DAN Booth

Interim Report

2021

contents

## Aims and Objectives

### Aim

The key aim of this project is 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. They 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 Spring 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

To begin with, I looked up the Rebrickable API [2] that contains data for Lego sets, that I am going to use, that can search through to find a Lego set and the pieces within this 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 with 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 another call to the API needs to be performed. 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 and features for the system, for example 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.

My questionnaire received a reasonable number of responses (20 responses) and from the results of this questionnaire (see **Appendix B**), I was able to decide on certain features and requirements for my project.

Question 1 showed me that the majority of users (75% see **Appendix B**) would like to use the digital checklist on both PC/Laptop and mobile devices, which helped inform me that my digital checklist for pieces in a Lego Set should run on both these types of systems.

I also learnt from questions answers to 2 and 3 (see **Appendix B**) 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 accidentally decreased how many pieces they needed so are missing one, or the opposite where they increase the number they need but 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 answers from question 3 (see **Appendix B**) also show some people currently use the 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 certain pieces needed but not see the original number (like BrickLink see above). 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.

Overall, the results of questions 2 and 3 (see **Appendix B**) has helped me find and research similar software and helped give me ideas on what would be useful to use from them.

The results of question 4 (see **Appendix B**) provided lots of useful information about how users would like to search for a Lego set. Some of the answers were very conclusion for example 19 people said that searching by ‘Age Range’ for a Lego Set was not need showing me that there is no demand for this search parameter. Likewise, all 20 respondents stated they would like to be able to search by ‘Set Number’.

The answers to the ‘Set Pieces’ section of question 4 (see **Appendix B**) are quite varied, with 8 responses saying it was ‘not needed’, which was one more the ‘Filter by’ (7 responses) and one less than ‘Sort by’ (9 responses). This shows filter and sort for ‘Set Pieces’ only just make up the majority of responses showing that maybe this is not a key requirement when searching for a set but would still be useful.

Overall conclusions that I can draw from question 4 is that overall users would like to search by ‘set number’ and ‘set name’ when trying to locate a Lego Set. They would also like to filter by the ‘year made’, ‘theme’ and ‘Set Pieces’ as well as sorting by ‘Theme’, ‘Year Made’ and ‘Set Pieces’.

The results of Question 5 (see **Appendix B**) show that most people find it important/very important that a digital checklist the Lego pieces can be sorted by colour and type of piece, showing this should be a key feature of my digital checklist. However, being able to filter by Lego pieces type and colour are shown to be not as important and therefore are not as key to people. Finally, having a link to buy a missing Lego piece and being able to scan Lego pieces in a set to see if they are there and then check them off, have very mixed answers showing they should be nothing more than optional for the checklist.

Question 6 results (see **Appendix B**) shows that it is very important to most people (17 people) to be able to save progress on a checklist making this a key requirement for the system. Being to view and download instructions is also important to most users as is being able to save Lego sets, they own to a list meaning this is also key. The responses to having a favourites list for Lego sets are very mixed but mostly positive showing that it would be nice to have but not key.

From the responses to question 6 I went and found an API (Brickset API [7]) that I can use to 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.

Finally, the results of question 7 (see **Appendix B**) where users are allowed to add any ideas for any other features gave some useful ideas. For example, being able to also scan bricks with a webcam as well as a mobile for PC/Laptop users. As well as if there are multiple Lego pieces of the same type and colour on the checklist being able to record the specific number of these found. Finally, another feature suggested was to import and export XML files like a Bricklink [4] wanted list. I took these suggestions into account when designing my requirements.

## 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 that allows users to search a list of Lego sets. Can search by set number and text search (e.g. set name), and filter and sort by year made, set theme.
* Must be able to ‘check’ piece off the checklist, showing how many more of that piece are remaining
* Must show on the checklist (for Lego pieces in a set) 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 and type of a Lego piece
* Must be usable with and without a user account
* Must be able to save progress on a checklist

### Nice to have Features:

* Additional search parameter to sort and filter by number of pieces in a Lego set
* Additional search parameter to sort alphabetically by Set Name
* View instructions for a Lego set
* Download instructions for a Lego set
* Filter checklist by the colour of a piece
* Filter checklist by type of a piece
* 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

* Users with an account can create lists for Lego sets and save 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
* Scan Lego pieces with phone camera/webcam to check if a piece is in the list
  + If it is in the set (and not already enough of them), there is an 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
* Import and export a checklist as XML in the format of a Bricklink wanted list [4]

