

INDEX

TOPICS	Page No's
➤ Certificates	
➤ Acknowledgement	
➤ Abstract	
➤ Figures/Tables	
CHAPTER-1: INTRODUCTION	1
CHAPTER-2: LITERATURE SURVEY	2-3
CHAPTER-3: SYSTEM ANALYSIS	4
3.1 Existing System	4
3.2 Proposed System	5
CHAPTER-4: SYSTEM REQUIREMENTS	
4.1 Hardware Requirements	6
4.2 Software Requirements	6
CHAPTER-5: SYSTEM STUDY	
5.1 Feasibility Study	7
5.2 Feasibility Analysis	7-9
CHAPTER-6: SYSTEM ARCHITECTURE	
6.1 Data Flow Diagram	10
6.2 UML Diagrams	10-14
CHAPTER-7: INPUT AND OUTPUT DESIGN	
7.1 Input Design	15-17
7.2 Output Design	17-18
CHAPTER-8: IMPLEMENTATION	
8.1 MODULES	19
8.1.1 Module Description	19

CHAPTER-9: SOFTWARE ENVIRONMENT

9.1 JAVA **20-41**

9.2 Source Code **42-53**

CHAPTER-10: RESULTS/DISCUSSIONS

10.1 System Test **54-56**

10.2 Output Screens **57-66**

CHAPTER-11: CONCLUSION **67**

CHAPTER-12: REFERENCES/BIBLIOGRAPHY **68**

Abstract

Most of the largescale business organizations have revolutionized their business standards. As a part of this a lot of automation has taken place. On a statistical approach we find that in the Global Market even a small retail store running any business plays a key role. One among the business models available in the market is B2B model. B2B model is nothing but Business to Business model. The general objective of the proposed web application is to support small retail stores enhance their business standards and help automate their business more effectively. Current System makes use of promotional pricing. The proposed web application targets businesses following the B2B marketing model. It helps both the wholesale market and organization who owns the application.

LIST OF FIGURES

FIGURES/TABLES

S.No	Figures/Tables	Page No.
1	Data Flow Diagrams	10
2	UML Diagrams	11-14
3	Compiling and interpreting java source code	21
4	Output Screens	57-66

CHAPTER-1

INTRODUCTION

Order managing system is a web based java application which is used to order entry and processing the order. This may reduce the paper and printing and distribution work which saves lot of time.

To manage all the orders in a proper way this system will be done via catalogs and websites. Customer will be given a username and password. Customer will get a reference number which is given to check the customer identity. Permanent id will be given after proper identification. Customer can place the order after the registration with his unique username and password.

Customer can get all the details about the product and he can give the order to a product online so that the customer can take a right choice. A SMS can be sent to the customer on the request to track the order. Order once placed can pass through many phases till it finally reaches the customer. At any point customer can check the status of the order online or over phone with the reference number.

CHAPTER-2

LITERATURE SURVEY

TITLE: - Real-Time Process Management System in a Restaurant by Sharing Food Order Information

AUTHOR: - Takeshi Shimmura; Takeshi Takenaka; Motoyuki Akamatsu

ABSTRACT: - In a full-service restaurant, it is crucial to share order information among staff in the dining room and kitchen. This paper introduces a real-time process management system for restaurants using an advanced point-of-sale (POS) system by which staff can share order information in real time. In this system, kitchen staff can check all customer orders by the dish that was ordered and the elapsed time of each order. Moreover, dining hall workers can grasp their customer situation with a monitor. By introducing this system to a restaurant, we confirmed that it can make preparation processes more efficient and reduce customers' claims.

TITLE: Order Management System for Time and Quantity Saving of Recipes Ingredients Using GPS Tracking Systems

AUTHOR: - Asghar Ali Shah, s. M. Ilyas

ABSTRACT: - Order management system is one of supply chain management system which has an important part related to the customer satisfaction and profit of the company. Order Management process is a software which is supportive for the businesses work hardware goods, where storeowner saves the records of sales and purchase. Mishandled record means disappointed clients, too much money tied up in storerooms, and slower sales. In the current scenario, people place the orders customers have to visit hotels or cafeterias to know about foods and then delivered order and pay. In this process time and manual work is required. Due to the enhancement in technology, it is very easy for us to buy all the things of daily use by ordering online on E-Commerce sites. But in this advanced era of technology, we did not have any application or site for buying ingredients of the recipe that we want to make. In this paper, we are saving money, time and Quantity of ingredients from wastage. In which we are providing Recipes and the Quantitative ingredients for that recipe within 1 to 2 hours. This system will provide quick service of order delivering to our consumer.

TITLE: Applying capacity-management science: the case of Browns Restaurants

AUTHOR: Brian Sill and Robert Decker

ABSTRACT: - Capacity-management science (CMS) is an analytical approach to restaurant management that continually monitors and aligns operations with competitive goals. By using CMS, managers are able to combine cause-and-effect thinking with specific measurements. The application of CMS follows several steps. These include the definition of parts of restaurant production, measurement of capacities of each component, analysis of historical demand, calculation of ideal capacity use, comparison of current usage with ideal use-levels and the formation of teams to rework processes.

CHAPTER-3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Before developing any application, a comprehensive and systematic research needs to be done so as to target the market needs. A number of wireless applications for restaurant ordering have been analyzed, developed and implemented in restaurants. These have been implemented using PDA's (Personal Digital Assistant), Windows Mobiles or Android Mobiles. Also many wireless technologies are available today. This application is developed specifically for medium and large scale restaurants using Wi-Fi (Wireless Fidelity) system and Android-4.0 as a working platform.

DISADVANTAGES

- Lack of inventory visibility. Maintaining accurate inventory and handling allocation are tasks that can make a big difference in building strong customer relationships
- Human error
- Security dangers
- Need for customer support
- Siloed operations and information.

3.2 PROPOSED SYSTEM

A web based ordering system named CaptainPad [2], is the wireless technology solution for automating the ordering system in the hotels and restaurants. Using a Captain Pad, orders can be sent directly from the table to the kitchen, which ensures that customers receive their orders faster. Developers used MS Dos, Win 3.11, Win95, 98 and Win NT, Win XP, Linux as operating systems, C, C++, Java, XML, HTML and PL/SQL as programming language and web based technologies like (Java Servlets, JSP, EJB, HTML, XML, Struts, and Hibernate). They also used My SQL and Oracle 8 for databases and for web servers JBoss, Apache and Tomcat. The whole menu is loaded in the CaptainPad device. With the use of CaptainPad the overall efficiency increases

Advantages

The advantages of an order management system include having the ability to keep your inventory and tracking in line, while also providing a seamless, front-facing experience for your customers.

CHAPTER-4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS:

- Processor - Pentium –III
- RAM - 256 MB (min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

4.2 SOFTWARE REQUIREMENTS:

- Operating System : Windows95/98/2000/XP
- Application Server : Tomcat5.0/6.X
- Front End : HTML, Jsp
- Scripts : JavaScript.
- Server side Script : Java Server Pages.
- Database : MySQL 5.0
- DatabaseConnectivity : JDBC

CHAPTER-5

SYSTEM STUDY

5.1 Feasibility Study:

Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time.

5.2 Feasibility Analysis

There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

OPERATIONAL FEASIBILITY

User-friendly

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

Reliability

The package will pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

Security

The web server and database server should be protected from hacking, virus etc

Portability

The application will be developed using standard open source software (Except Oracle) like Java, tomcat web server, Internet Explorer Browser etc these software will work both on Windows and Linux o/s. Hence portability problems will not arise.

Availability

This software will be available always.

Maintainability

The system called the ewheelz uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses My-Sql, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

ECONOMIC FEASILITY

The computerized system takes care of the present existing system's data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports.

