**Industrial Production. Data Management Application Software.**

**Software Requirements Specification.**

# **Table of Contents**

[**Table of Contents**](#_gq3iceqftwld) **1**

[**Software Requirements Specification**](#_2l01mwh28i7d) **2**

[1. Introduction](#_ptaqo26tn1zk) 2

[1.1 Purpose](#_pkh8mvc3st34) 2

[1.2 Scope](#_fpjt6w86kyc7) 2

[1.3 Definitions, Acronyms, and Abbreviations](#_x3auuvus54p2) 2

[1.3 References](#_t504ayxtojn0) 2

[2. Overall description](#_p6otaz7vfdmq) 3

[2.1 Who are the users?](#_8ga3i0gxayvc) 3

[2.2 How will it help users?](#_4kaj4rlu85km) 3

[2.3 What kind of product is it?](#_wtsrnh9cympw) 3

[3. Specific Requirements](#_drukaivi4285) 4

[3.1 Functional Requirements](#_y5lklhhjv8xs) 4

[3.1.1 User Account Type](#_ydqc0vmawsj) 4

[3.1.2 User Login](#_z7xc77c1upv7) 4

[3.1.3 Salesman Menu](#_tsodnbcqa8nt) 4

[3.1.4 Provider Menu](#_1cnjzeu2lczr) 5

[3.1.5 Deliver Menu](#_qdrz3x5f2p91) 6

[3.2 Reliability & Availability](#_lt43gff400px) 7

[3.3 Security Requirements](#_affi5nwsiwf3) 7

[3.4 Interfaces](#_64w7oc81dmyb) 7

[3.4.1 User Interface](#_onad6v3dofl5) 7

[3.4.2 Communication Interface](#_t7t3nea5zl7e) 7

[4. Implementation](#_nxiy9llohun7) 8

[4.1 Main class](#_k27ju0l5o0xz) 8

[4.2 File Handling](#_9ina70e97agf) 9

[4.3 Main Menu](#_4qbf1puhzs3k) 12

[4.4 Some models for data structure](#_8p3pnma9xpm3) 14

[Conclusion](#_3uijhvxqq3tr) 15

# **Software Requirements Specification**

## **1. Introduction**

This project is built for educational purposes to receive and improve software development skills.

### **1.1 Purpose**

Current software is developed for simplicity of store workflow. It automates tasks like

* tracking the amount of products in the store
* tracking the amount of products which have been sold
* tracking the amount of products to deliver
* tracking the price, title and all the necessary information about the products
* adding new products
* ordering a new product

and all the tasks are secured from unwanted users by implementing the authorization system.

The product will be used by store owners or staff. They will work with the database using the user interface provided by the software.

### **1.2 Scope**

Any store which is interested in better business management would be interested in installing this kind of product. Stores could use it to handle their products flow by storing them in a database and automating the selling, ordering and new product adding process.

### **1.3 Definitions, Acronyms, and Abbreviations**

| User Interface (UI) | The point of user interaction with the software. |
| --- | --- |
| Database | An organized collection of structured information stored in a computer system. |
| CLI | Command Line Interface is a command line program that uses text inputs to execute different tasks. |

### **1.3 References**

* [Maven Documentation](https://maven.apache.org)
* [Java Documentation (Javatpoint)](https://www.javatpoint.com)
* [GSON (JSON Converter)](https://github.com/google/gson)

## **2. Overall description**

### **2.1 Who are the users?**

The product is supposed to be used by store owners and staff. They will manage their products by adding them into a database. They will work with the database using the software application with a graphical user interface, which will simplify the workflow.

Users need to be authorized to have access to the application. Once they’ve managed to have it they can use the soft. The functionality will depend on the staff’s position.

### **2.2 How will it help users?**

It will help stores to better manage their data and to keep track of their data flow. Users can view the information about the products, delete products, add products etc. depending on account type.

### **2.3 What kind of product is it?**

That is a new product which doesn’t depend on existing products. That is going to be a desktop application which will be supported on Windows, Linux and Mac. Companies will be able to download the product from our website after purchasing it.

## 

## **3. Specific Requirements**

### **3.1 Functional Requirements**

This section contains functional requirements for the software. It describes the features that the application provides.

#### **3.1.1 User Account Type**

* The system will display account type menu
* The system will ask user to choose an account type
* The system will accept user input in terminal
* The system will display an error message in case of wrong input



#### **3.1.2 User Login**

* The system will display authorization menu
* The system will allow user to enter the username and password
* The system will verify username and password
* In case of error the system will display an error message



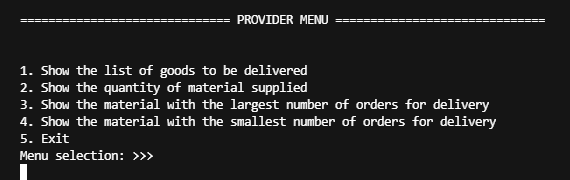
#### **3.1.3 Salesman Menu**

* The system will display a new window depending on account type
* The system will allow the user to choose from the menu
* The system will give an option to show the list of all products to sell
* The system will give an option to search for products
* The system will give an option to show the sales report
* The system will give an option to sell products
* The system will give an option to order a product
* The system will give an option to remove a product
* The system will give an option to exit
* The system will display data in a table