## Outline of Specification and Design

Chart

Description automatically generated with low confidence

Figure - High-Level Architecture

## Planning and Timescales

Table

Gantt charts

## 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

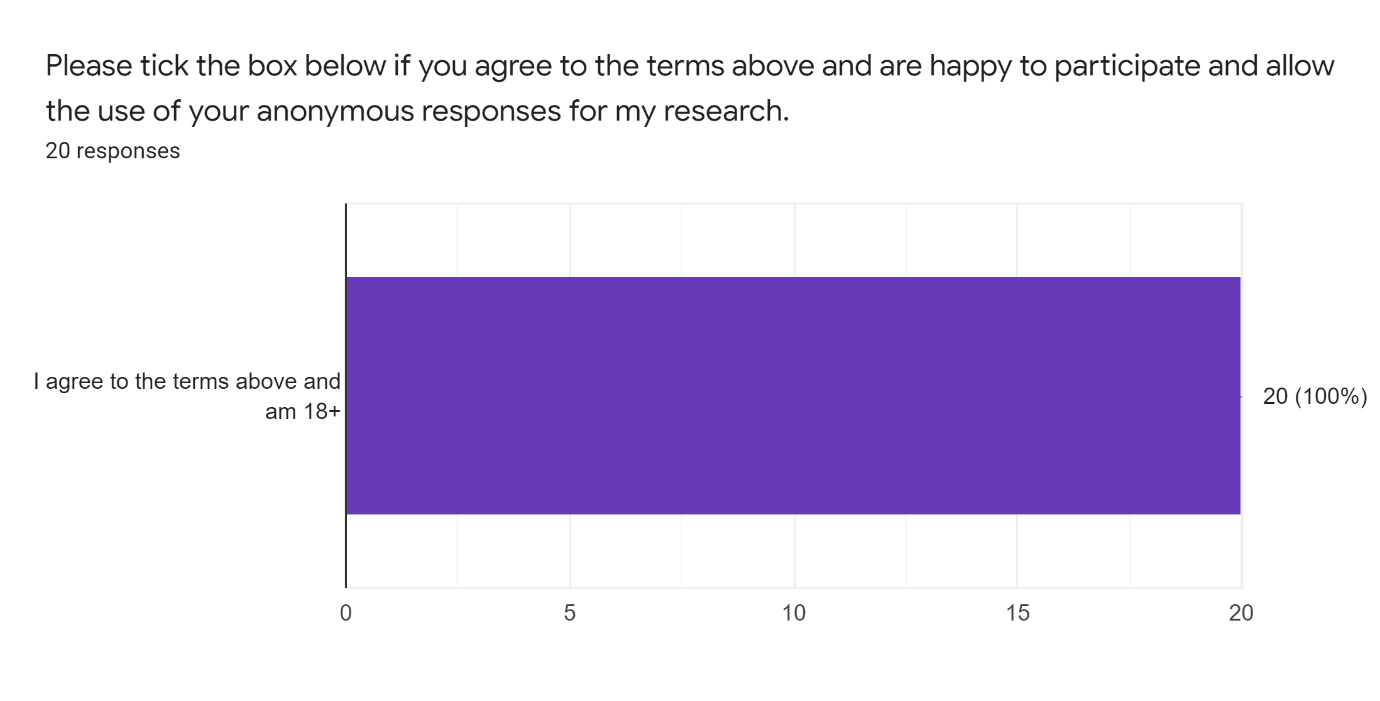
Description automatically generatedText

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

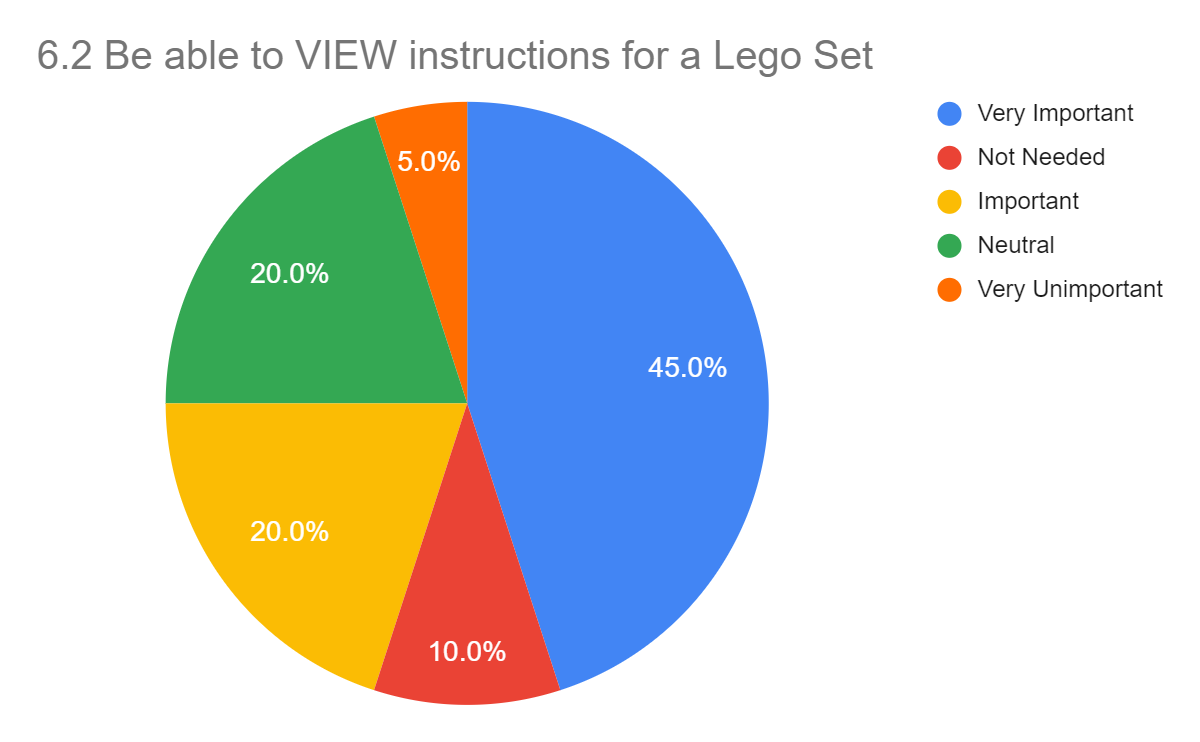
Description automatically generated

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

Chart, pie chart

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated