



Smart-Inventory

Demo 2



Our Client

Amazon Warehouse Services

About Smart Inventory

What is Smart Inventory?

Smart Inventory is a user-friendly, web-based application that streamlines and automates core inventory processes.

Who?

- Business Owners/Managers (Administrators):
- Employees (Inventory Controllers/End-Users):



Our Team



Tristan Sutherland
Project Manager



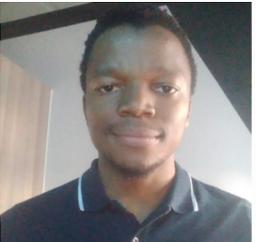
Bryce Sukhdeo



Hawa Ibrahim



Bouchaib Chraf



Thabang Kgaladi

Links To Documentation



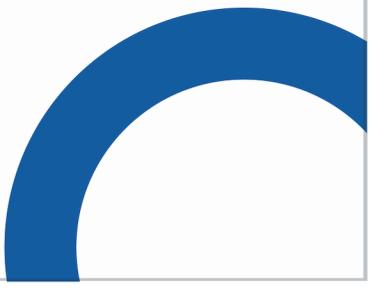
[**ReadMe**](#)
[**Requirement Specification**](#)
[**Architectural Specification**](#)
[**Design Specification**](#)
[**User Manual**](#)
[**Coding Standards**](#)
[**Service Contract**](#)
[**Testing Policy**](#)



Agenda



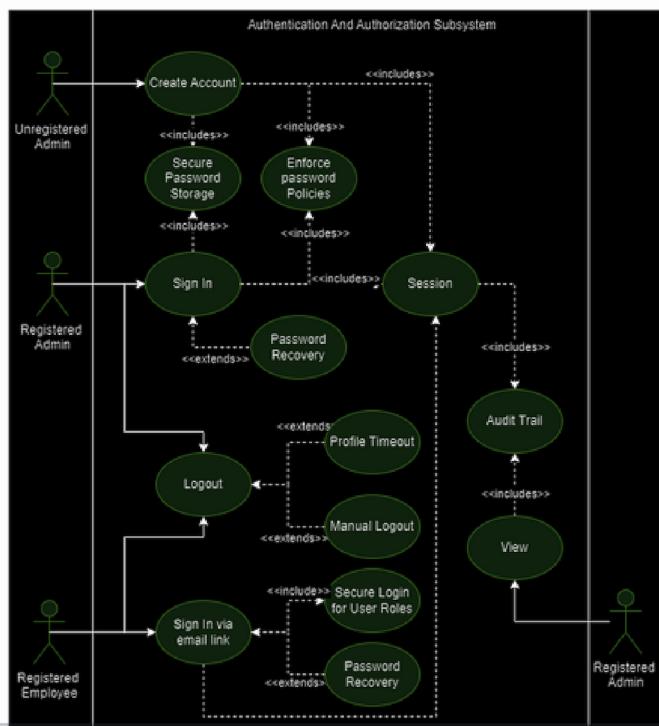
- 0 **Introduction**
- 1 **Documentation**
- 2 **Implementation Progress**
- 3 **Project Board**
- 4 **CI/CD**



