# Assignment 2 Report: Inventory Management System (Flask Application)

**Student Name:**

*Leon Barretto*

**Course:**

Assignment 2: Developing a Flask Application for Business Problem Solving

---

## 1️⃣Business Problem

Small businesses often struggle to manage inventory levels, track sales, and monitor stock efficiently. Manual tracking is prone to errors, difficult to update, and not scalable.

The goal of this project was to build a simple, easy-to-use Inventory Management System using Flask that:

- Allows businesses to add new inventory items.
- Keeps track of available stock.
- Records sales transactions and updates stock levels automatically.
- Provides real-time inventory visualizations.

---

## 2️⃣Project Scope and Purpose

This web application helps businesses solve inventory challenges by:

- Automating inventory management
- Reducing manual work
- Preventing stock errors
- Providing better visibility into stock levels through a dashboard

---

## 3️⃣System Design

### Data Structures:

- **Inventory data:**
  Stored in a JSON file (inventory.json) as a list of dictionaries.
- Each inventory item contains:
  - item_id
  - name
  - category
  - price
  - stock

### Flask Routes:

- / → Home page
- /inventory → View current inventory
- /add_item → Add new inventory items
- /sales → Record sales and reduce stock
- /dashboard → Display inventory visualization

### Functions Used:

- load_inventory(): Load data from inventory.json
- save_inventory(): Save updated inventory data
- Business logic functions inside Flask routes

### Flow Control:

- Conditional checks to handle stock levels.
- Loops to process inventory and update data.

**Visualization:**

- Implemented using Chart.js to display current stock levels as a bar chart.

**Storage:**

- JSON file-based storage was used instead of a database to keep the project simple and file-based.

---

## 4️⃣Implementation Summary

- Used Python 3 and Flask for backend logic.
- Used HTML templates (Jinja2) for frontend pages.
- Implemented forms for adding items and recording sales.
- Used Chart.js to create the dashboard visualization.
- No external APIs were required.

---

## 5️⃣Testing

- All application routes were manually tested.
- Tested:
  - Adding items
  - Sales processing
  - Stock updates
  - Handling insufficient stock scenarios
  - Visual dashboard accuracy

---

## 6️⃣Challenges and Learning

- Learned how to structure Flask routes to handle GET and POST requests.
- Learned how to process form data from HTML templates.
- Debugged issues with JSON file reading and writing.
- Faced challenges in rendering Jinja2 variables safely into JavaScript for Chart.js.
- Successfully resolved data serialization issues using Flask's tojson filter.

---

## 7 Conclusion

This project demonstrates how Python, Flask, and basic data visualization tools can be combined to solve real-world business problems.
The system is functional, flexible, and can be expanded further in the future with:

- Authentication
- Sales reporting
- CSV export
- More advanced error handling

---

**End of Report**