VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

A Project Report

WAREHOUSE INVENTORY MANAGEMENT SYSTEM

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in

Computer Science and Engineering

Submitted by

DANIYAL PARVEEZ

1BG15CS026

Vidyaya Amrutham Ashnuthe

B.N.M. Institute of Technology

(Approved by AICTE, Affiliated to VTU, ISO 9001:2008 Certified and NAAC accredited grade A Institution)
Postbox No. 7087, 27th Cross, 12th Main, Banashankari 2nd Stage, Bengaluru 560 070

Department of Computer Science and Engineering

2017 – 2018

B.N.M. Institute of Technology

(Approved by AICTE, Affiliated to VTU, ISO 9001:2008 Certified and NAAC accredited grade A Institution)
Postbox No. 7087, 27th Cross, 12th Main, Banashankari 2nd Stage, Bengaluru 560 070

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vidyaya Amrutham Ashnuthe

CERTIFICATE

Certified that the Project entitled Warehouse Inventory Management System carried out by Mr. **Daniyal Parveez** USN **1BG15CS026**, bonafide student of V Semester BE, **B.N.M Institute of Technology** in partial fulfillment for the Bachelor of Engineering in COMPUTER SCIENCE AND ENGINEERING of the **Visvesvaraya Technological University**, Belagavi during the year 2017-18. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report. The Project report has been approved as it satisfies the academic requirements in respect of Database Management System with Mini Project Laboratory prescribed for the said Degree.

Dr. Sahana D. Gowda
Professor and HOD
Department of CSE
BNMIT, Bengaluru

Name & Signature

Examiner 1:

Examiner 2:

Table of Contents

ABSTRACT

Warehouses in any sort of industry deal with large volumes of incoming and outgoing stock every day. These commercial entities are built primarily for the storage of goods which can include raw material, packing material, spare parts or finished goods. They are used by manufacturers, importers, exporters and wholesalers, transport businesses and customs, etc. As such, warehouses form the backbone of any production unit or transport network.

These warehouses, and the businesses/industries that rely on them need a cost-effective and efficient solution for keeping track of inventory levels, orders, sales and deliveries.

Any sort of mismanagement at any level can lead to problems like - understocking or overstocking of inventory, delay in deliveries, poor handling of shipments, etc. Such problems can affect the business significantly, leading to heavy financial loss. Considering the dominance of software technology in the current decade, it is only natural that any solution aimed at preventing this kind of mismanagement will involve a sophisticated software system. Our proposed Inventory Management System is among the class of systems that are deployed in typical industrial/warehouse environments to ensure the accurate and seamless management of inventory.

These systems provide sophisticated features for managing all kinds of data related to stock and transactions. Backed by powerful database management systems that are capable of holding large amounts of data efficiently and easily, coupled with easily navigable interfaces that readily provide access to a plethora of powerful functions, these systems are essential to ensure the smooth running of any business that relies on any sort of inventory handling on any scale.

The aim of this project is to develop a simple inventory management system for a fictional warehouse that serves a fictional electronics and sporting goods store. While the project doesn't seek to incorporate all the complex functionality found in commercial applications, it does serve to demonstrate how inventory management works on a smaller simplified scale. To keep things simple, many assumptions are made regarding how orders are placed, deliveries are handled, stock is managed, etc.

Given the nature of the application, it can currently only be used for our own fictional warehouse. However, with minor changes, it could be modified to work in any typical warehouse scenario.

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project.

I would like to thank **Shri. Narayan Rao R Maanay**, Secretary, BNMIT, Bengaluru for providing excellent academic environment in the college.

I would like to sincerely thank **Prof. T J Rama Murthy**, Director, BNMIT, Bengaluru for having extended his support and encouragement during the course of the work.

I would like to express my gratitude to **Dr. M S Suresh**, Dean, BNMIT, Bengaluru for his relentless support, guidance and assistance.

I would like to thank **Dr. Krishnamurthy G N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank **Dr. Sahana D Gowda**, Professor and Head of the Department of Computer Science and Engineering who has shared her opinions and thoughts which helped me in giving my presentation successfully.

Finally, I take this opportunity to extend my earnest gratitude and respect to my parents, teaching & non-teaching staffs of department and all my friends, for giving me valuable advices and support at all times in all possible ways

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Overview of Database Management Systems

A DBMS, or Database Management System is a system software utility used for the creation, easy manipulation and effective management of databases. They allow users to create, update, delete and access data stores efficiently and securely.

Database Management Systems provide: -

- Logical, structured organization of data.
- Data abstraction and independence.
- Data security.
- Concurrent access to data.
- Backup and Recovery capabilities.
- Robust data integrity capabilities.
- Logging and auditing of activity.
- Uniform administration procedures for data.

Relational Database Management Systems (or RDBMSs) are the most popular DBMSs used for financial records, logistical information, personnel data, and other applications. They present data as relations, and provided relational operators to manipulate this data.

1.2 Problem Statement

Warehouses in any industry deal with huge volumes of incoming/outgoing inventory every day. This inventory may involve raw materials, packaging material, spare part and components or simply finished goods.

Given the number of transactions and exchanges involved, there is a large amount of data to be managed. Tracking inventory levels, orders, sales and deliveries is extremely complex and time consuming, and any mismanagement can mean problems. Inventory Management Systems are employed in such places to effectively manage inventory data and daily

exchanges. This ensures product overstocking, outages and confusion are avoided, thus allowing for the smooth functioning of warehouses, and the industries that depend on them.

1.3 Objectives

The chief goal here is to implement an inventory management system that can be deployed in our fictional warehouse environment. Our fictional warehouse will stock products for a brands store that sells a range of electronics and sports apparel. The system developed should provide all essential functionalities in an easy-to-use package. The application should at the minimum be able to: -

- Keep track of all inventory (in our case, electronics and apparel).
- Keep record of incoming shipments.
- Allow orders to be placed.
- Maintain data on vendors.
- Monitor stock levels and generate alerts for dwindling/depleted stock.
- Allow stock to be moved to store as per demand.

CHAPTER 2

SYSTEM REQUIREMENTS

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software (Front end & Back end) & Hardware

Front End Requirements

- Windows 7 (32-bit/64-bit) or above Operating System
- Java SE Runtime Environment (JRE) 8 or above

Back End Requirements

- Oracle XE 11g
- Oracle SQL Developer Version 17
- Oracle Database 11g Release 2 JDBC Thin Driver (ojdbc6.jar)
- Java SE Development Kit (JDK) 8.0 or above
- JavaFX Scene Builder 8 or above
- Any standard Java IDE (Eclipse, IntelliJ IDEA, NetBeans, etc.)

Hardware Requirements

Requirement	Minimum	Recommended
Memory	512 MB	1 GB
Free Disk Space	300 MB	1 GB or more
Processor Speed	800 MHz	1.5 GHz and above
GPU	DirectX 9.0 compliant GPU with 128 MB VRAM	DirectX 9.0 compliant GPU with 256 MB VRAM and Hardware Acceleration

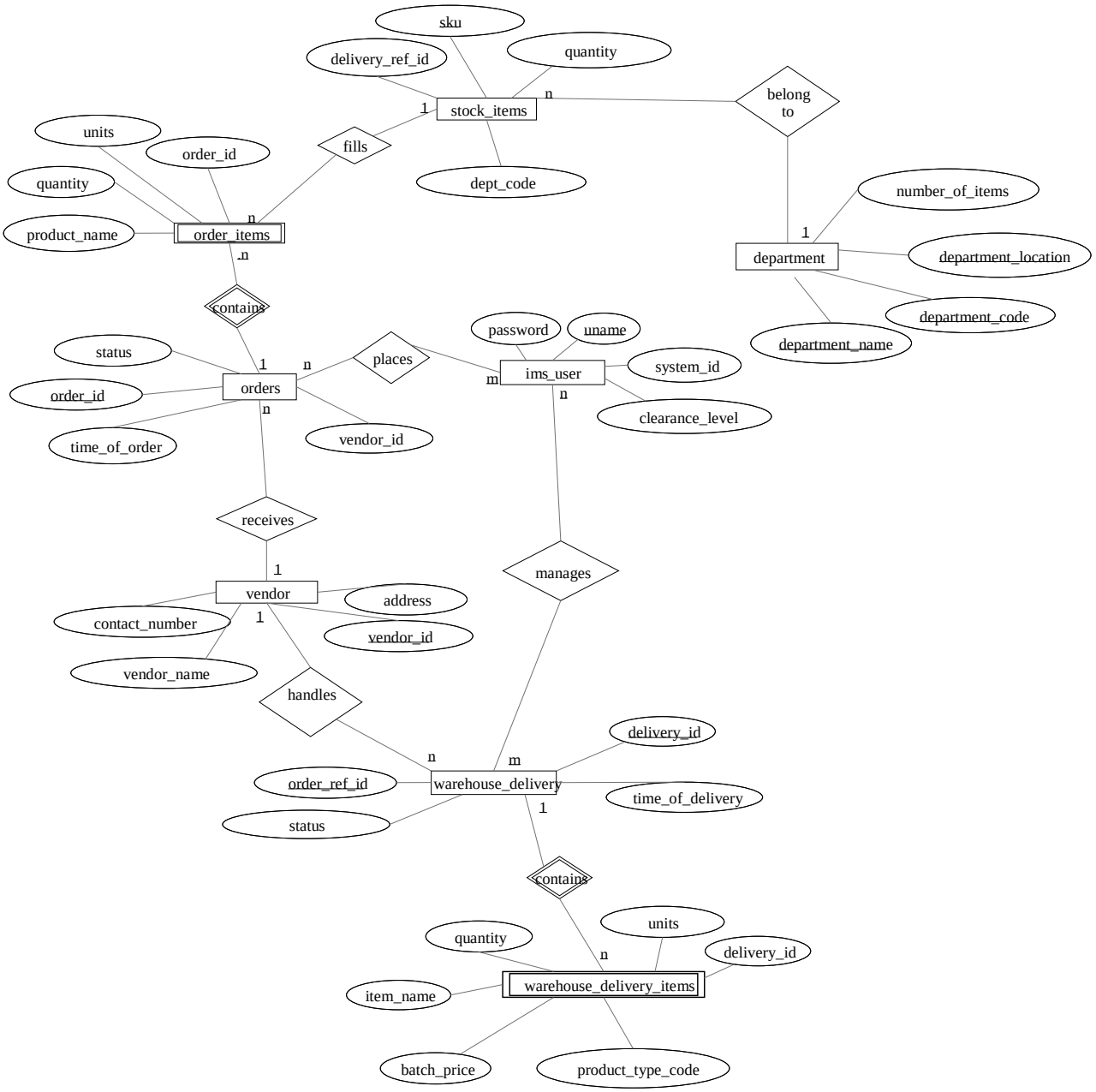
CHAPTER 3

SYSTEM DESIGN

CHAPTER 3

SYSTEM DESIGN

3.1 ER Diagram



3.2 Schema Diagram

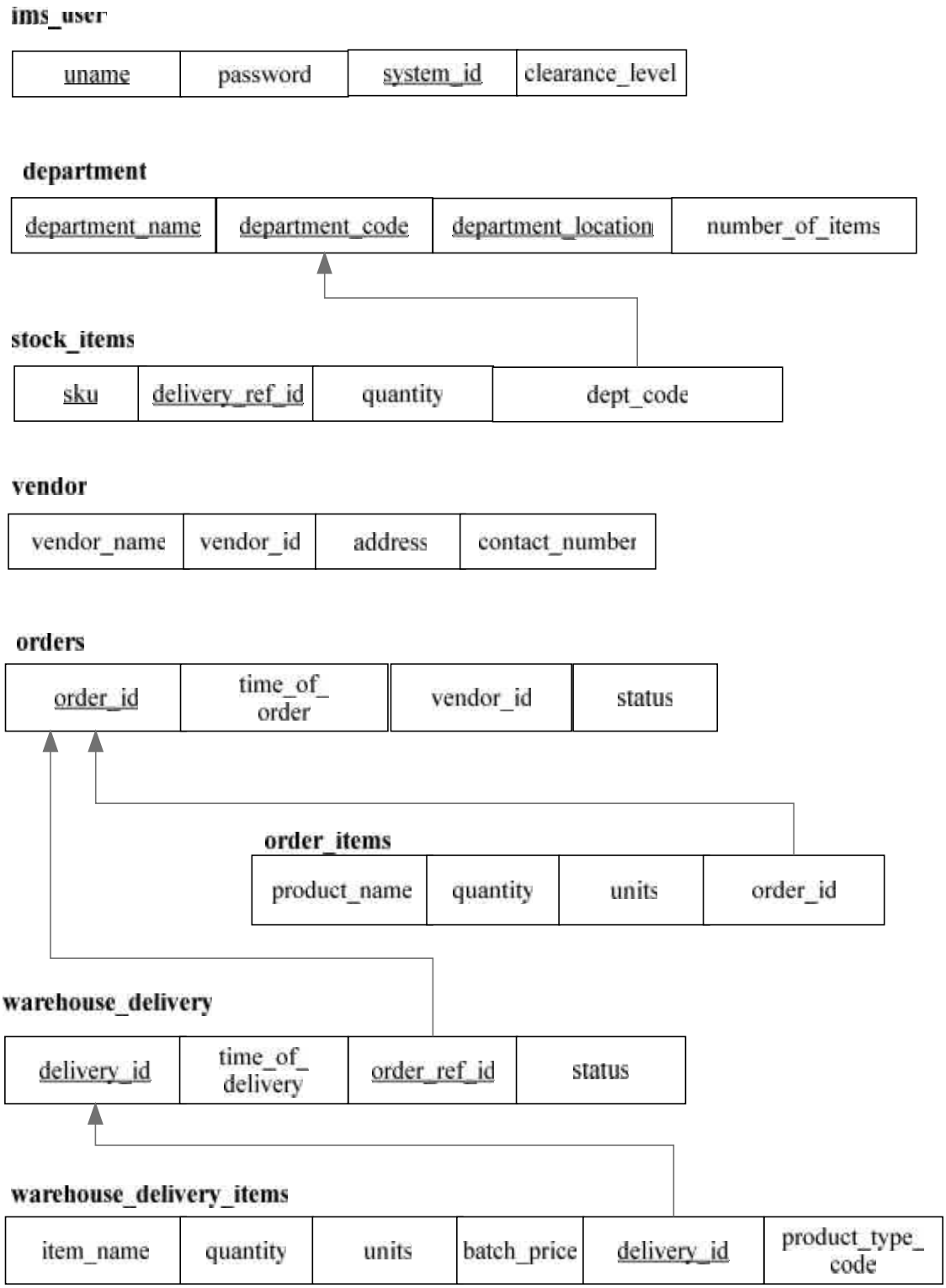


Figure 3.2: Schema Diagram

3.3 Overview of GUI

The GUI for this system has been developed using JavaFX. JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.

Salient features of the graphical interface for the application include: -

- Simple, clean and intuitive.
- Easily navigable.
- Separation of different functionalities into distinct modules, accessible via an intuitive menu.

Figure 3.3.1: Login Screen

Upon application launch, a simple login form validates user input against credentials stored in a database. Depending on input validity, the user is subsequently allowed access to the system, or an error message is flashed at the bottom.

Successfully logged in users are presented with the main menu panel (Figure 3.3.2). Here, the user has access to different modules. For instance, all users can access the "Manage Deliveries" module (Figure 3.3.3) to view delivery history, and create entries for new deliveries. The items associated with each delivery are automatically obtained from the corresponding order (Figure 3.3.4). The user can make changes to item details, as well as flag (and unflag) items to mark defective deliveries. Once necessary changes have been integrated, the delivery items can be moved to stock.

Figure 3.3.2: IMS Panel

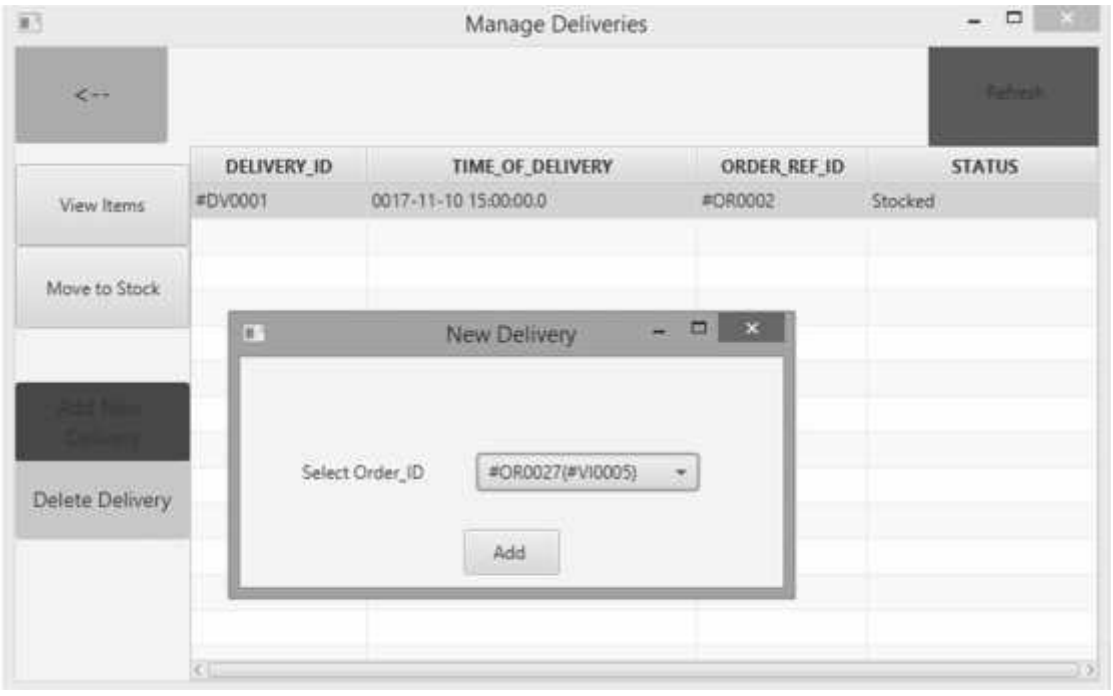


Figure 3.3.3: Delivery Page

Figure 3.3.4: Delivery Items Page

The "Manage Orders" module allows users to manage their orders as well as place new ones. Items can be added to or removed from orders that have not yet been delivered (Refer to Figure 3.3.5).

