**STOCK AND INVOICE MANAGMENT**

A Project Report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology In Computer Science Engineering

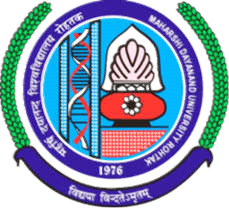
(Session: 2018-2021)

**SUPERVISOR : Submitted By :**

Dr. Sukhvinder Singh Deora Pooja Rani

Assistant Professor Roll No.: 18187

8th Semester



**UNIVERSITY INSTITUTE OF ENGINEERING & TECHNOLOGY, MDU, ROHTAK-124001**

**Declaration**

I **Pooja Rani** student of B Tech(8th semester). Hereby declare that the work detailed in this Project entitled **“MAHARSHI DAYANAND UNIVERSITY”** submitted to the Department UIET, Maharshi Dayanand University, Rohtak for the award of the **Bachelor of Technology in computer science** is our original work. We have neither plagiarized nor submitted this work for the award of any other degree. Period from 1st March 2021 to 10th May 2021 under the supervision of Mr. Yogoesh

Signature of Student \_\_\_\_\_\_\_\_\_\_\_\_\_

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge .

Internal Supervisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

External Supervisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of HOD \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### ACKNOWLEDGEMENT

With a deep sense of gratitude, we wish to express my sincere thanks to my guides, **Prof. (Dr.) Rajender Singh Chiller** (HOD) **and Dr.Sukhvinder Singh Deora**, Professor, C.S.A Department for giving us the opportunity to work under them on the project. We truly appreciate and value their esteemed guidance and encouragement from the beginning to end of this project. We are extremely grateful to him. We want to thank to all my teachers for providing a solid background for our studies and research thereafter. They have been great source of inspiration to us and we thank them from the bottom of my heart. We also want to thank our parents, who taught us the value of hard work by their own example. We would like to share this moment of happiness with our parents. They rendered us enormous support during the whole tenure of our stay in Integral University, Lucknow. Finally; we would like to thank our department for giving us the opportunity and platform to make our effort a successful one.

Signature :

Student : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Supervisor : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

External Supervisor : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Pooja Rani**

# TABLE OF CONTENTS

1. **Project Description 6**
2. **Requirement Analysis 7-13**

2.1 Introduction

2.2 Overall Description

1. **Problem Identification & Feasibility Study 14-16**
2. **Platform(Technology & Tools) 17-30**
3. **Proposed Work 31**
4. **Hardware & Software Specification 32**
5. **System Design 33-36**

7.1 Data Flow Diagrams

7.2 Entity Relationship Diagrams

7.3 Flowchart

1. **Snapshot of the Project 37-42**
2. **Testing 43-48**

**10. Maintenance 49**

**11. Implementation 50**

**12. Tables in Database 51-56**

1. **Coding Details 57-61**

1. **Future Scope 62**
2. **Conclusion 63**
3. **References 64**

**Project Description**

This project is easy user interface. The system utilizes the powerful database management, data retrieval and data manipulation. This project provides more ease for managing the data than manually maintaining in the documents. The project is useful for saving valuable time and reduces the huge paper work.

Stock and Invoice Management is a software where Stock details are been setup by an Employee and Invoice Details set up by accountant. This system is a hierarchical structure. It contains certain numbers of actors. Each will have their respective actions. A login id password Provide by company. For each of the actor to perform their action, they must be logged in with their respective ids. It is a window application managing invoice generated content. Earlier versions were to store the details of Stock. Here updating, retrieving actions are also been performed.

The proposed system is very user friendly, providing 2 actors. Account for generate invoices, employee for adding and updating Stocks.

# REQUIREMENT ANALYSIS

### **Introduction**

The following subsections are an overview of the entire Software Requirements Specification (SRS).

#### **Purpose**

This document provides the technical description of all software requirements of STOCK AND INVOICE MANAGMENT.

The document will not only define the product functions, user characteristics, constraints, and specific requirements of the system, but also serve as a basis for the Software Design Document that is prepared according to IEEE Std.

The objective of the software is to maintain information pertaining to the students with the purpose of :-

* Planned approach towards working
* Accuracy
* Reliability
* No Redundancy
* Immediate retrieval of information
* Immediate storage of information
* Easy to Operate

#### **Scope**

“STOCK AND INVOICE MANAGMENT” is a project with a mission of viewing and manipulating Stock information of any shop, industry and Organization in a Window-based environment. Thus, the overall system will consist of a Service Database System and Window Interface.

The Services Database System will supply the fundamental database structure of the entire system whereas Window Interface will provide a secure Window interface between the users and the database.

The Software aims to create a “paperless office” rather than using a traditional record keeping system.

Although this project is presently being designed specifically for Integral industry but there exist the possibility in future to upgrade it to general level.

The software will not only help the following levels of user in viewing the information but also each user can alongside update changes within their respective access limits.

* + Account Level
  + Stock Level

#### **Definition**

**Paperless Office:** refers to an integrated working environment where all the data and documentation is represented in electronic format.

Customers Personal Information: refers to personal records of individual services data along with his problems throughout the Product.

Traditional Record Keeping System: refers to a manual system where all records are kept on papers by manual in-charge.

#### Overview

This document is prepared in accordance with the IEEE Std, IEEE Recommended Practice for Software Requirements Specifications.

It also provides product perspectives, product functions, user characteristics, general constraints, and assumptions and dependencies of the system.

It will contain functional and performance requirements, design constraints, attributes and external interface requirements for the Software.

### **2.2 Overall Description**

This section describes the general factors that affect software and its requirements. In order to be easily understandable, this part of SS provides a background for the requirements.

#### **Product Perspective**

This software is a totally self-contained system. Also it is not dependent of any larger system.

#### **System Interfaces**

Since this Stock and Invoice Managements a standalone system, there is no system interface with any other system.

#### **User Interfaces**

The interfaces will involve check boxes, combo boxes, text boxes, and radio buttons. The combo boxes and the radio buttons will be used to prevent users from entering wrong type of information. They will also enable fast data entry. Text boxes will be controlled for avoiding invalid and inconsistent data.

Users can use “Tab” key to move cursor on screen items easily.

There will be two types of messages for constructive advice to the users: error and confirmation messages. There will be four types of error messages for application control: input, output, process and database.

#### **Communication Interfaces**

The default communication protocol for data transmission between server and the client is Transmission Control Protocol/ Internet Protocol (TCP/IP).

#### **Memory Constraints**

The client computer, which runs the web browser, should have enough physical memory to run this program.

#### **Product Functions**

* Customer Services
* Work order Details
* Item Master
* Job Work Details
* Generate Invoice
* Search Invoice Data
* Older Details Table

#### **User Characteristics**

Administrator – The administrator will hold full access to view as well as manipulate anywhere in the software and the information.

Employee- The other user like staff or faculty can view only their individual report and analyze their own performance based on that.

#### **Assumptions**

Every user will be having the appropriate hardware and software configuration as per the necessary requirement.

#### **Software Product Features**

The software is meant to generate a Invoices which will provide less paper work. Every invoice have a Automatic Generated Id and every Invoice No have Primary Key in database So the Employee cannot make 2 bills of the same number in it even by mistake.

#### **Performance Requirements**

The performance of the software will be as smooth as possible with special consideration on the following parameters-

* Planned approach towards working
* Accuracy
* Reliability
* No Redundancy
* Immediate retrieval of information
* Immediate storage of information
* Easy to Operate

#### **Design Constraints**

* GUI is only in English.
* Login and password is used for identification of user and there is no facility for guest.

#### Software System Attributes

**Reliability**

The system has to operate in a reliable manner with no scope for any flaws. This is to ensure efficient working and processing of information.

#### **Availability**

The site should be available all the time without any issues. A backup must be available for recovery issues so that the existing is not lost in case of any issue.

#### **Security**

The system has an authorization mechanism for users to identify their personal profiles. Therefore, different users will have different authorization levels to access the data. Data integrity for critical variables will also be checked.

#### **Maintainability**

The system can meet the changing requirements easily, since the infrastructure of the system would not need major changes. The requirements of the software while evolving, will be met by just adding new sub-functions. Therefore, the maintainability of the system would not be a complex issue.

#### **Portability**

All of the code which will be deployed at the window will be written in JAVA 8 and using SQL Server 2008 for database storage.

# PROBLEM IDENTIFICATION & FEASIBILITY STUDY

Feasibility Study begins when a problem is identified by managers and users of department. In this phase, the systems analyst visits the relevant department and starts preliminary investigations.

#### Objectives of Feasibility Study:

The main objectives of feasibility study are:

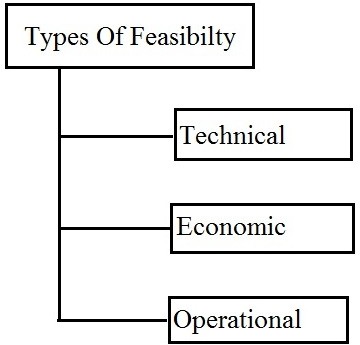
* To identify the deficiencies in the current system.
* To determine objectives of the proposed system.
* To acquire a sense of scope of the system.
* To identify the responsible users.
* To determine whether it is feasible to develop the system.

#### Steps in Feasibility Study:

Feasibility study is carried out in the following steps:

* Form a project team and appoint a project leader (Systems Analyst).
* Start preliminary investigation through different fact finding techniques.
* Prepare the systems flowcharts of the current system.
* Identify and describe the deficiencies in the current system.
* Determine objectives of the proposed system.
* Prepare the systems flowchart of the proposed system.
* Identify and enumerate the existing computer systems along with their technical specifications.
* Determine the cost and benefits of the proposed system.
* Identify the responsible users and determine the operational feasibility.
* Prepare the feasibility study report.
* Give the oral presentation of the feasibility study.

