

Contents

1 Analysis	2
1.1 Problem Identification	2
1.1.1 Problem Description	2
1.1.2 Stakeholders	2
1.1.3 Interview	2
1.1.4 Existing similar solutions	3
1.1.5 Features to be incorporated into solution	5
1.1.6 Feedback from stakeholders	5
1.2 Requirements	5
1.2.1 Stakeholder requirements	5
1.2.2 Software and hardware requirements	6
1.2.3 Success requirements	7
2 Design	8
2.1 User Interface Design	8
2.1.1 Usability Features	8
2.1.2 Feedback from stakeholder	8
2.2 Modular breakdown	8
2.3 Algorithms	8
2.4 Data Dictionary	8
2.5 Inputs and outputs	8
2.6 Validation	8
2.7 Testing	8
2.7.1 Methods	8
2.7.2 Test Plan	8
3 Implementation	8
3.1 First Iteration	8
4 Testing	8
5 Evaluation	8

1 Analysis

1.1 Problem Identification

1.1.1 Problem Description

Popular inventory management solutions are relatively expensive, and may be out of reach for individuals or small schools. Inventory systems have numerous benefits for businesses and individuals alike; a business may choose to track their supply levels where an individual may wish to catalogue their DVD collection.

My goal is to create a web-based application aimed at both businesses and individuals to manage inventory, with additional modern features such as automatic item re-ordering when stocks are running low.

Traditional inventory management solutions are typically single-user at best, whereas I intend to create a multi-user, collaborative environment.

In my view, an inventory system should be:

- Easy for end users to use.
- Cross platform
- Performant interface
- Efficient in terms of adding data
- Allow for easy cataloguing of inventory
- Allow for item scanning using QR codes / barcodes
- Be able to source data from external sources
- Support both consumable and non-consumable goods.

1.1.2 Stakeholders

Stakeholder	Description	Current Use	Requirements
Claire Foley	Senior Leadership Team at The Village Prep School	Library books; currently tracked on paper	TBD
TBD	Freelance photographer	Excel Spreadsheet	TBD

1.1.3 Interview

Question Set

- What would you consider your skill level to be regarding technology?
- Do you currently have a way to manage inventory?
 - If so, what is your current solution?
 - * What aspects of this solution do you like?
 - * What aspects of this solution do you dislike?
 - What features would you **require** in a custom solution?
 - What features would **enhance** your experience?

1.1.4 Existing similar solutions

InvenTree <https://inventree.org/>

Overview

InvenTree is an **open-source** inventory management system, providing *low level stock control and part tracking*. It uses a Python/Django database backend and provides both a **web-based interface** as well as a REST API for interacting with other services. InvenTree also has a powerful plugin system for custom applications and other extensions.

Below is a screenshot of the InvenTree homepage.

Build	Description	Project Code	Priority	Part	Progress	Status	Created	Issued by	Responsible	Target
BO0016	Making widgets for SO 0003	-	0	Widget Assembly Variant	0 / 75	Pending	2022-05-25	admin	-	-
BO0014	Making tables for SO 0003	-	0	Blue Square Table	0 / 100	Pending	2022-05-25	admin	-	-
BO0013	Required parts for Build 0010	-	0	TB3 Test Board 3	0 / 50	Pending	2022-05-25	admin	-	-
BO0011	Required parts for Build 0010	-	0	TB1 Test Board 1	25 / 50	Production	2022-05-25	admin	-	-
BO0010	Making a high level assembly part	-	0	MAST Master Assembly	0 / 50	Pending	2022-05-25	admin	-	-
BO0007	Making red square tables	-	0	Red Square Table	0 / 15	Production	2022-04-29	allaccess	-	-

Parts applicable to my solution

The core concept is similar (the application is web-based), but my solution will be more generalized that just stock control.

PartKeepr <https://partkeepr.org/>

The screenshot shows the PartKeepr web application interface. On the left is a 'Categories' sidebar with a tree view of electronic components. The main area displays a 'Parts List' for the category '0 Replimat > Connectors > JST PH (8 Part(s))'. The table lists various JST PH components with their descriptions, storage locations, status, condition, stock, minimum stock, average price, footprint, and internal ID.