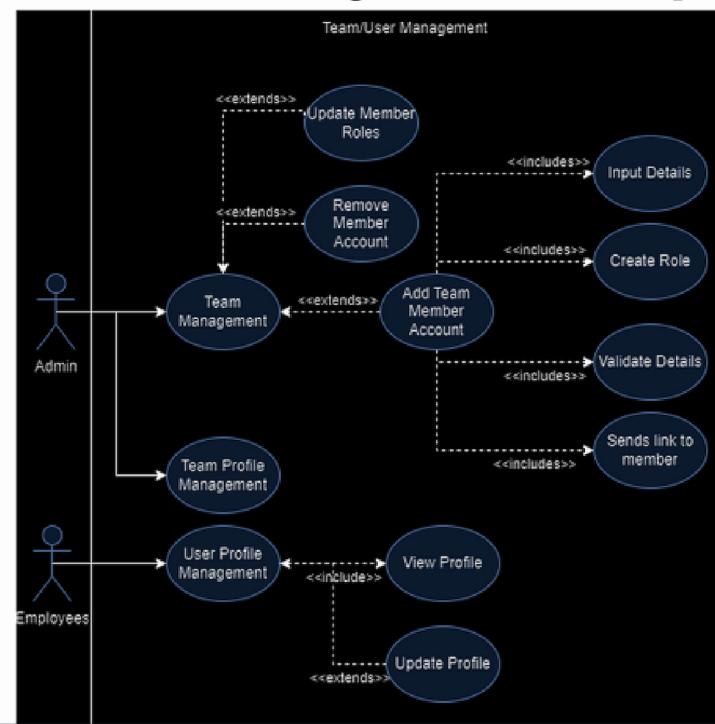
Subsystems

- 1. Authentication and Authorization Subsystem**
- 2. Team/User Management Subsystem**
- 3. Reporting Subsystem**
- 4. Inventory Management Subsystem**
- 5. Stock Request Subsystem**
- 6. Supplier Management Subsystem**
- 7. Order Placement Subsystem**