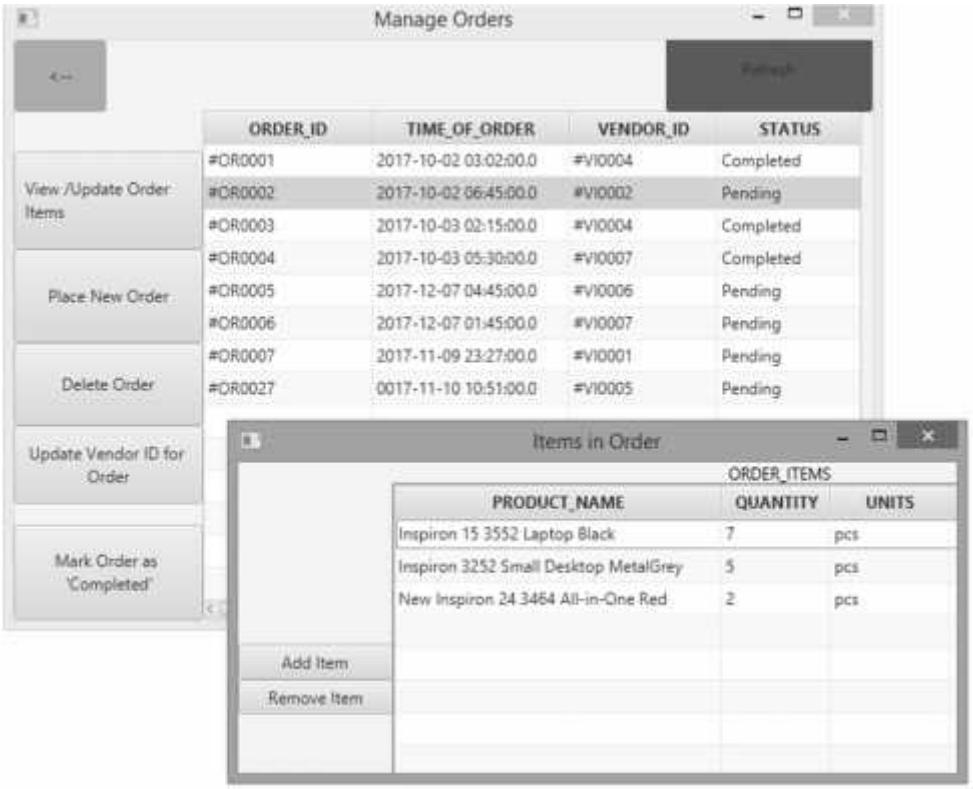


Figure 3.3.5: Orders Page and Order Items Page

The "View Vendors" module (Figure 3.3.6) simply displays the list of vendors and their associated details, while the "View Departments" module (Figure 3.3.7) simply displays a list of departments with their associated details

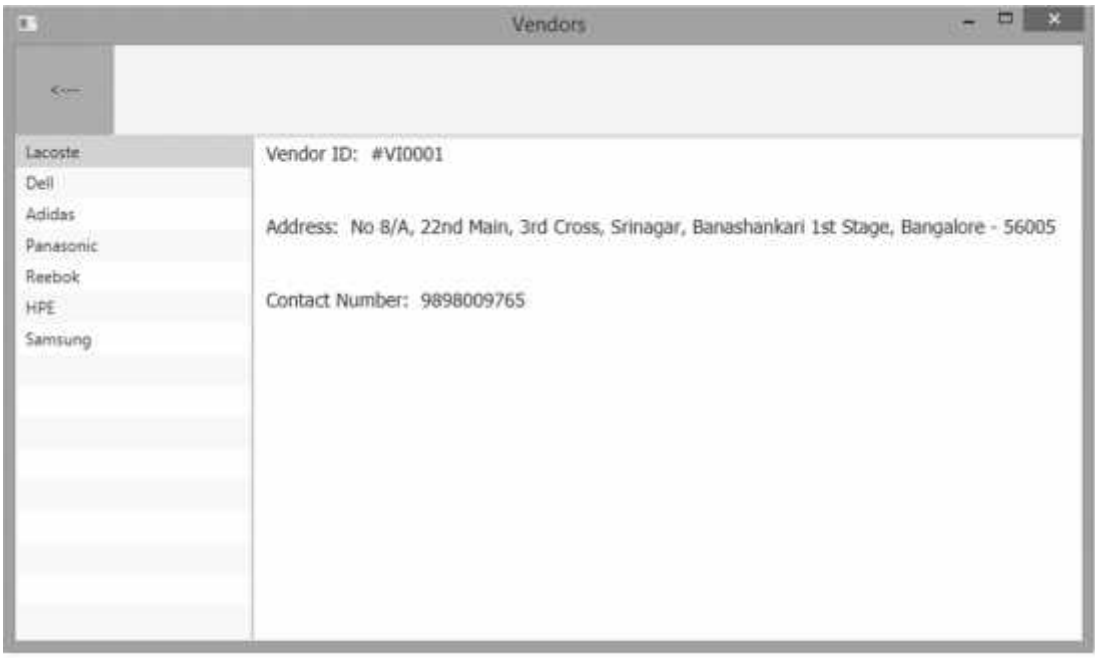


Figure 3.3.6: Vendors Page

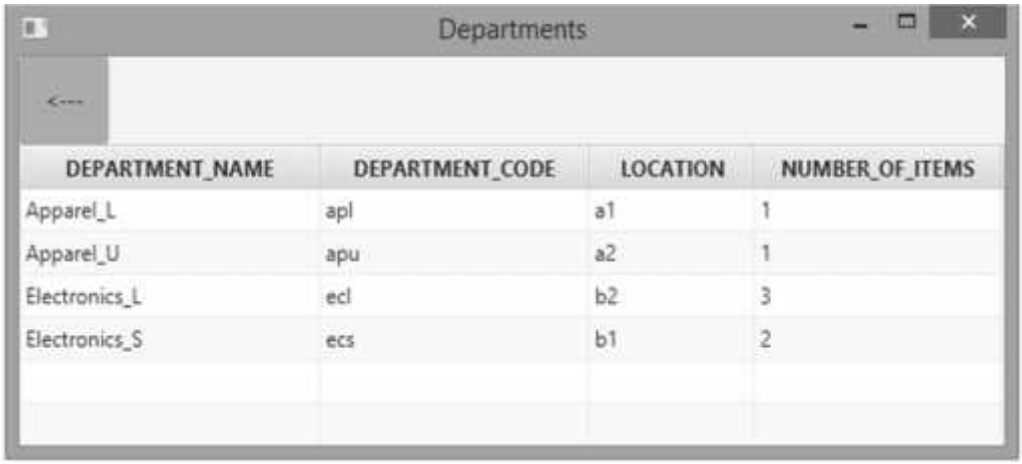


Figure 3.3.7: Departments Page

The "Manage Stocks" module displays items in stock (Figure 3.3.8). It also allows users to transfer a batch of items fully/partially from stock to store as per the need.

Figure 3.3.8: Stock Page

The "Query Box" present on the main panel is available only to users with administrator privileges, and allows them to execute a range of SQL queries (Figure 3.9).

