DAN Booth

Interim Report

2021

contents

## Aims and Objectives

### Aim

The key aim of this project to create a digital checklist for pieces in a Lego Set. This will be where users can search for a Lego Set and then click on the Lego set they would like to see the pieces for. User can then view all the pieces for this Lego set (like in the back of the instruction book), and they can check they have all the pieces when they are building the set again.

For example, you have a Lego set that you have taken apart and put all the pieces in a box along with other Lego pieces, and you would like to rebuild the set, you could do this easily using a digital checklist.

The target users for my project will be 18+ Lego builders/enthusiasts/collectors who are looking to build/rebuild a Lego set they own.

### Objectives

1. Write a Java program to connect to the Rebrickable API [2] and retrieve data
2. Learn how to turn JSON files into a Java class
3. Build an application using the Model-View-Controller framework
4. Design and create a cross-platform website (view) from which users can use the system
5. Design and implement a way for users to save progress on a checklist

Challenges I expect to face during the project are

* Linking the Rebrickable API [2] to a Java program
* Transforming data from a JSON file to a Java class
* Developing a cross-platform website

## Survey of Literature/Information Sources

I looked up the Rebrickable API [2] that contains data for Lego sets, that I can search through to find a Lego set and Leo pieces in the set. I also read the documentation for the API [3], finding out that it's a RESTful API (meaning I can use HTTP requests to access data) and to access the data I need an API key that is freely available with an account. Using the API you can request a Lego set directly using the Lego set unique number, or search using “A search term”, filter using “theme\_id (a number associated to a Lego theme e.g. Star Wars, that can be retrieved also using API), min\_year, max\_year, min\_parts, max\_parts” and order by a certain “field” (“set\_num”, “name”, “year”, “theme\_id”, “num\_parts”). Data is returned from the API in the form of JSON files, and a set returns “set\_num”, “name”, “year”, “theme\_id”, “num\_parts” and “set\_img\_url”, but to retrieve a JSON of all the pieces in a Lego set users have perform another call to the API. This returns a list containing each part however this cannot be ordered using the API.

I then performed some data collection on what my target users would like from a digital checklist for pieces in a Lego Set, via an online questionnaire (see **Appendix A**). Using this I could identify their key requirements for the system like where they would like to use the system, how they currently check they have all the pieces for a Lego set, other tools they use for research, how important certain features would be to them and if they have any other ideas for features.

- appendix questionnaire and results and findings

From my questionnaire results (see **Appendix B**) I found that Lego builders/enthusiasts/collectors who use a digital tool use the website Bricklink [4]. I found that on Bricklink users can add pieces from a Lego set to a “wanted list” and from there tick of parts you have. This shows the user how many pieces they need and how many they currently have found. However, this number easily be changed by accident which could cause issues. For example, users could believe they have all the pieces for a set but they accidently decreased how many of a pieces they needed so are missing one, or the opposite where they increase the number they need but actually have all of them. Users can’t filter pieces by colour or type making it difficult to find pieces, also when pieces are fully found they are not hidden from the list. Any pieces missing can easily show a list of possible locations to buy them. Most of these issues appear because the purpose of the tool is to buy pieces for a Lego set.

The results of the questionnaire also show some people currently use Rebrickable website [5]. On Rebrickable, which also provide the API I am going to use, users can find a Lego set by typing in the set number or searching by a text search (i.e. Set Name) and filter by a range of year released, range of the number of parts and also filter by themes. On the page of a set (e.g. this Lego Set [6]) users can see a list of all parts, the instructions, pictures of the Lego set, year released, number of parts etc. Here if the user has an account they can add the set parts to a List. On the list, the user added parts too, users can filter by piece colour, type (Category) and sort by colour, Hue, part, type (category) and price to buy the Lego piece. Users can see how many each piece is required as well as the colour and price to buy it, but to check a piece off the list, the user has to delete it from the list meaning you can’t undo the change, also users can change the number of a certain pieces needed but not see original number (like BrickLink). This is primarily due to the fact the tool is meant to help users buy Lego pieces for a set, also the same as Bricklink, but can be used as a makeshift checklist.

I also found an API (Brickset API [7]) that I can use retrieve Lego set instructions (as the current Rebrickable API cannot do this), but will only use this API for retrieving instructions as it does not contain data on pieces within a Lego set, which is a vital part of the project, so I will use Rebrickable API for this.