# Types of Feasibility



1. **Technical Feasibility:** During this study, the analyst identifies the existing computer systems of the concerned department and determines whether these technical resources are sufficient for the proposed system or not. If they are not sufficient, the analyst suggests the configuration of the computer systems that are required. The analyst generally pursues two or three different configurations which satisfy the key technical requirements but which represent different costs. During technical feasibility study, financial resources and budget is also considered. The main objective of technical feasibility is to determine whether the project is technically feasible or not, provided it is economically feasible.
2. **Economic Feasibility:** Economic Feasibility the most important study that determines the cost and benefits of the proposed system and compares with the budget. The cost of the project should not outweigh the budget. The cost of the project includes the cost of hardware, software, development and implementation. Cost/benefit analysis is the common method to determine the benefits that are expected from the proposed system and compare them with the costs expected to spend on development of the system.
3. **Operational Feasibility:** When it is found that the project is both economic and technical feasible, the next step is to determine whether it is operationally feasible or not. During operational feasibility study, it is operationally feasible or not. During operational feasibility study, it is determined whether the system will operate in the way that user wants or not. Operational feasibility depends upon human resources for the development and implementation of the system. It is considered whether the qualified or experienced manpower is available for development and implementation of the system or not. User involvement is more required in determining the operational feasibility.

# PLATFORM (TECHNOLOGIES/TOOLS)

**The JAVA Netbeans IDE and the J V M**

The.Netbeans IDE is an integrated and managed environment for the development and execution of your code. This lesson is an introduction to the Netbeans IDE, the philosophy behind it, and how it works.

## Overview of the Netbeans IDE

The Netbeans IDE is a managed type-safe environment for application development and execution. The Netbeans IDE manages all aspects of your progra1n’s execution. It allocates memory for the storage of data and instructions, grants or denies the appropriate permissions to your application, initiates and manages application execution, and manages the reallocation of memory from resources that are no longer needed. The Netbeans IDE consists of two main components: the common language runtime and the Netbeans IDE class library.

The common language runtime can be thought of as the environment that manages code execution. It provides core services, such as code compilation, memory allocation, thread management, and garbage collection. Through the common type system (CTS), it enforces strict type-safety and ensures that code is executed in a safe environment by also enforcing code access security.

The Netbeans IDE class library provides a collection of useful and reusable types that are designed to integrate with the common language runtime. The types provided by the Netbeans IDE are object-oriented and fully extensible, and they allow you to seamlessly integrate your applications with the Netbeans IDE.

# Languages and the Netbeans IDE

The .Netbeans IDE is designed for cross-language compatibility, which means, simply, that JAVA components can interact with each other no matter what supported language they were written in originally. So, an application written in Netbeans IDE might reference a dynamic-link library (DLL) file written in JAVA. This language interoperability extends to full object-oriented inheritance.

An IDE is much more than a text editor. The NetBeans Editor indents lines, matches words and brackets, and highlights source code syntactically and semantically. It lets you easily refactor code, with a range of handy and powerful tools, while it also provides code templates, coding tips, and code generators.

Keeping a clear overview of large applications, with thousands of folders and files, and millions of lines of code, is a daunting task. NetBeans IDE provides different views of your data, from multiple project windows to helpful tools for setting up your applications and managing them efficiently, letting you drill down into your data quickly and easily, while giving you versioning tools via Subversion, Mercurial, and Git integration out of the box

# The Structure of a JAVA Application

**Structure of a java program** is the standard format released by Language developer to the Industry programmer. Sun Micro System has prescribed the following **structure** for the **java** programmers for developing **java application**. A package is a collection of classes, interfaces and sub-packages

* Identity information, such as the assembly’s name and version number
* A list of all types exposed by the assembly
* A list of other assemblies required by the assembly
* A list of code access security instructions, including permissions required by the assembly and permissions to be denied the assembly

Each assembly has one and only one assembly manifest, and it contains all the description information for the assembly. However, the assembly manifest can be contained in its own file or within one of the assembly’s modules.

An assembly contains one or more modules. A module contains the code that makes up your application or library, and it contains metadata that describes that code. When you compile a project into an assembly, your Code is converted from high-level code to ML. Because all managed code is first converted to IL code, applications written in different languages can easily interact. Each module also contains a number of types. Types are templates that describe a set of data encapsulation and functionality.

Each type is described to the common language runtime in the assembly. A type can contain fields. Properties and methods, each of which should be related to a common functionality. For example, you might have a class that represents a bank account. It contains fields, properties and methods related to the functions needed to implement a but account. A field represents storage of a particular type of data. One field might store the name of an account holder. For example. Properties use similar to fields but properties usually provide some kind of validation when data is set or retrieved You might have a property that represents an account balance. When an attempt is made to change the value the property can check to see if the attempted change is greater than a predetermined limit. If the value is greater than the limit, the property does not allow the change. Methods represent behavior, such as actions taken on data stored within the class or changes to the user interface. Continuing with the bank account example. you might have a Transfer method that transfers a balance from a checking account to a savings account, or an Alert method that warns users when their balances fall below a predetermined level.

# Compilation and Execution of a JAVA Application

In Java, programs are not compiled into executable files; they are compiled into bytecode (as discussed earlier) which the JVM (Java Virtual Machine) then executes at runtime. Java source code is compiled into bytecode when we use the javac compiler. The bytecode gets saved on the disk with the file extension .class. When the program is to be run, the bytecode is converted, using the just-in-time (JIT) compiler. The result is machine code which is then fed to the memory and is executed.

Java code needs to be compiled twice in order to be executed:

1. Java programs need to be compiled to bytecode.
2. When the bytecode is run, it needs to be converted to machine code.

The Java classes/bytecode are compiled to machine code and loaded into memory by the JVM when needed the first time. This is different from other languages like C/C++ where programs are to be compiled to machine code and linked to create an executable file before it can be executed.

In Java, if you have used any reference to any other java object, then the class for that object will be automatically compiled, if that was not compiled already. These automatic compilations are nested, and this continues until all classes are compiled that are needed to run the program. So it is usually enough to compile only the high level class, since all the dependent classes will be automatically compiled.

# NETBEANS IDE

Writing graphics applications in Java using Swing can be quite a daunting experience which requires understanding of some large libraries, and fairly advanced aspects of Java. In a graphical system, a windowing toolkit is usually responsible for providing a framework to make it relatively painless for a graphical user interface (GUI) to render the right bits to the screen at the right time. Both the AWT (abstract windowing toolkit) and Swing provide such a framework. In this report, we designed and developed a simple painter project used to enable a user to draw any shape and any integrated graphic with any color using FreeHand (move the mouse using your hand to draw any shape and specify the coordinate in JPanel). Several tools such as Undo and Redo process, Clear JPanel, Set Background Color & set Foreground Color, Save paint (Panel) to file ( \*. JPG; \*. GIF; \*.\* ) and Open paint from image file are considered. The purpose of this project is to give you practice with graphical user interface programming in Java. This project implemented using the components from Java's awt and swing library in the Java programming language (NetBenas IDE 8.2). As the final result of our project is enabling you to use FreeHand to draw as an easy way to draw the Circle, Line, Rectangle, Square, and Oval, and integrated graphics such as a car, a street, a football stadium, traffic signals and others. Keywords: NetBeans IDE 8.2, JSwing, GUI

.

### DESIGN:

#### JAVA :

Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications with a reported 10 million users Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low level facilities than either of them. Java [3] can be used to write applications and applets. A Java application is similar to any other high-level language program: It can only be compiled and then run on the same machine. An applet is compiled on one machine, stored on a server in binary, and can be sent to another machine over the Internet to be interpreted by a Java-aware browser. Java comes with a large library of ready-made classes and objects. The key difference between Java 1.0 and 1.1 was in this library. Similarly, Java 2.0 has a very much larger library for handling user interfaces (Swing by name) but only small changes

#### JFrame and JPanel:

In a Java GUI program, each GUI component in the interface is represented by an object in the program. One of the most fundamental types of component is the window. Windows have many behaviors. They can be opened and closed. They can be resized..

#### Components and Layout:

Another way of using a JPanel is as a container to hold other components. In our project, we used NetBeans IDE 7.2.1 to create all components in JFrame and JPanel..

#### Framework Class Library:

The Framework Class Library (FCL) is a library of functionality available to all languages using .Netbeans IDE. The FCL provides classes that encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation, and so on.

#### Simplified deployment:

Netbeans IDE includes design features and tools which help manage the installation of computer software to ensure it does not interfere with previously installed software, and it conforms to security requirements.

#### Security:

The design addresses some of the vulnerabilities, such as buffer overflows, which have been exploited by malicious software. Additionally JAVA provides a common security model for all applications

### ARCHITECTURE:

#### Common Language Infrastructure (CLI):

The purpose of the Common Language Infrastructure (CLI) is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability. By implementing the core aspects of Netbeans IDE within the scope of the CLI, this functionality will not be tied to a single language but will be available across the many languages supported by the framework.

#### SECURITY:

The Java security APIs span a wide range of areas. Cryptographic and public key infrastructure (PKI) interfaces provide the underlying basis for developing secure applications. Interfaces for performing authentication and access control enable applications to guard against unauthorized access to protected resources. The JDK is designed with a strong emphasis on security. At its core, the Java language itself is type-safe and provides automatic garbage collection, enhancing the robustness of application code. A secure class loading and verification mechanism ensures that only legitimate Java code is executed. The Java security architecture includes a large set of application programming interfaces (APIs), tools, and implementations of commonly-used security algorithms, mechanisms.

#### CLASS LIBRARY:

Netbeans IDE includes a set of standard class libraries. The class library is organized in a hierarchy of namespaces. Most of the built-in APIs are part of either System.\* or Microsoft.\* namespaces. These class libraries implement a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. .NET class libraries are available to all CLI compliant languages. Netbeans IDE class library is divided into two parts: the Framework Class Library and the Framework Class Library

The Framework Class Library (FCL) includes a small subset of the entire class library and is the core set of classes that serve as the basic API of the Common Language Runtime.