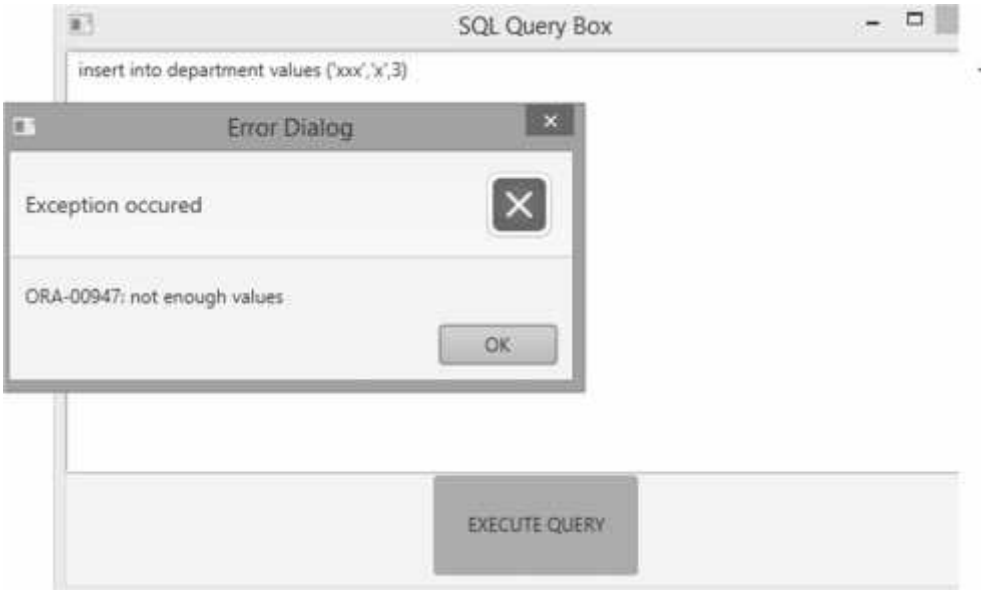


Figure 3.3.9: SQL Query Page

The text box in the center of the IMS panel displays the status level of all the batches in stock (Figure 3.10).

Figure 3.3.10: IMS main panel (with stock status)

3.4 Normalization

Normalization is the process of minimizing redundant data from a database to ensure more logical organization of data and to eliminate insertion, deletion and update anomalies. The most commonly used normal forms are First normal form(1NF), Second normal form(2NF), Third normal form(3NF) and Boyce & Codd normal form (BCNF).

First Normal Form (1NF): As per the rule of first normal form, an attribute (column) of a table should hold only atomic values. E.g.:-

PRODUCT_NAME	QUANTITY	UNITS	ORDER_ID
Lacoste Red Poloneck Shirt, Lacoste Blue Poloneck Shirt	7, 5	pcs	#OR0002
Inspiron 3252 Small Desktop MetalGrey	5	pcs	#OR0002
New Inspiron 24 3464 All-in-One Red	2	pcs	#OR0002
Panasonic rp-ht211 headphones Maroon	5	pcs	#OR0003

Figure 3.4.1: Sample Order_Items table

Consider the sample table above. Consider the 1st row. Since the 1st row has two entries under product names and two entries under quantity. its no longer atomic. Applying 1NF, we come up with: -

PRODUCT_NAME	QUANTITY	UNITS	ORDER_ID
Lacoste Red Poloneck Shirt		7 pcs	#OR0002
Lacoste Blue Poloneck Shirt		5 pcs	#OR0002
Inspiron 3252 Small Desktop MetalGrey		2 pcs	#OR0002
New Inspiron 24 3464 All-in-One Red		5 pcs	#OR0003

Figure 3.4.2: Order_Items table with 1NF applied

Second Normal Form (2NF): A table is said to be in 2NF if both the following conditions hold: -

- Table is in 1NF (First normal form)
- Non-key attributes (attribute that is not part of any candidate key) are fully functionally dependent on primary key, and not functionally dependent on a proper subset of the primary key.

E.g. Consider a table that is in 1NF but not in 2NF: -

ORDER_ID	VENDOR_ID	VENDOR_TOKEN	PRODUCT_NAME	TIME_OF_ORDER	STATUS
#OR0001	#VI0004	1	Dell Inspiron	02-OCT-17 03:02:00	Completed
#OR0001	#VI0002	2	HP Envy 15	02-OCT-17 06:45:00	Completed

Figure 3.4.3: Orders table

Here, the candidate keys are ORDER_ID and VENDOR_TOKEN. As seen, PRODUCT_NAME field is functionally dependent on ORDER_ID, but not on VENDOR_TOKEN. Hence, 2NF is violated, This can be fixed by having 2 tables as: -

ORDER_ID	VENDOR_ID	TIME_OF_ORDER	STATUS
#OR0001	#VI0004	02-OCT-17 03:02:00	Completed
#OR0001	#VI0002	02-OCT-17 06:45:00	Completed

Figure 3.4.4: Orders table

VENDOR_TOKEN	PRODUCT_NAME
1	Dell Inspiron
2	HP Envy 15

Figure 3.4.5: Order_Items table

Third Normal Form (3NF): A database is in third normal form if it satisfies the following conditions: -

- It is in second normal form
- There is no transitive functional dependency

E.g. Consider a table that is in 2NF but not in 3NF: -

	SKU	QUANTITY	DELIVERY_REF_ID	DEPT_NAME	DEPT_CODE
1	Panasonic/#OR0001/Mens Colorblock Cotton Pique Red/1...	10	#DV0001	Apparel_U	apu
2	Panasonic/#OR0001/Mens Slim Fit Polo Shirt Blue/15/'...	15	#DV0001	Apparel_U	apu
3	Dell/#OR0002/Inspiron 15 3552 Laptop Black/7/'14-Nov...	7	#DV0002	Electronics_L	ecl
4	Dell/#OR0002/Inspiron 3252 Small Desktop MetalGrey /...	5	#DV0002	Electronics_L	ecl
5	Samsung/#OR0004/Samsung Galaxy Note 8 (sm-910c) Whit...	45	#DV0006	Electronics_S	ecs
6	Samsung/#OR0004/Samsung Galaxy On5 Gold/10/'14-Nov-1...	10	#DV0006	Electronics_S	ecs

Figure 3.4.6: Stock_Items table

Here, SKU determines DEPT_NAME which in turn determines DEPT_CODE. This is an example of transitive functional dependency, and this violates 3NF. This can be fixed by having 2 separate tables.

	SKU	QUANTITY	DELIVERY_REF_ID	DEPT_NAME	DEPT_CODE
1	Panasonic/#OR0001/Mens Colorblock Cotton Pique Red/1...	10	#DV0001	Apparel_U	apu
2	Panasonic/#OR0001/Mens Slim Fit Polo Shirt Blue/15/'...	15	#DV0001	Apparel_U	apu
3	Dell/#OR0002/Inspiron 15 3552 Laptop Black/7/'14-Nov...	7	#DV0002	Electronics_L	ecl
4	Dell/#OR0002/Inspiron 3252 Small Desktop MetalGrey /...	5	#DV0002	Electronics_L	ecl
5	Samsung/#OR0004/Samsung Galaxy Note 8 (sm-910c) Whit...	45	#DV0006	Electronics_S	ecs
6	Samsung/#OR0004/Samsung Galaxy On5 Gold/10/'14-Nov-1...	10	#DV0006	Electronics_S	ecs

Figure 3.4.7: Stock_Items table

DEPARTMENT_NAME	DEPARTMENT_CODE
Apparel_L	apl
Apparel_U	apu
Electronics_S	ecs
Electronics_L	ecl

Figure 3.4.8: Department table

CHAPTER 4

IMPLEMENTATION

CHAPTER 4

IMPLEMENTATION

4.1 Table Creation

```
1) CREATE TABLE DEPARTMENT
( DEPARTMENT_NAME VARCHAR2(45) NOT NULL,
DEPARTMENT_CODE VARCHAR2(3) NOT NULL,
LOCATION VARCHAR2(45) NOT NULL,
NUMBER_OF_ITEMS NUMBER(3) DEFAULT 0,
PRIMARY KEY (DEPARTMENT_NAME, DEPARTMENT_CODE, LOCATION));

2) CREATE TABLE IMS_USER
  ( UNAME VARCHAR2(20) NOT NULL,
    PASSWORD VARCHAR2(15) NOT NULL,
    SYSTEM_ID VARCHAR2(20) NOT NULL,
    CLEARANCE_LEVEL NUMBER(1) NOT NULL,
    PRIMARY KEY(UNAME, SYSTEM_ID) );

 3) CREATE TABLE VENDOR
     ( VENDOR_NAME VARCHAR2(45) NOT NULL,
       VENDOR_ID VARCHAR2(45) NOT NULL,
       ADDRESS VARCHAR2(130) NOT NULL,
       CONTACT_NUMBER NUMBER(19) DEFAULT NULL,
       PRIMARY KEY(VENDOR_ID, VENDOR_NAME)  );

 4) CREATE TABLE ORDERS
    ( ORDER_ID VARCHAR2(45) NOT NULL,
      TIME_OF_ORDER DATE NOT NULL,
      VENDOR_ID VARCHAR2(45) NOT NULL,
```

STATUS VARCHAR2(10) NOT NULL,

PRIMARY KEY(ORDER_ID),

CONSTRAINT VENDOR_FK FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR (VENDOR_ID) ON DELETE SET NULL);

5) CREATE TABLE ORDER_ITEMS(

PRODUCT_NAME VARCHAR2(75) NOT NULL, QUANTITY NUMBER(10) DEFAULT 0,

UNITS VARCHAR2(6) NOT NULL,ORDER_ID VARCHAR2(45) NOT NULL,

CONSTRAINT ORDER_ID_FK FOREIGN KEY (ORDER_ID) REFERENCES ORDERS (ORDER_ID) ON DELETE CASCADE);

