

Contents

1	Analysis	2
1.1	Problem Identification	2
1.1.1	Problem Description	2
1.1.2	Stakeholders	2
1.1.3	Why is it suitable to a computational solution?	3
1.2	Investigation	3
1.2.1	Preparation for interview	3
1.2.2	Interviews	3
1.2.3	Summary of interviews	3
1.3	Research	4
1.3.1	Existing similar solutions	4
1.3.2	Features to be incorporated into solution	7
1.3.3	Limitations of the solution	7
1.3.4	Feedback from stakeholders	7
1.4	Requirements	7
1.4.1	Stakeholder requirements	7
1.4.2	Software and hardware requirements	8
1.4.3	Success requirements	9
2	Design	10
2.1	User Interface Design	10
2.1.1	Usability Features	10
2.1.2	Feedback from stakeholder	10
2.2	Modular breakdown	10
2.3	Algorithms	10
2.4	Data Dictionary	10
2.5	Inputs and outputs	10
2.6	Validation	10
2.7	Testing	10
2.7.1	Methods	10
2.7.2	Test Plan	10
3	Implementation	11
3.1	First Iteration — Initial Backend and Database	11
3.1.1	Introduction	11
4	Testing	11
5	Evaluation	11

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 requirements are further discussed for each stakeholder in the Stakeholder Requirements section.

Stakeholder	Description	Requirements	Capability
Claire Foley	Senior Leadership Team at The Village Prep School	Ability to manage library books. Admin and supervision of other users carrying out librarian tasks	Well-versed in computer use, at least when it comes to intuitive and well designed interfaces. Would struggle with a non-intuitive interface design.
Ella	"Head Librarian" (Pupil) at the Village Prep School	Ability to check in and out library books. User of the system; requires interface that is appropriate for her age.	Beginner user of technology, proficient in mobile applications on phones and tablets only. Rarely uses a laptop or desktop computer.
Generic Gear Rental Shop	Photography gear for hire business	Ability to manage business inventory in a fast and efficient manner.	Proficient with computers.

1.1.3 Why is it suitable to a computational solution?

1.2 Investigation

1.2.1 Preparation for 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?

1.2.2 Interviews

1.2.3 Summary of interviews

1.3 Research

1.3.1 Existing similar solutions

InvenTree <https://inventree.org/>

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	-	-

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.

Parts applicable to my solution

- Web-based application
The application will be web-based.
- Modern, Relatively simple user interface
InvenTree offers a relatively simple and intuitive user interface.

Parts not applicable to my solution

- Stock control and part tracking specific features
I am looking to implement a system that is capable of being far more generalized than just part tracking, although the system will have features for library book tracking.

PartKeepr <https://partkeepr.org/>

The screenshot displays the PartKeepr web application. On the left is a 'Categories' sidebar with a tree view of electronic components like Connectors, Electronics, and Semiconductors. The main area shows a 'Parts List' table with columns for Name, Description, Storage Location, Status, Condition, Stock, Min. Stock, Avg. Price, Footprint, and Internal ID. The table is filtered to show parts under the 'Root Category > 0 Replimat > Connectors' path. The bottom status bar indicates the user is logged in as 'admin' on 9/19/2023.