The Framework Class Library (FCL) is a superset of the FCL classes and refers to the entire class library that ship with .Netbeans IDE. It includes an expanded set of libraries, including Windows

Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation among others. The FCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to the standard libraries of Java.

#### Memory Management:

Netbeans IDE CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Instantiations of .NET types (objects) are allocated from the managed heap; a pool of memory managed by the CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection. Netbeans IDE includes a garbage collector which runs periodically, on a separate thread

**JAVA**

Writing graphics applications in Java using Swing can be quite a daunting experience which requires understanding of some large libraries, and fairly advanced aspects of Java. In a graphical system, a windowing toolkit is usually responsible for providing a framework to make it relatively painless for a graphical user interface (GUI) to render the right bits to the screen at the right time. Both the AWT (abstract windowing toolkit) and Swing provide such a framework. In this report, we designed and developed a simple painter project used to enable a user to draw any shape and any integrated graphic with any color using FreeHand (move the mouse using your hand to draw any shape and specify the coordinate in JPanel). Several tools such as Undo and Redo process, Clear JPanel, Set Background Color & set Foreground Color, Save paint (Panel) to file ( \*. JPG; \*. GIF; \*.\* ), and Open paint from image file are considered. The purpose of this project is to give you practice with graphical user interface programming in Java. This project implemented using the components from Java's awt and swing library in the Java programming language (NetBenas IDE 8.2). As the final result of our project is enabling you to use FreeHand to draw as an easy way to draw the Circle, Line, Rectangle, Square, and Oval, and integrated graphics such as a car, a street, a football stadium, traffic signals and others. Keywords: NetBeans IDE 8.2, JSwing, GUI.**.**

#### Features:

#### Object Oriented

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

#### Platform Independent

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

#### Simple

Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

#### Secure

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

#### Architecture-neutral

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

#### Portable

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

#### Robust

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

#### Multithreaded

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

#### Interpreted

Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

#### High Performance

With the use of Just-In-Time compilers, Java enables high performance.

#### Distributed

Java is designed for the distributed environment of the internet.

#### Dynamic

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

# BACK END: SQL-SERVER 2008

Microsoft SQL-SERVER 2008 is a full featured relational database management system that offers a variety of administrative tools to easy the burdens of database development & administration.

Enterprise manager is the main administrative console for sql server installation. We can perform high level administrative function that affects one or more servers, schedule common maintenance tasks or create & modify the structure of individual database. Query analyzer offers a quick & dirty method for performing query against sql server database. SQL profiler provides a window into inner working of our database. Service manager is used to Ms-SQL servers, Ms-Distributed transaction co-coordinator & sql server agent processes. Data transformation services provides an extremely data between a Microsoft sql server installation & large variety of other formats.

#### Features:

* Database mirroring
* Online restore opportunity
* Online indexing operations
* SQL-server management studio
* Native XML support
* Security enhancement

# PROPOSED WORK

Stock and Invoice Management is a software where Stock details are been setup by an Employee and Invoice Details set up by accountant. This system is a hierarchical structure. It contains certain numbers of actors. Each will have their respective actions. A login id password Provide by company. For each of the actor to perform their action, they must be logged in with their respective ids. It is a window application managing invoice generated content. Earlier versions were to store the details of Stock. Here updating, retrieving actions are also been performed.

The proposed system is very user friendly, providing 2 actors. Account for generate invoices, employee for adding and updating Stocks.

### Advantages of proposed system:

* It’s user-friendly.
* Each of the actors has been provided with register id, with which he has to log on to perform the required action.
* Customer’s data is highly secured in this software.
* Only the administrator has a right to access all database.
* Employee has a right to retrieve the details of a Customer.

# Hardware & Software Requirements

### Software:

**Front End :** JAVA 8 (Netbeans IDE 8.2)

**Back End :** SQL Server 2008

**Documentation Tool :** Microsoft Office

### Hardware:

**Processor :** 1 GHz Processor

**RAM :** 1 GB (minimum recommended)

**Hard Disk :** 20 GB (minimum recommended)

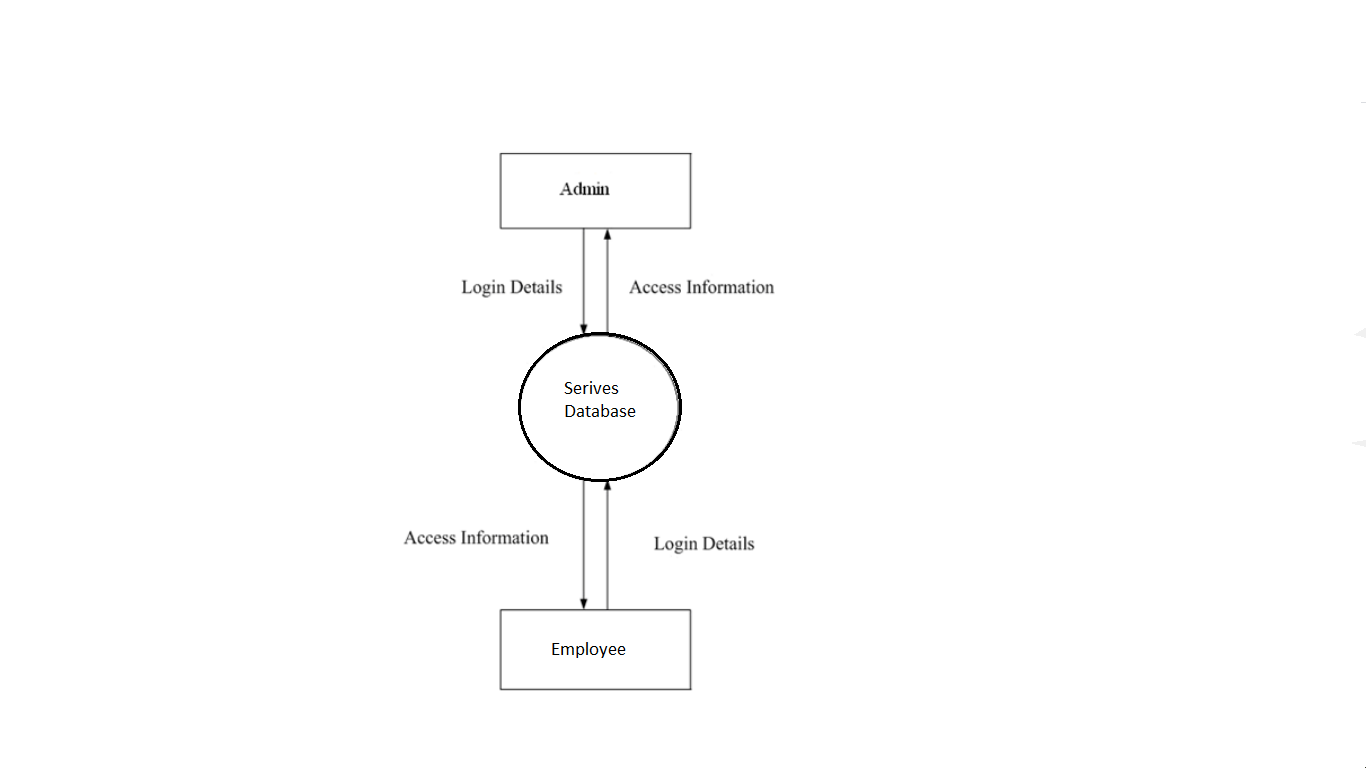
**Monitor :** 11”inch DISPLAY or more

**Keyboard :** 108 key normal

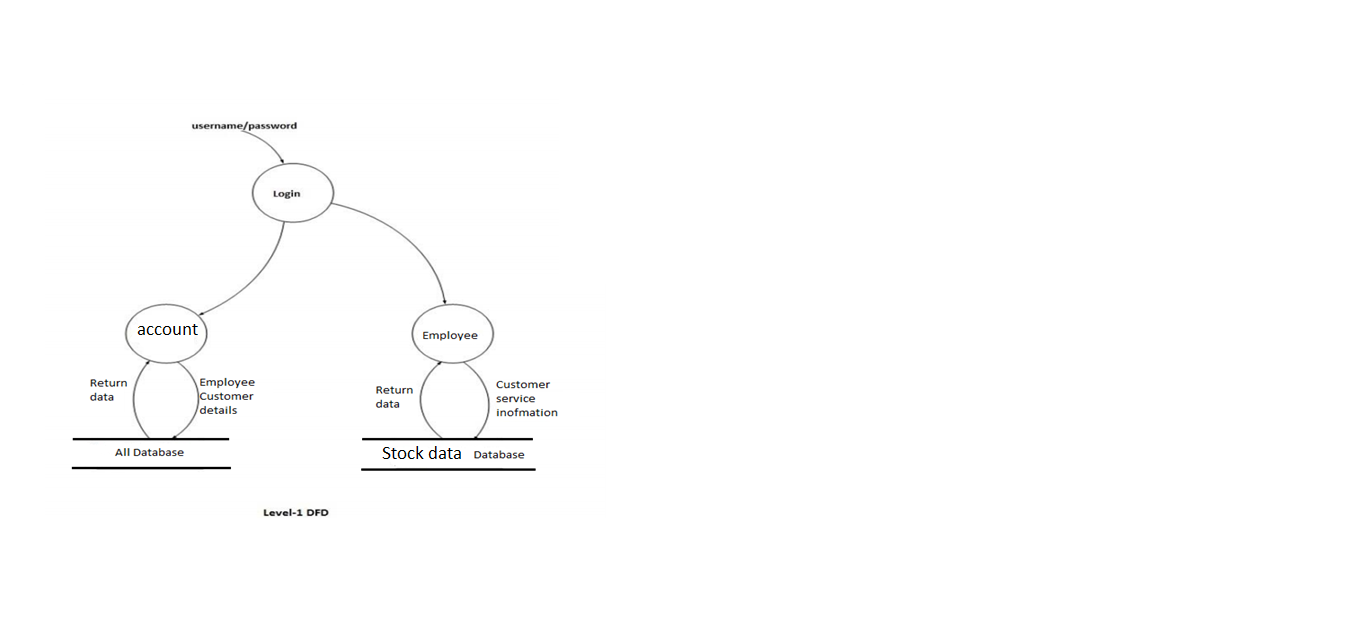
# System Design

# Data Flow Diagram

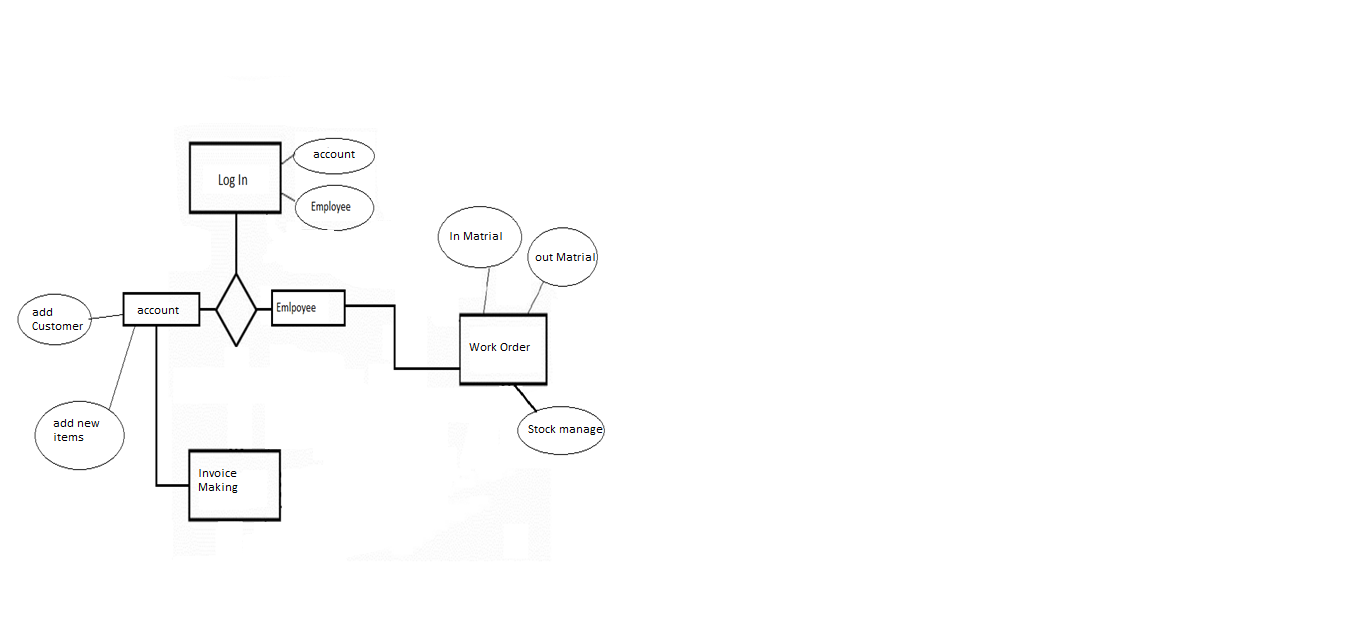
**0 Level DFD**



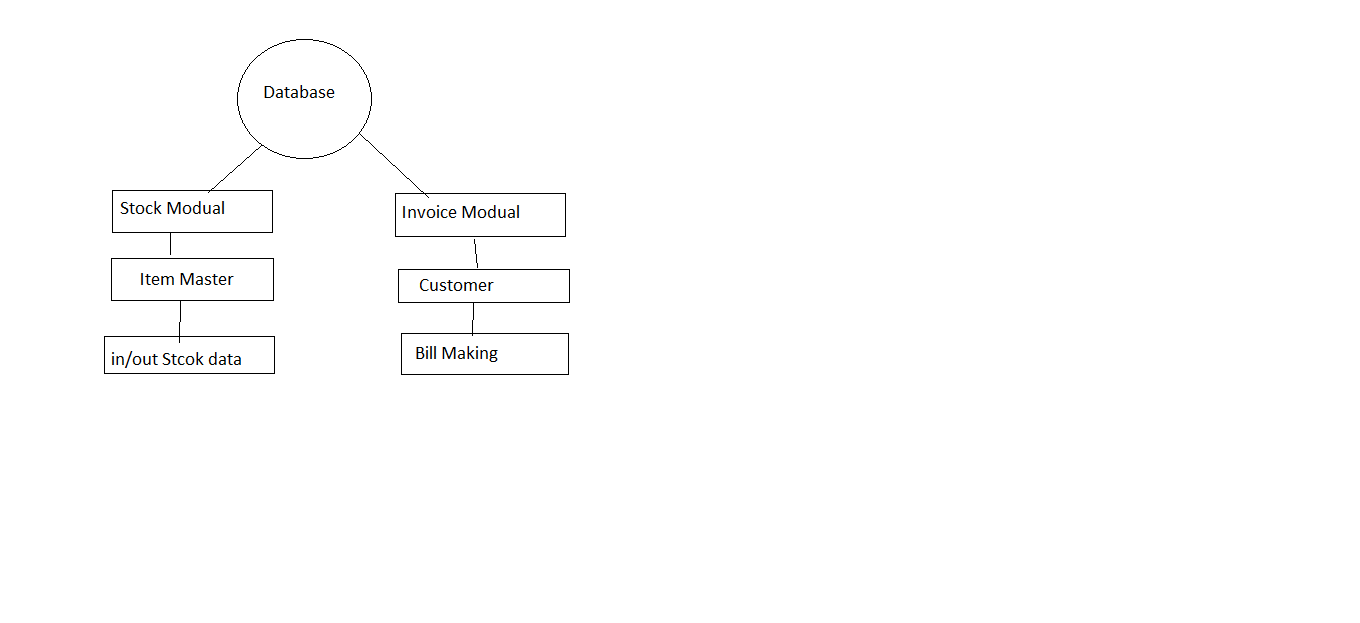
**1 Level DFD**



**E R Diagram**

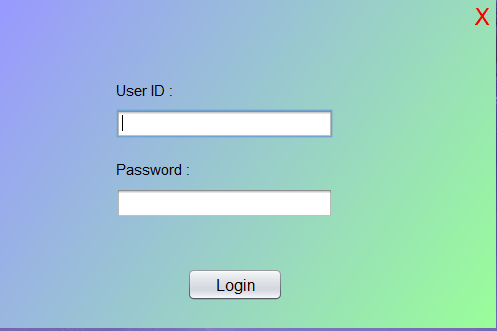


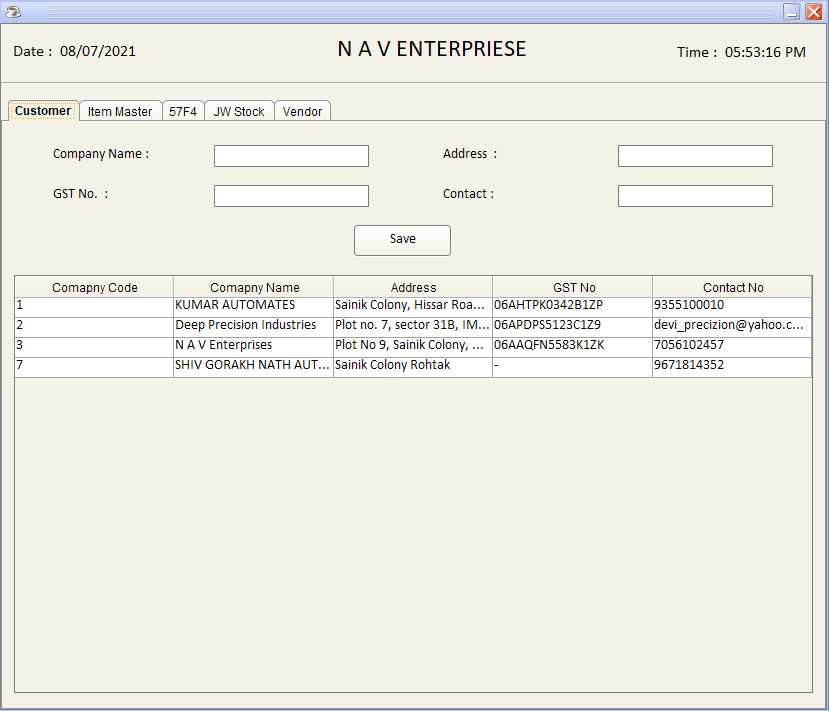
**Flow Chart**



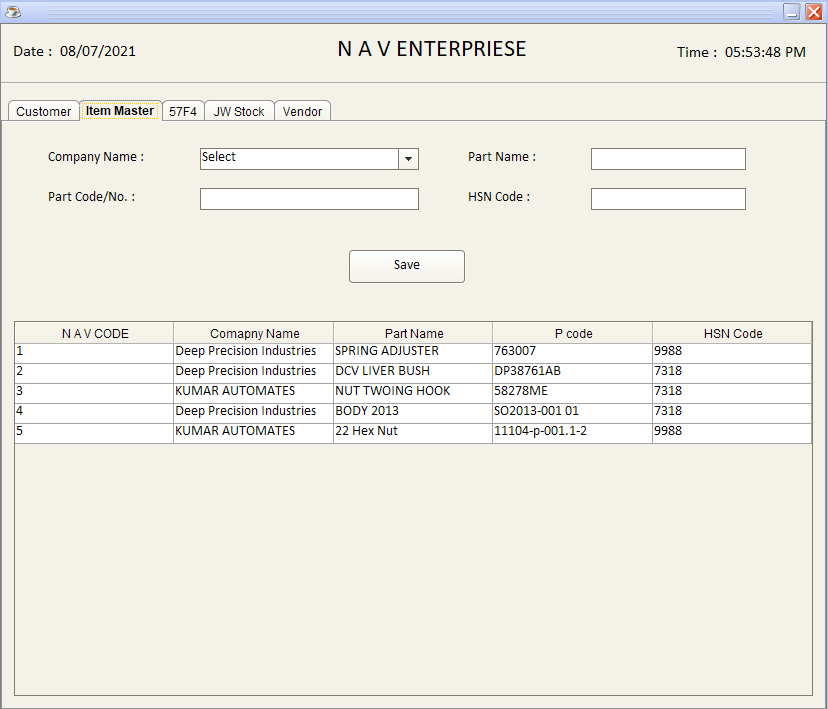
# SNAPSHOT OF PROJECT

Login Page

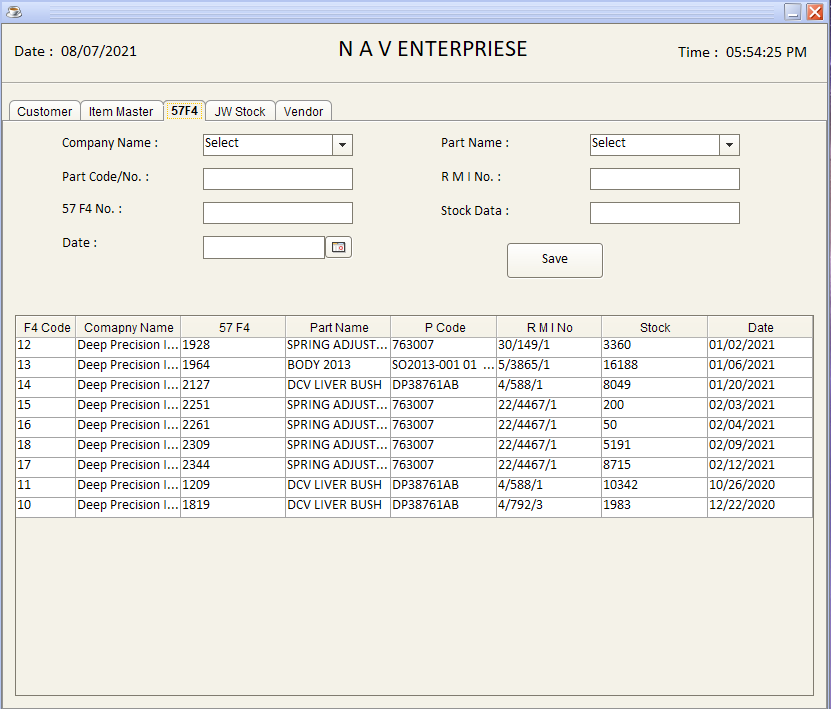
****

Add Customer

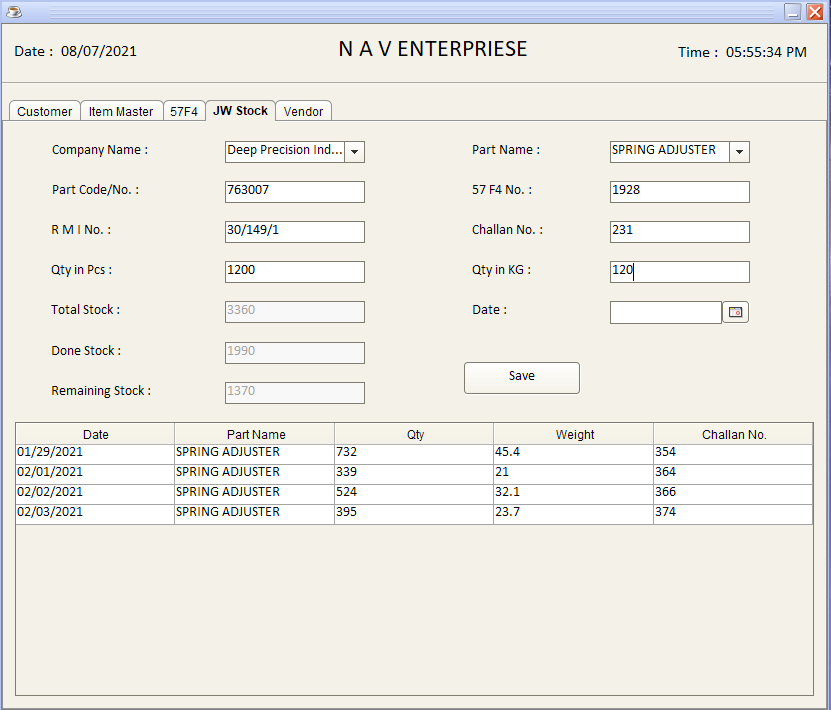
**Item Master**

****

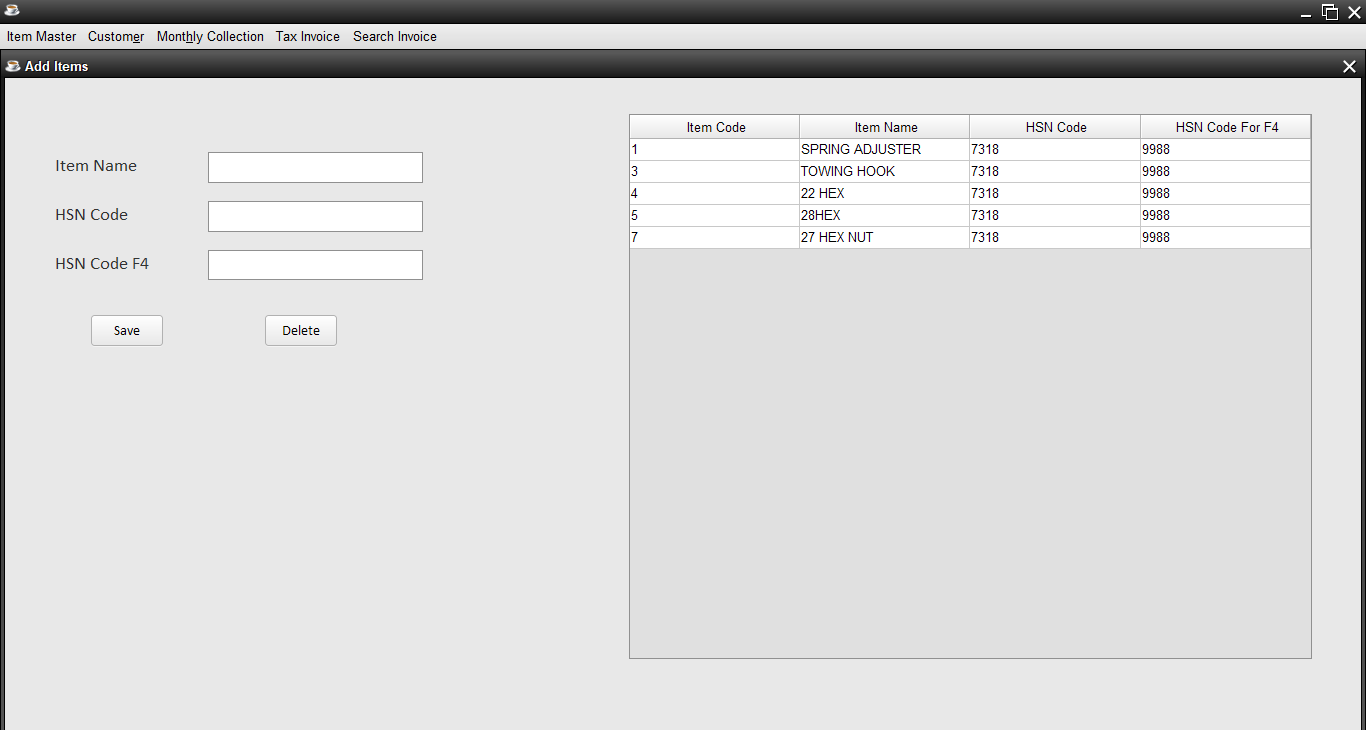
**Work order**

****

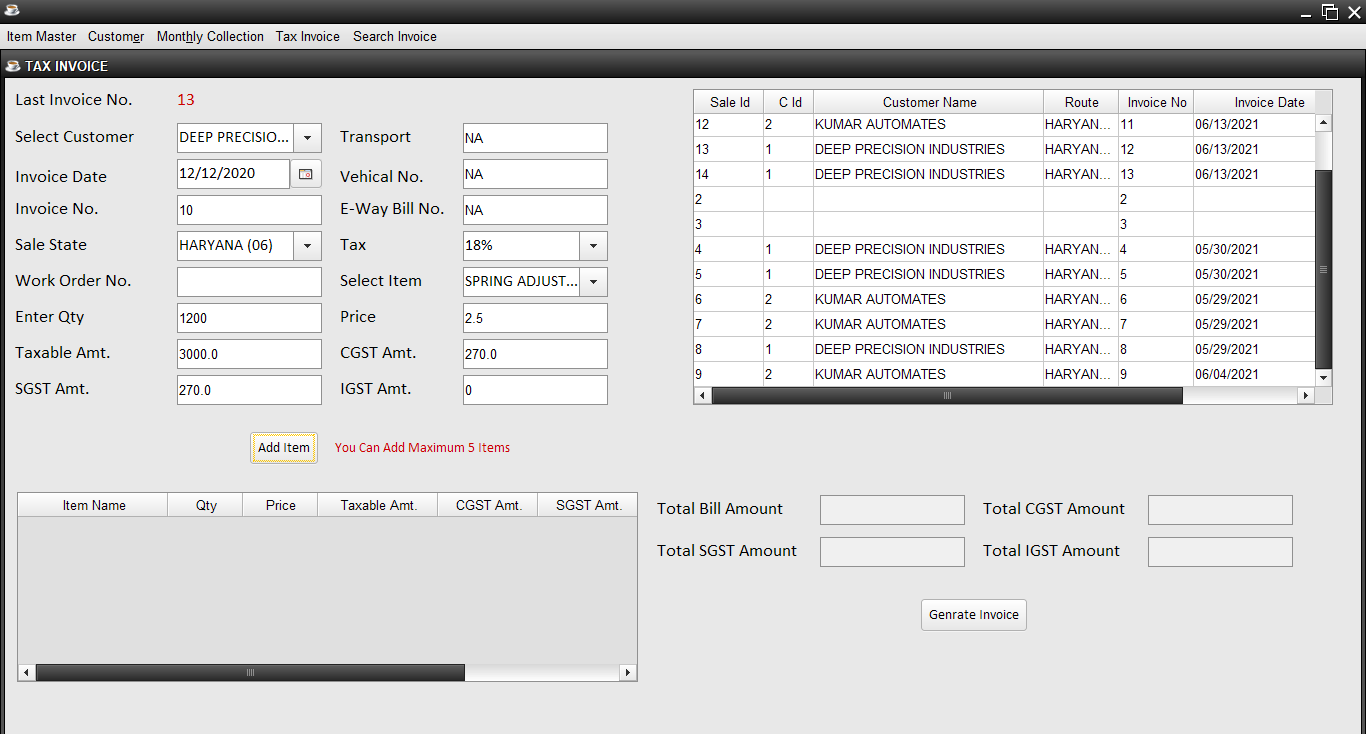
**Job work Order Stock**

****

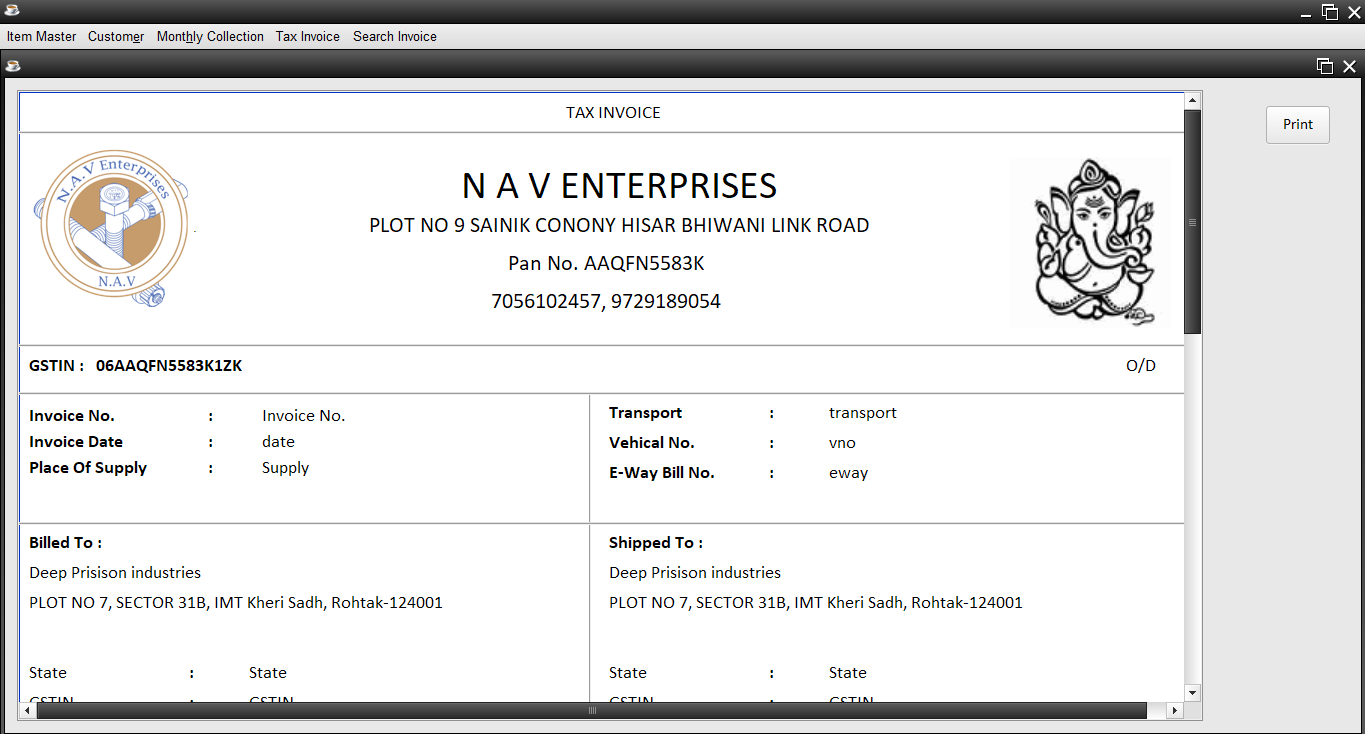
**Item Master For Invoice**

****

**Generate Invoice**

****

**Print Preview**

****

**TESTING**

**Software testing** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs(errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

* meets the requirements that guided its design and development,
* works as expected,
* can be implemented with the same characteristics,
* and satisfies the needs of stakeholders.

#### Static vs. dynamic testing

#### There are many approaches to software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment.

**The box approach**

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

#### **White-Box testing**

White-box testing(also known as clear box testing, glass box testing, transparent box testing and s**t**ructural testing ) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in‐circuit testing (ICT).

#### **Black-box testing**

Black-box testingtreats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all‐pairs testing, state transition tables, decision table testing, fuzz testing, model‐based testing, use case testing, exploratory testing and specification-based testing.

**Specification-based testing:**

aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specification and requirements, i.e., what the application is supposed to do.

It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non‐functional, though usually functional. Ad hoc testing and exploratory testing are important methodologies for checking software integrity, because they require less preparation time to implement, while the important bugs can be found quickly. In adhoc testing, where testing takes place in an improvised, impromptu way, the ability of a test tool to visually record everything that occurs on a system becomes very important