6) CREATE TABLE WAREHOUSE_DELIVERY(

DELIVERY_ID VARCHAR2(45) NOT NULL,

TIME_OF_DELIVERY DATE NOT NULL, ORDER_REF_ID VARCHAR2(45) NOT NULL,

STATUS VARCHAR2(20) NOT NULL,

PRIMARY KEY(DELIVERY_ID),

CONSTRAINT ORDER_REF_ID_FK FOREIGN KEY (ORDER_REF_ID) REFERENCES ORDERS(ORDER_ID) ON DELETE SET NULL);

7) CREATE TABLE WAREHOUSE_DELIVERY_ITEMS (

ITEM_NAME VARCHAR2(45) NOT NULL,

QUANTITY NUMBER(10, 0) NOT NULL,

UNITS VARCHAR2(6) NOT NULL,

BATCH_PRICE NUMBER(10, 2) NOT NULL,

DELIVERY_ID VARCHAR2(45) NOT NULL,

PRODUCT_TYPE_CODE VARCHAR2(5) NOT NULL,

PRIMARY KEY(ITEM_NAME),

CONSTRAINT DELIVERY_ID_FK FOREIGN KEY (DELIVERY_ID) REFERENCES WAREHOUSE_DELIVERY(DELIVERY_ID) ON DELETE SET NULL);

8) CREATE TABLE STOCK_ITEMS (

SKU VARCHAR2(120) NOT NULL,

DELIVERY_REF_ID VARCHAR2(45) NOT NULL,

QUANTITY NUMBER(10, 0) NOT NULL,

DEPT_CODE VARCHAR2(3) NOT NULL,

PRIMARY KEY(STOCK_ITEMS),

CONSTRAINT DELIVERY_ID_REF_FK FOREIGN KEY (DELIVERY_REF_ID) REFERENCES WAREHOUSE_DELIVERY(DELIVERY_ID) ON DELETE SET NULL,

CONSTRAINT DEPT_FK FOREIGN KEY (DEPT_CODE) REFERENCES DEPARTMENT(DEPARTMENT_CODE) ON DELETE SET NULL);

4.2 Description of Table

1) desc department;

```
Name            Null?    Type
--------------- -------- ------------
DEPARTMENT_NAME NOT NULL VARCHAR2(45)
DEPARTMENT_CODE NOT NULL VARCHAR2(3)
LOCATION        NOT NULL VARCHAR2(45)
NUMBER_OF_ITEMS          NUMBER(38)
```

Figure 4.2.1

2) desc ims_user;

```
Name            Null?    Type
--------------- -------- ------------
UNAME           NOT NULL VARCHAR2(20)
PASSWORD        NOT NULL VARCHAR2(15)
SYSTEM_ID       NOT NULL VARCHAR2(20)
CLEARANCE_LEVEL NOT NULL NUMBER(38)
```

Figure 4.2.2

3) desc orders;

```
Name           Null?    Type
-------------- -------- ------------
ORDER_ID       NOT NULL VARCHAR2(45)
TIME_OF_ORDER  NOT NULL DATE
VENDOR_ID      NOT NULL VARCHAR2(45)
STATUS         NOT NULL VARCHAR2(10)
```

Figure 4.2.3

4) desc order_items;

```
Name          Null?    Type
------------- -------- ------------
PRODUCT_NAME  NOT NULL VARCHAR2(75)
QUANTITY      NOT NULL NUMBER(10)
UNITS                  VARCHAR2(6)
ORDER_ID      NOT NULL VARCHAR2(45)
```

Figure 4.2.4

5)desc warehouse_delivery;

```
Name              Null?    Type
----------------- -------- ------------
DELIVERY_ID       NOT NULL VARCHAR2(45)
TIME_OF_DELIVERY  NOT NULL DATE
ORDER_REF_ID      NOT NULL VARCHAR2(45)
STATUS            NOT NULL VARCHAR2(20)
```

Figure 4.2.5

6) desc warehouse_delivery_items;

```
Name               Null?    Type
------------------ -------- ------------
ITEM_NAME          NOT NULL VARCHAR2(45)
QUANTITY           NOT NULL NUMBER(10)
UNITS              NOT NULL VARCHAR2(6)
BATCH_PRICE        NOT NULL NUMBER(10,2)
DELIVERY_ID        NOT NULL VARCHAR2(45)
PRODUCT_TYPE_CODE  NOT NULL VARCHAR2(5)
```

Figure 4.2.6

7) desc vendor;

```
Name           Null?    Type
-------------- -------- -------------
VENDOR_NAME    NOT NULL VARCHAR2(45)
VENDOR_ID      NOT NULL VARCHAR2(45)
ADDRESS        NOT NULL VARCHAR2(130)
CONTACT_NUMBER          NUMBER(19)
```

Figure 4.2.7

8) desc stock_items;

```
Name            Null?    Type
--------------- -------- -------------
SKU             NOT NULL VARCHAR2(120)
DELIVERY_REF_ID NOT NULL VARCHAR2(45)
QUANTITY        NOT NULL NUMBER(10)
DEPT_CODE       NOT NULL VARCHAR2(3)
```

Figure 4.2.8

4.3 Populated Tables

1) DEPARTMENT

	DEPARTMENT_NAME	DEPARTMENT_CODE	LOCATION	NUMBER_OF_ITEMS
1	Apparel_L	apl	a1	1
2	Apparel_U	apu	a2	1
3	Electronics_S	ecs	b1	2
4	Electronics_L	ecl	b2	3

Figure 4.3.1

2) IMS_USER

UNAME	PASSWORD	SYSTEM_ID	CLEARANCE_LEVEL
admin	admin	#SYS034	1
root	root	#SYS010	2
sys	sys	#SYS005	2

Figure 4.3.2

3) ORDER_ITEMS

PRODUCT_NAME	QUANTITY	UNITS	ORDER_ID
New Inspiron 24 3464 All-in-One Red	2	pcs	#OR0002
Pansonic bt-50gcs earbuds Blue	3	pcs	#OR0003
Panasonic rp-ht211 headphones Maroon	5	pcs	#OR0003
Inspiron 3252 Small Desktop MetalGrey	5	pcs	#OR0002
HP 240 G6 Notebook Black	5	pcs	#OR0005
HP 24es 60.45 cm(23.8) Display MetalGrey	5	pcs	#OR0005
Inspiron 15 3552 Laptop Black	7	pcs	#OR0002
Mens Colorblock Cotton Pique Red	10	pcs	#OR0001
Samsung Galaxy On5 Gold	10	pcs	#OR0004
Mens Slim Fit Polo Shirt Blue	15	pcs	#OR0001
Unisex Adidas Running Ankle Socks Blue	20	pairs	#OR0006
Unisex Adidas Blackcat Cap Black	25	pcs	#OR0006
Mens Adidas Adi Classic Backpack Black	25	pcs	#OR0006

Figure 4.3.3

4) ORDERS

ORDER_ID	VENDOR_ID	TIME_OF_ORDER	STATUS
#OR0001	#VI0004	02-OCT-17 03:02:00	Completed
#OR0002	#VI0002	02-OCT-17 06:45:00	Completed
#OR0003	#VI0004	03-OCT-17 02:15:00	Pending
#OR0004	#VI0007	03-OCT-17 05:30:00	Completed
#OR0005	#VI0006	07-DEC-17 04:45:00	Pending
#OR0006	#VI0007	07-DEC-17 01:45:00	Pending

Figure 4.3.4

5) STOCK_ITEMS

SKU	QUANTITY	DELIVERY_REF_ID	DEPT_CODE
Panasonic/#OR0001/Mens Colorblock Cotton Pique Red/1...	10	#DV0001	apu
Panasonic/#OR0001/Mens Slim Fit Polo Shirt Blue/15/'...	15	#DV0001	apu
Dell/#OR0002/Inspiron 15 3552 Laptop Black/7/'14-Nov...	7	#DV0002	ecl
Dell/#OR0002/Inspiron 3252 Small Desktop MetalGrey /...	5	#DV0002	ecl
Samsung/#OR0004/Samsung Galaxy Note 8 (sm-910c) Whit...	45	#DV0006	ecs
Samsung/#OR0004/Samsung Galaxy On5 Gold/10/'14-Nov-1...	10	#DV0006	ecs

Figure 4.3.5

6) VENDOR

VENDOR_NAME	VENDOR_ID	ADDRESS	CONTACT_NUMBER
Lacoste	#VI0001	No 8/A, 22nd Main, 3rd Cross, Srinagar, Banashankari ...	9898009765
Dell	#VI0002	No.15, Ground Floor, Wood Street, Richmond Road, Bang...	8557546712
Adidas	#VI0003	No 10/A, Chandra Kiran Building, Kasturba Road, Banga...	8444319661
Panasonic	#VI0004	No. 124, Silver Jubilee Park Road, S.P. Road, Bangalo...	9731611987
Reebok	#VI0005	690, Next To-Sony World, 80 Feet Rd, Koramangala 4th ...	8786879000
HPE	#VI0006	2 FL, No.24, Salarpuria Arena Building, Hosur Main Ro...	8898112332
Samsung	#VI0007	# 1000 Sri vari Plaza, 80 Feet Road, 7th main Jakasan...	8861768798

Figure 4.3.6

7) WAREHOUSE_DELIVERY

DELIVERY_ID	TIME_OF_DELIVERY	ORDER_REF_ID	STATUS
#DV0002	14-NOV-17 19:44:00	#OR0002	Stocked
#DV0003	14-NOV-17 19:44:00	#OR0003	Arrived
#DV0004	14-NOV-17 19:44:00	#OR0005	Arrived
#DV0005	14-NOV-17 19:44:00	#OR0006	Arrived
#DV0001	14-NOV-17 19:44:00	#OR0001	Stocked
#DV0006	14-NOV-17 19:44:00	#OR0004	Stocked

Figure 4.3.7

8) WAREHOUSE_DELIVERY_ITEMS

ITEM_NAME	QUANTITY	UNITS	BATCH_PRICE	DELIVERY...	PRODUCT_TYPE_CODE
Mens Slim Fit Polo Shirt Blue	15	pcs	9500	#DV0001	apu
Inspiron 15 3552 Laptop Black	7	pcs	525000	#DV0002	ecl
Inspiron 3252 Small Desktop ...	5	pcs	600000	#DV0002	ecl
! New Inspiron 24 3464 All-i...	2	pcs	25000	#DV0002	ecl
Panasonic rp-ht211 headphone...	5	pcs	0	#DV0003	x
Pansonic bt-50gcs earbuds Blue	3	pcs	0	#DV0003	x
HP 240 G6 Notebook Black	5	pcs	0	#DV0004	x
Mens Colorblock Cotton Pique...	10	pcs	1800	#DV0001	apu
HP 24es 60.45 cm(23.8) Displ...	5	pcs	0	#DV0004	x
Mens Adidas Adi Classic Back...	25	pcs	0	#DV0005	x
Unisex Adidas Blackcat Cap B...	25	pcs	0	#DV0005	x
Unisex Adidas Running Ankle ...	20	pairs	0	#DV0005	x
Samsung Galaxy Note 8 (sm-91...	45	pcs	3600000	#DV0006	ecs
Samsung Galaxy On5 Gold	10	pcs	100000	#DV0006	ecs

Figure 4.3.8

4.4 SQL Triggers and Stored Procedures

1) Trigger to generate new delivery id: -

```
create or replace TRIGGER "DPZ"."GENERATENEWDELIVERYID"
before insert on warehouse_delivery
for each row
begin
 SELECT '#DV' || TO_CHAR(DELIVERYSEQ.NEXTVAL, 'FM0000') INTO :NEW.DELIVERY_ID FROM DUAL;
end;
```

2) Trigger to generate new order id: -

```
create or replace TRIGGER "DPZ"."GENERATENEWORDERID"
before insert on orders
for each row

begin
   SELECT '#OR' || TO_CHAR(ORDERSEQ.NEXTVAL, 'FM0000') INTO :NEW.ORDER_ID FROM DUAL;
end;
```

3) Trigger to update item quantity in case of duplicate product name entry: -

```
create or replace TRIGGER "DPZ"."UPDATE_QTY"
before insert on order_items
for each row
DECLARE
  oldQty integer:=0;
  product_name varchar2(45);
  pragma autonomous_transaction;
  CURSOR itemRows is
```

```
   select trim(upper(product_name)),quantity from order_items where
trim(upper(product_name)) = trim(upper(:NEW.product_name))
   and order_id = :NEW.order_id;
begin
 open itemRows;
   fetch itemRows into product_name, oldQty;
   if itemRows%rowcount = 1 then
      update order_items set quantity = oldQty + :NEW.quantity where
trim(upper(PRODUCT_NAME)) = trim(upper(:NEW.product_name));
      commit;
      Raise_Application_Error (-20100, 'Item with same name already exists. Quantity of
original item will be updated...');
   end if;
end;
```

4) Stored Procedure to update number of items for each department: -

```
create or replace PROCEDURE "NUMOFITEMS" as
num number;
dept_code varchar2(20);
CURSOR counts IS
   select count(*), dept_code from stock_items
   group by dept_code;
BEGIN
 open counts;
 loop
   fetch counts into num, dept_code;
   exit when counts%notfound;
   if counts%rowcount = 0 then
   update department set NUMBER_OF_ITEMS = 0;
```

```
    exit;
    end if;
    DBMS_OUTPUT.PUT_LINE(dept_code||' '||num);
    update department set NUMBER_OF_ITEMS = num where DEPARTMENT_CODE =
dept_code;
end loop;
end numofitems;
```

4.5 Database Connectivity

To establish a connection with our database, we make user of a JDBC (Java DataBase Connectivity) Driver. JDBC is an API for the Java Programming language via which we can execute SQL queries and call stored procedures from within the JAVA language itself. We make use of the following code segment to connect to our database: -

```
try
{
   Class.forName("oracle.jdbc.OracleDriver");
   Connection db=DriverManager.getConnection(url,uname,pass);
   System.out.println("Connection to db successful.");
}
catch(Exception e)
{
   e.printStackTrace();
 }
```

Here, the ‘url’, ‘uname’, and ‘pass’ variables store our database url, username and password respectively.

We can then use Statement or PreparedStatement objects to execute SQL statements. Termination of database connection is achieved with

```
   db.close();
```

where ‘db’ is our Connection Object.

CHAPTER 5

RESULTS

CHAPTER 5

RESULTS

The front-end offers a number of different functions, each of which yields a different result. Consider some examples: -

