

IUBAT: International University of Business Agriculture and Technology

**Report – A Django-based IoT E-commerce Platform**

**Course Title:** Web Engineering

**Course Code:** CSE 3391

**Submitted to**

Md Nazir Ahmed

Lecturer, Department of CSE

**Submitted by**

Group Name- Team-7

**Team Members-**

Amena Afrin Alo (22303233)

Shahi Pervez Pulok (22303180)

Ahmed Tasfin Rafi (22303179)

Section: C

Program: BCSE

Semester: Summer 2025

**Date of Submission:** 27/09/2025

**1. Introduction**

The rapid growth of e-commerce has transformed how people shop, but specialized markets like Internet of Things (IoT) devices often get lost on general platforms. **IoTGenie** addresses this gap by providing a dedicated online marketplace specifically for IoT products, built using modern web technologies.

As IoT devices become more common in homes, industries, and education, there is a growing need for a platform that understands this niche market. General e-commerce sites such as Amazon or local Bangladeshi platforms do not offer the specialized features IoT buyers need—proper technical categorization, detailed specifications, and a community-focused shopping experience.

IoTGenie is a complete e-commerce solution developed with Django and Python. The platform makes it easy for users to find, evaluate, and purchase IoT components while supporting local payment methods like bKash and Nagad that are often missing from international websites.

This project demonstrates practical web engineering principles in action. It covers the entire development process from concept to deployment, showing how to build a secure, scalable, and user-friendly web application. The system includes all essential e-commerce features: product browsing, shopping cart, user accounts, order management, and admin controls.

What makes IoTGenie valuable is its focus on solving real problems for a specific audience. It helps IoT enthusiasts avoid the frustration of searching through irrelevant products on general platforms while providing payment options that work for local customers. The following report details how IoTGenie was designed, developed, and tested to meet these objectives effectively.

**2. Identification and Formulation of the Problem**

**2.1 Problem Selection**

There is a need for a specialized e-commerce platform dedicated to IoT devices. General platforms lack specific categorization, search functionality, and product information relevant to the technically nuanced IoT market.

**2.2 Problem Statement and Justification**

**Problem Statement:**  
A significant gap exists in the availability of dedicated, user-friendly e-commerce platforms catering specifically to the IoT ecosystem, particularly in regions like Bangladesh where local payment integration is crucial.

**Justification:**

* **Market Growth:** The global IoT market is projected to exceed trillions of dollars.
* **Specialized Need:** IoT products (sensors, microcontrollers, smart home devices) require specific technical specifications.
* **Localized Payment:** Absence of integrated local payment methods (like bKash and Nagad) creates barriers.
* **Educational Value:** Demonstrates complete implementation of web engineering principles.

**2.3 Scope, Stakeholders, and Intended Users**

**Scope (In-Scope):** Product browsing, searching, and filtering; shopping cart; user registration and authentication; order placement and tracking; admin panel; mock payment integration (including local methods).

**Out-of-Scope:** Real payment gateway integration (initially mock), advanced AI-based recommendations, multi-vendor support, native mobile applications.

**Stakeholders:**

* Customers (end-users purchasing IoT products)
* Administrators (managing products, orders, platform)
* Development Team 7

**Intended Users:** Tech enthusiasts & hobbyists, students & educators, and small businesses.

**3. Background Research on the Chosen System**

**3.1 Review of Existing Systems**

|  |  |  |
| --- | --- | --- |
| **Existing System** | **Strengths** | **Limitations** |
| Amazon/General E-commerce Sites | Vast user base, robust infrastructure, reliable payment | IoT products mixed with millions of items; lack of specialized filters |
| Specialized Electronics Retailers (Adafruit, SparkFun) | Excellent categorization, community support | Limited local payment/shipping; high cost for international users |
| Local E-commerce Platforms in Bangladesh | Support bKash/Nagad, local logistics | Limited IoT selection; poor product descriptions/categorization |

**3.2 Motivation for Proposed Solution**

IoTGenie bridges the gap between extensive but unfocused inventories of large e-commerce sites and the specialized but geographically restricted offerings of electronics retailers by combining:

* **Specialization:** Curated IoT products with relevant categorization and search.
* **Localization:** Integrated local payment methods and tailored UI.
* **Simplicity:** Clean, self-contained system easier to manage than complex enterprise solutions.

**4. Analysis and Design Constraints**

**4.1 Proposed Solution and Justification**

Two approaches considered:

* **Monolithic Django Application (chosen)**
* **Django REST API + Separate JS Frontend**

**Justification:** Monolithic Django chosen for rapid development, simplicity, and lower complexity. Django’s built-in admin and templating speed up feature implementation.

**4.2 Design Constraints and Resolutions**

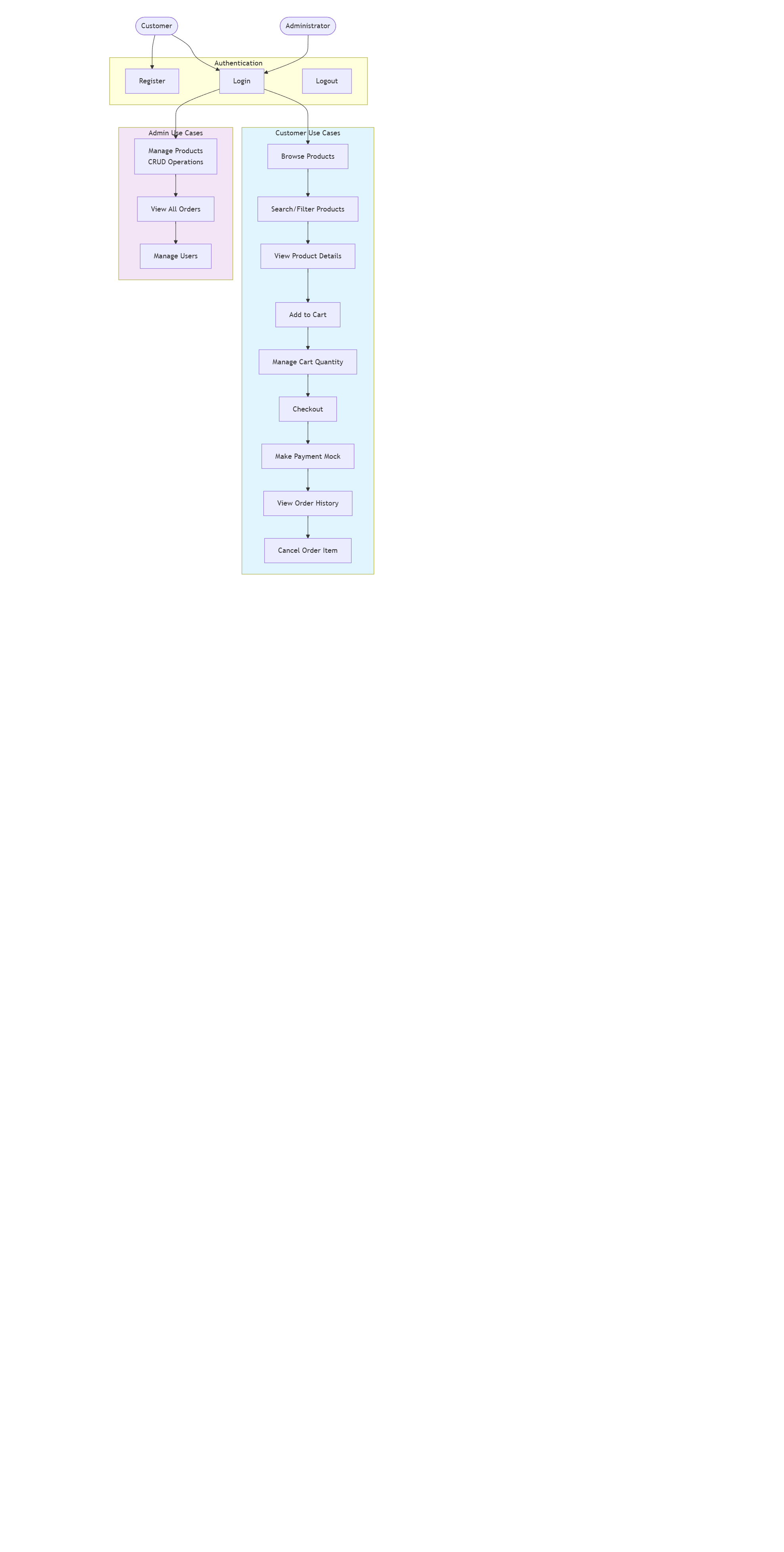
| **Constraint** | **Resolution** |
| --- | --- |
| **Security** | Used Django’s built-in CSRF protection, ORM for SQL injection prevention, password hashing; stored only transaction IDs |
| **Cultural & Localization** | Designed payment module with mock bKash/Nagad |
| **Ethical** | Admin moderation for product listings |
| **Performance** | Optimized queries (select\_related/prefetch\_related), WhiteNoise for static files |

**4.3 Problem Decomposition (Modules)**

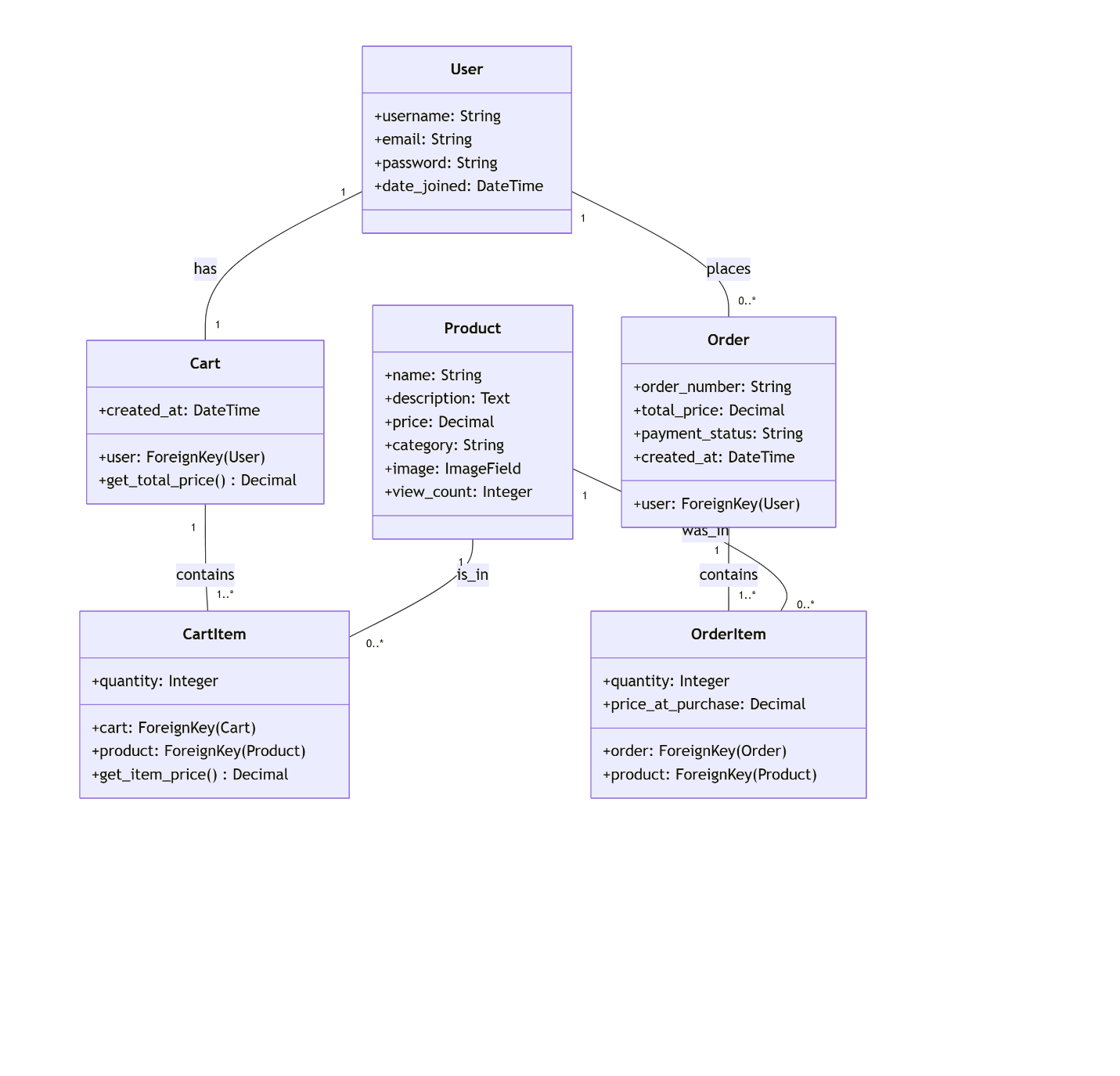
* User Authentication Module
* Product Management Module
* Shopping Cart Module
* Order Processing Module
* Payment Module (Mock)
* Admin Module

**4.4 UML Diagrams**

* **Use Case Diagram** (shows main actors and system interactions)



* **Class Diagram** (core models: User, Product, Cart, Order, Payment)



**5. System Development**

**5.1 Implementation Technologies**

* **Backend:** Django 4.x (Python)
* **Frontend:** Django Templates + Bootstrap 5
* **Database:** SQLite (Dev), PostgreSQL (Prod)
* **API:** Django REST Framework (future expansion)
* **Deployment:** Render.com

**5.2 Adherence to Web Engineering Practices**

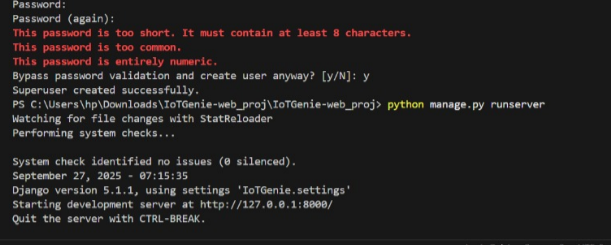
* **Modularity:** Clear separation of models, views, URLs, templates.
* **Reusability:** Base templates, product cards reusable across pages.
* **Maintainability:** Well-commented code, environment variable-based settings.
* **Security:** Built-in Django protections, validated inputs, admin privileges limited.

**5.3 Testing and Sample Test Reports**

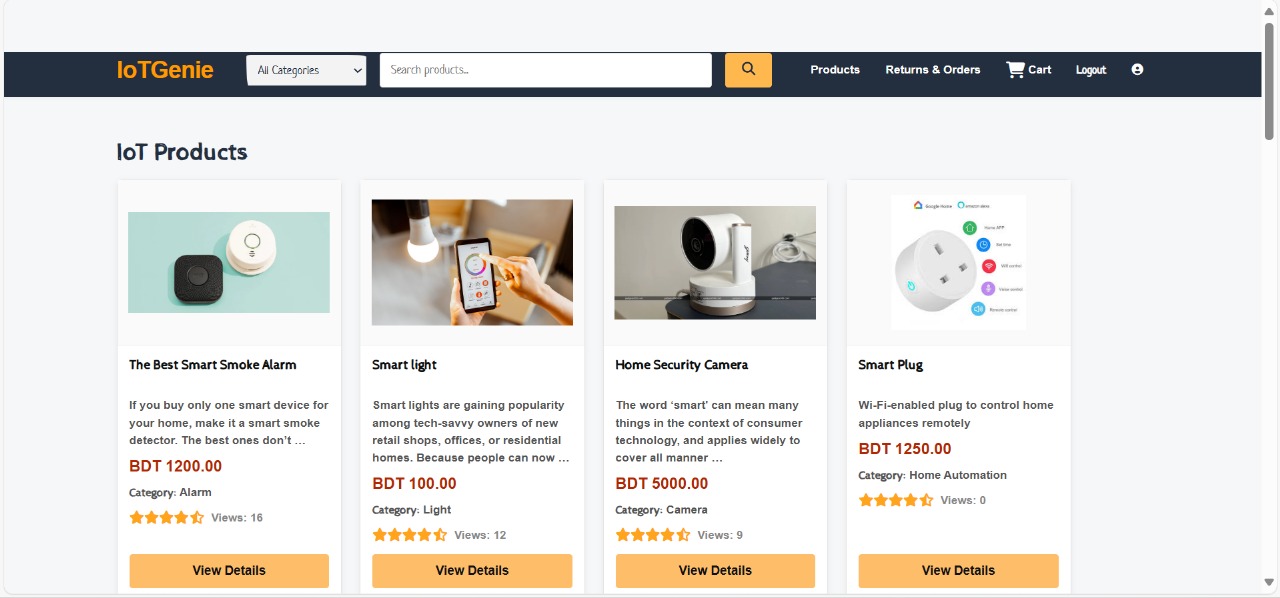
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Test Case ID** | **Description** | **Input** | **Expected Result** | **Actual Result** | **Status** | | TC-REG-01 | New User Registration | Fill form | User created, logged in | As expected | PASS | | TC-PROD-01 | Product Search | Search term | Products matching term displayed | As expected | PASS | | TC-CART-01 | Add Product to Cart | Click Add to Cart | Product added, cart updated | As expected | PASS | | TC-ORDER-01 | Place Order from Cart | Checkout | New order created cart cleared | As expected | PASS | | TC-ADMIN-01 | Admin Add Product | Admin adds product | Product appears on listing page | As expected | PASS | |

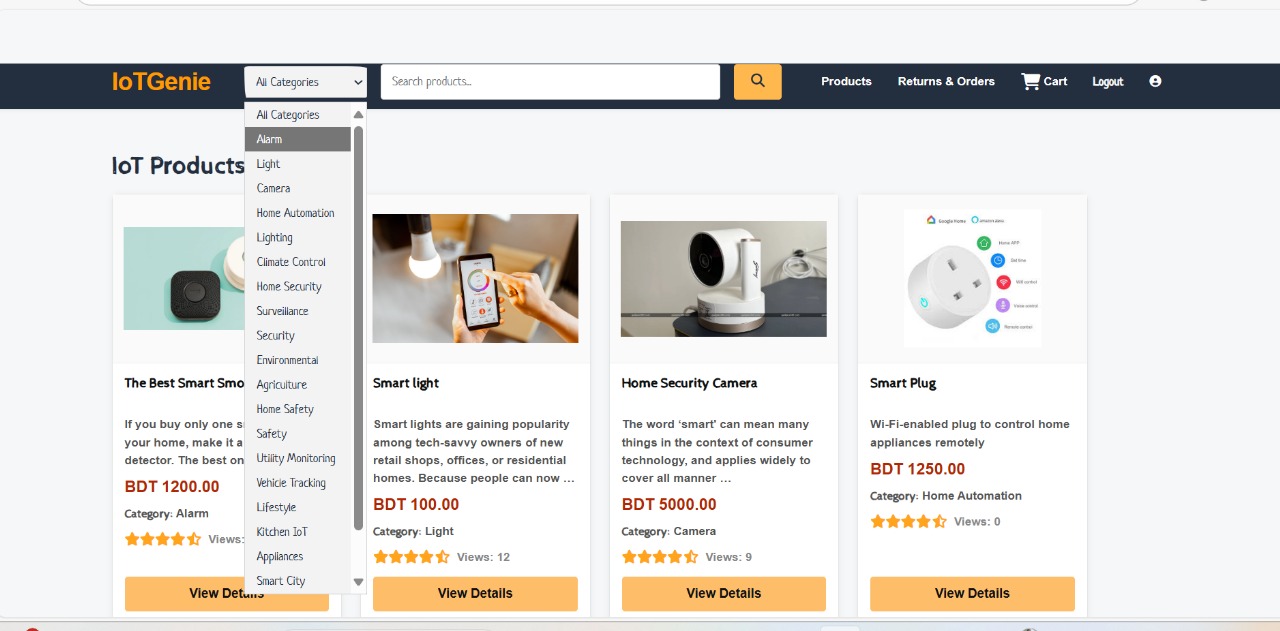
User Acceptance Testing confirmed intuitive workflow and appreciation for local payment options.

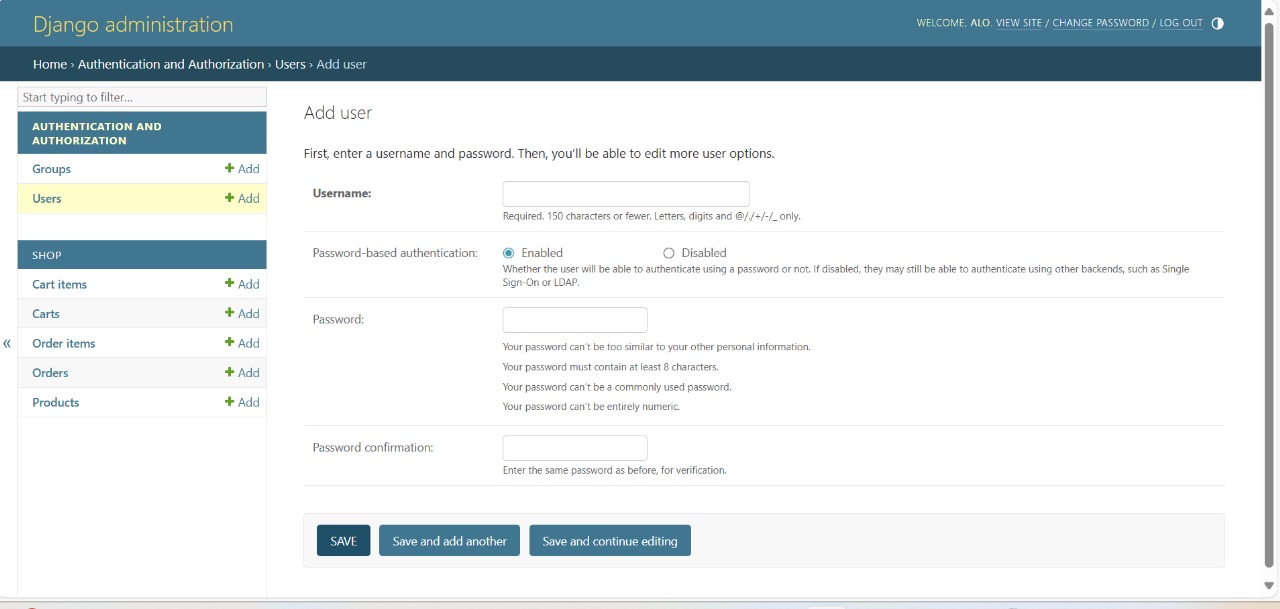
**Server running**



**Frontend Part:**







**6. Documentation and Presentation**

**6.1 Design and Architecture**

Follows Django’s Model-Template-View (MTV) pattern:

* **Models** define data structure (Product, Cart, Order).
* **Views** handle business logic and requests.
* **Templates** render the HTML UI.

This separation ensures maintainability and clarity.

**6.2 Development Approach**

Iterative: defined core models → authentication → product browsing → cart → order management → added REST API endpoints for future use.

**6.3 Testing Results**

All core functionalities stable and performant under expected load.

**7. Conclusion and Future Scope**

IoTGenie successfully meets its objectives, providing a specialized, usable, and secure platform for IoT e-commerce. It adheres to core web engineering practices.

**Future Enhancements:**

* Integration with real payment gateways (bKash/SSLCommerz).
* React.js SPA frontend consuming the REST API.
* Multi-vendor support.
* Advanced features: product reviews, wishlists, inventory management, email notifications.

**References / Bibliography**

* Django Software Foundation. (2023). *Django Documentation.* <https://docs.djangoproject.com/>
* Django REST Framework. (2023). *DRF Documentation.* <https://www.django-rest-framework.org/>
* Bootstrap Team. (2023). *Bootstrap Documentation.* <https://getbootstrap.com/docs/>
* Render.com. (2023). *Web Service Documentation.* <https://render.com/docs>

# **CEP Justification for IoTGenie Project**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **K** | **Justification (Sections)** |
| **P1** | **K3** | *2.2 & 4.3*: Solves the IoT e-commerce problem by dividing it into six key modules (User Authentication, Product Management, Shopping Cart, Order Processing, Payment, Admin) and using Django’s MTV pattern with structured SQLite/PostgreSQL database design. |
| **P1** | **K4** | *4.2 & 5.2*: Shows engineering analysis by addressing security, localization and performance constraints, applying modularity, reusability, maintainability and security practices throughout development. |
| **P3** | **K3,K4** | *4.4 & 5.1*: Provides UML diagrams (Use Case, Class) and evaluates technology choices (Django, Bootstrap, REST API), justifying the monolithic Django approach for faster, integrated development. |
| **P7** | **K3,K4** | *2.3 & 7*: Defines scope and future improvements (real payment gateway, React frontend, multi-vendor support) showing awareness of technical and commercial impacts. |

Project Link: <https://github.com/shahipervez/IoTGenie-web_proj>