/12/22 |  |
| Evaluation (05/01/23 – 07/01/23) | | | | |
| 19. | Evaluation Report | 05/01/23 | 07/01/23 |  |

## Project Plan – Final version

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | | Start Date | End date | Resources Required |
| Analysis | | | | |
| 1. | Description of problem | 27/08/22 | 29/08/22 |  |
| 2. | Create user survey | 31/08/22 | 05/09/22 | Survey monkey – create account |
| 3. | Constraints | 20/09/22 | 21/09/22 |  |
| 4. | UML Use Case diagram |  |  | Using Lucidchart |
| 5. | Create persona and user stories |  |  |  |
| 6. | Requirements specification: end-user req. |  |  |  |
| 7. | Requirements specification: functional req. |  |  |  |
| 8. | Project plan |  |  | Using Gantt Project |
| Design | | | | |
| 9. | Pseudocode design |  |  |  |
| 10. | UML class diagram |  |  |  |
| 11. | Project Design |  |  |  |
| 12. | User-Interface Design |  |  |  |
| Implementation | | | | |
| 13. | Implementation |  |  |  |
| 14. | Research and development of new skills |  |  |  |
| 15. | Log of ongoing testing |  |  |  |
| End Testing | | | | |
| 16. | Final Test Plan |  |  | Using pytest |
| 17. | Requirements Testing |  |  |  |
| 18. | Testing with personas and test cases | 20/12/22 |  |  |
| Evaluation | | | | |
| 19. | Evaluation Report | 05/01/23 | 07/01/23 |  |

# Design

Pseudocode Design

UML Class Design

Project Design

User-Interface Design

Appendix 1 – User Survey Results

Q1

Chart, bar chart

Description automatically generated

Results – Most users

Q3

Graphical user interface, text, application, email, website

Description automatically generated

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 users play videogames a fair amount.

Q3

Chart

Description automatically generated

Results – Most end users would want the program to display feedback on their game.

Q4Chart

Description automatically generated

Results – Most users would want the program to try and help them improve.

Q5

Chart, bar chart

Description automatically generated

Q6

Chart, bar chart

Description automatically generated

Q7

Chart, bar chart

Description automatically generated

Appendix 2

Appendix 3