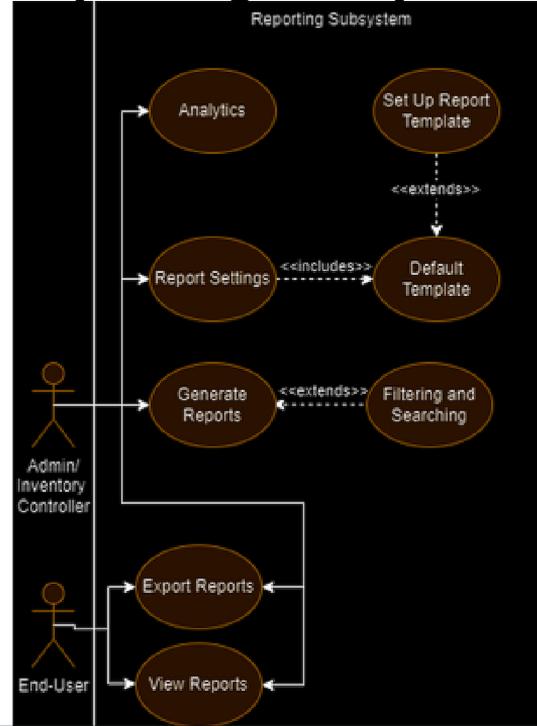
Authorization/ Authentication Subsystem



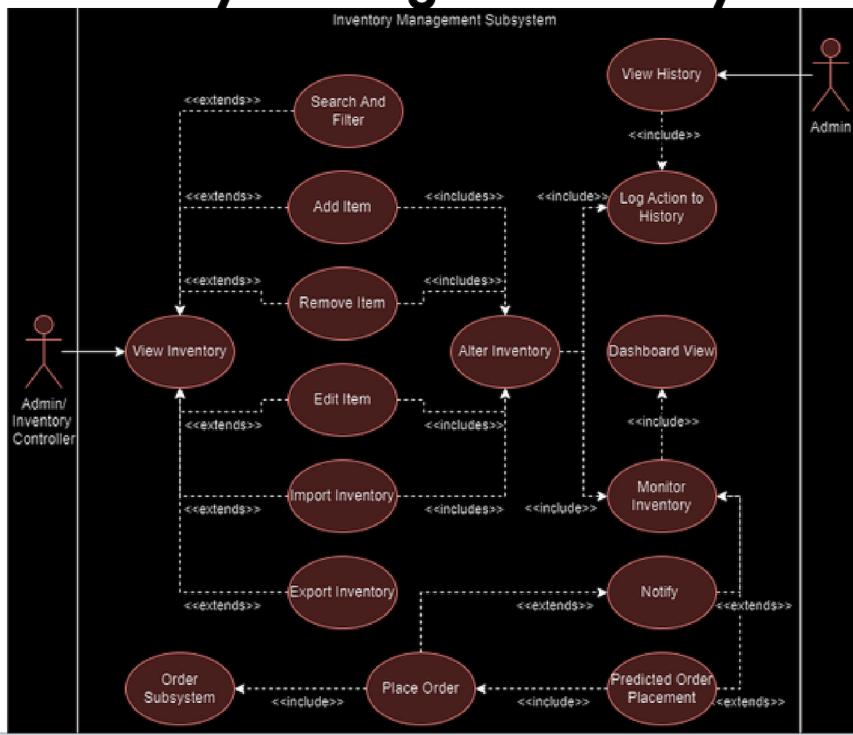
Team/User Management Subsystem



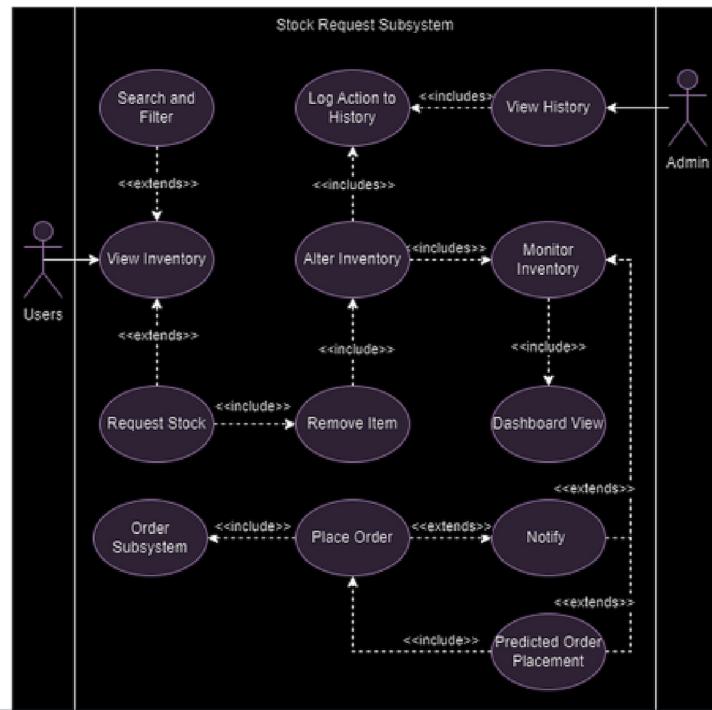
Reporting Subsystem



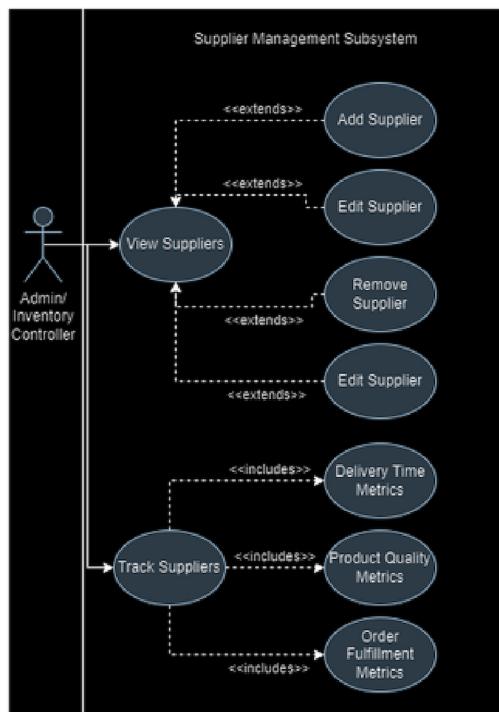
Inventory Management Subsystem



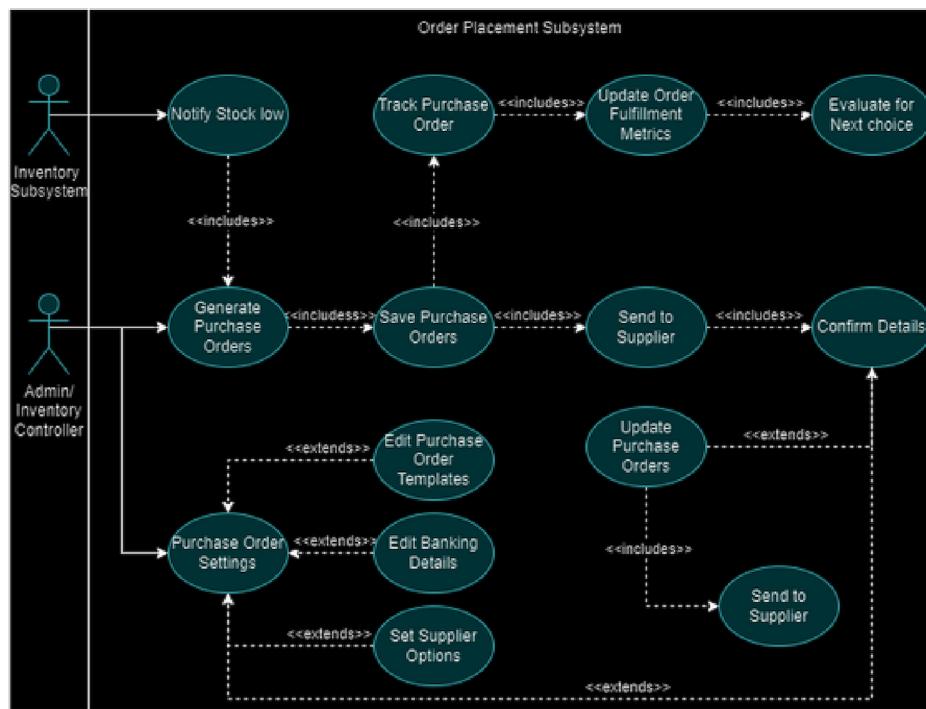
Stock Request Subsystem



Supplier Management Subsystem



Order Placement Subsystem



Architectural Design

Quality Requirements

- Reliability
- Scalability
- Security
- Usability
- Maintainability

Architectural Constraints

A core architectural constraint for this project is the requirement to utilize only open-source libraries.

How to quantify security

- * Role based access control should be prevalent in all ends of the app including help and settings.

How to quantify usability

- * A success rate of at least 70% will be expected of our app when tested.
- * A time of achieving a simple task should take no more than 2 minutes for a new user to achieve.
- * A 1-5 scale of how satisfied a user was using the app. An average higher than 3 is the goal.

How to quantify reliability

- * A mean time between failures of two times within the same 30 days for two hours.
- * Mean time to repair of less than five hours is our expectation.