Name	Description	Storage Location	Status	Condition	Stock	Min. Stock	Avg. Price	Footprint	Internal ID
Root Category > 0 Replimat (4 Part(s))									
100R 0805 145	0805W8F1501TSE	REPLIMAT			50 pcs	0 pcs	0.00€		1826 (#1eq)
100R 1/8W THT 5%		REPLIMAT			100 pcs	0 pcs	0.00€		1827 (#1er)
BAT85		REPLIMAT			12 pcs	0 pcs	0.00€		1839 (#173)
KW10 Microswitch		REPLIMAT			0 pcs	0 pcs	0.00€		1830 (#1eu)
Root Category > 0 Replimat > Connectors (2 Part(s))									
Pin Header 2x4p 90°		REPLIMAT			0 pcs	0 pcs	0.00€		1849 (#1fd)
Socket Header 2x4p		REPLIMAT			0 pcs	0 pcs	0.00€		1848 (#1fc)
Root Category > 0 Replimat > Connectors > JST PH (8 Part(s))									
JST PH Crimp Contact		REPLIMAT			1270 pcs	0 pcs	0.00€		1847 (#1fb)
JST PH Housing 2p		REPLIMAT			160 pcs	0 pcs	0.00€		1844 (#1fB)
JST PH Housing 3p		REPLIMAT			140 pcs	0 pcs	0.00€		1845 (#1f9)
JST PH Housing 4p		REPLIMAT			370 pcs	0 pcs	0.00€		1846 (#1fa)
JST PH THT 2p 90°		REPLIMAT			80 pcs	0 pcs	0.00€		1840 (#1f4)
JST PH THT 3p		REPLIMAT			57 pcs	0 pcs	0.00€		1843 (#1f7)
JST PH THT 3p 90°		REPLIMAT			40 pcs	0 pcs	0.00€		1842 (#1f6)
JST PH THT 4p		REPLIMAT			49 pcs	0 pcs	0.00€		1853 (#1fh)
Root Category > 0 Replimat > Connectors > JST XH (9 Part(s))									
XH Crimp Contacts		REPLIMAT			223 pcs	0 pcs	0.00€		1831 (#1ev)
XH Housing 2p		REPLIMAT			67 pcs	0 pcs	0.00€		1816 (#1eq)
XH Housing 3p		REPLIMAT			57 pcs	0 pcs	0.00€		1817 (#1eh)
XH Housing 4p		REPLIMAT			52 pcs	0 pcs	0.00€		1818 (#1ei)
XH THT 2p		REPLIMAT			50 pcs	0 pcs	0.00€		1815 (#1ef)
XH THT 2p 90°		REPLIMAT			75 pcs	0 pcs	0.00€		1811 (#1eb)
XH THT 3p		REPLIMAT			56 pcs	0 pcs	0.00€		1813 (#1ed)
XH THT 4p		REPLIMAT			21 pcs	0 pcs	0.00€		1814 (#1ee)
XH THT 4p 90°		REPLIMAT			31 pcs	0 pcs	0.00€		1812 (#1ec)
Root Category > 0 Replimat > Connectors > MF3.0 (13 Part(s))									

Overview

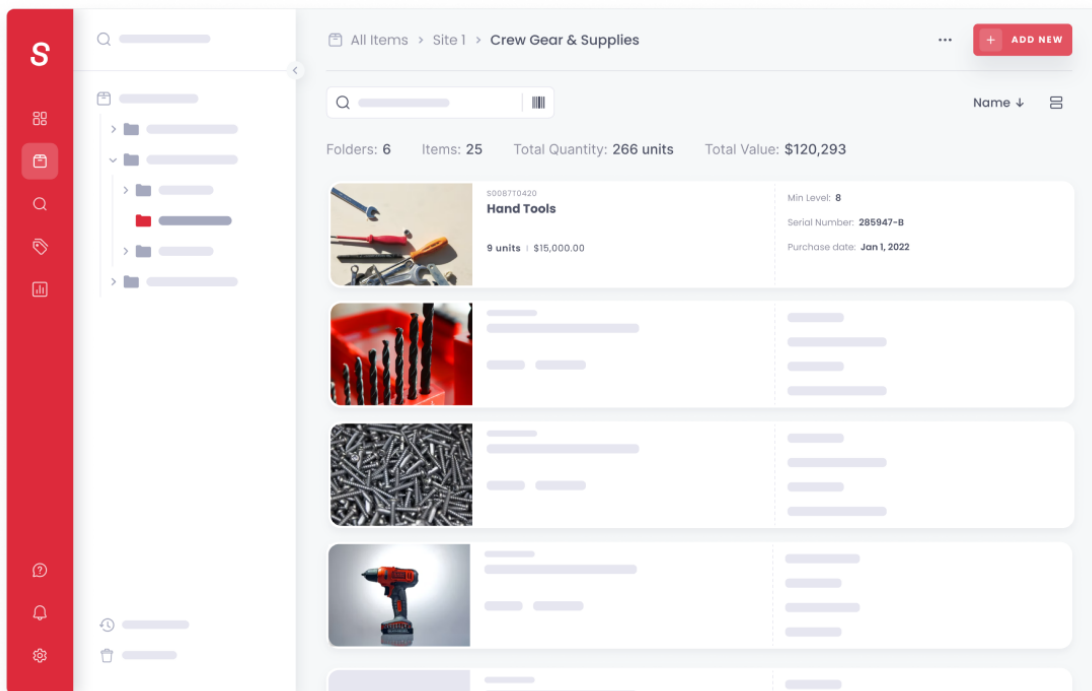
PartKeepr is an open-source inventory management system with a focus on electronic components. It is designed around four main principles:

