# Final Report - Medical Store Inventory System

Team: IntelliWare

Team Members: Mustafa Iqbal (Team Lead), Amaar Khan (Scrum Master), Saad Nadeem (Developer)

## 1. Project Introduction

The Medical Store Inventory System is a web-based solution designed to manage and optimize the inventory, sales, and reporting processes of a pharmacy or medical store. The project is built using Python and Flask, with a focus on providing secure, scalable, and user-friendly interfaces for store managers, pharmacists, and cashiers. This system helps ensure efficient stock monitoring, reduction of expired items, and automated transaction processing.

## 2. Functional and Non-Functional Requirements

Functional Requirements:

* Inventory Management (Pharmacist & Store Manager)  
  • Add, update, and delete medicine records with details such as name, price, quantity, category, and expiry.  
  • Automatically remove expired medicines and update stock.  
  • Monitor real-time stock levels and receive low-stock alerts.  
  • View and search the complete inventory with visual stock indicators.
* **Sales Operations (Cashier & Store Manager)**  
  • Process sales, generate invoices, and update stock accordingly.  
  • Validate stock availability before sale completion.  
  • Generate daily, weekly, or monthly sales and inventory reports.  
  • Export inventory and sales data to CSV for external analysis.
* **Performance & Business Insights**  
  • View monthly revenue statistics and top-selling products.  
  • Analyse sales trends by category to guide inventory planning.

### Non-Functional Requirements

**Usability**  
• Responsive UI with dark mode and clear visual cues.  
• User-friendly error and confirmation messages.

**Performance**  
• Page load time under 3 seconds; report generation under 5 seconds.  
• Support at least 100 concurrent users without performance degradation.

**Reliability**  
• Input data validation and atomic database transactions.  
• Regular automatic data backups to ensure data safety.

**Security**  
• Hashed password storage and auto session timeouts.  
• Role-based access control for all users.

**Maintainability**  
• Modular codebase, consistent naming conventions, and complete documentation.

**Compatibility & Scalability**  
• Cross-browser and multi-device responsive design.  
• Scalable to support growing inventory and transaction volume.

## 3. User Stories

**Amaar's User Stories**

1. **As a pharmacist, I want to add new medicines to inventory**  
   *So that I can track stock availability and update pricing.*
2. **As a pharmacist, I want to update medicine details**  
   *So that I can ensure accurate information is maintained in the system.*
3. **As a pharmacist, I want to view the entire inventory**  
   *So that I can track all available medicines at any time.*
4. **As a pharmacist, I want to delete medicines (that have no sales records)**  
   *So that discontinued or irrelevant products are removed from the system.*
5. **As a pharmacist, I want to create sales records**  
   *So that customer purchases can be processed with billing and stock updates.*

**Saad's User Stories**

1. **As a store manager, I want to view comprehensive inventory reports**  
   *So that I can understand current stock status and manage restocking.*
2. **As a store manager, I want to monitor stock levels**  
   *So that I can avoid shortages and overstocking.*
3. **As a store manager, I want to receive low stock notifications**  
   *So that I can take timely actions for restocking.*
4. **As a store manager, I want to view monthly revenue data**  
   *So that I can track business financial performance.*
5. **As a store manager, I want to see top-selling products**  
   *So that I can make better purchasing decisions based on demand.*

**Mustafa's User Stories**

1. **As a cashier, I want to check medicine availability before processing a sale**  
   *So that I ensure requested items are in stock.*
2. **As a cashier, I want to view the inventory to check stock details**  
   *So that I can inform customers accurately during checkout.*
3. **As a user, I want to see stock status indicators (well-stocked, low-stock, out of stock)**  
   *So that I can assess inventory conditions at a glance.*
4. **As a user, I want medicines categorized by type**  
   *So that I can quickly find and manage them.*
5. **As a user, I want to see medicine expiration dates**  
   *So I can identify and remove expired products in a timely manner.*

4. Product Backlog – Pharmacy Management System

**Team Members**

* **Amaar**
* **Saad**
* **Mustafa**

**Backlog Items**

**1. Add New Medicines**  
**Description:** Pharmacist can add medicines with details like name, category, price, quantity, and expiry date.  
**Priority:** High  
**Story Points:** 5  
**Assigned To:** Amaar

**2. Update Medicine Details**  
**Description:** Pharmacist can update existing medicine information such as price, quantity, or expiry.  
**Priority:** High  
**Story Points:** 3  
**Assigned To:** Amaar

**3. View Inventory**  
**Description:** Pharmacist can view the current list of medicines in stock.  
**Priority:** High  
**Story Points:** 3  
**Assigned To:** Amaar

**4. Delete Medicines**  
**Description:** Pharmacist can delete medicines from the inventory, provided they have no associated sales.  
**Priority:** Medium  
**Story Points:** 5  
**Assigned To:** Amaar

**5. Create Sales Records**  
**Description:** Pharmacist can create a sales entry, which reduces stock and generates a receipt.  
**Priority:** High  
**Story Points:** 8  
**Assigned To:** Amaar

**6. View Inventory Reports**  
**Description:** Manager can view detailed reports on current stock and inventory changes.  
**Priority:** High  
**Story Points:** 5  
**Assigned To:** Saad

**7. Monitor Stock Levels**  
**Description:** Manager can see quantities of medicines across different categories.  
**Priority:** High  
**Story Points:** 3  
**Assigned To:** Saad

**8. Low Stock Notifications**  
**Description:** System alerts the manager when stock for any medicine goes below a set threshold.  
**Priority:** High  
**Story Points:** 5  
**Assigned To:** Saad

**9. View Monthly Revenue**  
**Description:** Manager can view monthly revenue generated through sales.  
**Priority:** Medium  
**Story Points:** 8  
**Assigned To:** Saad

**10. View Top-Selling Products**  
**Description:** Display a list of top-selling products in a given time range.  
**Priority:** Medium  
**Story Points:** 5  
**Assigned To:** Saad

**11. Check Medicine Availability Before Sale**  
**Description:** Cashier can check stock before finalizing a sale.  
**Priority:** High  
**Story Points:** 3  
**Assigned To:** Mustafa

**12. Cashier Views Inventory**  
**Description:** Cashier can view medicine stock details during billing.  
**Priority:** Medium  
**Story Points:** 2  
**Assigned To:** Mustafa

**13. Stock Status Indicators**  
**Description:** System highlights medicines as well-stocked, low-stock, or out-of-stock.  
**Priority:** Medium  
**Story Points:** 5  
**Assigned To:** Mustafa

**14. Categorize Medicines by Type**  
**Description:** Medicines are grouped by type such as tablet, syrup, injection, etc.  
**Priority:** Low  
**Story Points:** 3  
**Assigned To:** Mustafa

**15. Show Expiry Dates**  
**Description:** Display expiry date of each medicine and allow filtering by soon-to-expire items.  
**Priority:** Medium  
**Story Points:** 3  
**Assigned To:** Mustafa

5. Sprint 1 Backlog – Pharmacy Management System

**1. Add New Medicines**

* **Description: Pharmacist can add medicines with details like name, category, price, quantity, and expiry date.**
* **Story Points: 5**
* **Assigned To: Amaar**

**2. Update Medicine Details**

* **Description: Pharmacist can update existing medicine information.**
* **Story Points: 3**
* **Assigned To: Amaar**

**3. View Inventory**

* **Description: Pharmacist can view all medicines in stock.**
* **Story Points: 3**
* **Assigned To: Amaar**

**4. Check Medicine Availability Before Sale**

* **Description: Cashier can check stock before finalizing a sale.**
* **Story Points: 3**
* **Assigned To: Mustafa**

**5. Cashier Views Inventory**

* **Description: Cashier can view medicine stock details during billing.**
* **Story Points: 2**
* **Assigned To: Mustafa**

**6. Monitor Stock Levels**

* **Description: Manager can monitor quantities of medicines across categories.**
* **Story Points: 3**
* **Assigned To: Saad**