1) Adding a new order: -

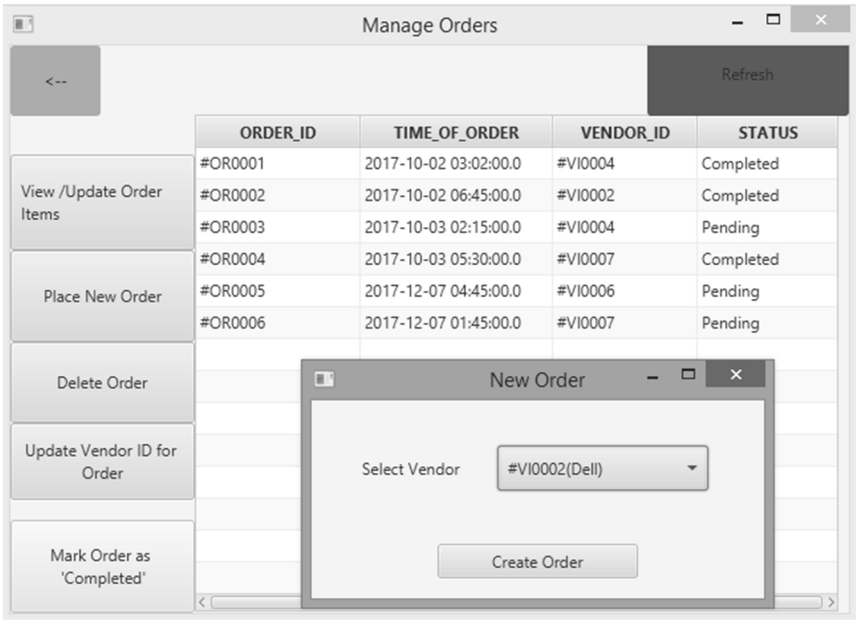


Figure 5.1: Before Adding Order

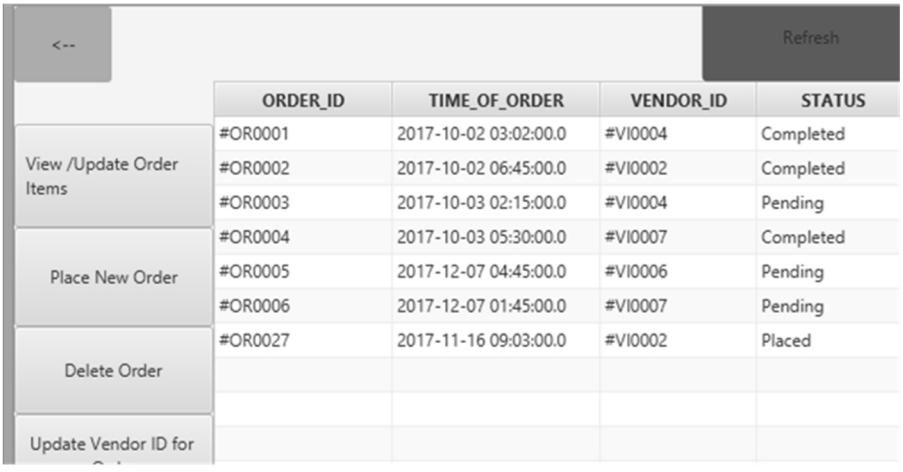


Figure 5.2: After Adding Order

2) Inserting Item into Order:-

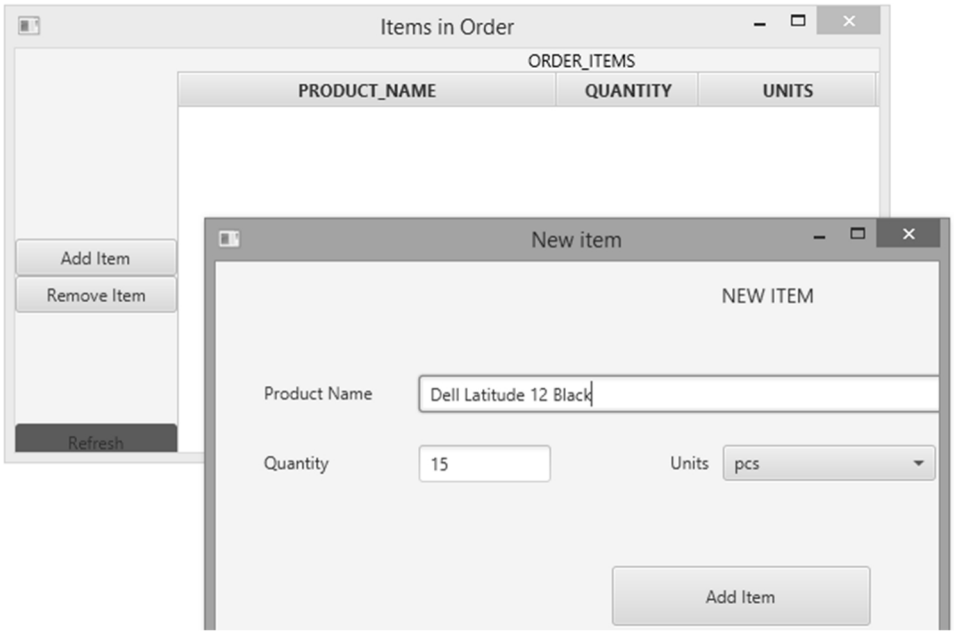


Figure 5.3: Before Adding Item to Order

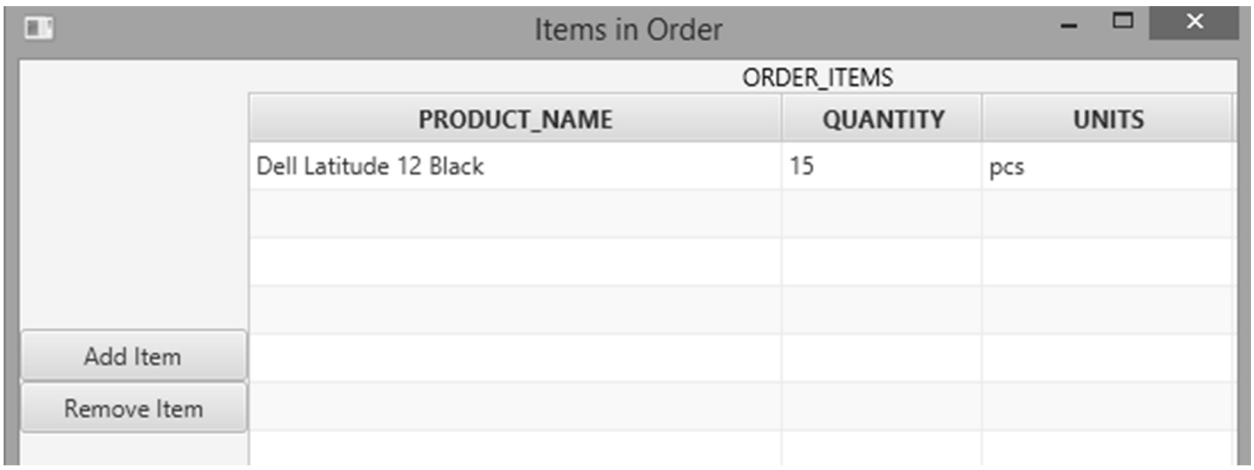


Figure 5.4: After Adding Item to Order

3) Creating new delivery entry for order that's just been delivered: -

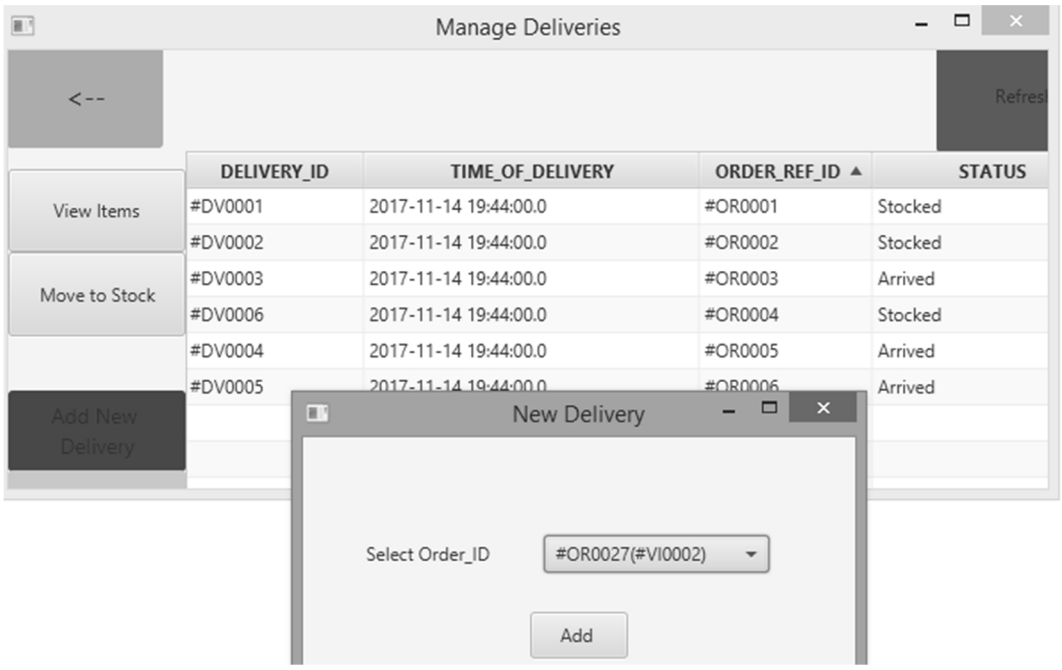


Figure 5.5: Before Creating Entry for New Delivery

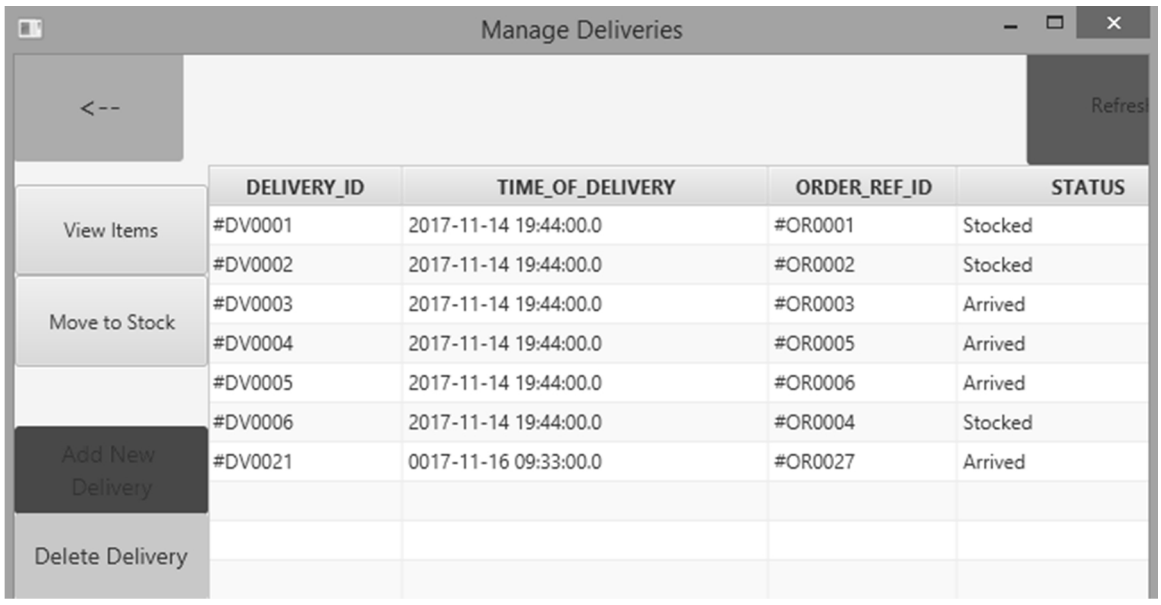


Figure 5.6: After Creating Entry for new Delivery

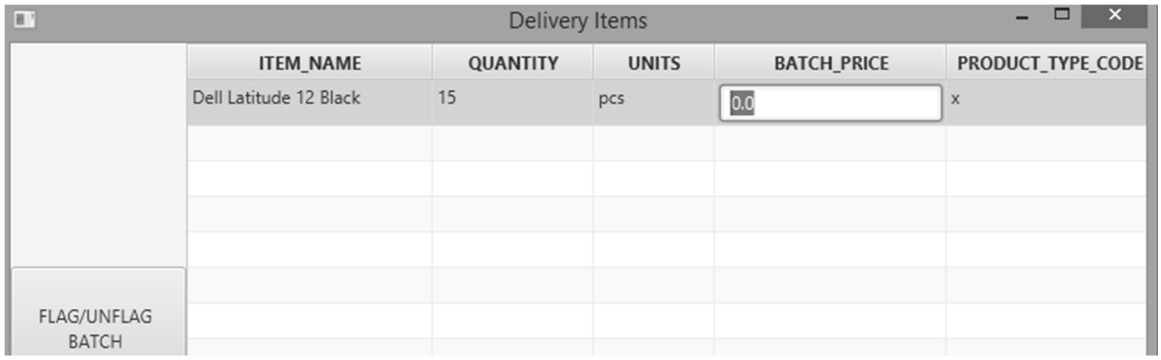


Figure 5.7: Items that have been imported from Order_Items for new delivery entry #DV0021

4) Stock_Items table after moving deliveries "#DV0001" and "#DV0002" to stock: -

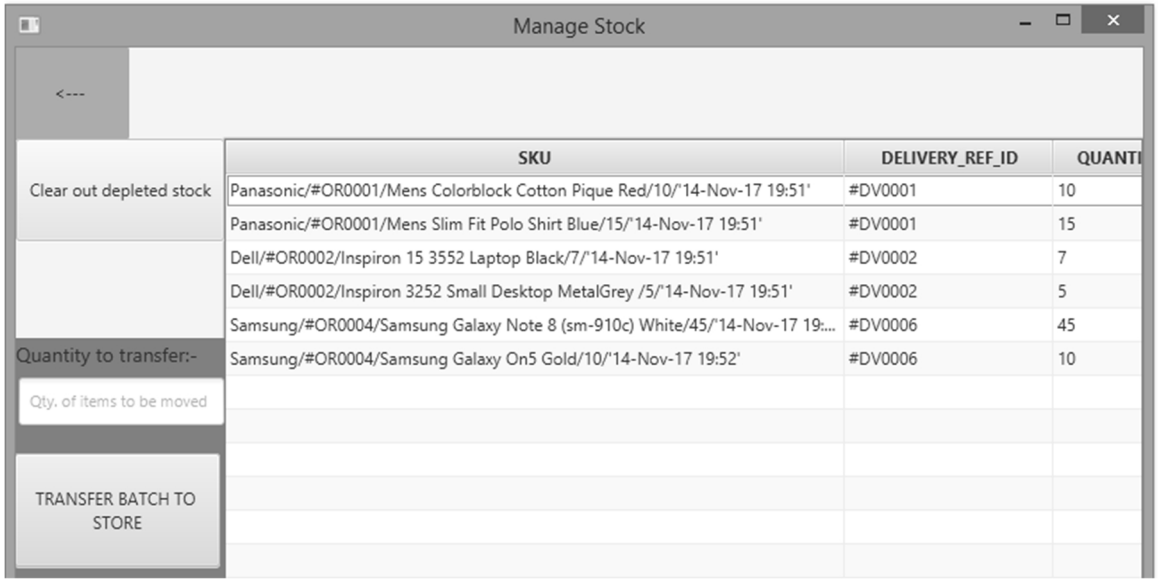


Figure 5.8: Stock_Items table

5) Stock_Items table after moving some batch items to store and clearing out depleted stock: -

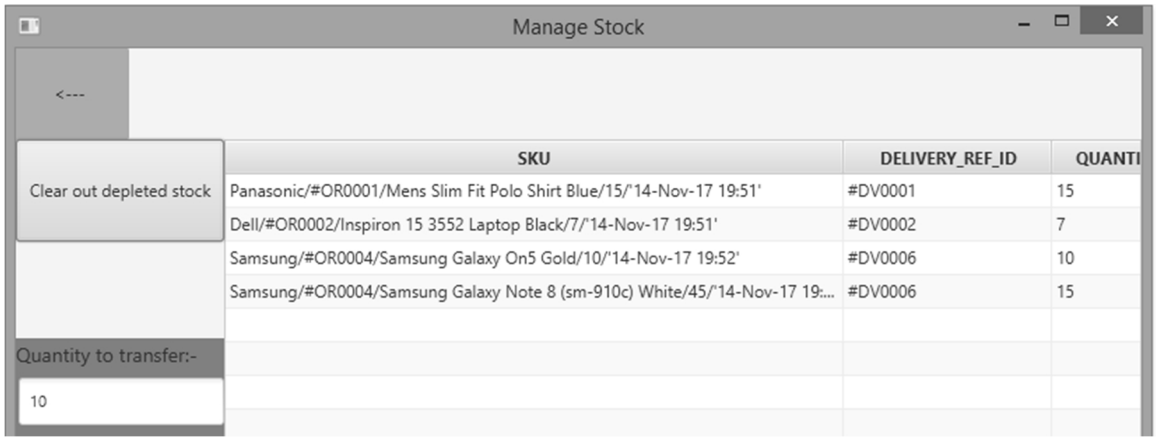


Figure 5.9: Stock_Items table

6) Executing query via query box: -

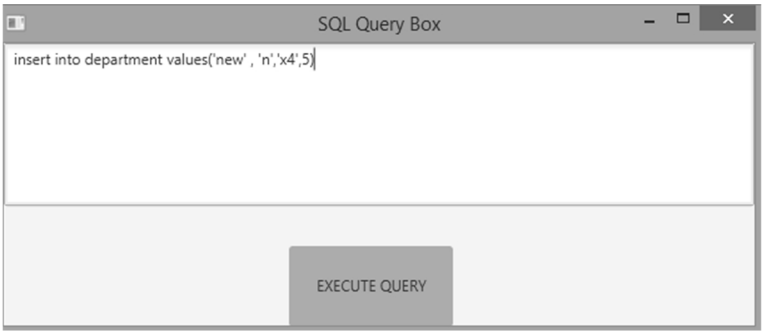


Figure 5.10: Query Box

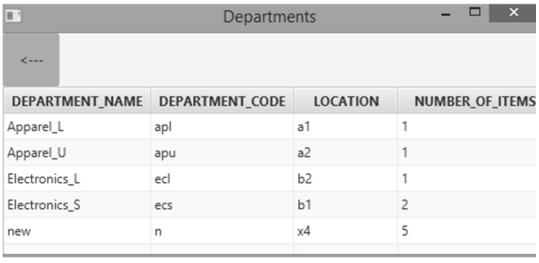


Figure 5.10: Resulting Departments Table

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENTS

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

Our simple inventory management system for our fictional warehouse is capable of handling orders and deliveries. Once a delivery is complete, user can fill in and modify details for item names, quantities, batch prices and product type codes, as well mark defective/unexpected item batches. The delivery items can then be moved to stock, where SKUs for each batch are automatically generated. Also, users can initiate transfer of items from stock to store – something required in a scenario where a store runs out of a batch of items.

This is more or less how inventory management works in the real word. Of course, the process is larger and more sophisticated with several other factors coming into play – many more products, vendors/suppliers and transactions are involved. However, considering the fact that it is a college mini-project, our application does a good job of demonstrating how inventory management works.

Nevertheless, there is still room for enhancements. Some important features that could later be included to bring it closer to the level of a real world commercial application are: -

1. Using the charts module in JavaFX to display charts (line chart, bar chart, pie chart, etc.) that demonstrate trends in orders and deliveries. Such data is actually an integral part of sophisticated inventory management systems, where they are used to study changes in market conditions and customer behavior.
2. Adding capability to manage defective product batches. Items may be damaged during delivery, or in transit from warehouse to store. Functionality to record such situation is helpful.
3. Allowing the users to generate reports and summaries in excel or pdf format.
4. Allowing user to export related data to spreadsheet applications like Microsoft Excel.
5. Adding functionality to take offline backup of data, as well as allowing users to backup data to the cloud.
6. Adding functionality to log changes (with timestamp).
7. Adding simple functionality to: -
    - add, update or delete vendors from front end.
    - add, update or delete departments from front end.

    Commercial systems offer such functionality that allows users to perform a great deal from the front end.