## Requirements

### Key Features:

* Must be usable as a website on both mobile and PC/laptops
* Must display a list of all Lego sets stored in Rebrickable API [2]
* Must have a search feature for that allows users to search a list of Lego sets. Can search by set number and text search (e.g. set name), filter by year made, theme (e.g. Star Wars), number of pieces in set and sort by year made, set pieces, theme
* Must be able to ‘check’ piece off the checklist, showing how many more of that piece are remaining (able to undo if wrong brick clicked) 🡨 may change
* The checklist must show a picture of the piece, with correct colour, as well as an alternative text description including piece name and colour
* Must be able to sort a checklist by colour of a piece
* Must be able to sort a checklist by type of a piece
* Must be usable with and without a user account

### Nice to have Features:

* Must be able to save progress on a checklist
* View instructions for a Lego set
* Filter checklist by colour of a piece
* Filter checklist by type of a piece
* Scan Lego pieces with phone camera to check if brick is in list
  + If it is in the set (and not already enough of them), If Lego piece scanned is in set, option to check pieces off the Digital Checklist
  + If in the set but already have all that type of piece needed, it will inform the user of this
  + If not in set it will inform the user of this
  + If in the set but already enough it will inform the user
* Link to buy a missing piece from a Lego Set
* Users can create an account
* Users with an account can save sets they own to a ‘Sets Owned List’, so they can easily find them later

### Optional Features

* Download instructions for a Lego set
* Users with an account can create a Lego set list and save Lego sets to them, so they can easily find them later (Sets can be in multiple lists)
* Searchable sets owned List and favourites list, like the main search feature
* There is also a mobile application

## Outline of Specification and Design

mvc

Sql

## Planning and Timescales

Table

## References

1. Del
2. "Rebrickable API | Rebrickable - Build with LEGO", *Rebrickable.com*. [Online]. Available: https://rebrickable.com/api/. [Accessed: 17- Nov- 2021]
3. "Rebrickable API Documentation | Rebrickable - Build with LEGO", *Rebrickable.com*, 2020. [Online]. Available: https://rebrickable.com/api/v3/docs/?key=15b84a4cfa3259beb72eb08e7ccf55df. [Accessed: 15- Nov- 2021]
4. "BrickLink - Buy and sell LEGO Parts, Sets and Minifigures", *Bricklink.com*. [Online]. Available: <https://www.bricklink.com/v2/main.page>. [Accessed: 17- Nov- 2021]
5. "Rebrickable | Rebrickable - Build with LEGO", *Rebrickable.com*. [Online]. Available: https://rebrickable.com/. [Accessed: 17- Nov- 2021]
6. "LEGO Set 75280-1 501st Legion Clone Troopers (2020 Star Wars) | Rebrickable - Build with LEGO", *Rebrickable.com*, 2021. [Online]. Available: https://rebrickable.com/sets/75280-1/501st-legion-clone-troopers. [Accessed: 17- Nov- 2021]
7. Huw, "API version 3 documentation", *Brickset.com*, 2020. [Online]. Available: https://brickset.com/article/52664/api-version-3-documentation. [Accessed: 17- Nov- 2021]
8. Del

## Appendix A

Questionnaire for what user would want from a digital checklist for pieces in a Lego Set.Text, letter