**7. Show Expiry Dates**

* **Description: System displays expiry dates and allows filtering.**
* **Story Points: 3**
* **Assigned To: Mustafa**

Sprint 2 Backlog – Pharmacy Management System

**1. Delete Medicines**

* **Description: Pharmacist can delete medicines with no sales.**
* **Story Points: 5**
* **Assigned To: Amaar**

**2. Create Sales Records**

* **Description: Pharmacist can record sales, update stock, and generate receipts.**
* **Story Points: 8**
* **Assigned To: Amaar**

**3. View Inventory Reports**

* **Description: Manager views stock and change history.**
* **Story Points: 5**
* **Assigned To: Saad**

**4. Low Stock Notifications**

* **Description: Manager receives alerts for low inventory.**
* **Story Points: 5**
* **Assigned To: Saad**

**5. View Monthly Revenue**

* **Description: Manager can view revenue by month.**
* **Story Points: 8**
* **Assigned To: Saad**

**6. View Top-Selling Products**

* **Description: Manager sees top-selling products by demand.**
* **Story Points: 5**
* **Assigned To: Saad**

**7. Stock Status Indicators**

* **Description: System highlights stock levels visually.**
* **Story Points: 5**
* **Assigned To: Mustafa**

**8. Categorize Medicines by Type**

* **Description: Medicines grouped by form (tablet, syrup, etc).**
* **Story Points: 3**
* **Assigned To: Mustafa**

**6. Project Plan – Medical Store Inventory System**

**Project Title:**

Medical Store Inventory System

**Team Name:**

Intelliware

**Team Members:**

* **Mustafa Iqbal** – Team Lead
* **Amaar Khan** – Scrum Master
* **Saad Nadeem** – Developer

**Project Timeline:**

| **Deliverable** | **Start Date** | **End Date** |
| --- | --- | --- |
| Deliverable 1 – Basic System | February 13 | February 28 |
| Deliverable 2 – Advanced System | March 6 | March 23 |

**Project Objectives:**

* Develop an efficient and user-friendly inventory system for medical stores.
* Enable pharmacists, managers, and cashiers to manage inventory, process sales, and generate reports.
* Ensure scalability, modularity, and responsiveness for better performance and user experience.

**Key Technologies:**

* **Backend Framework:** Flask (Python)
* **Database:** SQLite
* **UI Design:** Figma
* **Data Visualization:** Chart.js
* **Version Control:** GitHub
* **Task Management:** Trello

**Development Methodology:**

* **Agile Scrum Framework** will be followed.
* Two **Sprints** will be conducted:
  + Sprint 1: Focus on core functionality (inventory management, basic UI, sales processing).
  + Sprint 2: Emphasis on advanced features (report generation, notifications, UI enhancements).

**Roles & Responsibilities:**

* **Mustafa Iqbal (Team Lead)**
  + Oversee task delegation
  + Review requirements and documentation
  + Contribute to UI mockups and reporting modules
* **Amaar Khan (Scrum Master)**
  + Manage sprint planning and daily stand-ups
  + Track progress using Trello
  + Support testing and documentation
* **Saad Nadeem (Developer)**
  + Implement Flask backend and database
  + Build inventory, sales, and reporting modules
  + Handle system validations and integration

**Milestones & Deliverables:**

**Deliverable 1 – Basic System**

* Requirement gathering and planning
* SRS and UI mockups
* Flask app setup and Inventory Module
* Unit testing and basic documentation

**Deliverable 2 – Advanced System**

* Sales and Reporting Modules
* Stock validation and notifications
* UI enhancements and dark mode
* Final testing and project submission

6.Project Plan

**Software Project Plan**

**Project: Medical Store Inventory System**

**Team: IntelliWare  
Team Members:**

* **Mustafa Iqbal – Team Lead**
* **Amaar Khan – Scrum Master**
* **Saad Nadeem – Developer**

**Project Timeline**

* **Deliverable 1: Feb 13 – Feb 28**
* **Deliverable 2: Mar 6 – Mar 23**

**Work Breakdown Structure (WBS)**

**Level 1: Project**

* **1. Medical Store Inventory System**

**Level 2: Deliverables**

* **1.1 Project Planning**
* **1.2 Deliverable 1 – Basic System**
* **1.3 Deliverable 2 – Advanced System**

**Level 3: Components**

**1.1 Project Planning**

* **1.1.1 Requirement Gathering**
* **1.1.2 Team & Tool Setup**

**1.2 Deliverable 1 – Basic System**

* **1.2.1 SRS & UI Mockups**
* **1.2.2 Flask App Initialization**
* **1.2.3 Inventory Module**
* **1.2.4 Testing & Documentation**

**1.3 Deliverable 2 – Advanced System**

* **1.3.1 Reporting Module**
* **1.3.2 Sales & Stock Validation**
* **1.3.3 UI Enhancements**
* **1.3.4 Final Testing & Submission**

**Level 4: Activities (Work Packages)**

**1.1.1 Requirement Gathering**

* **Interview stakeholders**
* **Document requirements**
* **Review & get approval**

**1.1.2 Team & Tool Setup**

* **Set up GitHub repo**
* **Configure Trello board**
* **Install and configure development tools**

**1.2.1 SRS & UI Mockups**

* **Create Software Requirements Specification**
* **Design UI mockups in Figma**
* **Review and finalize**

**1.2.2 Flask App Initialization**

* **Create Flask project structure**
* **Set up routing**
* **Integrate SQLite**

**1.2.3 Inventory Module**

* **Add Medicine feature**
* **Update Medicine feature**
* **Delete Medicine feature**
* **Input validation**

**1.2.4 Testing & Documentation**

* **Write unit tests with Pytest**
* **Create user documentation**
* **Peer review**

**1.3.1 Reporting Module**

* **Design report layout**
* **Implement filter logic**
* **Display with Chart.js**

**1.3.2 Sales & Stock Validation**

* **Implement sales entry logic**
* **Realtime validation logic**
* **Integrate with stock records**

**1.3.3 UI Enhancements**

* **Add error handling**
* **Implement dark mode**
* **Improve UI responsiveness**

**1.3.4 Final Testing & Submission**

* **Conduct peer testing**
* **Fix reported bugs**
* **Prepare final submission package**

**Level 5: Task Assignments**

**Mustafa Iqbal**

* **Interview stakeholders**
* **Write SRS**
* **Design UI mockups**
* **Implement UI enhancements**

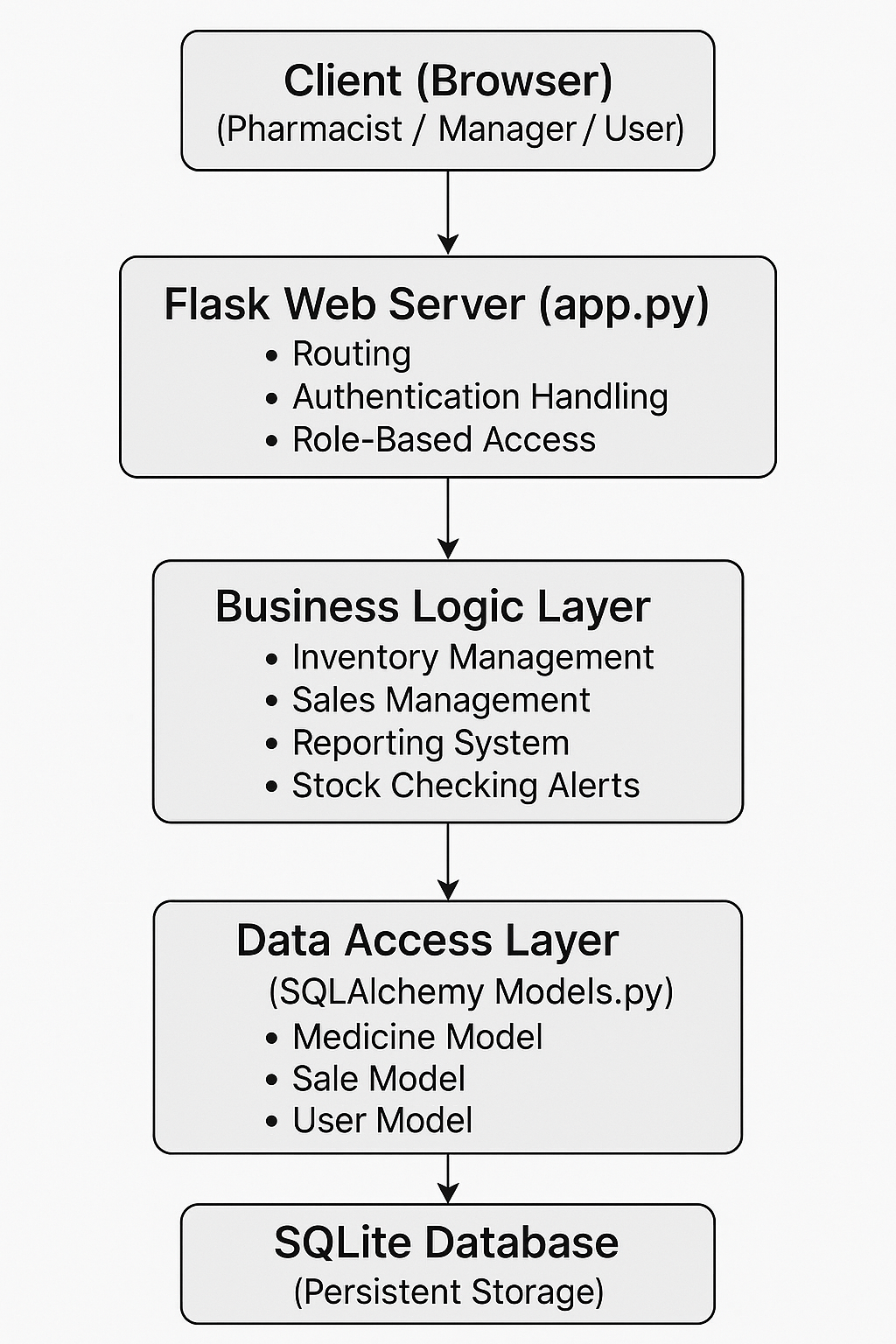
**Amaar Khan**

* **Configure Trello**
* **Write documentation**
* **Perform final testing**

**Saad Nadeem**

* **Set up Flask project**
* **Build inventory module**
* **Develop reporting & validation modules**

7Architecture Diagram



8. Design (all sprint 3 items)

**Sprint 3 Focus:**

Sprint 3 primarily addresses **final improvements, advanced reporting features, bug fixing, UI enhancements, and project finalization**.

**Sprint 3 Items and Design Details**

| **Feature** | **Design Description** | **Assigned To** |
| --- | --- | --- |
| **Reporting Module Finalization** | - Integrate Chart.js visualizations for sales and inventory reports. - Implement dynamic filters (by date, product category). - Enable export of reports in CSV format. | Saad Nadeem |
| **Sales and Stock Validation Enhancement** | - Implement real-time stock checking at sale time. - Block sale if stock insufficient. - Update stock levels immediately after each sale. - Trigger low-stock notifications when thresholds are crossed. | Saad Nadeem |
| **UI Enhancements** | - Add dark mode toggle using Bootstrap 5 and custom CSS. - Ensure all pages are fully responsive across devices. - Implement clear and consistent user error/confirmation messages. - Refine overall UI for better usability and aesthetics. | Mustafa Iqbal |
| **Final System Testing & Bug Fixing** | - Conduct complete black-box and white-box testing. - Perform peer review of functionality. - Log and fix any identified bugs or UI inconsistencies. - Validate all modules (inventory, sales, reports). | Amaar Khan |
| **Documentation and Final Submission Preparation** | - Update Software Requirements Specification (SRS). - Capture actual system screenshots. - Update the User Manual. - Finalize and organize all documentation for submission. | Amaar Khan |