#### **Grey-box testing**

Grey-box testinginvolves having knowledge of internal data structures and algorithms for purposes of designing tests, while executing those tests at the user, or black-box level. The tester is not required to have full access to the software's source code. Manipulating input data and formatting output do not qualify as grey- box, because the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

**TESTING LEVELS**

There are generally four recognized levels of tests: unit testing, integration testing, system testing, and acceptance testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined are unit-, integration-, and system testing that are distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

**Unit Testing**

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white- box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

#### **Integration testing**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

**Component interface testing**

The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a timestamp logged to allow analysis of thousands of cases of data passed between units for days or weeks.

#### **System testing**

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements. For example, a system test might involve testing a log-on interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then log-off.

**Regression testing**

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, as degraded or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working, correctly, stops working as intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re- running previous sets of test-cases and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features.

#### **Alpha testing**

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

**Beta testing**

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

#### **Functional vs non-functional testing**

Functional testing refers to activities that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work."

Non-functional testing refers to aspects of the software that may not be related to a specific function or user action, such as scalability or other performance, behavior under certain constraints, or security. Testing will determine the breaking point, the point at which extremes of scalability or performance leads to unstable execution. Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users

#### **Top-down and bottom-up**

**Bottom Up Testing** is an approach to integrated testing where the lowest level components (modules, procedures, and functions) are tested first, then integrated and used to facilitate the testing of higher level components. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. The process is repeated until the components at the top of the hierarchy are tested. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

**Top Down Testing** is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

In both, method stubs and drivers are used to stand-in for missing components and are replaced as the levels are completed.