- Fast Part Searching
- Ability to add complete part database
- Keeping track of stock
- Ease of use

Parts applicable to my solution

Like PartKeepr, I hope to implement a web-based interface. However, I am using a different approach as my solution will not be tailored specifically to electronic components.

Sortly <https://www.sortly.com/solutions/inventory-management-software/>



Overview

Sortly is a proprietary cloud-based inventory management system with a focus on small businesses and individuals.

It has two plans available, an always free plan with limited functionality and a paid plan with a more complete feature-set.

Parts applicable to my solution

I hope to implement the following features from Sortly:

- Web based interface
 - Allows for easy access.
- Barcode support
 - Allows end users to print off QR codes to stick to items
 - Which can be scanned in-app to easily perform actions on the item.
- Real-time reporting insights
 - Allows for added insight into usage patterns for particular units.

1.1.5 Features to be incorporated into solution

1.1.6 Feedback from stakeholders

1.2 Requirements

1.2.1 Stakeholder requirements

1.2.2 Software and hardware requirements

System Requirements

Hardware	Justification
Input Device (<i>Touchscreen or Keyboard+Mouse</i>)	An input device will help users navigate through the program as it will have a graphical user interface (GUI). This will allow the user to interact with buttons, menus and icons.

Software Requirements

Software	Justification
Operating System (<i>Windows, MacOS, Linux, ChromeOS, iOS, iPadOS, Android</i>)	An operating system is required to... TBD
A web browser (<i>eg. Chrome, Firefox, Microsoft Edge, Safari</i>)	TBD

1.2.3 Success requirements

The overall objectives for the system.

To measure the overall effectiveness of the system, targets must be set before writing the program. These targets will help in the evaluation stage to determine whether our objectives have been met. These objectives will be **SMART**, i.e:

- **Specific**
What objective needs to be accomplished?
- **Measurable**
How can we quantify this objective?
How will the success of this objective be measured? (quantitatively or qualitatively)
- **Achievable**
Is this objective achievable and realistic? If so, how do you plan to achieve them?
- **Relevant**
How does this objective benefit the end-users of this application as a whole?
Why has this goal been set?
- **Timely**
Can this objective be completed within an appropriate time frame?
At what stage in the software development lifecycle will you start implementing this goal?
In which order will any sub-objectives be completed?

The Project's SMART Objectives

1. **To produce a solution for cataloguing a school library and recording users and books borrowed**
At the end of the project, I will evaluate against my success criteria and determine whether this objective has been met. On the software side, I will be using React, Expo and PostgreSQL. This objective will be the main objective for this project. This objective must be completed by March 2024.
2. **To produce a solution including a database that can store details of books, borrowers, loans and returns**
3. **To produce an intuitive and easy to use solution**
I will evaluate my success on this objective by having a new user without any prior training or advice use the system and try to carry out a number of tasks without any assistance. If the user is able to successfully complete the tasks I will consider the system to be intuitive and easy to use and therefore this objective satisfied. To achieve this I will design my system to have a consistent layout based on **Material Design 2**, (<https://m2.material.io/>) the design language used by Google products and many apps running on the Android operating system. I will also use language that is a) appropriate for the situation the product will be deployed in (with young children) and b) easy to understand (so that children can interact with the system) I will also use meaningful error messages so that the user has a clear understanding of the problem that has occurred.
4. **To produce a solution that features a fully searchable catalogue**
5. **To produce a solution that features reporting for overdue and/or lost books**
6. **To produce a solution that includes a curated "suggested reading list" for each borrower**

2 Design

2.1 User Interface Design

2.1.1 Usability Features

2.1.2 Feedback from stakeholder

2.2 Modular breakdown

2.3 Algorithms

2.4 Data Dictionary

2.5 Inputs and outputs

2.6 Validation

2.7 Testing

2.7.1 Methods

2.7.2 Test Plan

3 Implementation

3.1 First Iteration

4 Testing

5 Evaluation