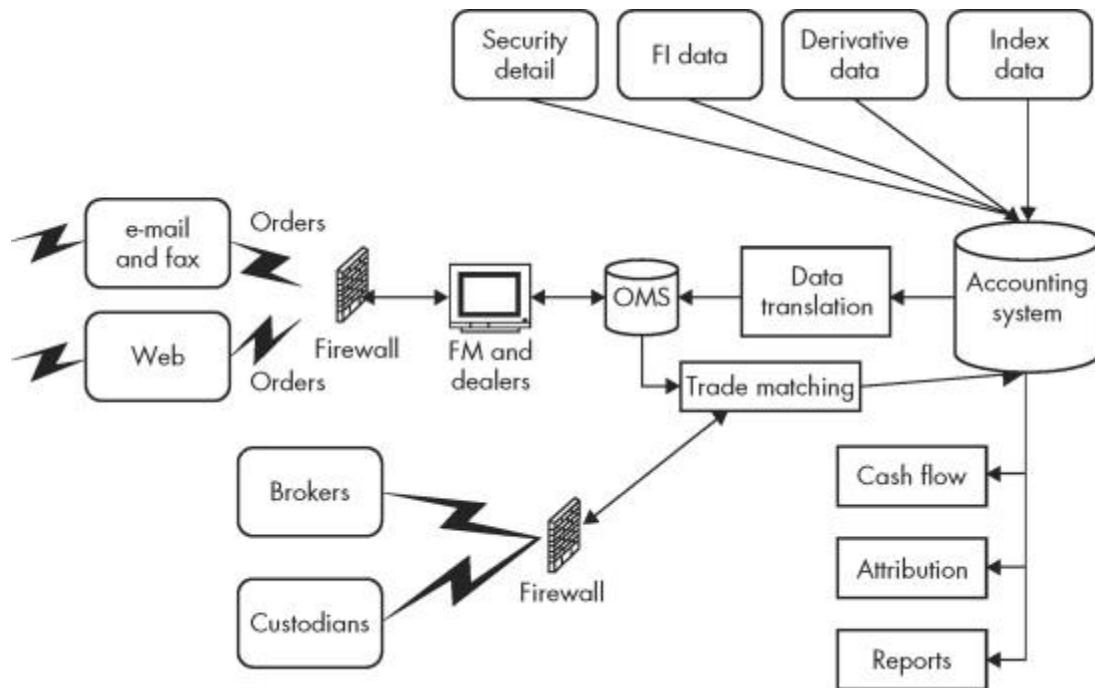
It should be built as a web based application with separate web server and database server. This is required as the activities are spread through out the organization customer wants a centralized database. Further some of the linked transactions take place in different locations.

Open source software like TOMCAT, JAVA, Mysql and Linux is used to minimize the cost for the Customer.

CHAPTER-6

SYSTEM ARCHITECTURE

6.1 Data Flow Diagram



6.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

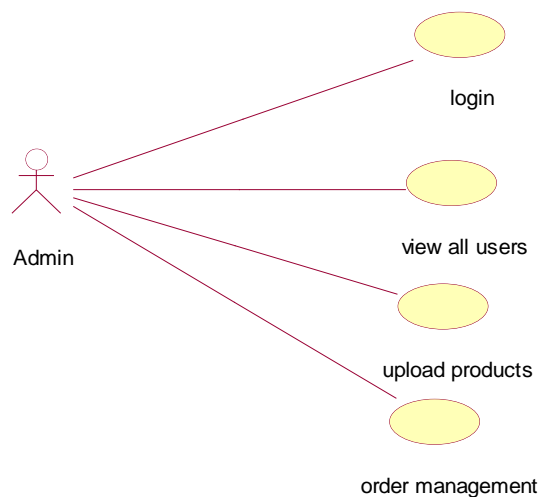
GOALS:

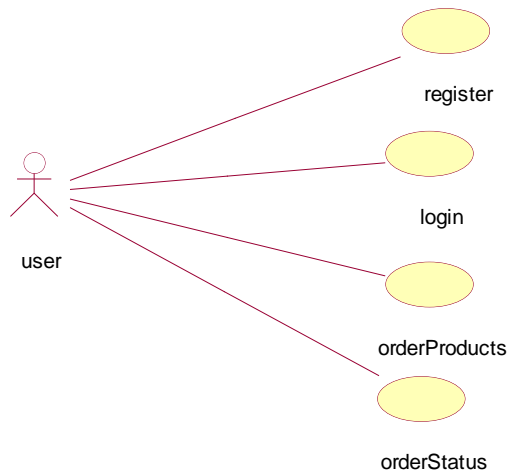
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

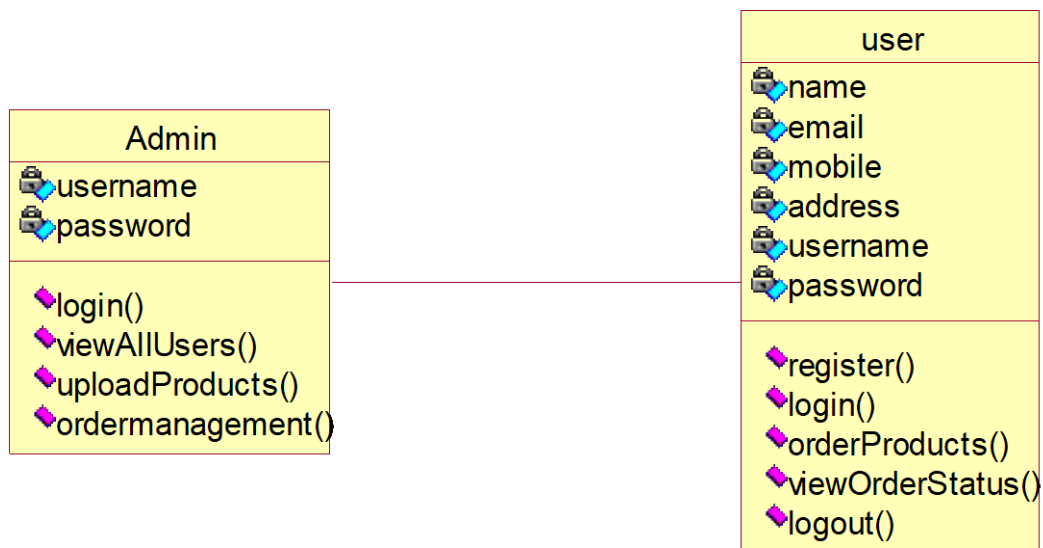
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.





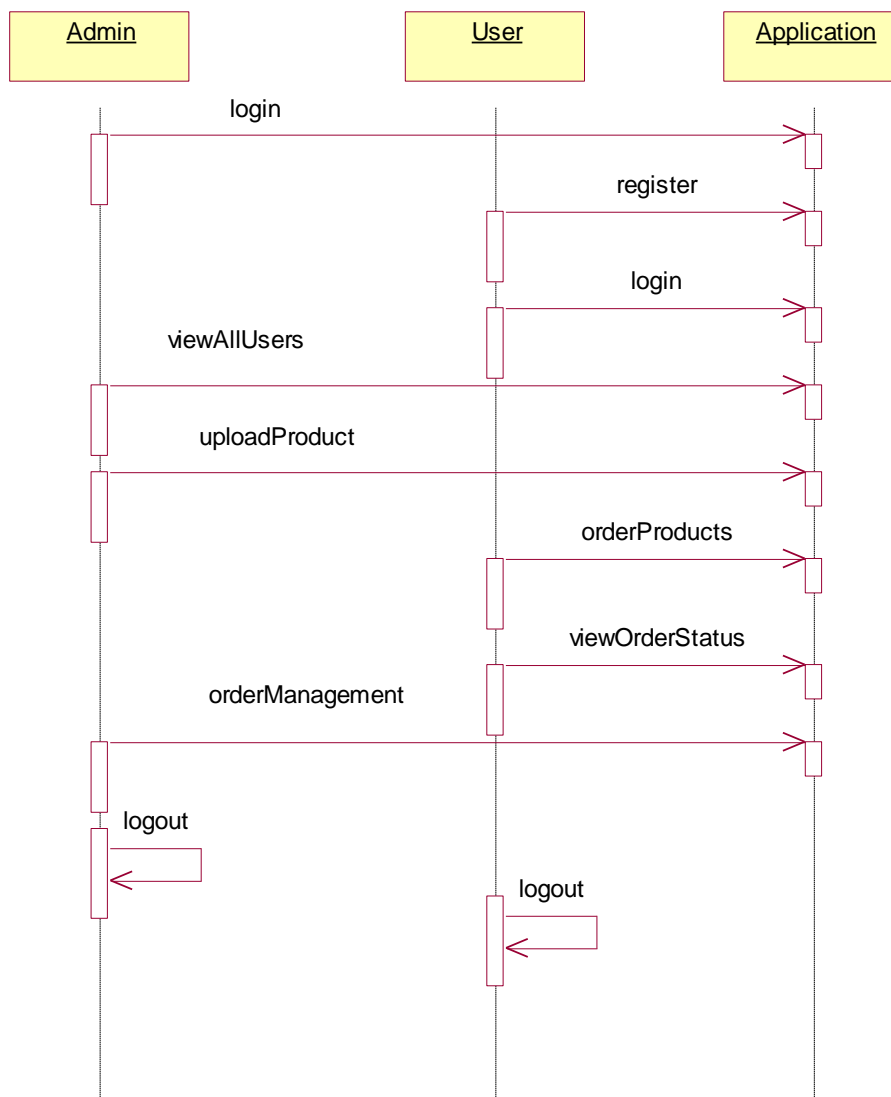
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



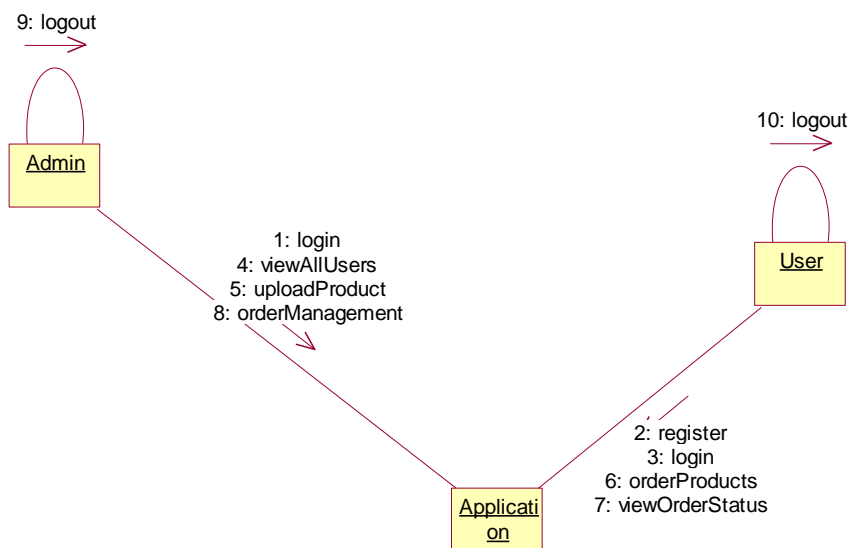
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



Collaboration

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.



CHAPTER-7

INPUT AND OUTPUT DESIGN

7.1 INPUT DESIGN

To provide flexibility to the users, the interfaces have been developed that are accessible through a browser. The GUI'S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

7.1.1 INPUT OBJECTIVES

The 'administrative user interface' concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. These interfaces help the administrators with all the transactional states like Data insertion, Data deletion and Date updation along with the extensive data search capabilities.

The 'operational or generic user interface' helps the end users of the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information in a customized manner as per the included flexibilities

7.1.2 INPUT & OUTPOUT REPRESENTETION

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

7.1.3 INPUT STAGES:

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

7.1.4 INPUT TYPES:

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

7.1.5 INPUT MEDIA:

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use

- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

7.2 OUTPUT DESIGN:

In general are:

- External Outputs whose destination is outside the organization.
- Internal Outputs whose destination is within organization and they are the User's main interface with the computer. Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly with the system.

7.2.1 OUTPUT DEFINITION

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

For Example

- Will decimal points need to be inserted
- Should leading zeros be suppressed.

7.2.2 OUTPUT MEDIA:

In the next stage it is to be decided that which medium is the most appropriate for the output.

The main considerations when deciding about the output media are:

- The suitability for the device to the particular application.
- The need for a hard copy.
- The response time required.
- The location of the users
- The software and hardware available.

Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The main outputs desired according to the requirement specification are:

The outputs were needed to be generated as a hard copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

CHAPTER-8

IMPLEMENTATION

8.1 MODULES

1. Admin
2. User

8.2 MODULES DESCRIPTION

In this Application, there are two modules Admin & User. They must need to register into the application to run it successfully. After Successfully registering into the application he/she can perform the following functions.

Admin:

Here the admin can directly login into the application and the admin also perform some actions like, view all profiles, upload products, order management.

The above are the operations going to be done by the Admin

User

In this application the User should register with the application and login into the application. After he gets login into the application, the User can perform the following operations like Order products, order status.

CHAPTER-9

SOFTWARE ENVIRONMENT

9.1 About Java

Initially the language was called as “oak” but it was renamed as “java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language
- Java is cohesive and consistent
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control

Finally Java is to Internet Programming where C was to System Programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. In the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Applications and applets.

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one created using C or C++ .Java's ability to create Applets makes it important. An Applet is an application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

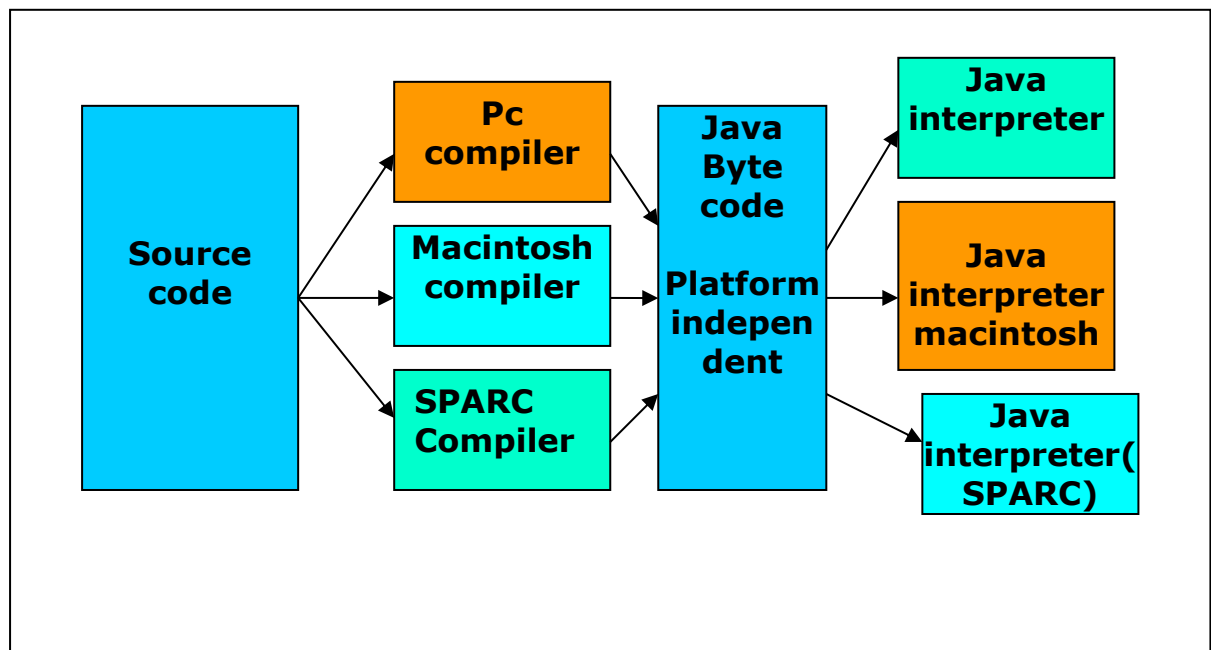
Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

Compiling and interpreting java source code.



During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

Simple:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer. Learning Java will oriented features of C++ . Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

Object oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank state. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

Robust

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs. Was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and runtime.

Java virtually eliminates the problems of memory management and deal location, which is completely automatic. In a well-written Java program, all run-time errors can and should be managed by your program.

Servlets/JSP

INTRODUCTION

A Servlet Is a generic server extension. a Java class that can be loaded

Dynamically to expand the functionality of a server. Servlets are commonly used with web servers. Where they can take the place CGI scripts.

A servlet is similar to proprietary server extension, except that it runs inside a Java Virtual Machine (JVM) on the server, so it is safe and portable

Servlets operate solely within the domain of the server.

Unlike CGI and Fast CGI, which use multiple processes to handle separate program or separate requests, separate threads within web server process handle all servlets. This means that servlets are all efficient and scalable.

Servlets are portable; both across operating systems and also across web servers. Java Servlets offer the best possible platform for web application development.

Servlets are used as replacement for CGI scripts on a web server, they can extend any sort of server such as a mail server that allows servlets to extend its functionality perhaps by performing a virus scan on all attached documents or handling mail filtering tasks.

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming including inextensible scripting solutions platform-specific API's and incomplete interface.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side. What applets are to the client-side are object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, pluggable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example an HTTP servlet can be used to generate dynamic HTML content when you use servlets to do dynamic content you get the following advantages:

- They're faster and cleaner than CGI scripts
- They use a standard API (the servlet API)
- They provide all the advantages of Java (run on a variety of servers without needing to be rewritten)

Attractiveness of Servlets:

They are many features of servlets that make them easy and attractive to use these include:

- Easily configure using the GUI-based Admin tool]
- Can be Loaded and Invoked from a local disk or remotely across the network.
- Can be linked together or chained, so that one servlet can call another servlet, or several servlets in sequence.
- Can be called dynamically from within HTML, pages using server-side include-tags.
- Are secure-even when downloading across the network, the servlet security model and servlet and box protect your system from unfriendly behavior.,

Advantages of the servlet API

One of the great advantages of the servlet API is protocol independent. It assumes nothing about:

- The protocol being used to transmit on the net
- How it is loaded
- The server environment it will be running in
- These quantities are important, because it allows the Servlet API to be embedded in many different kinds of servers. There are other advantages to the servlet API as well These include:
- It's extensible-you can inherit all your functionality from the base classes made available to you
- It's simple small, and easy to use.

Features of Servlets:

- © Servlets are persistent. Servlet are loaded only by the web server and can maintain services between requests.

- Ⓢ Servlets are fast. Since servlets only need to be loaded once, they offer much better performance over their CGI counterparts.
- Ⓢ Servlets are platform independent.
- Ⓢ Servlets are extensible Java is a robust, object-oriented programming language, which easily can be extended to suit your needs.
- Ⓢ Servlets are secure
- Ⓢ Servlets are used with a variety of client.

Servlets are classes and interfaces from two packages, `javax.servlet` and `javax.servlet.http`. The `javax.servlet` package contains classes to support generic, protocol-independent servlets. The classes in the `javax.servlet.http` package to and HTTP specific functionality extend these classes

Every servlet must implement the `javax.servlet` interface. Most servlets implement it by extending one of two classes, `javax.servlet.GenericServlet` or `javax.servlet.http.HttpServlet`. A protocol-independent servlet should subclass `GenericServlet`, while an HTTP servlet should subclass `HttpServlet`, which is itself a subclass of `GenericServlet` with added HTTP-specific functionality.

Unlike a Java program, a servlet does not have a `main()` method. Instead, the server in the process of handling requests invokes certain methods of a servlet. Each time the server dispatches a request to a servlet, it invokes the servlet's `service()` method,

A generic servlet should override its `service()` method to handle requests as appropriate for the servlet. The `service()` accepts two parameters: a request object and a response object. The request object tells the servlet about the request, while the response object is used to return a response.

In contrast, an HTTP servlet usually does not override the `service()` method. Instead, it overrides `doGet()` to handle GET requests and `doPost()` to handle POST requests. An HTTP servlet can override either or both of these methods. The `service()` method of `HttpServlet` handles the setup and dispatching to all the `doXXX()` methods, which is why it usually should not be overridden.

The remainders in the `javax.servlet` and `javax.servlet.http` package are largely support classes. The `ServletRequest` and `ServletResponse` classes in `javax.servlet` provide access to generic server requests and responses while `HttpServletRequest` and `HttpServletResponse` classes in `javax.servlet` provide access to generic server requests and responses while `HttpServletRequest` and `HttpServletResponse` in `javax.servlet.http` provide access a HTTP requests and responses. The `javax.servlet.http` provide contains an `HttpSession` class that provides built-in session tracking functionality and `Cookie` class that allows quickly setup and processing `HttpCookies`.

Loading Servlets:

Servlets can be loaded from their places. From a directory that is on the CLASSPATH. The CLASSPATH of the JavaWebServer includes `service root/classes/`, which is where the system classes reside

From the `<SERVICE_ROOT/servlets/directory`. This is not in the server's classpath. A class loader is used to create servlets from this directory. New servlets can be added-existing servlets can be recompiled and the server will notice these changes. From a remote location. For this a code base like <http://nine.eng/classes/foo/> is required in addition to the servlet's class name. Refer to the admin Gui docs on servlet section to see how to set this up.

Loading Remote Servlets

Remote servlets can be loaded by:

- Configuring the admin Tool to setup automatic loading of remote servlets.
- Selectiong up server side include tags in .html files
- Defining a filter chain Configuration

Invoking Servlets

A servlet invoker is a servlet that invokes the “server” method on a named servlet. If the servlet is not loaded in the server, then the invoker first loads the servlet (either from local disk or from the network) and then invokes the “service” method. Also like applets, local servlets in the server can be identified by just the class name. In other words, if a servlet name is not absolute, it is treated as local.

A Client can Invoke Servlets in the Following Ways:

- The client can ask for a document that is served by the servlet.
- The client (browser) can invoke the servlet directly using a URL, once it has been mapped using the SERVLET ALIASES Section of the admin GUI
- The servlet can be invoked through server side include tags.
- The servlet can be invoked by placing it in the servlets/directory
- The servlet can be invoked by using it in a filter chain

The Servlet Life Cycle:-

The Servlet life cycle is one of the most exciting features of Servlets. This life cycle is a powerful hybrid of the life cycles used in CGI programming and lower-level NSAPI and ISAPI programming.

The servlet life cycle allows servlet engines to address both the performance and resource problems of CGI and the security concerns of low level server API programming.

Servlet life cycle is highly flexible. Servers have significant leeway in how they choose to support servlets. The only hard and fast rule is that a servlet engine must conform to the following life cycle contract:

- Create and initialize the servlets
- Handle zero or more service from clients
- Destroy the servlet and then garbage collects it.

It's perfectly legal for a servlet to be loaded, created and initialized in its own JVM, only to be destroyed and garbage collected without handling any client request or after handling just one request.

The most common and most sensible life cycle implementations for HTTP servlets are:

Single java virtual machine and stateless persistence.

Init and Destroy:-

Just like Applets servlets can define `init()` and `destroy()` methods. A servlet's `init(ServletConfig)` method is called by the server immediately after the server constructs the servlet's instance. Depending on the server and its configuration, this can be at any of these times

- ⌚ When the server starts
- ⌚ When the servlet is first requested, just before the `service()` method is invoked
- ⌚ At the request of the server administrator

In any case, `init()` is guaranteed to be called before the servlet handles its first request

The `init()` method is typically used to perform servlet initialization creating or loading objects that are used by the servlet in handling of its request. In order to providing a new servlet any information about itself and its environment, a server has to call a servlet's `init()` method and pass an object that implements the `ServletConfig` interface.

This `ServletConfig` object supplies a servlet with information about its initialization parameters. These parameters are given to the servlets and are not associated with any single request. They can specify initial values, such as where a counter should begin counting, or default values, perhaps a template to use when not specified by the request,

The server calls a servlet's `destroy()` method when the servlet is about to be unloaded. In the `destroy()` method, a servlet should free any resources it has acquired that will not be garbage collected. The `destroy()` method also gives a servlet a chance to write out its unsaved, cached information or any persistent information that should be read during the next call to `init()`.

Session Tracking:

HTTP is a stateless protocol, it provides no way for a server to recognize that a sequence of requests is all from the same client. This causes a problem for application such as shopping cart applications. Even in chat application server can't know exactly who's making a request of several clients.

