**REPUBLIQUE DU CAMEROUN**

**PAIX – TRAVAIL – PATRIE**

**\*\*\*\*\*\*\*\*\*\***

**UNIVERSITE DE BAMENDA**

**\*\*\*\*\*\*\*\*\*\***

**FACULTE DES SCIENCES**

**\*\*\*\*\*\*\*\*\*\***

**Centre** **de cybersécurité et de cryptologie mathématique**

**\*\*\*\*\*\*\*\*\*\*\***

**REPUBLIC OF CAMEROON**

**PEACE – WORK – FATHERLAND**

**\*\*\*\*\*\*\*\*\*\***

**UNIVERSITY OF BAMENDA**

**\*\*\*\*\*\*\*\*\*\***

**FACULTY OF SCIENCE**

**\*\*\*\*\*\*\*\*\*\***

**Centre for Cybersecurity and Mathematical Cryptology**

**\*\*\*\*\*\*\*\*\*\*\***

+\*\*\*\*\*\*\*

**School: NAHPI**

**Centre for Cybersecurity and Mathematical cryptology**

**Year 4**

**COURSE CODE: CYBE6215**

**COURSE TITLE: APPLICATION DEVELOPMENT WITH PYTHON**

**DATE: 05/07/2025**

**PRESENTED BY: AKWA MABEL NTSEH**

**Course Lecturer: Dr. Evrad Kamtchoum**

A REPORT FOR AN INVENTORY MANAGEMENT SYSTEM

ACADEMIC YEAR: 2024/2025

**📝 Project Report: Grocery Inventory Management System**

### **🛠️ Tech Stack**

* **Frontend**: React (JavaScript)
* **Backend**: Django (Python)
* **Database**: PostgreSQL
* **Development Environment**: VS Code
* **Deployment**: (e.g., Heroku, Render, or custom VPS)

### 🎯 **Project Overview**

The Grocery Inventory Management System is a full-stack web application designed to help businesses efficiently manage grocery stock levels, monitor item availability, and perform secure transactions. The application enforces industry-standard security best practices across the frontend and backend layers.

### ✅ **Implemented Security Features**

#### 1. **Authentication**

* Utilized Django’s built-in authentication system.
* React frontend handles login/signup via secure forms.
* Authenticated sessions maintained with HTTP-only cookies or JWT tokens.

#### 2. **Authorization**

* User roles defined using Django Groups and Permissions.
* Route-level protection implemented in React.
* Restricted access to admin dashboards and inventory management features.

#### **3. Secure Session Management**

* Sessions stored securely using Django’s session framework.
* SESSION\_COOKIE\_SECURE and SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE enabled.
* React does not store sensitive info in localStorage.

#### 4. **Input Validation and Sanitization**

* Backend input validated using Django forms and serializers.
* React uses client-side validation with Yup and React Hook Form.
* All user inputs sanitized to prevent XSS and SQL injection.

#### **5. Encryption**

* Passwords hashed using Django’s default PBKDF2 algorithm.
* All sensitive data transferred over HTTPS.
* Optionally, sensitive database fields are encrypted.

#### **6. No Hard Coding**

* All secrets, API keys, and DB credentials are stored in .env files.
* Environment variables loaded using python-decouple or django-environ.

#### 7. **CSRF Protection**

* Django’s CSRF middleware enabled by default.
* CSRF tokens included in all POST requests from React using cookies.
* Axios or Fetch used to append CSRF tokens in headers.

#### 8. **Logging and Monitoring**

* Django logging configured to log errors, warnings, and security events.
* Authentication events (e.g., login failures) recorded.
* External tools like Sentry (optional) for error tracking and performance monitoring.

#### 9. **Account Lockout and Rate Limiting**

* django-axes integrated to block IPs after multiple failed login attempts.
* Rate limiting applied using django-ratelimit.
* React handles simple client-side delays for repeated failed submissions.

#### **10**. **Secure Coding Practices**

* Avoided use of eval, raw SQL, and insecure file handling.
* Used django-csp to implement Content Security Policy headers.
* Backend and frontend dependencies scanned using pip-audit and npm audit.

#### **11. Error Handling and Information Disclosure**

* Custom 404 and 500 pages configured.
* DEBUG = False in production.
* Detailed error messages hidden from users and logged securely.

#### **12. Password Security and HTTPS**

* Passwords must meet strength requirements via Django’s validators.
* HTTPS enforced using SECURE\_SSL\_REDIRECT = True.
* SSL certificate setup via Let’s Encrypt or cloud platform.
* React only interacts with the backend over secure HTTPS endpoints.

Below is the link for my github repository where we can actually view all the codes of

this project. It is made public so that anyone can actually access and view the project

<https://github.com/mabel-123/inventory123/tree/main/inventory123>

### 📦 **Conclusion**

This Grocery Inventory Management System is designed with a strong emphasis on **security**, **data integrity**, and **user privacy**. The system aligns with best practices for modern web development and can be scaled and extended for use in real-world retail and inventory settings.