Description automatically generated

Table

Description automatically generatedTable

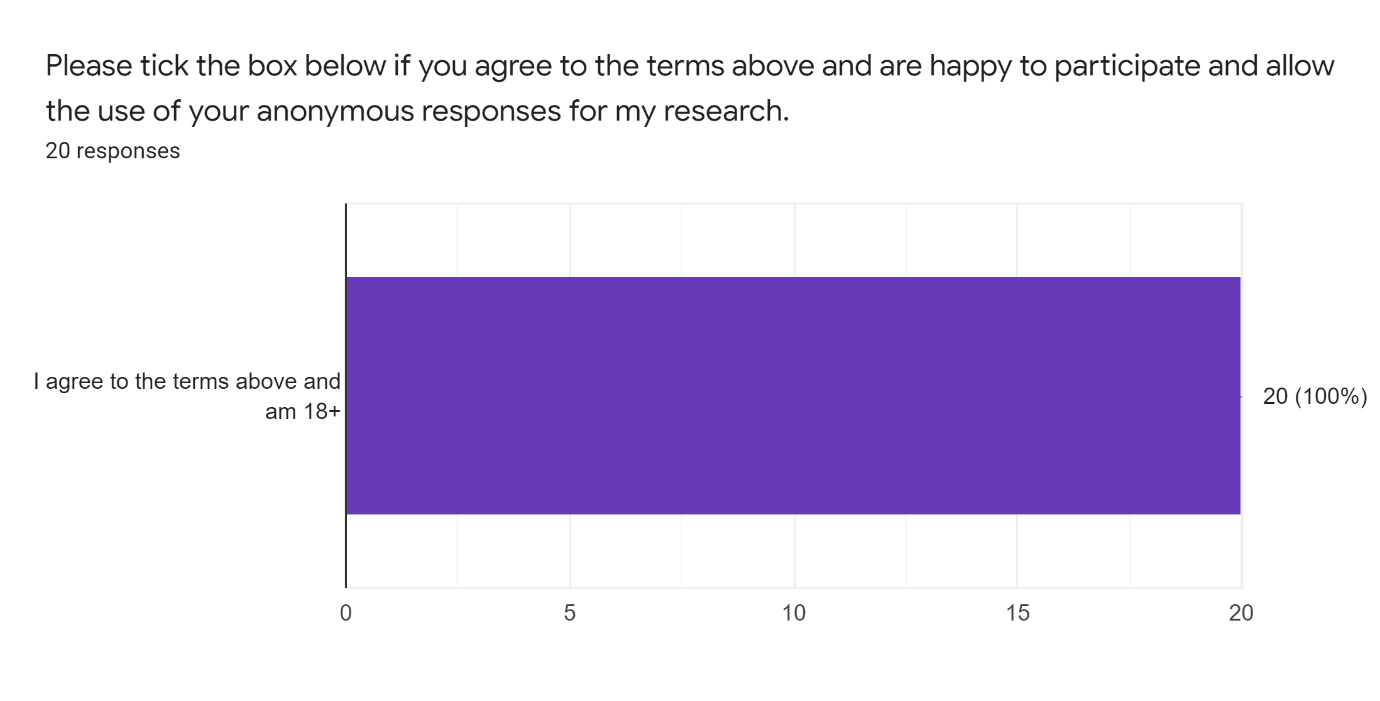
Description automatically generatedText

Description automatically generated

## Appendix B

Results from my questionnaire (see **Appendix A**).

Letter of Consent Result



Chart, pie chart

Description automatically generated

Chart, pie chart

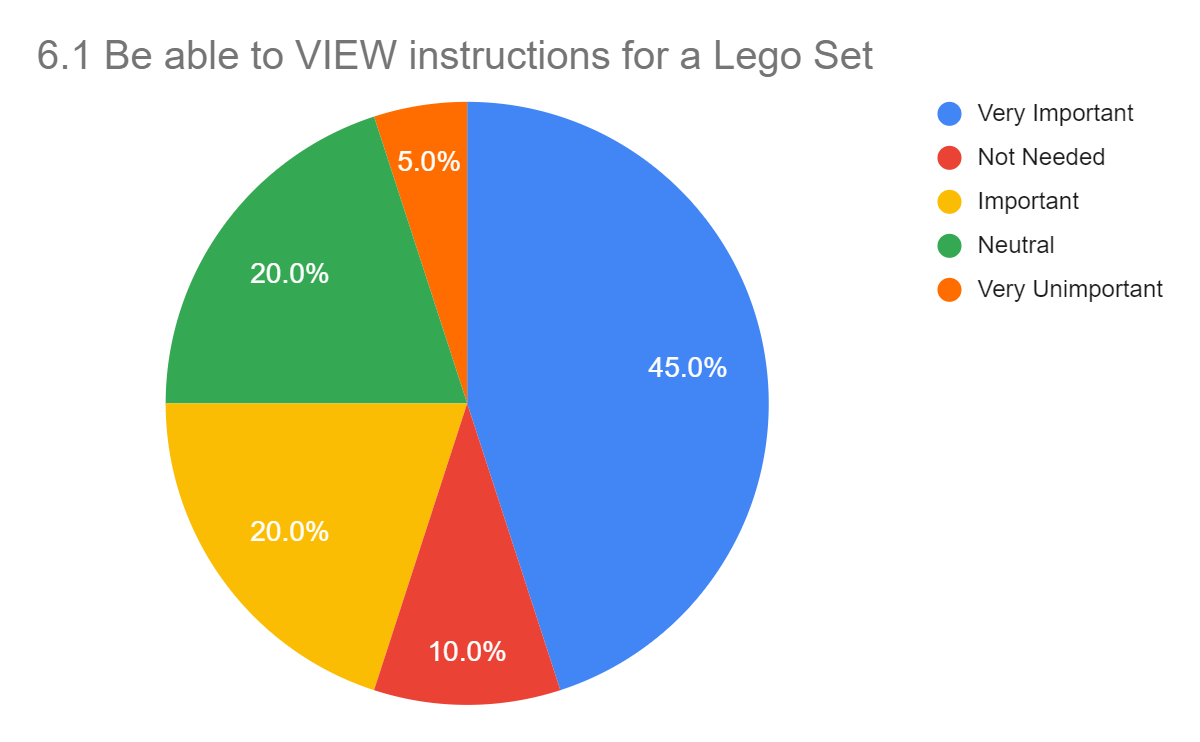
Description automatically generatedGraphical user interface, text, application

Description automatically generatedChart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated



6. How important would the following features be to you in a Digital Checklist for Pieces in a Lego Set ?

Chart, pie chart

Description automatically generated

Chart, pie chart

Description automatically generated

Chart, pie chart

Description automatically generated

Chart, pie chart

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated