

Contents

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

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 am to create a multi-user, collaborative environment.

1.1.2 Interview

1.1.3 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

- concept is similar (web-based), but I'm doing a different approach.
- not indented for stock control

PartKeepr <https://partkeepr.org/>

The screenshot displays the PartKeepr web application. On the left is a 'Categories' sidebar with a tree view showing a hierarchy from 'Root Category' down to specific components like 'JST PH' and 'MF3.0'. The main area is titled 'Parts List' and shows a table of parts. The table has columns for Name, Description, Storage Location, Status, Condition, Stock, Min. Stock, Avg. Price, Footprint, and Internal ID. The parts are grouped by category, such as 'Root Category > 0 Replimat > Connectors > JST PH (8 Part(s))'. The status of all parts is 'REPLIMAT'. The stock column shows various quantities, and the avg. price is consistently 0.006. The internal IDs are unique for each part.

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	0.006		1826 (#1ec)
100R 1/8W THT 5%		REPLIMAT			100 pcs	0 pcs	0.006		1827 (#1er)
BAT85		REPLIMAT			12 pcs	0 pcs	0.006		1839 (#1f3)
KW10 Microswitch		REPLIMAT			0 pcs	0 pcs	0.006		1830 (#1eu)
Root Category > 0 Replimat > Connectors (2 Part(s))									
Pin Header 2x4p 90°		REPLIMAT			0 pcs	0 pcs	0.006		1849 (#1td)
Socket Header 2x4p		REPLIMAT			0 pcs	0 pcs	0.006		1848 (#1tc)
Root Category > 0 Replimat > Connectors > JST PH (8 Part(s))									
JST PH Crimp Contact		REPLIMAT			1270 pcs	0 pcs	0.006		1847 (#1tb)
JST PH Housing 2p		REPLIMAT			160 pcs	0 pcs	0.006		1844 (#1rb)
JST PH Housing 3p		REPLIMAT			140 pcs	0 pcs	0.006		1845 (#1r9)
JST PH Housing 4p		REPLIMAT			370 pcs	0 pcs	0.006		1846 (#1ta)
JST PH THT 2p 90°		REPLIMAT			80 pcs	0 pcs	0.006		1840 (#1f4)
JST PH THT 3p		REPLIMAT			57 pcs	0 pcs	0.006		1843 (#1f7)
JST PH THT 3p 90°		REPLIMAT			40 pcs	0 pcs	0.006		1842 (#1f6)
JST PH THT 4p		REPLIMAT			49 pcs	0 pcs	0.006		1853 (#1th)
Root Category > 0 Replimat > Connectors > JST XH (9 Part(s))									
XH Crimp Contacts		REPLIMAT			223 pcs	0 pcs	0.006		1831 (#1ev)
XH Housing 2p		REPLIMAT			67 pcs	0 pcs	0.006		1816 (#1eg)
XH Housing 3p		REPLIMAT			57 pcs	0 pcs	0.006		1817 (#1eh)
XH Housing 4p		REPLIMAT			52 pcs	0 pcs	0.006		1818 (#1ei)
XH THT 2p		REPLIMAT			50 pcs	0 pcs	0.006		1815 (#1ef)
XH THT 2p 90°		REPLIMAT			75 pcs	0 pcs	0.006		1811 (#1eb)
XH THT 3p		REPLIMAT			56 pcs	0 pcs	0.006		1813 (#1ed)
XH THT 4p		REPLIMAT			21 pcs	0 pcs	0.006		1814 (#1ee)
XH THT 4p 90°		REPLIMAT			31 pcs	0 pcs	0.006		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

1.1.4 Features to be incorporated into solution

1.1.5 Feedback from stakeholders

1.2 Requirements

1.2.1 Stakeholder requirements

1.2.2 Software and hardware requirements

1.2.3 Success requirements

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