Name	Description	Storage Location	Status	Condition	Stock	Min. Stock	Avg. Price	Footprint	Internal ID
Root Category > 0 Replimat (4 Part(s))									
100R 0805 1K5	0805W8F1501T5E	REPLIMAT			50 pcs	0 pcs	0.00E		1826 (#1eq)
100R 1/8W THT 5%		REPLIMAT			100 pcs	0 pcs	0.00E		1827 (#1er)
BAT85		REPLIMAT			12 pcs	0 pcs	0.00E		1839 (#1f3)
KW10 Microswitch		REPLIMAT			0 pcs	0 pcs	0.00E		1830 (#1eu)
Root Category > 0 Replimat > Connectors (2 Part(s))									
Pin Header 2x4p 90°		REPLIMAT			0 pcs	0 pcs	0.00E		1849 (#1ld)
Socket Header 2x4p		REPLIMAT			0 pcs	0 pcs	0.00E		1848 (#1lc)
Root Category > 0 Replimat > Connectors > JST PH (8 Part(s))									
JST PH Crimp Contact		REPLIMAT			1270 pcs	0 pcs	0.00E		1847 (#1th)
JST PH Housing 2p		REPLIMAT			160 pcs	0 pcs	0.00E		1844 (#18)
JST PH Housing 3p		REPLIMAT			140 pcs	0 pcs	0.00E		1845 (#19)
JST PH Housing 4p		REPLIMAT			370 pcs	0 pcs	0.00E		1846 (#1fa)
JST PH THT 2p 90°		REPLIMAT			80 pcs	0 pcs	0.00E		1840 (#14)
JST PH THT 3p		REPLIMAT			57 pcs	0 pcs	0.00E		1843 (#17)
JST PH THT 3p 90°		REPLIMAT			40 pcs	0 pcs	0.00E		1842 (#16)
JST PH THT 4p		REPLIMAT			49 pcs	0 pcs	0.00E		1853 (#1th)
Root Category > 0 Replimat > Connectors > JST XH (9 Part(s))									
XH Crimp Contacts		REPLIMAT			223 pcs	0 pcs	0.00E		1831 (#1ev)
XH Housing 2p		REPLIMAT			67 pcs	0 pcs	0.00E		1816 (#1eq)
XH Housing 3p		REPLIMAT			57 pcs	0 pcs	0.00E		1817 (#1eh)
XH Housing 4p		REPLIMAT			52 pcs	0 pcs	0.00E		1818 (#1ei)
XH THT 2p		REPLIMAT			50 pcs	0 pcs	0.00E		1815 (#1ef)
XH THT 2p 90°		REPLIMAT			75 pcs	0 pcs	0.00E		1811 (#1eb)
XH THT 3p		REPLIMAT			56 pcs	0 pcs	0.00E		1813 (#1ed)
XH THT 4p		REPLIMAT			21 pcs	0 pcs	0.00E		1814 (#1ee)
XH THT 4p 90°		REPLIMAT			31 pcs	0 pcs	0.00E		1812 (#1ec)
Root Category > 0 Replimat > Connectors > MF3.0 (13 Part(s))									

Overview

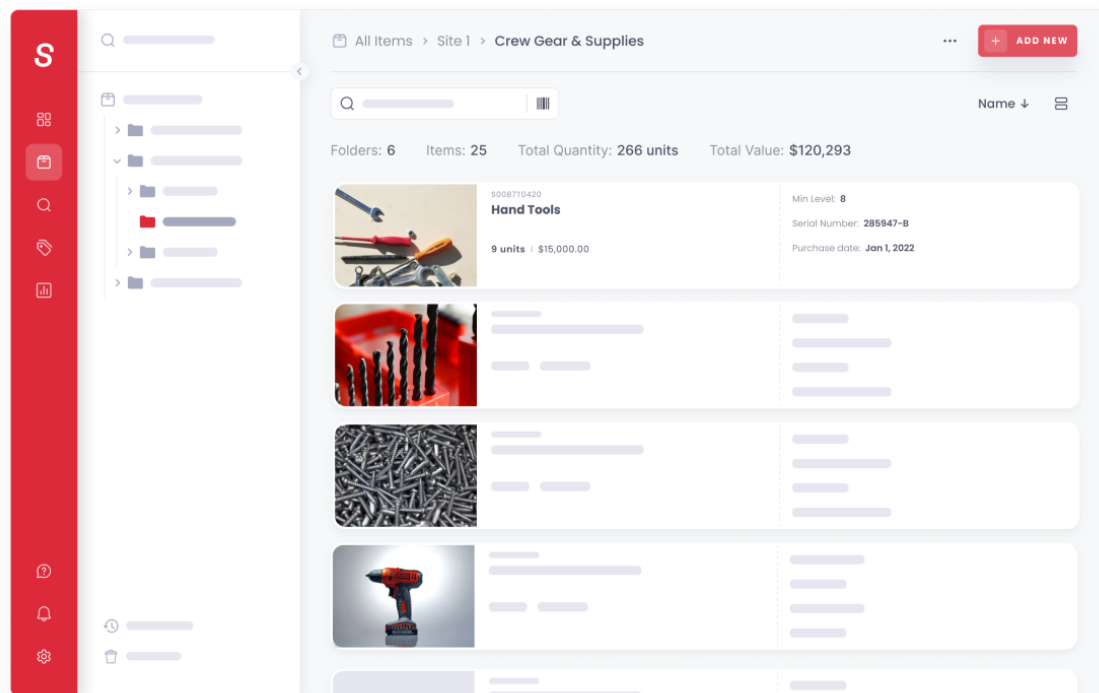
PartKeepr is an open-source inventory management system with a focus on electronic components. It is designed around four main principles:

- 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

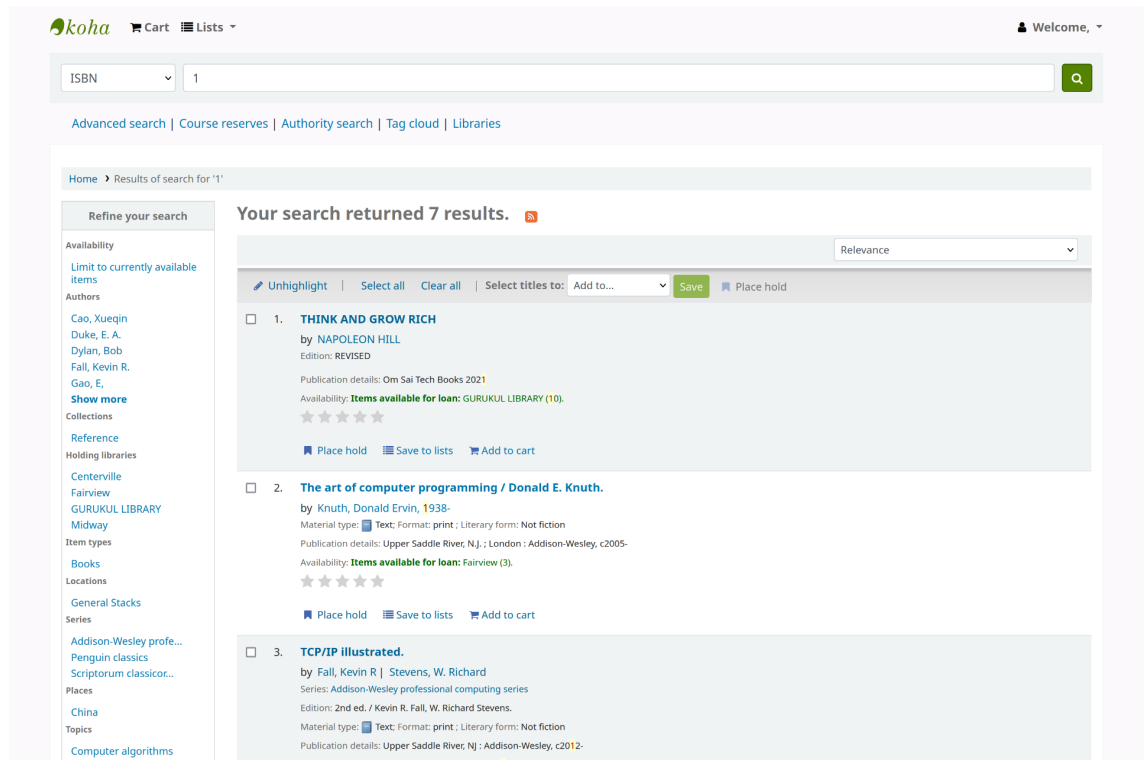
Sortly is a proprietary cloud-based inventory management system with a focus on small businesses and individuals.

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.