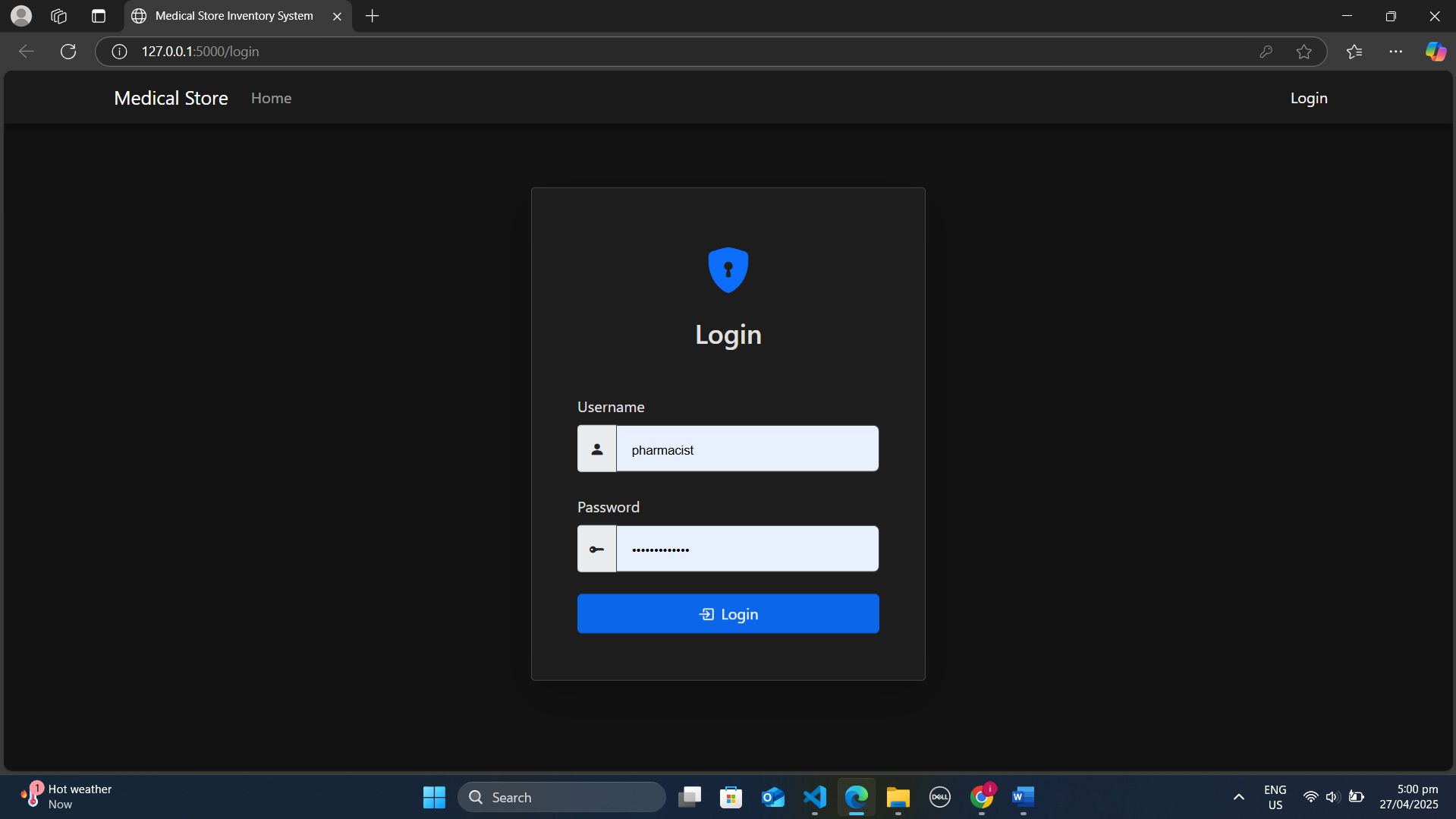
**Key Tools and Technologies Used**

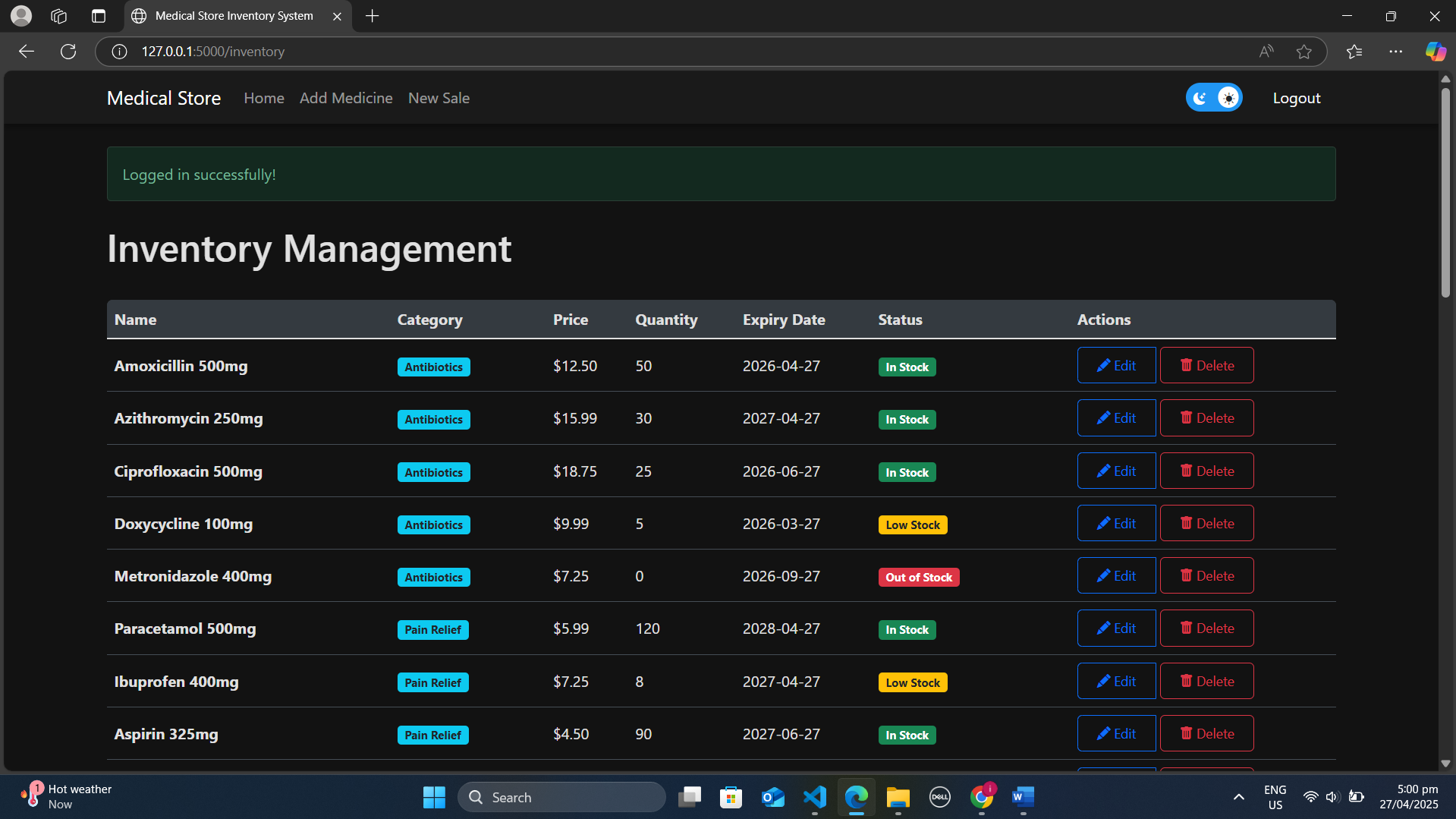
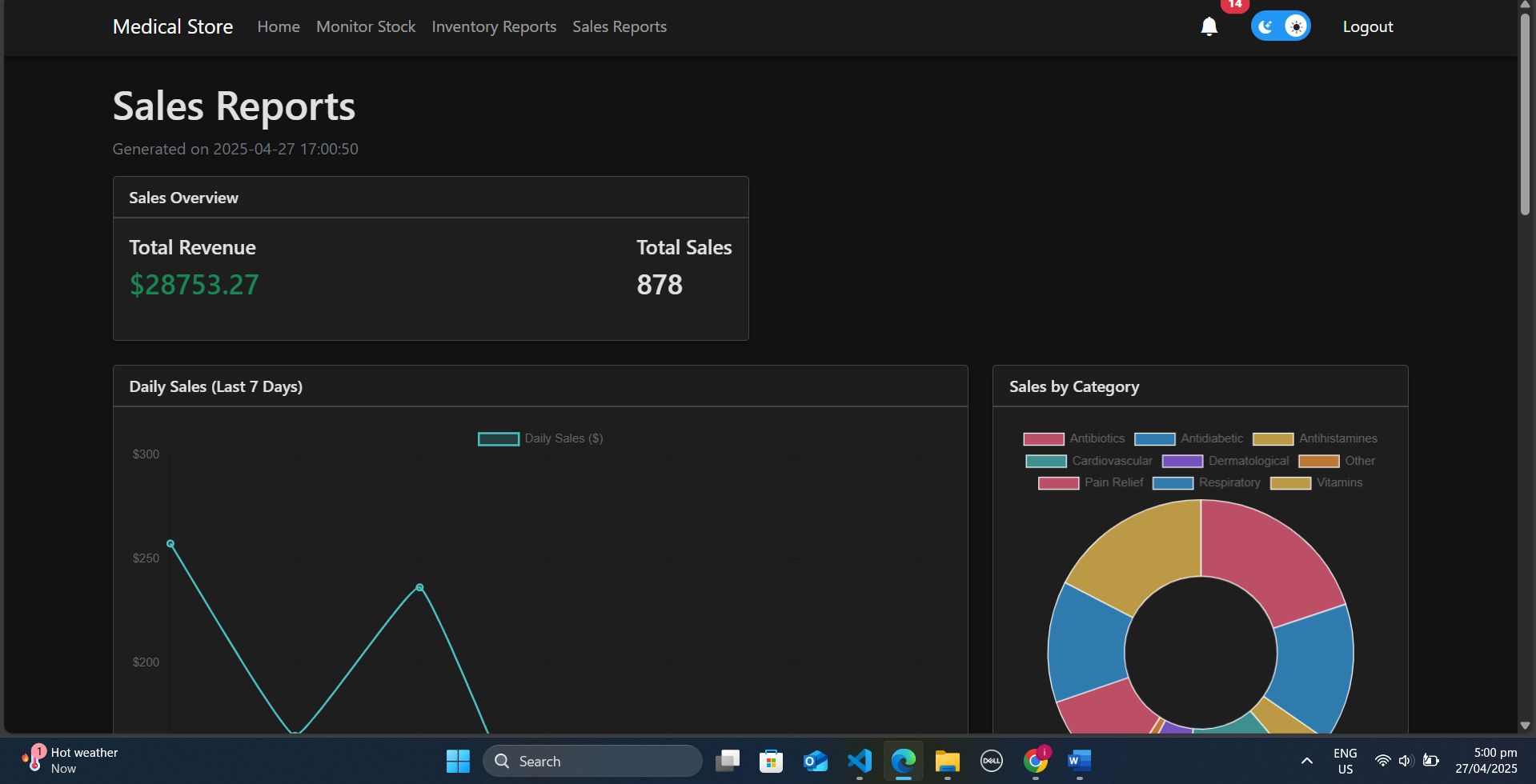
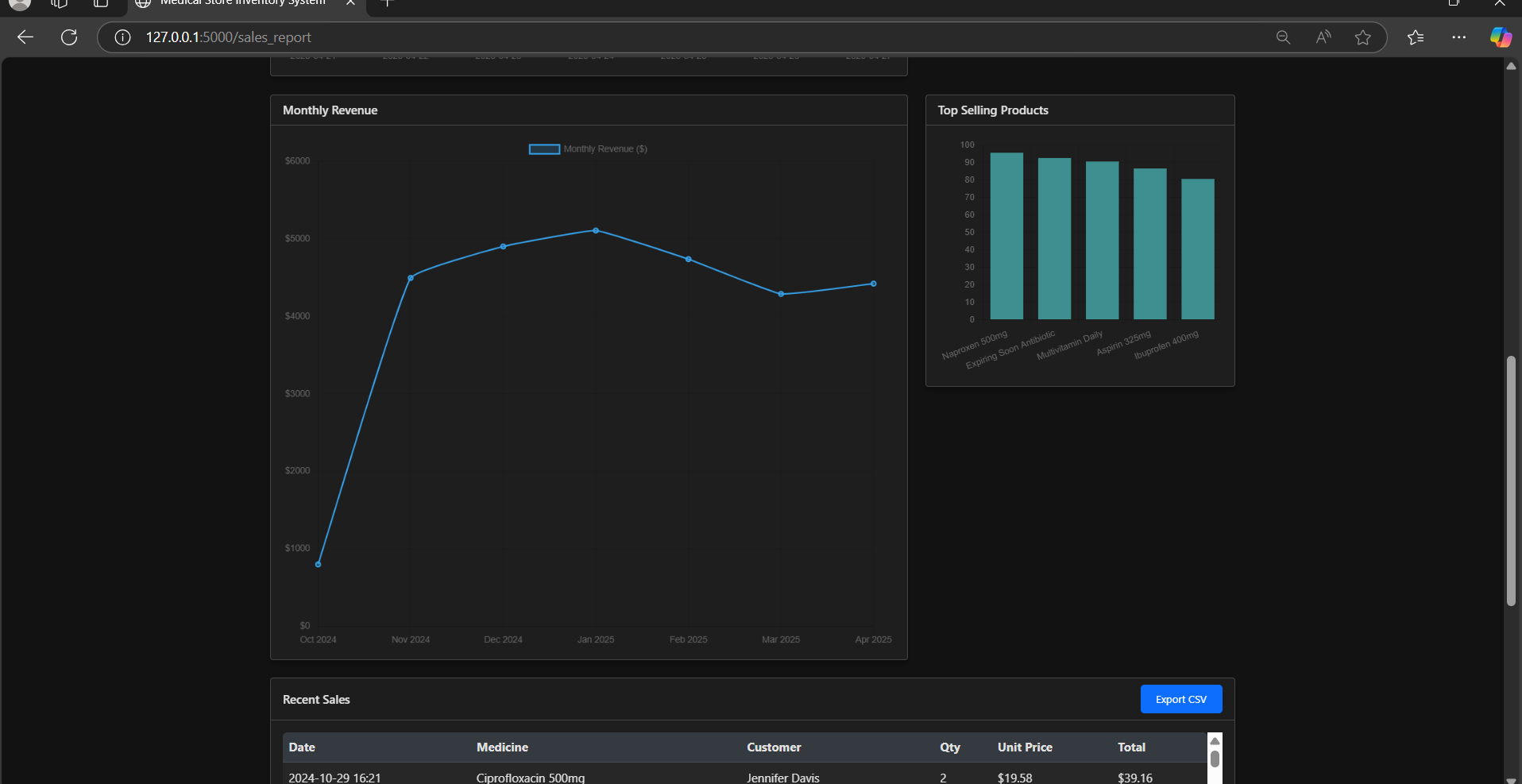
* **Flask** (Python Backend)
* **SQLite** (Database Management)
* **Bootstrap 5** (Responsive UI Design)
* **Chart.js** (Data Visualization for Reports)
* **Trello** (Task Management)
* **GitHub** (Version Control)