The solution for this is for client to introduce itself as it makes each request, Each clients needs to provide a unique identifier that lets the server identify it, or it needs to give some information that the server can use to properly handle the request, There are several ways to send this introductory information with each request Such as:

USER AUTHORIZATION:

One way to perform session tracking is to leverage the information that comes with User authorization. When a web server restricts access to some of its resources to only those clients that log in using a recognized username and password. After the client logs in, the username is available to a servlet through `getRemoteUser ()`

We can use the username to track the session. Once a user has logged in, the browser remembers her username and resends the name and password as the user views new pages on the site. A servlet can identify the user through her username and they're by Track her session.

The biggest advantage of using user authorization to perform session tracking is that it's easy to implement. Simply tell the protect a set of pages, and use `getRemoteUser()` to identify each client. Another advantage is that the technique works even when the user accesses your site from or exists her browser before coming back.

The biggest disadvantage of user authorization is that it requires each user to register for an account and then log in in each time the starts visiting your site. Most users will tolerate registering and lagging in as a necessary evil when they are accessing sensitive information, but its all overkill for simple session tracking. Other problem with user

authorization is that a user cannot simultaneously maintain more than one session at the same site.

Hidden Form Fields:

One way to support anonymous session tracking is to use hidden form fields. As the name implies, these are fields added to an HTML form that are not displayed in the client's browser. They are sent back to the server when the form that contains them is submitted.

In a sense, hidden form fields define constant variables for a form. To a servlet receiving a submitted form, there is no difference between a hidden field and a visible field.

As more and more information is associated with a client's session. It can become burdensome to pass it all using hidden form fields. In these situations it's possible to pass on just a unique session ID that identifies a particular client's session.

That session ID can be associated with complete information about its session that is stored on the server.

The advantage of hidden form fields is their ubiquity and support for anonymity. Hidden fields are supported in all the popular browsers, they demand no special server requirements, and they can be used with clients that haven't registered or logged in.

The major disadvantage with this technique, however, is that it works only for a sequence of dynamically generated forms. The technique breaks down immediately with static documents, emailed documents, bookmarked documents, and browser shutdowns.

URL Rewriting:

URL rewriting is another way to support anonymous session tracking. With URL rewriting every local URL the user might click on is dynamically modified, or rewritten, to include extra information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific URL change. Due to the limited space available in rewriting a URL, the extra information is usually limited to a unique session.

Each rewriting technique has its own advantage and disadvantage.

Using extra path information works on all servers, and it works as a target for forms that use both the Get and Post methods. It does not work well if the servlet has to use the extra path information as true path information.

The advantages and disadvantages of URL rewriting closely match those of hidden form fields. The major difference is that URL rewriting works for all dynamically created documents, such as the Help servlet, not just forms. With the right server support, custom URL rewriting can even work for static documents.

Persistent Cookies:

A fourth technique to perform session tracking involves persistent cookies. A cookie is a bit of information sent by a web server to a browser that can later be read back from that browser. When a browser receives a cookie, it saves the cookie and then sends the cookie back to the server each time it accesses a page on that server, subject to certain rules. Because a cookie's value can uniquely identify a client, cookies are often used for session tracking.

Persistent cookies offer an elegant, efficient easy way to implement session tracking. Cookies provide as automatic an introduction for each request as we could hope for. For each request, a cookie can automatically provide a client's session ID or perhaps a list of client's performance. The ability to customize cookies gives them extra power and versatility.

The biggest problem with cookies is that browsers don't always accept cookies. Sometimes this is because the browser doesn't support cookies. More often it's because

The browser doesn't support cookies. More often it's because the user has specifically configured the browser to refuse cookies.

The power of servlets:

The power of servlets is nothing but the advantages of servlets over other approaches, which include portability, power, efficiency, endurance, safety elegance, integration, extensibility and flexibility.

Portability:

As servlets are written in Java and conform to a well defined and widely accepted API, they are highly portable across operating systems and across server implementation.

We can develop a servlet on a windows NT machine running the java web server and later deploy it effortlessly on a high-end Unix server running apache. With servlets we can really “write once, serve every where”

Servlet portability is not the stumbling block it so often is with applets, for two reasons

First,Servlet portability is not mandatory i.e. servlets has to work only on server machines that we are using for development and deployment

Second, servlets avoid the most error-prone and inconstancy implemented portion of the java languages.

Power:

Servlets can harness the full power of the core java. API's: such as Networking and Url access, multithreading, image manipulation, data compression, data base connectivity, internationalization, remote method invocation(RMI) CORBA connectivity, and object serialization, among others,

Efficiency And Endurance:

Servlet invocation is highly efficient, Once a servlet is loaded it generally remains in the server's memory as a single object instance, There after the server invokes the servelt to handle a request using a simple, light weighted method invocation .Unlike the CGI, there's no process to spawn or interpreter to invoke, so the servlet can begin handling the request almost immediately, Multiple, concurrent requests are handled the request almost immediately. Multiple, concurrent requests are handled by separate threads, so servlets are highly scalable.

Servlets in general are enduring objects. Because a servlets stays in the server's memory as a single object instance. it automatically maintains its state and can hold onto external resources, such as database connections.

Safety:

Servlets support safe programming practices on a number of levels.

As they are written in java, servlets inherit the strong type safety of the java language. In addition the servlet API is implemented to be type safe. Java's automatic garbage collection and lack of pointers mean that servlets are generally safe from memory management problems like dangling pointers invalid pointer references and memory leaks.

Servlets can handle errors safely, due to java's exception – handling mechanism. If a servlet divides by zero or performs some illegal operations, it throws an exception that can be safely caught and handled by the server.

A server can further protect itself from servlets through the use of java security manager. A server can execute its servlets under the watch of a strict security manager.

Elegance:

The elegance of the servlet code is striking. Servlet code is clean, object oriented modular and amazingly simple one reason for this simplicity is the served API itself. Which includes methods and classes to handle many of the routine chores of servlet development. Even advanced to operations like cookie handling and session tracking tracking are abstracted into convenient classes.

Integration:

Servlets are tightly integrated with the server. This integration allows a servlet to cooperate with the server in two ways. for e.g.: a servlet can use the server to translate file paths, perform logging, check authorization, perform MIME type mapping and in some cases even add users to the server's user database.

Extensibility and Flexibility:

The servlet API is designed to be easily extensible. As it stands today the API includes classes that are optimized for HTTP servlets. But later it can be extended and

optimized for another type of servlets. It is also possible that its support for HTTP servlets could be further enhanced.

Servlets are also quite flexible, Sun also introduced java server pages. which offer a way to write snippets of servlet code directly within a static HTML page using syntax similar to Microsoft's Active server pages(ASP)

9.2 JDBC

What is JDBC?

any relational database. One can write a single program using the JDBC API, and the JDBC is a Java API for executing SQL statements. (As a point of interest JDBC is a trademarked name and is not an acronym; nevertheless, Jdbc is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java Programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API

Using JDBC, it is easy to send SQL statements to virtually any program will be able to send SQL statements to the appropriate database. The combination of Java and JDBC lets a programmer write it once and run it anywhere.

What Does JDBC Do?

Simply put, JDBC makes it possible to do three things

- Establish a connection with a database
- Send SQL statements
- Process the results
- JDBC Driver Types
- The JDBC drivers that we are aware of this time fit into one of four categories
- JDBC-ODBC Bridge plus ODBC driver
- Native-API party-java driver
- JDBC-Net pure java driver
- Native-protocol pure Java driver

An individual database system is accessed via a specific JDBC driver that implements the `java.sql.Driver` interface. Drivers exist for nearly all-popular RDBMS systems, though few are available for free. Sun bundles a free JDBC-ODBC bridge driver with the JDK to allow access to a standard ODBC data sources, such as a Microsoft Access database, Sun advises against using the bridge driver for anything other than development and very limited development.

JDBC drivers are available for most database platforms, from a number of vendors and in a number of different flavours. There are four driver categories

Type 01-JDBC-ODBC Bridge Driver

Type 01 drivers use a bridge technology to connect a java client to an ODBC database service. Sun's JDBC-ODBC bridge is the most common type 01 driver. These drivers implemented using native code.