#### 

#### **3.1.4 Provider Menu**

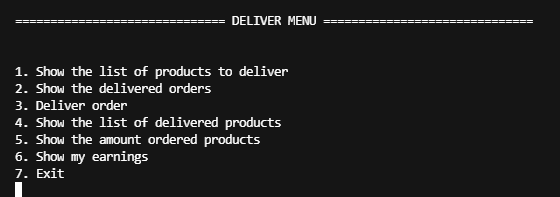
* The system will display a new window depending on account type
* The system will allow the user to choose from the menu
* The system will give an option to show the list of goods to be delivered
* The system will give an option to show the amount of products supplied
* The system will give an option to show the product with the largest number of orders for delivery
* The system will give an option to show the product with the smallest number of orders for delivery
* The system will give an option to exit



#### 

#### **3.1.5 Deliver Menu**

* The system will display a new window depending on account type
* The system will allow the user to choose from the menu
* The system will give an option to show the list of products to deliver
* The system will give an option to show the delivered orders
* The system will give an option to deliver the order
* The system will give an option to show the list of delivered products
* The system will give an option to show the earnings
* The system will give an option to exit



### **3.2 Reliability & Availability**

* The database should be hosted on a secure server and require authorization to access the database.
* The database should be accessible through internet so application can have access to the database anywhere with internet connection

### **3.3 Security Requirements**

This section describes security requirements to the software. It is necessary to prevent unwanted access to the system.

* Usernames and passwords have to be encrypted
* Database should be behind a firewall

### **3.4 Interfaces**

#### **3.4.1 User Interface**

* The user interface should be implemented using Java and any Java tools for building user interfaces or as a CLI.
* The user interface should not be complicated and easy to use and understand.

#### **3.4.2 Communication Interface**

* Application should establish connections with database using TCP/IP protocol

## **4. Implementation**

Here is some code for the application implementation:

### **4.1 Main class**

package com.boorsoft;

import java.util.ArrayList;

import com.boorsoft.components.AccountHandler;

import com.boorsoft.components.AccountTypeHandler;

import com.boorsoft.components.menus.Menu;

import com.boorsoft.helpers.FileHandler;

import com.boorsoft.models.Person;

import com.boorsoft.models.AccountType;

public class App {

public static void main(String[] args) throws Error {

ArrayList<Person> accounts = new ArrayList<Person>();

ArrayList<AccountType> accountTypes = new ArrayList<AccountType>();

new AccountTypeHandler(accountTypes);

new AccountHandler(accounts, accountTypes);

FileHandler fileHandler = new FileHandler();

fileHandler.init();

Menu menu = new Menu(accounts, accountTypes);

menu.displayStartingMessage();

}

}

### 

### 

### 

### **4.2 File Handling**

package com.boorsoft.helpers;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.lang.reflect.Type;

import java.util.ArrayList;

import com.boorsoft.models.DeliveredProduct;

import com.boorsoft.models.GoodModel;

import com.boorsoft.models.SoldProduct;

import com.boorsoft.models.ToSell;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import com.google.gson.stream.JsonReader;

public class FileHandler {

public void init() {

try {

File goods = new File(Constants.goodsPath);

File toSell = new File(Constants.toSellPath);

File sold = new File(Constants.soldPath);

File delivered = new File(Constants.deliveredPath);

if (goods.createNewFile() ||

toSell.createNewFile() ||

sold.createNewFile() ||

delivered.createNewFile()) System.out.println("File created.");

else System.out.println("File already exists.");

} catch(IOException e) {

System.out.println("File error: " + e.getMessage());

}

}

public static ArrayList<GoodModel> getGoods() throws FileNotFoundException {

JsonReader reader = new JsonReader(new FileReader(Constants.goodsPath));

Type objectsType = new TypeToken<ArrayList<GoodModel>>() {}.getType();

ArrayList<GoodModel> data = new Gson().fromJson(reader, objectsType);

if (data == null) {

*// System.out.println("No data found.");*

return new ArrayList<GoodModel>();

}

return data;

}

public static ArrayList<ToSell> getToSell() throws FileNotFoundException {

JsonReader reader = new JsonReader(new FileReader(Constants.toSellPath));

Type objectsType = new TypeToken<ArrayList<ToSell>>() {}.getType();

ArrayList<ToSell> data = new Gson().fromJson(reader, objectsType);

if (data == null) {

*// System.out.println("No data found.");*

return new ArrayList<ToSell>();

}

return data;

}

public static ArrayList<SoldProduct> getSold() throws FileNotFoundException {

JsonReader reader = new JsonReader(new FileReader(Constants.soldPath));