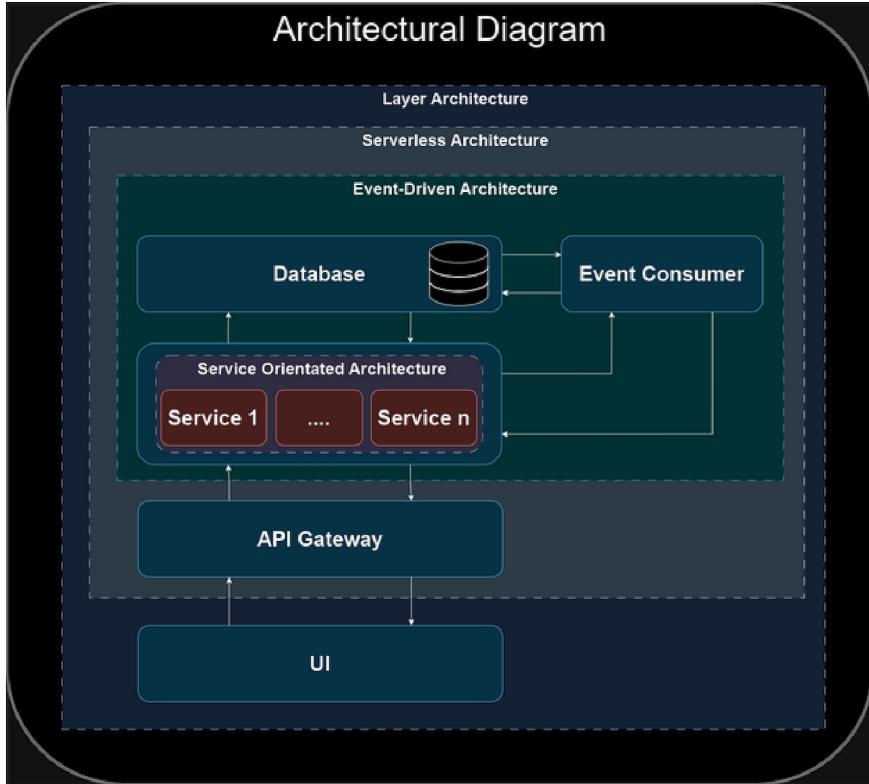
How to quantify scalability

- * The system is designed to handle 40 user requests per second.
- * The system must also have at least an 80% success rate with more than 20 users.

How to quantify Maintainability

- * Fixing a bug should not cause more than two hours of un-needed time due to un-maintainable code.
- * Adding a new feature should not cause more than two hours of un-needed time due to un-maintainable code.

Architectural Diagram



Architectural Styles

- Layered Architecture
- Service-Oriented Architecture (SOA)
- Event-Driven Architecture (EDA)
- Serverless Architecture

Latest Features

1. Team Management

- Add team member.
- Remove team member.
- Assign roles.
- Edit roles.
- Edit member details
- Export Table
- Search and sort table

2. Inventory Management

- Add Item
- Remove Item
- Remove Multiple Items
- Edit Item
- Export Table
- Search and sort table
- Request Items

4. Supplier Management

- Add Supplier
- Remove Supplier
- Edit supplier details
- Export Table
- Search and sort table

5. Dashboard

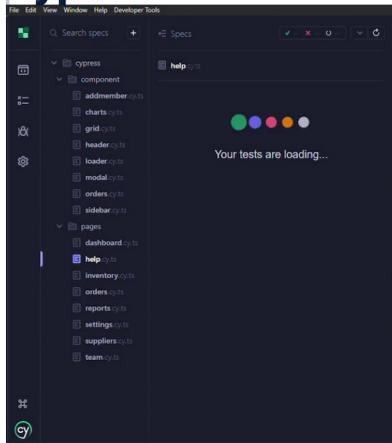
- Add Widget(graph, summary, table)
- Remove widget
- Reposition widget
- Save state

6. Settings

- Change Password
- Change details(name, surname, email)
- Delete Account

Tests - Cypress, Postman and AWS test event

Cypress



AWS Event tests

A screenshot of the AWS Lambda function logs. The top bar shows tabs for 'index.mjs', 'Environment Var', and 'Execution result'. A status indicator on the right says 'Status: Succeeded'. The main area displays the 'Execution results' for a test named 'createInventoryTest'. It shows the 'Response' object, which includes a 'statusCode': 201 and a 'body' containing JSON data: {"inventoryID": "6de0d130-8fe5-4fd9-866d-3edccfcec643"}, followed by a detailed 'Function Logs' section. The logs show a 'START RequestId: e93e71fd-39f1-4340-ab33-c9a5d2963d97 Version: \$LATEST' message, followed by an 'INFO Received event' message with the same JSON body. Subsequent log entries show the function processing the event, including 'pathParameters', 'proxy', 'stageVariables', 'headers', 'multiValueHeaders', and 'multiValueQueryStringParameters' details. The logs conclude with an 'END RequestId: e93e71fd-39f1-4340-ab33-c9a5d2963d97' message.

```
index.mjs Environment Var Execution result Status: Succeeded
Execution results
Test Event Name
createInventoryTest
Response
(
  "statusCode": 201,
  "body": "{\"inventoryID\": \"6de0d130-8fe5-4fd9-866d-3edccfcec643\"}"
)
Function Logs
START RequestId: e93e71fd-39f1-4340-ab33-c9a5d2963d97 Version: $LATEST
2024-02-11T11:12:49.540Z - e93e71fd-39f1-4340-ab33-c9a5d2963d97 INFO Received event: {
  "body": "{\"productID\": \"6a9c12a1-22fc-4ff6-92ad-bc1c86c3466f\", \"description\": \"Sample Product\", \"quantity\": 100, \"sku\": \"resource\": \"/(proxy)\"}"
}
  "pathParameters": null,
  "proxy": "/inventory",
  "stageVariables": null,
  "headers": {
    "Accept": "application/json",
    "Content-Type": "application/json"
  },
  "multiValueHeaders": {
    "Accept": [
      "application/json"
    ],
    "Content-Type": [
      "application/json"
    ]
  },
  "multiValueQueryStringParameters": null
)
END RequestId: e93e71fd-39f1-4340-ab33-c9a5d2963d97
REPORT RequestId: e93e71fd-39f1-4340-ab33-c9a5d2963d97 Duration: 10ms
```