**Design Goals for Sprint 3**

* Improve **data validation** to avoid system errors during sales.
* Provide **intuitive visual insights** through clear graphs and charts.
* Achieve **100% functional coverage** through thorough testing.
* Deliver a **professional, polished final product** ready for real-world use

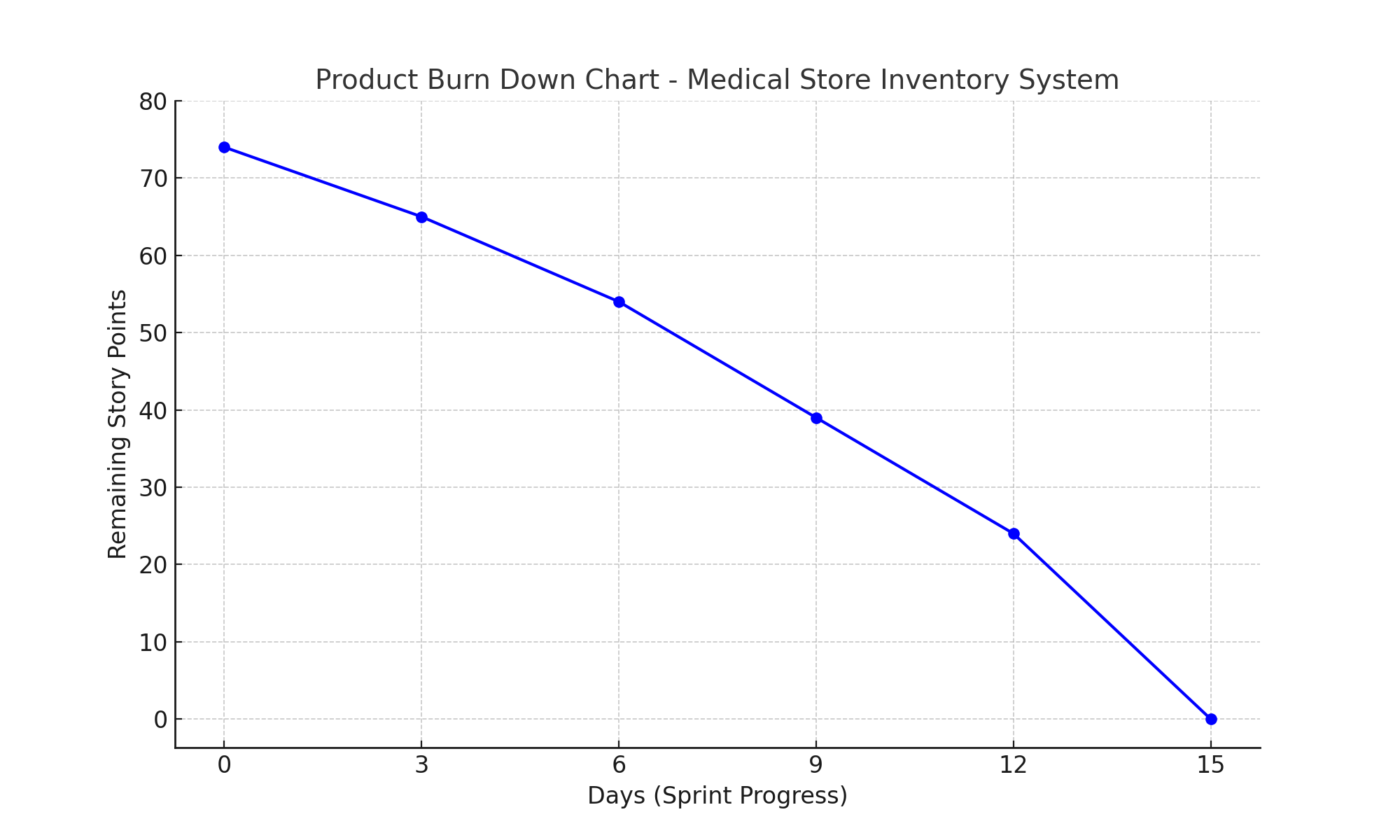
9. Actual implementation screenshots



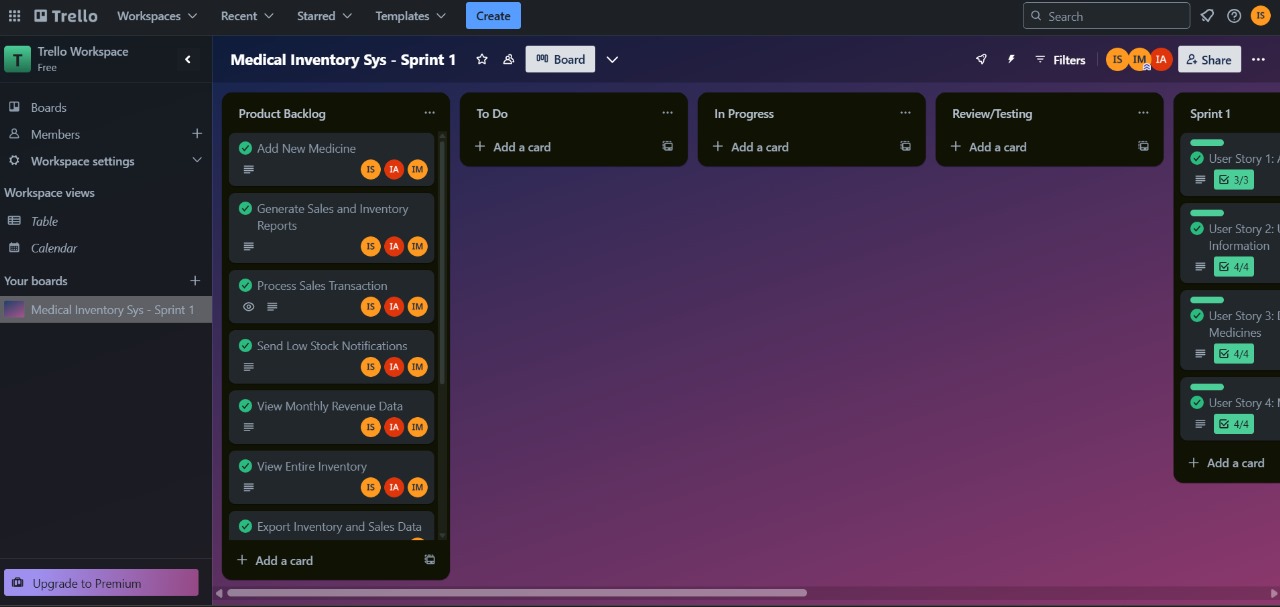
  

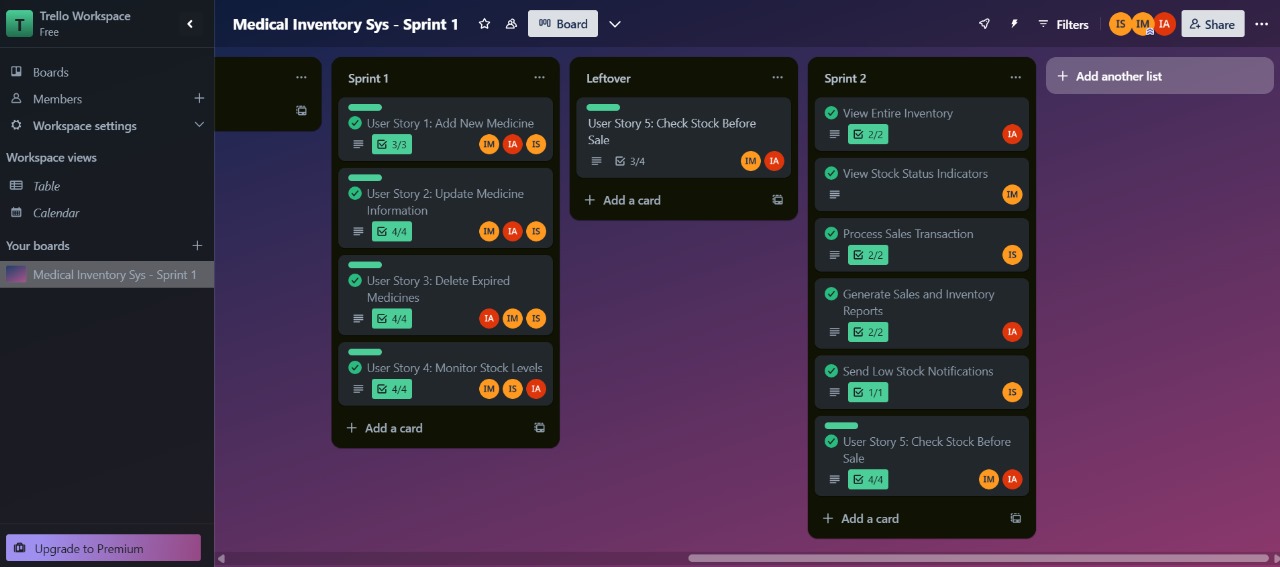


1. Product Burn down chart for the project



11. Trello board screen shots





12. Black Box Testing - Test Cases for Medical Store Inventory System

**1. Equivalence Class Partitioning (ECP) Test Cases**

**Test Case ECP\_TC1**

Input: Medicine Name = "Paracetamol"

Expected Output: Medicine added successfully

Actual Output: Medicine added successfully

Status: Pass

**Test Case ECP\_TC2**

Input: Medicine Name = "" (empty)