Type 02-Native-API party-java Driver

Type 02 drivers wrap a thin layer of java around database-specific native code libraries for Oracle databases, the native code libraries might be based on the OCI(Oracle call Interface) libraries, which were originally designed for `c/c++` programmers, Because type-02 drivers are implemented using native code. in some cases they have better performance than their all-java counter parts. They add an element of risk, however, because a defect in a driver's native code section can crash the entire server

Type 03-Net-Protocol All-Java Driver

Type 03 drivers communicate via a generic network protocol to a piece of custom middleware. The middleware component might use any type of driver to provide the actual database access. These drivers are all java, which makes them useful for applet deployment and safe for servlet deployment

Type-04-native-protocol All-java Driver

Type 04 drivers are the most direct of the lot. Written entirely in java, Type 04 drivers understand database-specific networking. protocols and can access the database directly without any additional software

JDBC-ODBC Bridge

If possible use a Pure Java JDBC driver instead of the Bridge and an ODBC driver. This completely eliminates the client configuration required by ODBC. It also eliminates the potential that the Java VM could be corrupted by an error in the native code brought in by the Bridge (that is, the Bridge native library, the ODBC driver manager library, library, the ODBC driver library, and the database client library)

WHAT IS The JDBC-ODBE Bridge ?

The JDBC-ODBC Bridge is a Jdbc driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge is implemented as the sun.jdbc.odbc Java package and contains a native library used to access ODBC. The Bridge is joint development of Intersolv and Java Soft

9.3 Oracle

Oracle is a relational database management system, which organizes data in the form of tables. Oracle is one of many database servers based on RDBMS model, which manages a seer of data that attends three specific things-data structures, data integrity and data manipulation.

With oracle cooperative server technology we can realize the benefits of open, relational systems for all the applications. Oracle makes efficient use of all systems resources, on all hardware architecture; to deliver unmatched performance, price performance and scalability. Any DBMS to be called as RDBMS has to satisfy Dr.E.F.Codd's rules.

Features of Oracle:

Portable

The Oracle RDBMS is available on wide range of platforms ranging from PCs to super computers and as a multi user loadable module for Novel NetWare, if you develop application on system you can run the same application on other systems without any modifications.

Compatible

Oracle commands can be used for communicating with IBM DB2 mainframe RDBMS that is different from Oracle, which is Oracle compatible with DB2. Oracle RDBMS

is a high performance fault tolerant DBMS, which is specially designed for online transaction processing and for handling large database applications.

Multithreaded Server Architecture

Oracle adaptable multithreaded server architecture delivers scalable high performance for very large number of users on all hardware architecture including symmetric multiprocessors (sumps) and loosely coupled multiprocessors. Performance is achieved by eliminating CPU, I/O, memory and operating system bottlenecks and by optimizing the Oracle DBMS server code to eliminate all internal bottlenecks.

Oracle has become the most popular RDBMS in the market because of its ease of use

- Client/server architecture.
- Data independence.
- Ensuring data integrity and data security.
- Managing data concurrency.
- Parallel processing support for speed up data entry and online transaction processing used for applications.
- DB procedures, functions and packages.

Dr.E.F.Codd's Rules

These rules are used for valuating a product to be called as relational database management systems. Out of 12 rules, a RDBMS product should satisfy at least 8 rules + rule called rule 0 that must be satisfied.

RULE 0: Foundation Rule

For any system to be advertised as, or claimed to be relational DBMS should manage database with in it self, with out using an external language.

RULE 1: Information Rule

All information in relational database is represented at logical level in only one way as values in tables.

RULE 2: Guaranteed Access

Each and every data in a relational database is guaranteed to be logically accessibility by using to a combination of table name, primary key value and column name.

RULE 3: Systematic Treatment of Null Values

Null values are supported for representing missing information and inapplicable information. They must be handled in systematic way, independent of data types.

RULE 4: Dynamic Online Catalog based Relation Model

The database description is represented at the logical level in the same way as ordinary data so that authorized users can apply the same relational language to its interrogation as they do to the regular data.

RULE 5: Comprehensive Data Sub Language

A relational system may support several languages and various models of terminal use. However there must be one language whose statement can express all of the following:

Data Definitions, View Definitions, Data Manipulations, Integrity, Constraints, Authorization and transaction boundaries.

RULE 6: View Updating

Any view that is theoretical can be updatable if changes can be made to the tables that effect the desired changes in the view.

RULE 7: High level Update, Insert and Delete

The capability of handling a base relational or derived relational as a single operand applies not only retrieval of data also to its insertion, updating, and deletion.

RULE 8: Physical Data Independence

Application program and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access method.

RULE 9: Logical Data Independence

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.

RULE 10: Integrity Independence

Integrity constraints specific to particular database must be definable in the relational data stored in the catalog, not in application program.

RULE 11: Distributed Independence

Whether or not a system supports database distribution, it must have a data sub-language that can support distributed databases without changing the application program.

RULE 12: Non Sub-Version

If a relational system has low level language, that low language cannot use to subversion or by pass the integrity rules and constraints expressed in the higher level relational language.

Oracle supports the following Codd's Rules

- Rule 1: Information Rule (Representation of information)-YES.
- Rule 2: Guaranteed Access-YES.
- Rule 3: Systematic treatment of Null values-YES.
- Rule 4: Dynamic on-line catalog-based Relational Model-YES.
- Rule 5: Comprehensive data sub language-YES.
- Rule 6: View Updating-PARTIAL.
- Rule 7: High-level Update, Insert and Delete-YES.
- Rule 8: Physical data Independence-PARTIAL.
- Rule 9: Logical data Independence-PARTIAL.
- Rule 10: Integrity Independence-PARTIAL.
- Rule 11: Distributed Independence-YES.
- Rule 12: Non-subversion-YES.

9.3.1 HTML

Hypertext Markup Language (HTML), the languages of the world wide web(WWW), allows users to produces web pages that included text, graphics and pointer to other web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML(Standard Generalized Markup Language),but

Specialized to hypertext and adapted to the Web. The idea behind Hypertext one point to another point. We can navigate through the information based on out interest and preference. A markup language is simply a series of items enclosed within the elements should be displayed.

Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

Html can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop

HTML provides tags(special codes) to make the document look attractive.

HTML provides are not case-sensitive. Using graphics,fonts,different sizes, color, etc.. can enhance the presentation of the document. Anything

That is not a tag is part of the document it self.

Basic Html Tags:

<code><!-- --></code>	Specific Comments.
<code><A>.....</code>	Creates Hypertext links.
<code>.....</code>	Creates hypertext links.
<code><Big>.....</Big></code>	Formats text in large-font
<code><Body>.....</Body></code>	contains all tags and text in the Html-document
<code><Center>.....</Center></code>	Creates Text
<code><DD>.....</DD></code>	Definition of a term.
<code><TABLE>.....</TABLE></code>	creates table
<code><Td>.....</Td></code>	indicates table data in a table.

<Tr>.....</Tr>	designates a table row
<Th>.....</Th>	creates a heading in a table.

ADVANTAGES:-

- ▶ A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
- ▶ HTML is platform independent

HTML tags are not case-sensitive.

JAVA SCRIPT

The Java Script Language

JavaScript is a compact, object-based scripting language for developing client and server internet applications. Netscape Navigator 2.0 interprets JavaScript statements embedded directly in an HTML page. and Livewire enables you to create server-based applications similar to common gateway interface(cgi) programs.

In a client application for Navigator, JavaScript statements embedded in an HTML Page can recognize and respond to user events such as mouse clicks form Input, and page navigation.

For example, you can write a JavaScript function to verify that users enter valid information into a form requesting a telephone number or zip code. Without any network transmission, an Html page with embedded Java Script can interpret the entered text and alert the user with a message dialog if the input is invalid or you can use JavaScript to perform an action (such as play an audio file, execute an applet, or communicate with a plug-in) in response to the user opening or exiting a page.

