Project 1: Management System

By: Cristian Verduzco

Developing a Backend System to mange inventory

Key Features:

- User authentication with sessions
- User-specific inventory management
- CRUD (Create, Read, Update, Delete) for inventory items
- Secure session handling and input validation

Technologies Used

Programming Language: Python

Framework: Flask (for web framework and routing)

Database: SQLite (chosen for simplicity and ease of setup)

Tools:

- **Postman**: For testing API endpoints
- **Flask-Login**: For user session management and authentication
- **Flask-Bcrypt**: For password hashing and security

Project Architecture

Authentication:

- User registration, login, and logout with Flask-Login
- Sessions and cookies for secure user tracking

Database Structure:

- **User Model**: Stores user data (username, email, password)
- **InventoryItem Model**: Stores inventory data (item name, description, quantity, price)

Blueprints: Organized routes with Flask blueprints for modularity

Authentication and Security

Authentication Flow:

- Registration: Stores user data with password hashing
- Login: Verifies user and initiates a session
- Logout: Clears session data

Security Measures:

- Password hashing with Flask-Bcrypt
- Sessions with Flask-Login
- Secure session handling with Flask's `SECRET_KEY`

CRUD Operations

Create, Read, Update, Delete: Implemented with routes in `inventory_routes.py`

- Create: Adds new inventory items
- Read: Fetches all items or a specific item for a logged-in user
- Update: Allows modification of an item's details
- Delete: Enables item removal

User-Specific Access: Ensures each user can only manage their own items

Testing and Validation

Testing with Postman:

- Step-by-step testing for each CRUD operation
- Verified session management and access control

Input Validation:

- Used JSON data format and basic error handling for API responses
- Flask-Login's `login_required` to secure routes

Team Contributions

My Role:

- Backend development (Flask, API routes)
- Database design (SQLite, model setup)
- Authentication setup (Flask-Login, Flask-Bcrypt)
- Testing and Validation (Postman, error handling)

Division of Work:

- Cristian Verduzco

Conclusion

Project Completion:

- Successfully met project requirements and tested each component

Potential Future Enhancements:

- Deployment to a cloud platform (e.g., Heroku)
- Enhanced validation and error handling
- Addition of automated testing or documentation