Type objectsType = new TypeToken<ArrayList<SoldProduct>>() {}.getType();

ArrayList<SoldProduct> data = new Gson().fromJson(reader, objectsType);

if (data == null) {

*// System.out.println("No data found.");*

return new ArrayList<SoldProduct>();

}

return data;

}

public static ArrayList<DeliveredProduct> getDelivered() throws FileNotFoundException {

JsonReader reader = new JsonReader(new FileReader(Constants.deliveredPath));

Type objectsType = new TypeToken<ArrayList<DeliveredProduct>>() {}.getType();

ArrayList<DeliveredProduct> data = new Gson().fromJson(reader, objectsType);

if (data == null) {

return new ArrayList<DeliveredProduct>();

}

return data;

}

public static void saveSoldProducts(ArrayList<SoldProduct> soldProducts) throws IOException {

FileWriter fileWriter = new FileWriter(Constants.soldPath);

fileWriter.write(new Gson().toJson(soldProducts, new TypeToken<ArrayList<SoldProduct>>() {}.getType()));

fileWriter.close();

}

public static void saveToSell(ArrayList<ToSell> toSell) throws IOException {

FileWriter fileWriter = new FileWriter(Constants.toSellPath);

fileWriter.write(new Gson().toJson(toSell, new TypeToken<ArrayList<ToSell>>() {}.getType()));

fileWriter.close();

}

public static void saveGoods(ArrayList<GoodModel> goods) throws IOException {

FileWriter fileWriter = new FileWriter(Constants.goodsPath);

fileWriter.write(new Gson().toJson(goods, new TypeToken<ArrayList<GoodModel>>() {}.getType()));

fileWriter.close();

}

public static void saveDelivered(ArrayList<DeliveredProduct> delivered) throws IOException {

FileWriter fileWriter = new FileWriter(Constants.deliveredPath);

fileWriter.write(new Gson().toJson(delivered, new TypeToken<ArrayList<DeliveredProduct>>() {}.getType()));

fileWriter.close();

}

}

### **4.3 Main Menu**

package com.boorsoft.components.menus;

import java.util.ArrayList;

import java.util.Scanner;

import com.boorsoft.models.AccountType;

import com.boorsoft.models.Person;

import com.boorsoft.components.AccountHandler;

import com.boorsoft.helpers.Error;

public class Menu {

ArrayList<Person> accounts = new ArrayList<Person>();

ArrayList<AccountType> accountTypes = new ArrayList<AccountType>();

public Menu(ArrayList<Person> accounts, ArrayList<AccountType> accountTypes) {

this.accounts = accounts;

this.accountTypes = accountTypes;

}

public void displayStartingMessage() {

Scanner scanner = new Scanner(System.in);

try {

System.out.println(

"\n============================== MAIN MENU ==============================");

System.out.println("Enter your account type: ");

System.out.println("1. Salesman \n2. Provider \n3. Deliver \n");

System.out.print("\n>> ");

int accountTypeInput = scanner.nextInt();

switch (accountTypeInput) {

case 1:

System.out.println("Welcome, dear salesman");

break;

case 2:

System.out.println("Welcome, dear provider");

break;

case 3:

System.out.println("Welcome, dear deliver");

break;

default:

scanner.close();

throw new Error("No account type found.");

}

System.out.println("\nEnter your credentials.");

System.out.print("\nLogin: ");

String login = scanner.next();

System.out.print("\nPassword: ");

String password = scanner.next();

AccountHandler.checkAuth(accounts, login, password);

AccountHandler.checkAccountType(accountTypeInput, accountTypes);

scanner.close();

} catch (Error e) {

System.out.println(e.getMessage());

}

}

}

### **4.4 Some models for data structure**

**User model**

package com.boorsoft.models;

public class Person {

public int id;

public String username;

public String password;

public AccountType accountType;

public Person(int id, String username, String password, AccountType accountType) {

this.id = id;

this.username = username;

this.password = password;

this.accountType = accountType;

}

}

**Good model (product)**

**package com.boorsoft.models;**

**import com.google.gson.Gson;**

**public class GoodModel {**

**public int id;**

**public String title;**

**public int amount;**

**public String deliveryDate;**

**public String orderDate;**

**public GoodModel(int id, String title, int amount, String deliveryDate, String orderDate) {**

**this.id = id;**

**this.title = title;**

**this.amount = amount;**

**this.deliveryDate = deliveryDate;**

**this.orderDate = orderDate;**

**}**

**public String toJson() {**

**return new Gson().toJson(this);**

**}**

**}**

## 

## **Conclusion**

The software was developed to improve the workflow of stores and solving problems by automating various tasks.

During the process of working on this project we’ve learned to use Maven to set up better Java projects, to work with JSON files in Java using the library GSON. We’ve improved knowledge of data structures and algorithms. Also we’ve learned to better work in a team using Git and to write SRS documents.

We have managed to finish this project despite lots of mistakes and difficulties. The course paper has let us receive a huge experience in application development.

# 