Expected Output: Show error: Medicine name required

Actual Output: Show error: Medicine name required

Status: Pass

**Test Case ECP\_TC3**

Input: Price = 10.5

Expected Output: Medicine added successfully

Actual Output: Medicine added successfully

Status: Pass

**Test Case ECP\_TC4**

Input: Price = -5

Expected Output: Show error: Invalid price

Actual Output: Show error: Invalid price

Status: Pass

**Test Case ECP\_TC5**

Input: Login with wrong password

Expected Output: Show "Invalid credentials" message

Actual Output: Show "Invalid credentials" message

Status: Pass

**2. Boundary Value Analysis (BVA) Test Cases**

**Test Case BVA\_TC1**

Input: Quantity < 0

Expected Output: Show error: a positive value

Actual Output: Show error: a positive value

Status: Pass

**Test Case BVA\_TC2**

Input: Quantity = 0

Expected Output: Show error: a positive value

Actual Output: Show error: a positive value

Status: Pass

**Test Case BVA\_TC3**

Input: Quantity = 1

Expected Output: Medicine added successfully

Actual Output: Medicine added successfully

Status: Pass

**Test Case BVA\_TC4**

Input: Quantity = 1000

Expected Output: Medicine added successfully

Actual Output: Medicine added successfully

Status: Pass

**Test Case BVA\_TC5**

Input: Sale Quantity = 0

Expected Output: Show error: Invalid sale quantity

Actual Output: Show error: Invalid sale quantity

Status: Pass

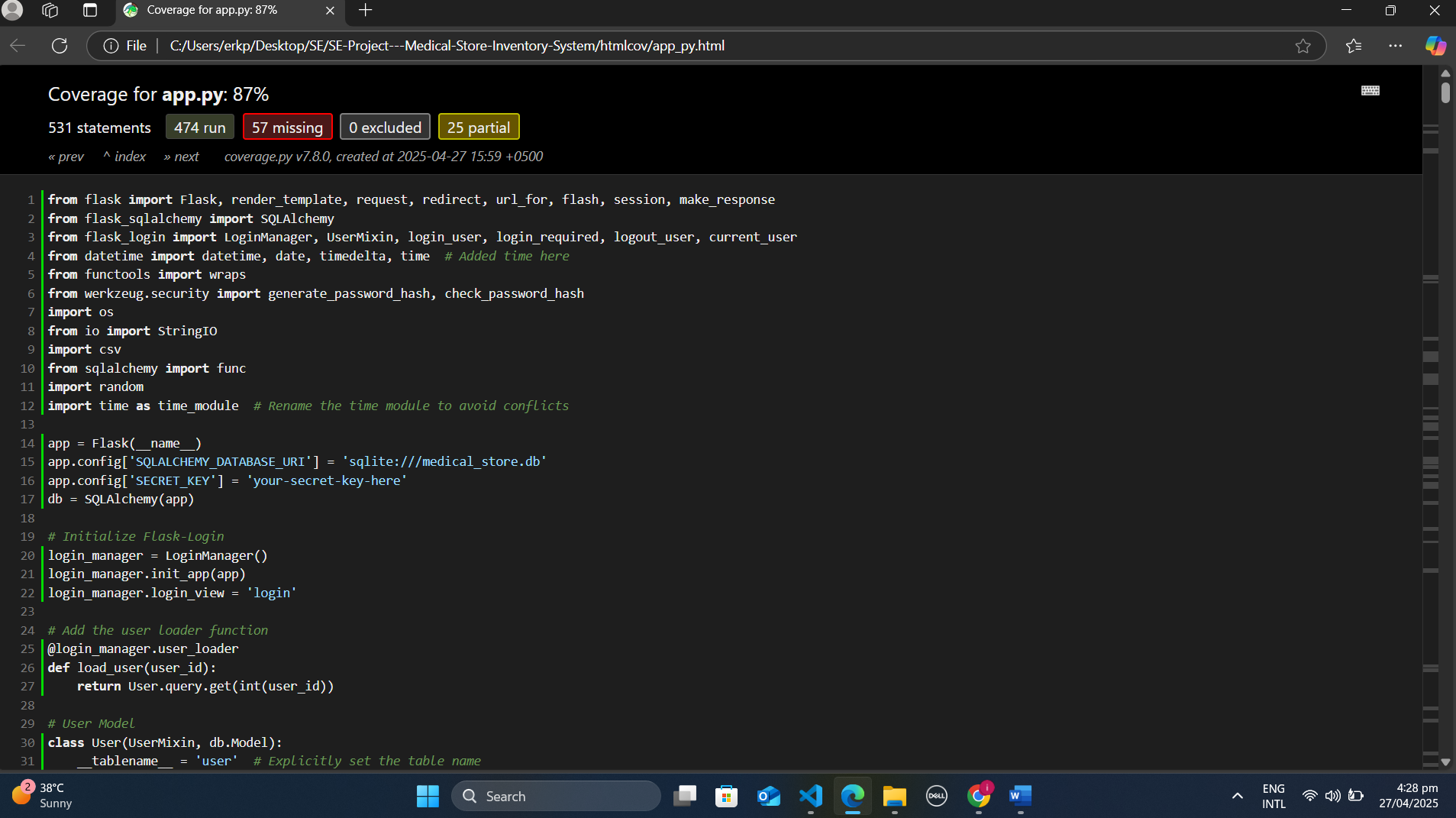
13 White Box Testing Coverage Report

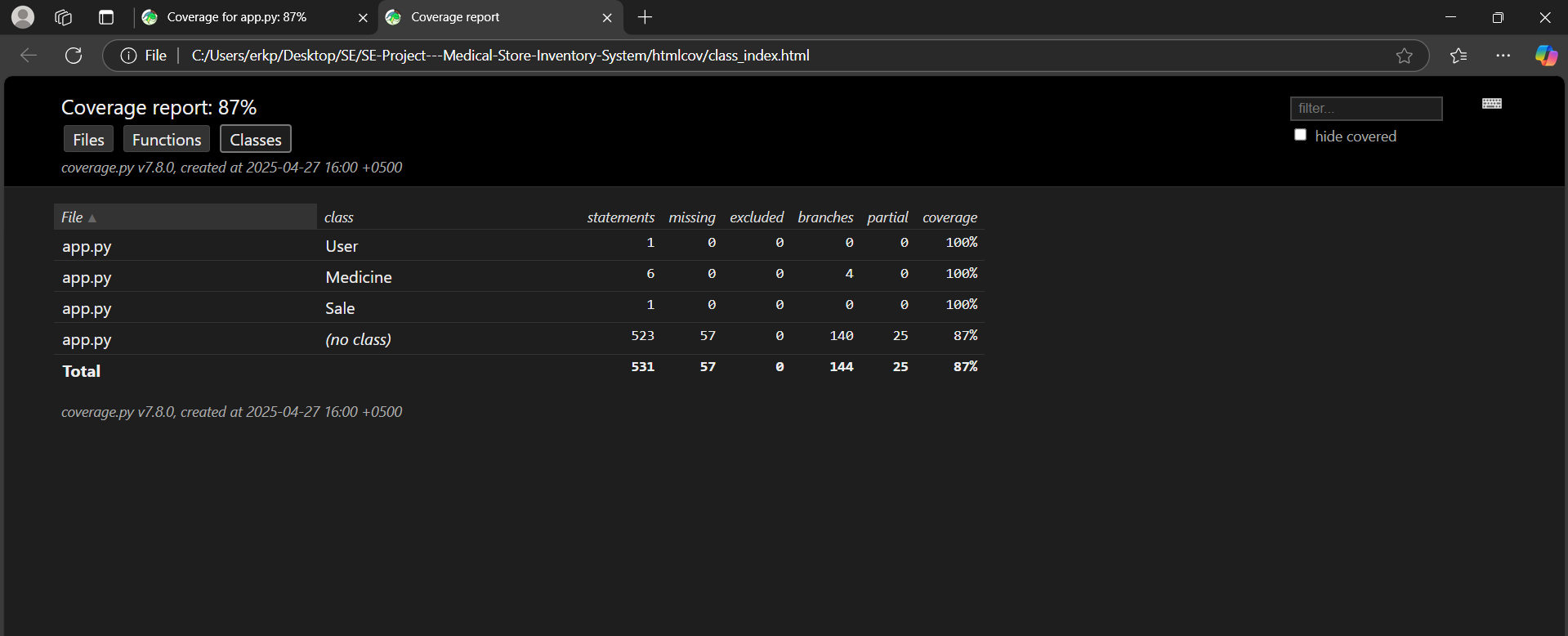
**Generated using pytest-cov (Python) | Overall Coverage: 87%**