Koha<https://koha-community.org/>**Overview****Parts applicable to my solution****1.3.2 Features to be incorporated into solution****1.3.3 Limitations of the solution****1.3.4 Feedback from stakeholders****1.4 Requirements****1.4.1 Stakeholder requirements**

1.4.2 Software and hardware requirements

System Requirements

Hardware	Justification
<i>Laptop/Desktop</i> Keyboard and Pointing Device (eg. Mouse)	For desktop or laptop computer users, a suitable input device is required in order to interact with the software. A pointing device (a mouse) is necessary in order to interact with the user interface, to perform actions such as clicking buttons, icons, and opening menus. A keyboard will be used to manually input data into the system.
<i>Tablet Device</i> Touchscreen	For tablet users, it would be impractical to expect the user to have access to a keyboard and or pointing device. Therefore, we must design the system to accept inputs from a touchscreen. This will be easier to use and more intuitive for tablet users. The touchscreen will be used to input data into the system and to interact with the user interface.
Dual-Core Processor (x86, ARM, RISC-V architectures)	A modern processor with sufficient resources to run an up-to-date web browser such as Chrome, Edge or Firefox is required in order to access the web-based interface.
2GB of RAM	Sufficient RAM is required to run the web browser, which can be a memory intensive task.
Monitor	To display the user interface.
Network Interface Card (NIC)	A Network Interface Card, or NIC, is required for the computer to be connected to a network, such as the Internet. This is required as the web interface will be hosted on a domain and server that is external to the user, that is to say, not on their local network.
Optional: Wireless Network Adapter	<i>A Wireless Network Adapter is an optional requirement, it will allow the user to connect to a wireless network in order to access the network or Internet so that they can access the external user interface.</i>
Optional: Camera	<i>A Camera is an optional requirement; devices with cameras will be able to scan barcodes or QR codes corresponding to inventory items and easily perform actions on them.</i>

Software Requirements

Talk about why I don't need much software since dependencies hosted on the server.

Software	Justification
Operating System (Windows, MacOS, Linux, ChromeOS, iOS, iPadOS, Android)	An operating system is required in order to run the web browser necessary to access the interface.
A web browser (eg. Chrome, Firefox, Microsoft Edge, Safari)	A web browser is necessary to access the interface as it will be primarily a web application.

1.4.3 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. This objective will benefit the end user as a intuitive and easy to use solution is critical to the usefulness of the project. If the end product is not easy to use, it is less likely to be used and accepted by my stakeholders. This objective will be worked on during the development process, and so will be completed by the time development concludes. I will mock-up a version of the user interface in the design stage and will continuously iterate on the user interface during development.
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**
7. **To produce a solution containing a user interface that can be accessed via a mobile device**

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 — Initial Backend and Database

3.1.1 Introduction

For my first iteration I decided to work on the backend and the database. I am writing the backend in Go. Go is a performant, statically typed high level language designed by Google. It is frequently used for backend development thanks to its performance and memory safety. I am going to use GraphQL as the query language for the frontend to interact with the backend. GraphQL is an open-source query and manipulation language designed for use in APIs. (Application Programming Interface). The backend will serve as an API which will interface with my database. I choose to use GraphQL as it is better suited for larger, more complex data sources, and supports querying for multiple different types of data at once, unlike REST. It is also something I was interested in learning more about as I have not designed a system using it before.

The first thing I decided to work on was user account creation. This would involve using GraphQL to receive data about the new user, generating the user ID, and then placing that data in a database. After doing some research (should I talk about the different options?) I decided to use the following three libraries for my project:

- **Gin**
Gin is a HTTP framework for Go. This allows me to expose my GraphQL endpoint over HTTP. After conducting some research, I choose to use gin as it is by far the fastest library available at the time of writing. It offers performance up to 40 times faster than its closest alternative, "Martini". Gin is a very popular framework, with over 72 thousand stars on GitHub. Using this library benefits me as it has been battle-tested by other users and includes end to end testing, making it more than suitable for my project. (wording isn't great, trying to say it has been tested already)
- **Gqlgen**
Gqlgen is a Go library designed to make building GraphQL servers easy and hassle-free. It is designed with a schema-first approach, meaning the developer (me) can simply define their API using the standard GraphQL Schema Definition Language. It prioritizes type safety, and has rudimentary validation support built-in. However, I will not be relying on this validation feature, I will create my own validation system that better suits my needs.
- **GORM** (Go Object Relational Mapping library)
GORM is a fully-features ORM library for Go. (explain what an ORM does)

(skip to 14.11.23, after voice memos)

4 Testing

5 Evaluation