Lego: Set Checklist Creator

DAN Booth

Interim Report

2021

# Table of Contents

## Aims and Objectives

I am currently completing a project to create a digital checklist for pieces in a Lego Set. This will be where people can search for a Lego Set and then click on the Set they would like pieces for. They will then be presented with all the pieces in this Lego set (like in the back of the instruction book), and can tick the pieces off the list when they are building the set again, or believe that they have the right pieces to build another set.

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.

I am going to produce a digital Checklist for pieces in a Lego Set, that will be a mobile and/or website application,

Challenges I expect to face during the project are

* Linking the rebrickable to a mobile application and/or a 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” ()

I performed some data collection on what features my target users would like from a digital checklist for pieces in a Lego Set, via a questionnaire. 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

-results and findings

From my research I found that Lego builders/enthusiasts/collectors who use a digital tool use the website Bricklink [4]. 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.

On the Rebrickable website , 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 Star Wars Set) 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 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.

On Rebrickable website [5], 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 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

## Outline of Specification and Design

### Key Features:

* A search feature that allows users to search by name, Set Number, Year Made, Theme (e.g. marvel, dc, star wars, Lego city, Indian Jones) to locate the required Lego set

### Possible Features:

* Tick off brick off the checklist, showing how many more of that brick are remaining (able to undo if wrong brick clicked)
* Filter brick colour and/or type when checking bricks off the list
* The checklist shows a picture of the piece (with right colour) as well as a description including piece name and colour
* Scan a brick with a phone camera
  + Tick off list if it is in the set (and not already enough of them)
  + If not in set it will inform the user
  + If in the set but already enough it will inform the user
* If a brick is missing the user can click a link to a website, where they can buy the missing piece/pieces
* Be able to view/download instructions for the current Lego set
* Be able to save current checklist progress so that the user can return to it at a later date
* Be able to favourite sets the user owns and add them to a favourites/my Lego sets list, that the user can also search
* Users can use the system with or without an account, but can only save checklist progress and favourite sets with an account.

## Planning and Timescales

## 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