# MAINTENANCE

**Software maintenance** in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that the majority, over 80%, of the maintenance effort is used for non-corrective actions. This perception is perpetuated by users submitting problem reports that in reality are functionality enhancements to the system. More recent studies put the bug-fixing proportion closer to 21%.

Software maintenance is a very broad activity that includes error correction, enhancements of capabilities, deletion of obsolete capabilities, and optimization. Because change is inevitable, mechanisms must be developed for evaluation, controlling and making modifications.

An integral part of software is the maintenance one, which requires an accurate maintenance plan to be prepared during the software development. It should specify how users will request modifications or report problems. The budget should include resource and cost estimates. A new decision should be addressed for the developing of every new system feature and its quality objectives. The software maintenance, which can last for 5–6 years (or even decades) after the development process, calls for an effective plan which can address the scope of software maintenance, the tailoring of the post delivery/deployment process, the designation of who will provide maintenance, and an estimate of the life-cycle costs. The selection of proper enforcement of standards is the challenging task right from early stage of software engineering which has not got definite importance by the concerned stakeholders.

**IMPLEMENTATION**

A **product software implementation method** is a systematically structured approach to effectively integrate a software based service or component into the work flow of an organizational structure or an individual end-user.

This entry focuses on the process modeling (Process Modeling) side of the implementation of “large” (explained in complexity differences) product software, using the implementation of Enterprise Resource Planning systems as the main example to elaborate on.

A product software implementation method is a blueprint to get users and/or organizations running with a specific software product.

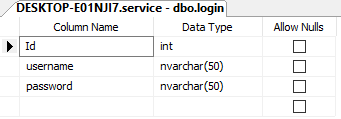
The method is a set of rules and views to cope with the most common issues that occur when implementing a software product: business alignment from the organizational view and acceptance from human view.

The implementation of product software, as the final link in the deployment chain of software production, is in a financial perspective of a major issue.

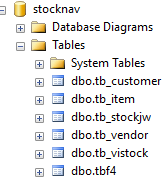
It is stated that the implementation of (product) software consumes up to 1/3 of the budget of a software purchase (more than hardware and software requirements together).

**TABLES IN DATABASE**

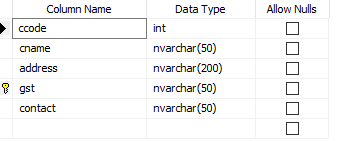
Login



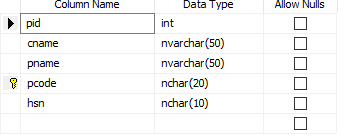
**Stock Database**



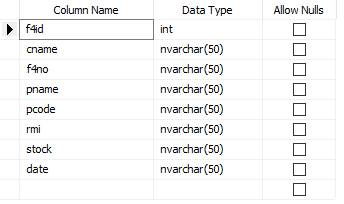
**Stock Customer Table**

****

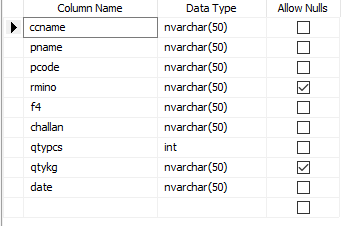
**Stock Item Master**

****

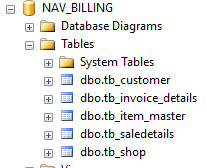
**Stock Work Order**

****

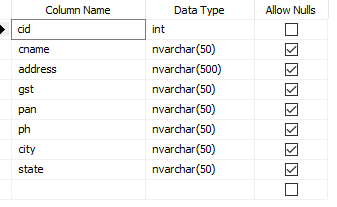
**Stock**

****

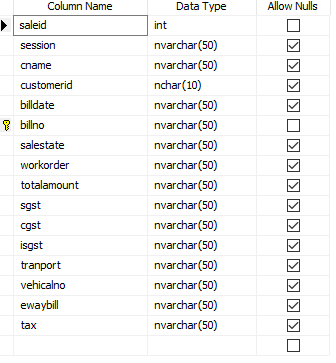
**Invoice Database**

****

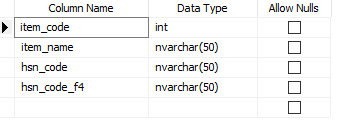
**Invoice Customer**



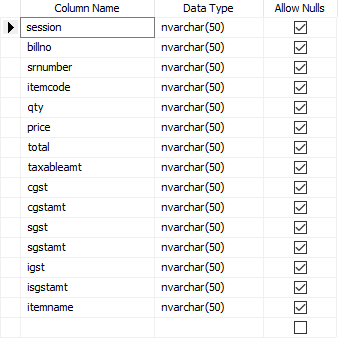
**Invoice**

****

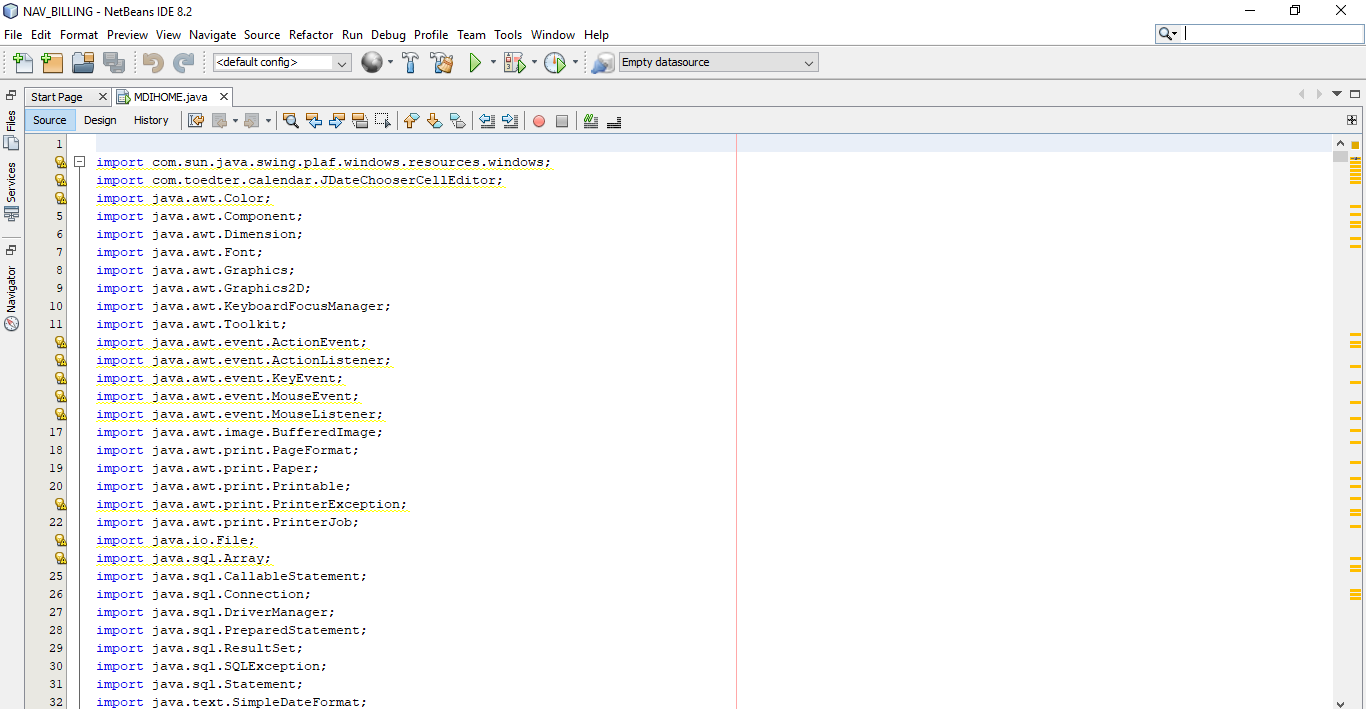
**Invoice Item Master**

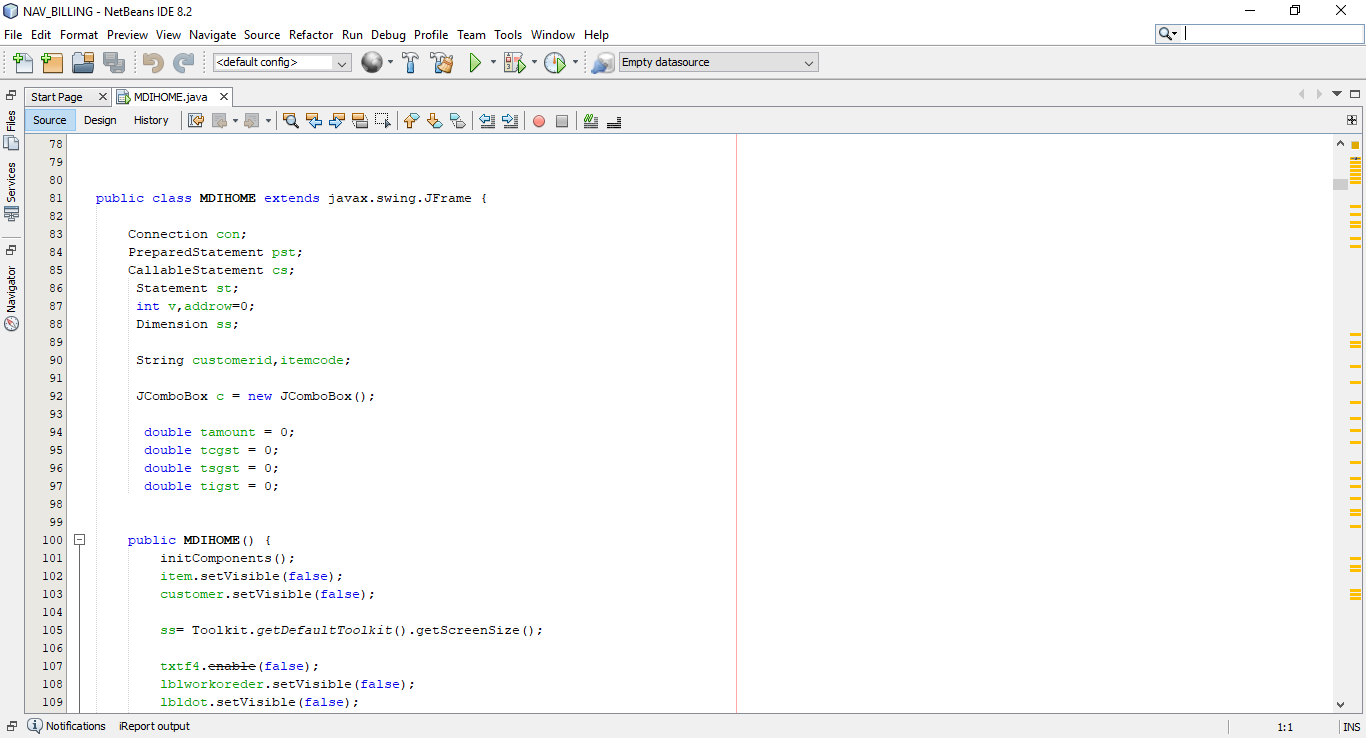


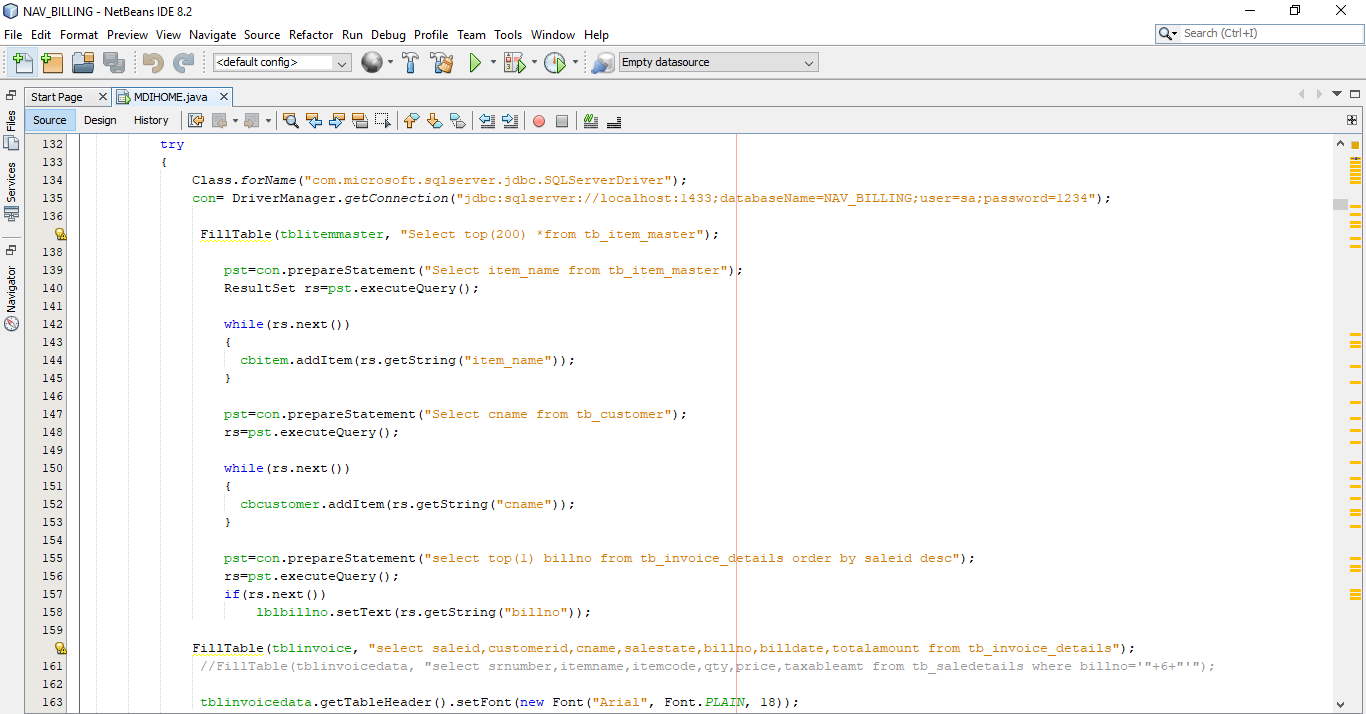
**Sale Details**

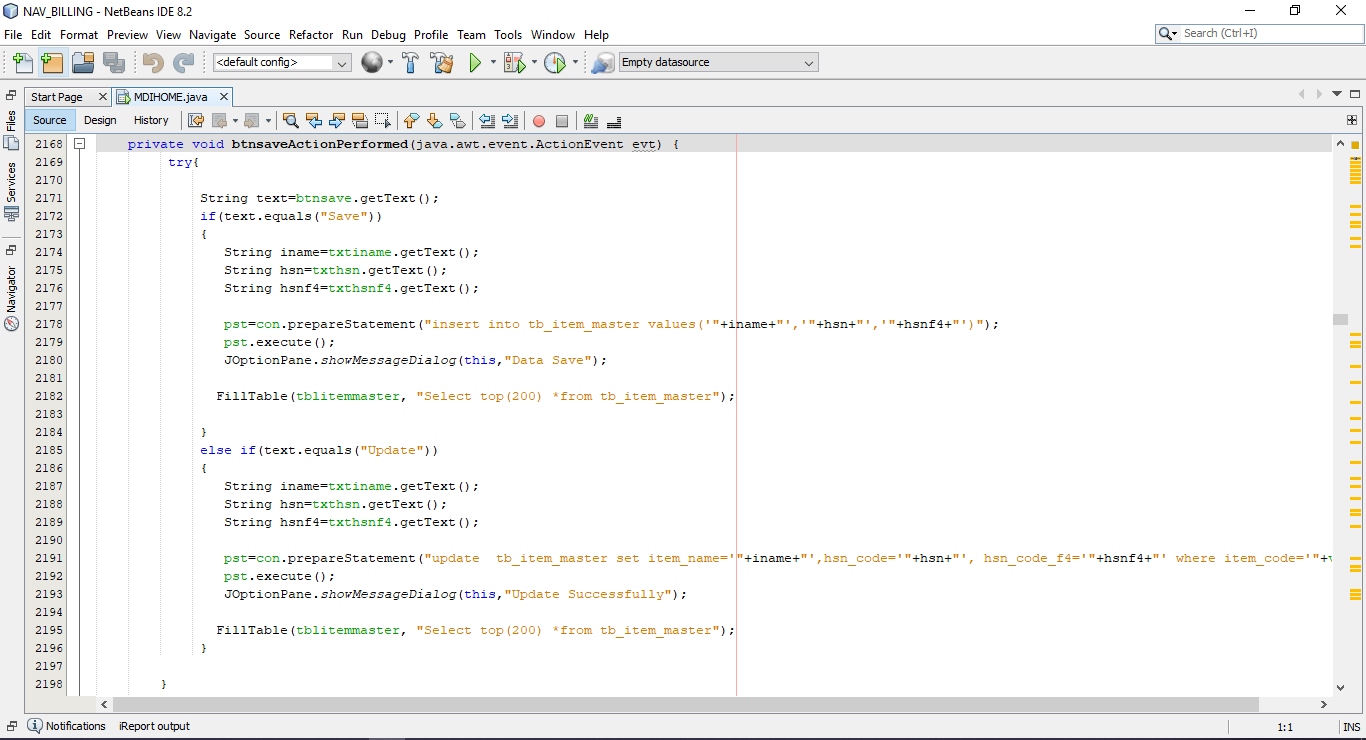
****

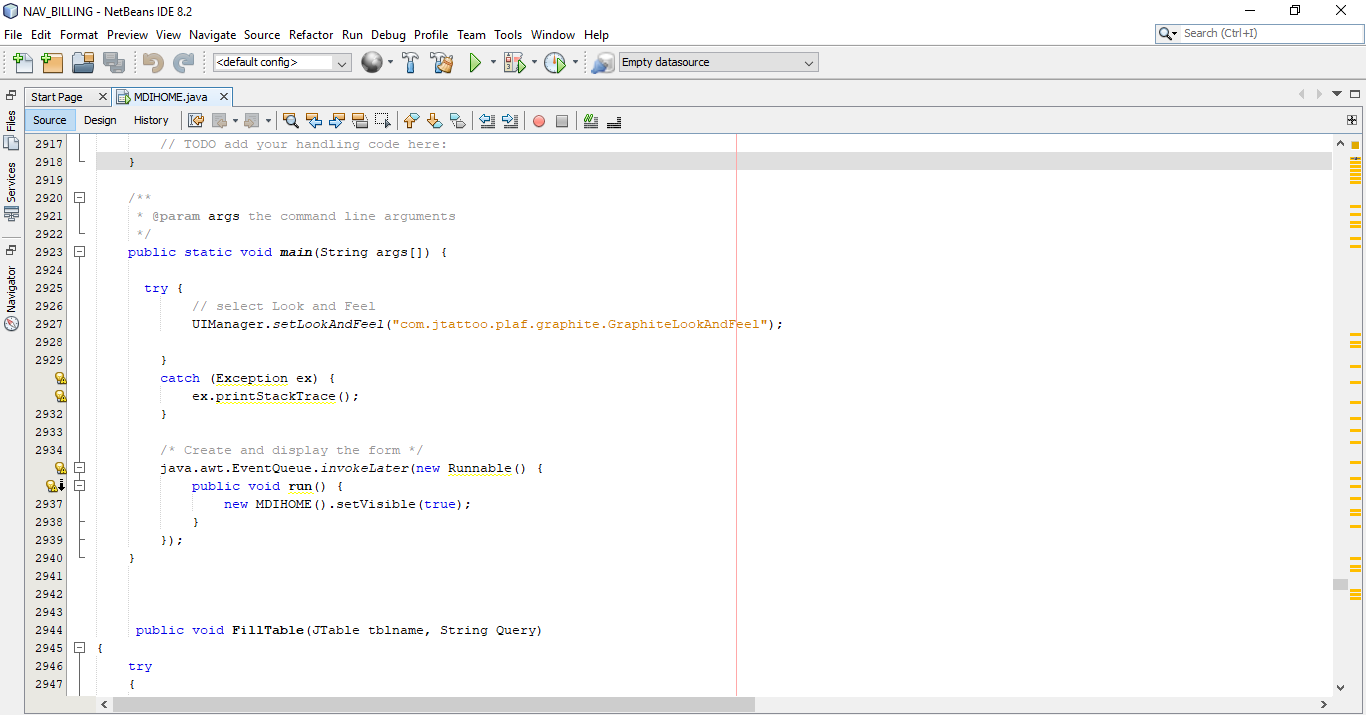
**Coding Details**





****

****

****

**Future Scope**

Such a window based platform has immense potential for future growth not only for students but also for faculty as well as department concerned in improving upon their efficiency of work with no or less burden. Some of the areas where future scope lies and if implemented would help a great deal to the students at large.

* Invoice generators calculate the total amount for you to prevent errors
* Processes done by humans are prone to error
* Less Paper Work

**Conclusion**

* The system provides an excellent support to the corporate client..
* Computerized surveying system provides an easy, fast access and support for the user.
* The usage of software increases the efficiency, decreases the effort.
* It has been thoroughly tested and implemented.

**References**

1. Pressman, Roger S, Software Engineering, fifth edition, McGraw-Hill Higher Education, 2001.
2. Elimasri | Navathe, Fundamentals of Database Systems, third edition, Addison-Wesley,2000.
3. Professional ASP.NET 4.0 in C# and VB by Bill Evegen Scott Hanselman Devin Rader.
4. Beginning ASP.NET 4.0 in C# by Matthew MacDonald.

Website Referred:

1. [www.google.com](http://www.google.com)
2. [www.wikipedia.com](http://www.wikipedia.com)
3. [www.w3schools.com](http://www.w3schools.com)
4. [www.aspforums.com](http://www.aspforums.com)
5. [www.forums.tutorialized.com](http://www.forums.tutorialized.com)
6. [www.youtube.com](http://www.youtube.com)