9.4 Source Code

9.4.1 User

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package order_management_system;

import com.database.Queries;
import java.sql.ResultSet;
import javax.swing.JOptionPane;

/**
 *
 * @author KishanVenky
 */
public class User extends javax.swing.JFrame {

    /**
     * Creates new form User
     */
    public User() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    username = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    password = new javax.swing.JPasswordField();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jPanel1.setBackground(new java.awt.Color(102, 102, 0));

    jLabel1.setFont(new java.awt.Font("Times New Roman", 3, 30)); // NOI18N
    jLabel1.setForeground(new java.awt.Color(255, 255, 255));
    jLabel1.setText("ORDER MANAGEMENT SYSTEM");

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(

        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.Alignment.TRAILING,
                jPanel1Layout.createSequentialGroup()
                    .addGap(128, 128, 128)
                    .addComponent(jLabel1)
                    .addGap(128, 128, 128))
    );
}
```

```
);  
jPanel1Layout.setVerticalGroup(
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
        .addGap(28, 28, 28)  
        .addComponent(jLabel1)  
        .addContainerGap(20, Short.MAX_VALUE))  
    );
```

```
jLabel2.setFont(new java.awt.Font("Tahoma", 3, 18)); // NOI18N  
jLabel2.setForeground(new java.awt.Color(153, 51, 0));  
jLabel2.setText("USER LOGIN HERE ");
```

```
jLabel3.setBackground(new java.awt.Color(0, 102, 102));  
jLabel3.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N  
jLabel3.setForeground(new java.awt.Color(0, 153, 153));  
jLabel3.setText("Mobile :");
```

```
username.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        usernameActionPerformed(evt);  
    }  
});
```

```
jLabel4.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N  
jLabel4.setForeground(new java.awt.Color(0, 153, 153));  
jLabel4.setText("Password :");
```

```
jButton1.setText("Login");  
jButton1.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton1ActionPerformed(evt);  
    }  
});
```



```

});

jButton2.setText("Reset");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jLabel5.setFont(new java.awt.Font("Tahoma", 0, 13)); // NOI18N
jLabel5.setForeground(new java.awt.Color(153, 0, 0));
jLabel5.setText("Don't Have An Account ?");

jLabel6.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
jLabel6.setForeground(new java.awt.Color(102, 102, 0));
jLabel6.setText("Register");
jLabel6.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jLabel6MouseClicked(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel5)

```

```

        .addGap(18, 18, 18)
        .addComponent(jLabel6)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup())

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel3)
            .addComponent(jLabel4))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,      36,
Short.MAX_VALUE)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
            .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(username)
            .addComponent(password))
            .addGroup(layout.createSequentialGroup()
                .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE,      78,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton2)))
            .addGap(268, 268, 268))))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()

```

```

        .addComponent(jPanel1,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(jLabel2,    javax.swing.GroupLayout.PREFERRED_SIZE,    62,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(34, 34, 34)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel3)
        .addComponent(username,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(41, 41, 41)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel4)
        .addComponent(password,    javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(33, 33, 33)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jButton1)
        .addComponent(jButton2))
        .addGap(27, 27, 27)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel5)
        .addComponent(jLabel6))
        .addGap(0, 81, Short.MAX_VALUE))
);

pack();

```

```
}// </editor-fold>
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    String UserName=username.getText();
    String Password=password.getText();

    if(UserName.length()<=0 || UserName==null){
        JOptionPane.showMessageDialog(null,"UserName field should not be empty");
    }
    if(Password.length()<=0 || Password==null){
        JOptionPane.showMessageDialog(null,"Password field should not be empty");
    }

    try{
        String query="select * from register where mobile='"+UserName+"'and
password='"+Password+"'";
        ResultSet r=Queries.getExecuteQuery(query);
        if(r.next()){
            String UName=r.getString("username");
            JOptionPane.showMessageDialog(null,"Login Successfull...!!!");
            UserHome oh=new UserHome();
            oh.setVisible(true);
            oh.getMobile(UserName,UName);
            dispose();
        }else{
            JOptionPane.showMessageDialog(null,"Username/Password Failed");
        }
    }catch(Exception e){
        System.out.println(e);
    }
}
```

```

}

private void usernameActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    username.setText("");
    password.setText("");
}

private void jLabel6MouseClicked(java.awt.event.MouseEvent evt) {
    OwnerRegister ow=new OwnerRegister();
    ow.setVisible(true);
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
    *
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
        javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }
}

```

```

        }
    }
} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new User().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;

```

```

private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JPanel jPanel1;
public javax.swing.JPasswordField password;
public javax.swing.JTextField username;
// End of variables declaration
}

```

9.4.2 Admin

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package order_management_system;

import com.database.Queries;
import java.sql.ResultSet;
import javax.swing.JOptionPane;

/**
 *
 * @author KishanVenky
 */
public class Owner extends javax.swing.JFrame {

    /**
     * Creates new form Owner
     */
    public Owner() {
        initComponents();
        setTitle("Admin LOGIN PAGE");
    }
}

```

```

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jDesktopPane1 = new javax.swing.JDesktopPane();
    jPanel1 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    mobile = new javax.swing.JLabel();
    username = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    Login = new javax.swing.JButton();
    password = new javax.swing.JPasswordField();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jPanel1.setBackground(new java.awt.Color(102, 102, 0));

    jLabel1.setFont(new java.awt.Font("Times New Roman", 3, 36)); // NOI18N
    jLabel1.setForeground(new java.awt.Color(255, 255, 255));
    jLabel1.setText("ORDER MANAGEMENT SYSTEM");

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addContainerGap(100, Short.MAX_VALUE))
    );
}

```



```

        .addGap(71, 71, 71)
        .addComponent(jLabel1)
        .addContainerGap(68, Short.MAX_VALUE))
    );

    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(33, 33, 33)
            .addComponent(jLabel1)
            .addContainerGap(23, Short.MAX_VALUE))
        );

    jLabel2.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N
    jLabel2.setForeground(new java.awt.Color(255, 255, 255));
    jLabel2.setText("ADMIN LOGIN");
    jLabel2.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    mobile.setFont(new java.awt.Font("Tahoma", 2, 20)); // NOI18N
    mobile.setForeground(new java.awt.Color(0, 204, 255));
    mobile.setText("User Name :");

    jLabel4.setFont(new java.awt.Font("Tahoma", 2, 20)); // NOI18N
    jLabel4.setForeground(new java.awt.Color(0, 204, 255));
    jLabel4.setText("Password :");

    Login.setBackground(new java.awt.Color(0, 204, 204));
    Login.setText("LOGIN");
    Login.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            LoginActionPerformed(evt);
        }
    });

```

CHAPTER-10

RESULTS/DISCUSSIONS

10.1 System Test

10.1.1 INTRODUCTION TO TESTING

Introduction to Testing:

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

10.1.2 TESTING IN STRATEGIES

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:






Unit Testing:

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

Each module can be tested using the following two Strategies:

Black Box Testing:

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been used to find errors in the following categories:

-  Incorrect or missing functions
-  Interface errors
-  Errors in data structure or external database access
-  Performance errors
-  Initialization and termination errors.

In this testing only the output is checked for correctness.

The logical flow of the data is not checked.

White Box testing :

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been used to generate the test cases in the following cases:

- ✓ Guarantee that all independent paths have been Executed.
- ✓ Execute all logical decisions on their true and false Sides.
- ✓ Execute all loops at their boundaries and within their operational bounds
- ✓ Execute internal data structures to ensure their validity.

Integrating Testing :

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

System Testing:

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

Acceptance Testing :

It is a pre-delivery testing in which the entire system is tested at the client's site on real world data to find errors.

