**Inventory Management System Documentation**

This document describes the architecture and functionality of the Inventory Management System, which includes a Python-based REST API and a Java-based GUI.

**Overview**

The Inventory Management System allows users to manage items in an inventory database. It supports CRUD (Create, Read, Update, Delete) operations through a Python Flask API and provides a user-friendly interface via a Java-based GUI.

**1. Python Flask API**

**Description**

The Python API handles the backend operations, including database interactions and CRUD functionality. Below is a summary of the key endpoints:

* **GET /api/inventory**: Fetch all items in the inventory.
* **POST /api/inventory**: Add a new item to the inventory.
* **PUT /api/inventory/<item\_id>**: Update an existing item.
* **DELETE /api/inventory/<item\_id>**: Delete an item from the inventory.

**Code**

from flask import Flask, jsonify, request

import mysql.connector

import logging

app = Flask(\_\_name\_\_)

# Configure logging

logging.basicConfig(level=logging.DEBUG)

# MySQL database configuration

db\_config = {

'host': 'localhost',

'user': 'root',

'password': 'root', # Add your MySQL password

'database': 'inventory\_db'

}

# Establish a database connection

def get\_db\_connection():

try:

connection = mysql.connector.connect(\*\*db\_config)

return connection

except mysql.connector.Error as err:

logging.error(f"Database connection error: {err}")

raise

@app.route('/api/inventory', methods=['GET'])

def get\_inventory():

try:

conn = get\_db\_connection()

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM inventory")

inventory = cursor.fetchall()

cursor.close()

conn.close()

return jsonify(inventory)

except Exception as e:

logging.error(f"Error fetching inventory: {e}")

return jsonify({'error': 'Unable to fetch inventory'}), 500

@app.route('/api/inventory', methods=['POST'])

def add\_item():

data = request.get\_json()

name = data.get('name')

quantity = data.get('quantity')

price = data.get('price')

if not name or not quantity or not price:

return jsonify({'error': 'Invalid data'}), 400

try:

conn = get\_db\_connection()

cursor = conn.cursor()

cursor.execute(

"INSERT INTO inventory (name, quantity, price) VALUES (%s, %s, %s)",

(name, quantity, price)

)

conn.commit()

cursor.close()

conn.close()

return jsonify({'message': 'Item added successfully'})

except Exception as e:

logging.error(f"Error adding item: {e}")

return jsonify({'error': 'Unable to add item'}), 500

@app.route('/api/inventory/<int:item\_id>', methods=['DELETE'])

def delete\_item(item\_id):

try:

conn = get\_db\_connection()

cursor = conn.cursor()

cursor.execute("DELETE FROM inventory WHERE id = %s", (item\_id,))

conn.commit()

cursor.close()

conn.close()

return jsonify({'message': 'Item deleted successfully'})

except Exception as e:

logging.error(f"Error deleting item: {e}")

return jsonify({'error': 'Unable to delete item'}), 500

@app.route('/api/inventory/<int:item\_id>', methods=['PUT'])

def update\_item(item\_id):

data = request.get\_json()

name = data.get('name')

quantity = data.get('quantity')

price = data.get('price')

if not name or not quantity or not price:

return jsonify({'error': 'Invalid data'}), 400

try:

conn = get\_db\_connection()

cursor = conn.cursor()

cursor.execute(

"UPDATE inventory SET name = %s, quantity = %s, price = %s WHERE id = %s",

(name, quantity, price, item\_id)

)

conn.commit()

cursor.close()

conn.close()

return jsonify({'message': 'Item updated successfully'})

except Exception as e:

logging.error(f"Error updating item: {e}")

return jsonify({'error': 'Unable to update item'}), 500

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**2. Java GUI**

**Description**

The Java GUI interacts with the Python API and provides a visual interface for users to perform inventory management tasks.

**Files and Responsibilities**

**AddItemFrame.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sia;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.PreparedStatement;

public class AddItemFrame extends JFrame {

public AddItemFrame() {

setTitle("Add Item");

setSize(400, 300);

setLayout(null);

JLabel lblName = new JLabel("Name:");

lblName.setBounds(50, 50, 100, 30);

JTextField txtName = new JTextField();

txtName.setBounds(150, 50, 200, 30);

JLabel lblQuantity = new JLabel("Quantity:");

lblQuantity.setBounds(50, 100, 100, 30);

JTextField txtQuantity = new JTextField();

txtQuantity.setBounds(150, 100, 200, 30);

JLabel lblPrice = new JLabel("Price:");

lblPrice.setBounds(50, 150, 100, 30);

...

**ApiClient.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sia;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.net.HttpURLConnection;

import java.net.URL;

public class ApiClient {

private static final String BASE\_URL = "http://127.0.0.1:5000/api/inventory";

// GET Request

public static String get(String endpoint) {

try {

URL url = new URL(BASE\_URL + endpoint);

HttpURLConnection connection = (HttpURLConnection) url.openConnection();

connection.setRequestMethod("GET");

int responseCode = connection.getResponseCode();

if (responseCode == 200) {

BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));

StringBuilder response = new StringBuilder();

String line;

...

