**Inventory Management System**

**Introduction**

The Inventory Management System is a Django-based web application designed to streamline inventory management, ensure real-time analytics, and enhance scalability. It leverages cloud services and big data tools like Hadoop and Apache Kafka to handle large datasets and provide real-time updates.

**Features**

**Authentication System**

* Secure login functionality using Django’s built-in authentication system.
* Prevents unauthorized access to sensitive data.

**CRUD Operations**

* **Create**: Add new inventory items.
* **Read**: View all items with stock balance, purchase price, and sale price.
* **Update**: Modify item details.
* **Delete**: Remove obsolete items.

**Dashboard**

* Displays analytics like:
  + Total Purchase Value
  + Total Sales Value
  + Total Profit
* Date selector for weekly and monthly insights.
* Real-time updates via Kafka integration.

**Dynamic Forms**

* Automatically update price and profit fields based on user input and item selection.

**Real-time Notifications**

* Publish stock transactions to Kafka topics.
* Real-time updates displayed on the dashboard.

**Architecture and Integration**

**Backend**

* **Framework**: Django
* Handles server-side logic, transaction processing, and database operations.

**Frontend**

* **Technologies**: HTML, CSS, and JavaScript
* Provides responsive and user-friendly interfaces.

**Database**

* **AWS RDS (MySQL)**
  + Robust, scalable data storage.
  + Multi-AZ deployment for high availability.

**Hadoop**

* Batch processing of large datasets using:
  + HDFS (Hadoop Distributed File System) for storage.
  + MapReduce jobs for calculating total sales and profits.

**Apache Kafka**

* Real-time data streaming for:
  + Publishing stock transactions.
  + Dashboard updates.

**Setup Instructions**

**Prerequisites**

* Python 3.8 or above
* Django 4.x
* Apache Kafka
* Hadoop (Local setup)
* AWS RDS MySQL (Configured)

**Configure Database**

* Update settings.py with your AWS RDS MySQL credentials.

**Run Migrations**

* python manage.py makemigrations
* python manage.py migrate

**Start the Server**

* python manage.py runserver

**Start Kafka**

* Configure and start Zookeeper.
* Start Kafka broker.
* Create Kafka topics for stock transactions.

**Hadoop Setup**

* Set up HDFS locally.
* Upload transaction\_data.csv to HDFS.
* Run the MapReduce job for analytics.

**Access Application**

* Open the application in a web browser at http://127.0.0.1:8000.

**Challenges and Solutions**

**Database Connectivity**

* **Challenge**: Errors during AWS RDS integration.
* **Solution**: Verified RDS endpoints, IAM roles, and security group settings.

**Real-Time Updates**

* **Challenge**: Setting up Kafka with Zookeeper.
* **Solution**: Debugged configurations using Python’s Kafka library.

**Big Data Processing**

* **Challenge**: Efficiently processing datasets with Hadoop.
* **Solution**: Optimized MapReduce logic and ensured proper HDFS configuration.