Deployment

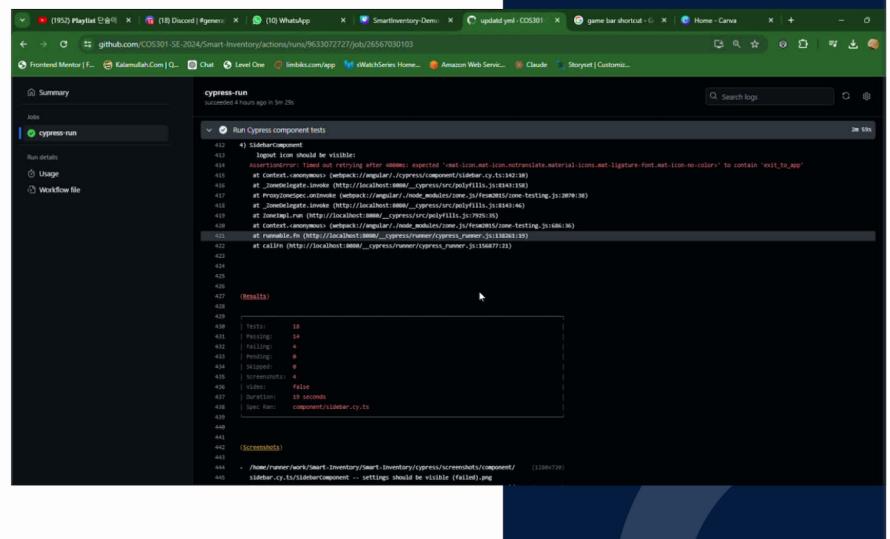
Deployment

- Set Trigger to deploy on push to main.
- Preview deployments through amplify sandbox to help enhance API and web app testing, creation and maintenance.

Testing

Tests

- Set Trigger on pushes to all branches.
 - When the trigger event is triggered it will run unit tests and e2e tests.



Linting

Linting

- Set Trigger on merge into main and develop.
- The automated linter ensures:
 - No syntax errors.
 - No coding standards violations.
 - Styling is readable.

The screenshot shows a CI/CD pipeline interface with a summary of a successful build. The build duration was 2m 7s. The pipeline steps include:

- Initial Setup
- Build Codebase (highlighted in blue)
- The end

Run details show the following steps:

- Set up job (1s)
- Checkout "COS301-SE-2024/Smart-Inventory" (2s)
- Set up Node v18.19.1 (4s)
- Clear npm cache (1s)
- Remove node_modules and package-lock.json (0s)
- Comment out outputs usage (0s)
- Install dependencies (3m 31s)
- Build Codebase (27s)

The final output log shows the build command and file sizes:

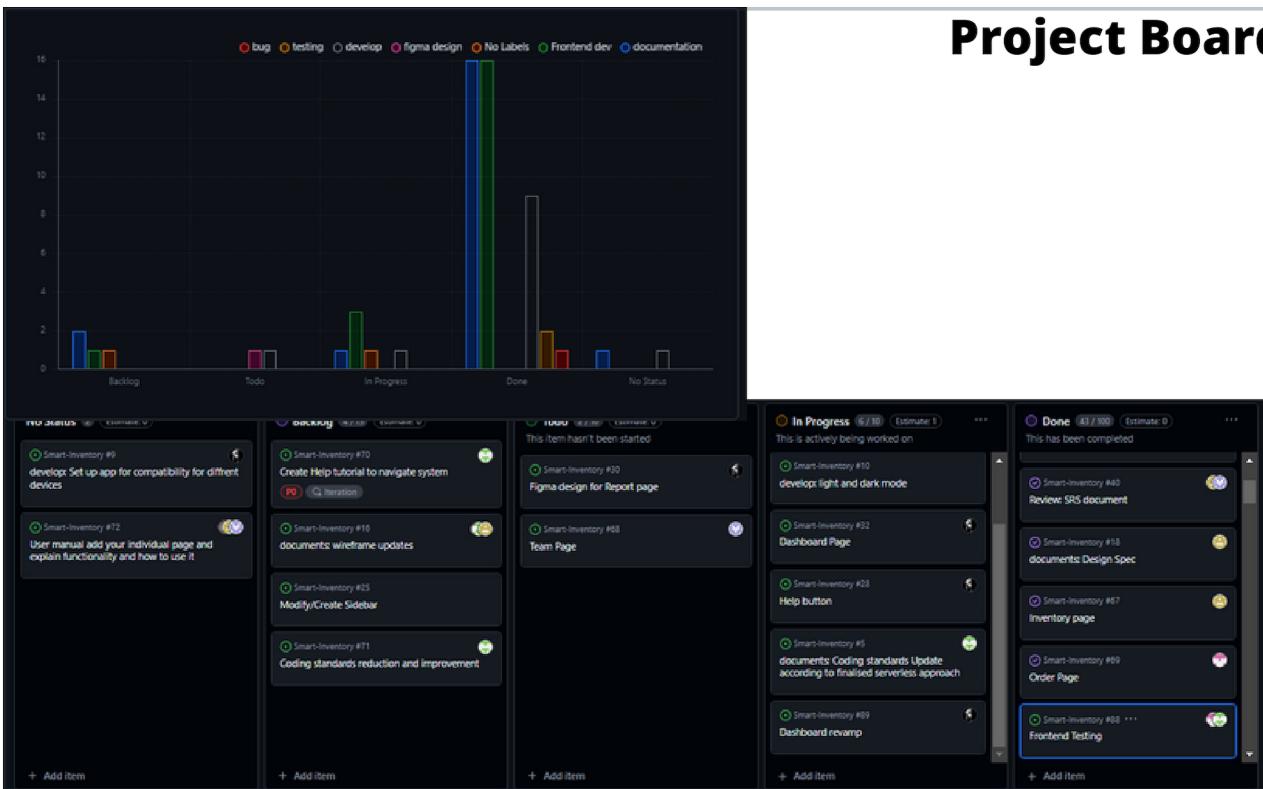
```
1 ► Run npm run build --configuration-production
2
3 > smart-inventory@0.0.0 build
4   > ng build --configuration=production
5
6 - Building...
7
8 Initial chunk files | Names      | Raw size | Estimated transfer size
9 main.30F21E5F.js    | main       | 3.89 MB  | 748.32 kB
10 styles-F6L7R7KX.css | styles     | 612.95 kB | 54.57 kB
11 chunk-SMAPWQJN.js  |          | 191.74 kB | 54.56 kB
12 polyfills-60AL5494.js | polyfills | 34.23 kB | 11.13 kB
13
14          | Initial total | 4.66 MB  | 908.95 kB
15
16 Lazy chunk files | Names      | Raw size | Estimated transfer size
```

Building

Building

- Set Trigger to build on push to main or develop branch.
- This helps ensure that no issues are around before the web app is deployed.

Project Board



THANK YOU!

QUESTIONS?