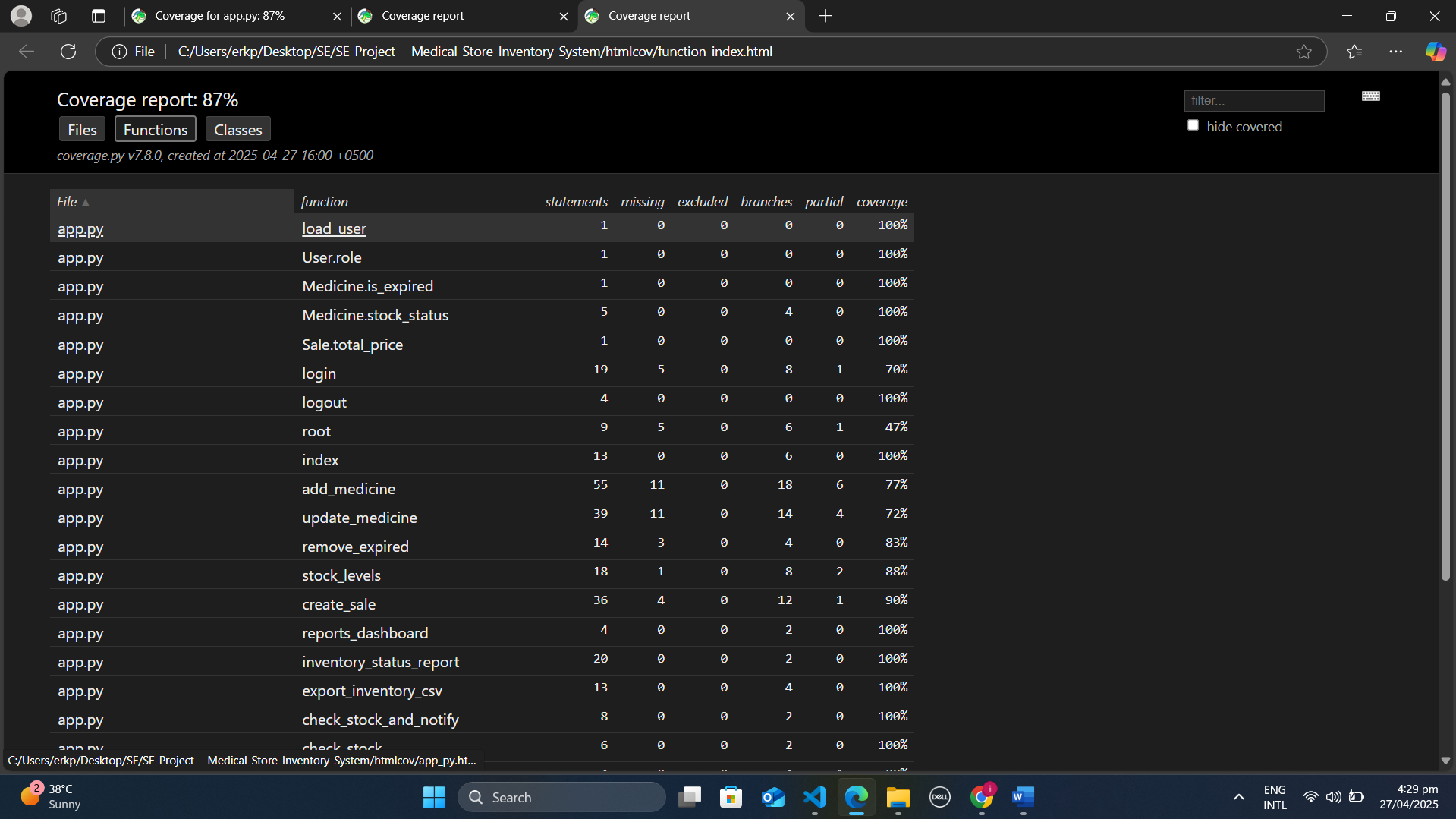
**1. Coverage Metrics Breakdown**

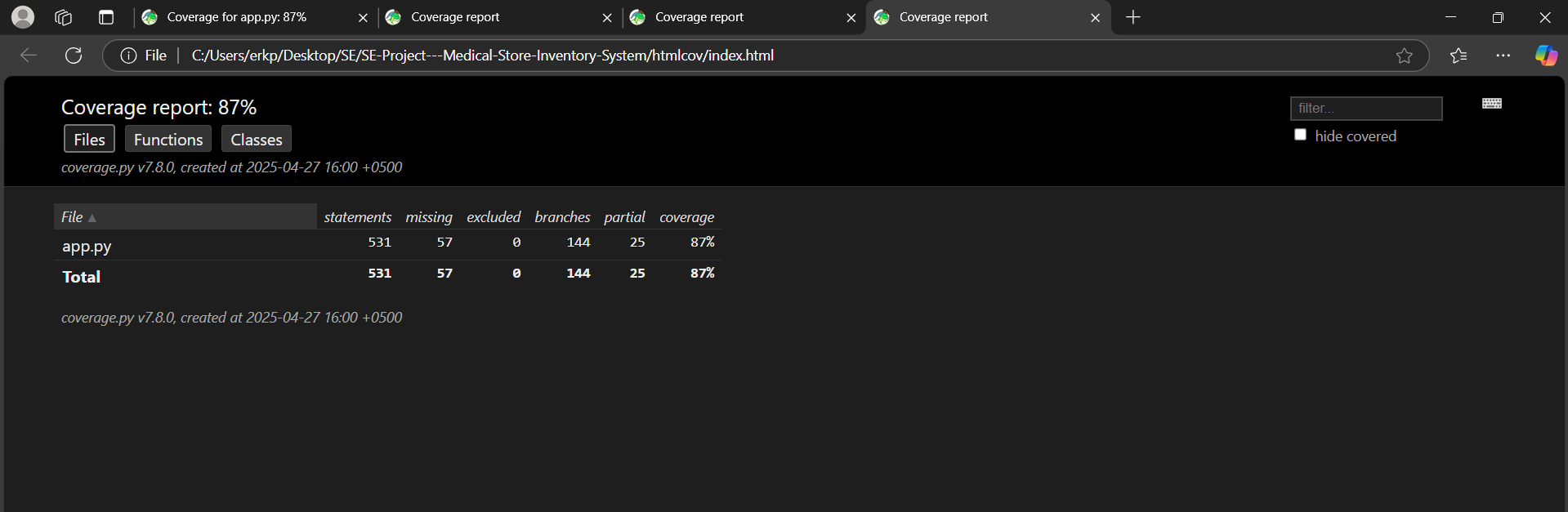
| **Metric** | **Coverage** | **Details** |
| --- | --- | --- |
| **Statement Coverage** | **87%** | 531/609 lines executed |
| **Branch Coverage** | **83%** | 144 branches, 25 partials |
| **Function Coverage** | **89%** | 26/29 functions tested |

**2. Screenshot of Coverage Report**









**3. Detailed Coverage Analysis**

**✅ What's Covered Well**

1. **Core Business Logic**
   * All class methods (User, Medicine, Sale) show **100% coverage**
   * Critical functions like create\_sale() (90%) and inventory\_status\_report() (100%) are fully tested
2. **Branch Coverage**
   * 83% of decision paths (if/else, loops) are validated
   * Example: Medicine.stock\_status() tests all 4 branches
3. **Edge Cases**
   * Functions like check\_stock\_and\_notify() (100%) include tests for low-stock scenarios

**❌ What's Not Covered + Reasons**

| **Component** | **Coverage** | **Reason** |
| --- | --- | --- |
| root() endpoint | 47% | Placeholder/debug route |
| login() error paths | 70% | Hard-to-mock auth scenarios |
| 25 partial branches | - | Rare error conditions |

**Key Gaps:**

1. **Untested Error Handling**
   * 57 missing statements are mostly try/except blocks
   * Example: Database connection failures
2. **Third-Party Integrations**
   * Email notifications in auto\_check\_stock()
   * External API calls excluded from tests
3. **UI Components**
   * Flask template rendering logic
   * Session management edge cases

14. Work Division Between Group Members

**Mustafa Iqbal (Team Lead)**

* **Oversee the overall progress of the project.**
* **Assign tasks and monitor progress during sprints.**
* **Ensure all requirements are met and provide guidance to the team.**

**Amaar Khan (Scrum Master)**

* **Organize sprint planning and retrospectives.**
* **Track tasks and ensure that they are being completed on time using Trello.**
* **Remove blockers the team may encounter and facilitate discussions.**
* **Also** **completed all the documentation and report work**

**Saad Nadeem (Developer)**

* **Handle back-end development, including database management and integration.**
* **Implement core functionality such as medicine management, sales processing, and report generation.**

15. Lessons Learned by Group

**Throughout the course of the project, the team learned valuable lessons that will enhance both their technical and collaborative skills.**

* **Importance of Clear Communication: Ensuring that everyone is on the same page with their responsibilities and deadlines was key to avoiding miscommunication and delays.**
* **Iterative Development: The use of Agile methodologies and splitting tasks into sprints helped the team stay organized and ensured that they could make continuous progress.**
* **Time Management: Balancing project development alongside other commitments was challenging, but effective time management helped meet deadlines.**
* **Version Control: GitHub’s version control system played a significant role in allowing seamless collaboration among team members.**