**DatabaseConnection.java**

package sia;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final String URL = "jdbc:mysql://localhost:3306/inventory\_db";

private static final String USER = "root";

private static final String PASSWORD = "root";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

}

**DeleteItemFrame.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sia;

import javax.swing.\*;

import java.sql.Connection;

import java.sql.PreparedStatement;

public class DeleteItemFrame extends JFrame {

public DeleteItemFrame() {

setTitle("Delete Item");

setSize(400, 200);

setLayout(null);

JLabel lblId = new JLabel("Item ID:");

lblId.setBounds(50, 50, 100, 30);

JTextField txtId = new JTextField();

txtId.setBounds(150, 50, 200, 30);

JButton btnDelete = new JButton("Delete Item");

btnDelete.setBounds(150, 100, 150, 30);

btnDelete.addActionListener(e -> {

int id = Integer.parseInt(txtId.getText());

try (Connection conn = DatabaseConnection.getConnection()) {

String sql = "DELETE FROM inventory WHERE id = ?";

PreparedStatem...

**InventoryManagementSystem.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sia;

import javax.swing.\*;

public class InventoryManagementSystem {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new MainMenu();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

});

}

}

**MainMenu.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sia;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class MainMenu extends JFrame {

public MainMenu() {

setTitle("Inventory Management System");

setSize(400, 300);

setLayout(null);

JButton btnAddItem = new JButton("Add Item");

btnAddItem.setBounds(50, 50, 150, 30);

btnAddItem.addActionListener(e -> {

new AddItemFrame().setVisible(true);

});

JButton btnUpdateItem = new JButton("Update Item");

btnUpdateItem.setBounds(50, 100, 150, 30);

btnUpdateItem.addActionListener(e -> {

new UpdateItemFrame().setVisible(true);

});

JButton btnDeleteItem = new JButton("Delete Item");

btnDeleteItem.se...

**UpdateItemFrame.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sia;

import javax.swing.\*;

import java.sql.Connection;

import java.sql.PreparedStatement;

public class UpdateItemFrame extends JFrame {

public UpdateItemFrame() {

setTitle("Update Item");

setSize(400, 300);

setLayout(null);

JLabel lblId = new JLabel("Item ID:");

lblId.setBounds(50, 50, 100, 30);

JTextField txtId = new JTextField();

txtId.setBounds(150, 50, 200, 30);

JLabel lblName = new JLabel("New Name:");

lblName.setBounds(50, 100, 100, 30);

JTextField txtName = new JTextField();

txtName.setBounds(150, 100, 200, 30);

JLabel lblQuantity = new JLabel("New Quantity:");

lblQuantity.setBounds(50, 150, 100, 30);

JTextField txtQuantity = new JTextField();

txtQuantity.setBoun...

**ViewInventoryFrame.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package sia;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

public class ViewInventoryFrame extends JFrame {

public ViewInventoryFrame() {

setTitle("View Inventory");

setSize(600, 400);

DefaultTableModel model = new DefaultTableModel();

JTable table = new JTable(model);

model.addColumn("ID");

model.addColumn("Name");

model.addColumn("Quantity");

model.addColumn("Price");

try (Connection conn = DatabaseConnection.getConnection()) {

String sql = "SELECT \* FROM inventory";

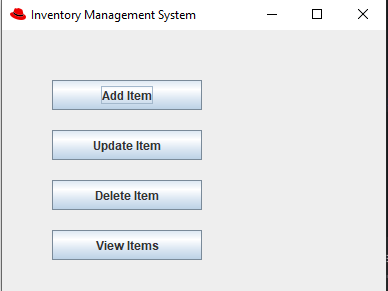
Statement stmt = conn.createStatement();

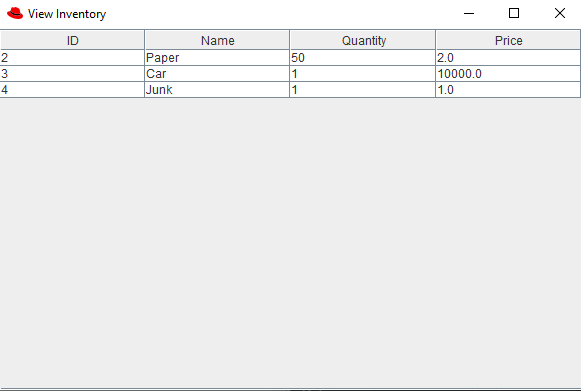
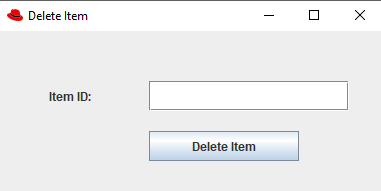
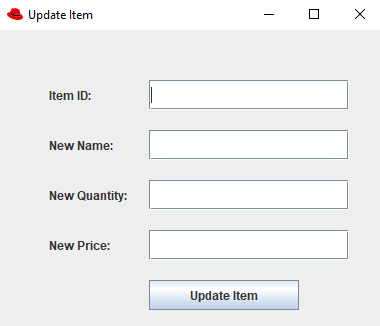
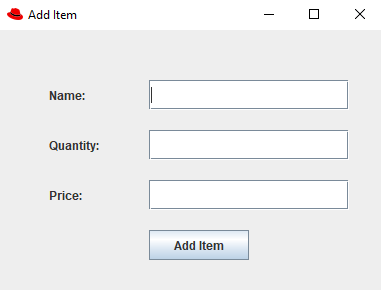
ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

...

**Pictures of the GUI**





**Database Schema**

The inventory database (inventory\_db) includes the following table:

**inventory**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INT | Primary key |
| name | VARCHAR | Name of the item |
| quantity | INT | Quantity of the item |
| price | FLOAT | Price of the item |

**How to Run**

**Python API**

1. Install dependencies: pip install flask mysql-connector-python.
2. Configure MySQL credentials in the Python file.
3. Run the API: python Product.py.

**Java GUI**

1. Compile the Java files: javac \*.java.
2. Run the main class: java InventoryManagementSystem.