Test Approach :

Testing can be done in two ways:

- Bottom up approach
- Top down approach

Bottom up Approach:

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

Top down approach:

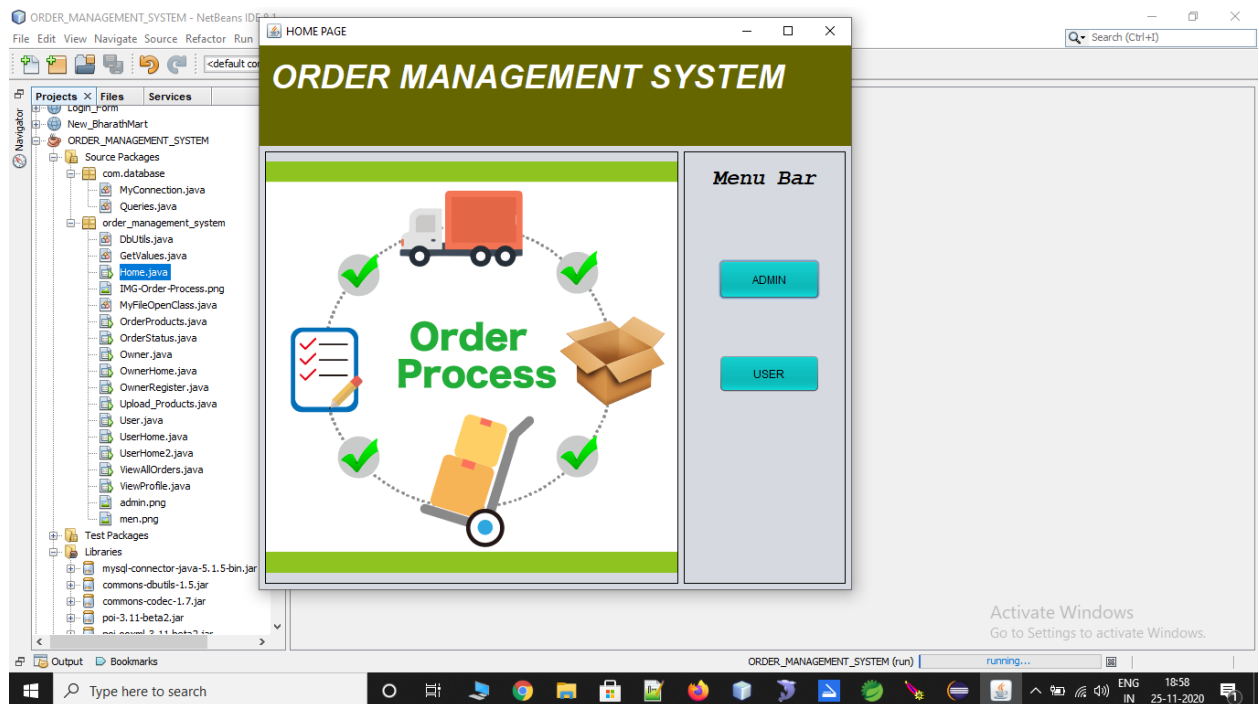
This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

Validation:

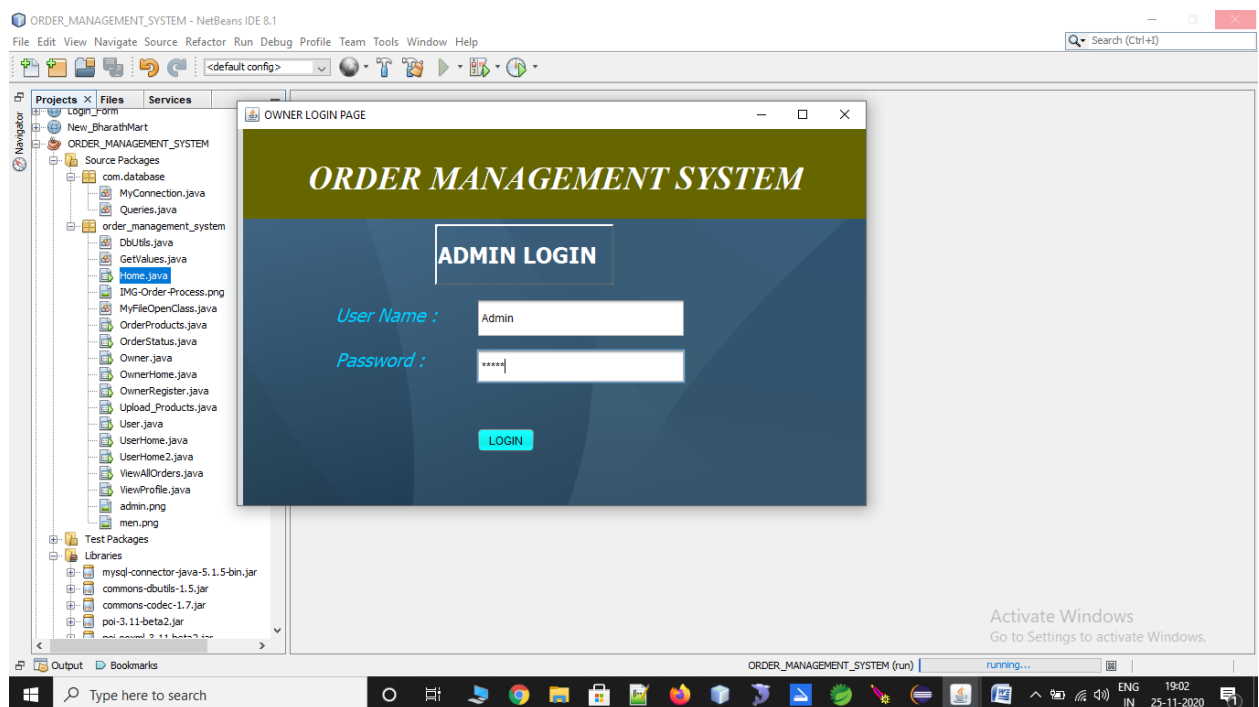
The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed

10.2 OUTPUT SCREENS

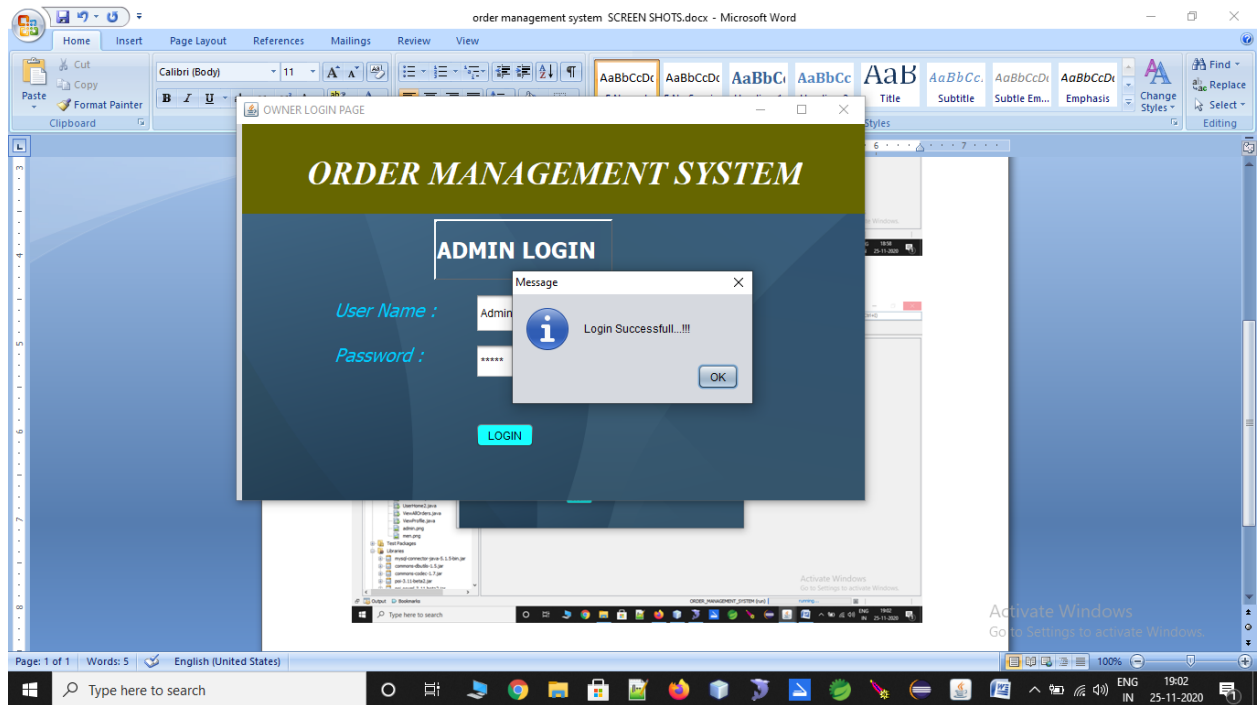
HOME SCREEN



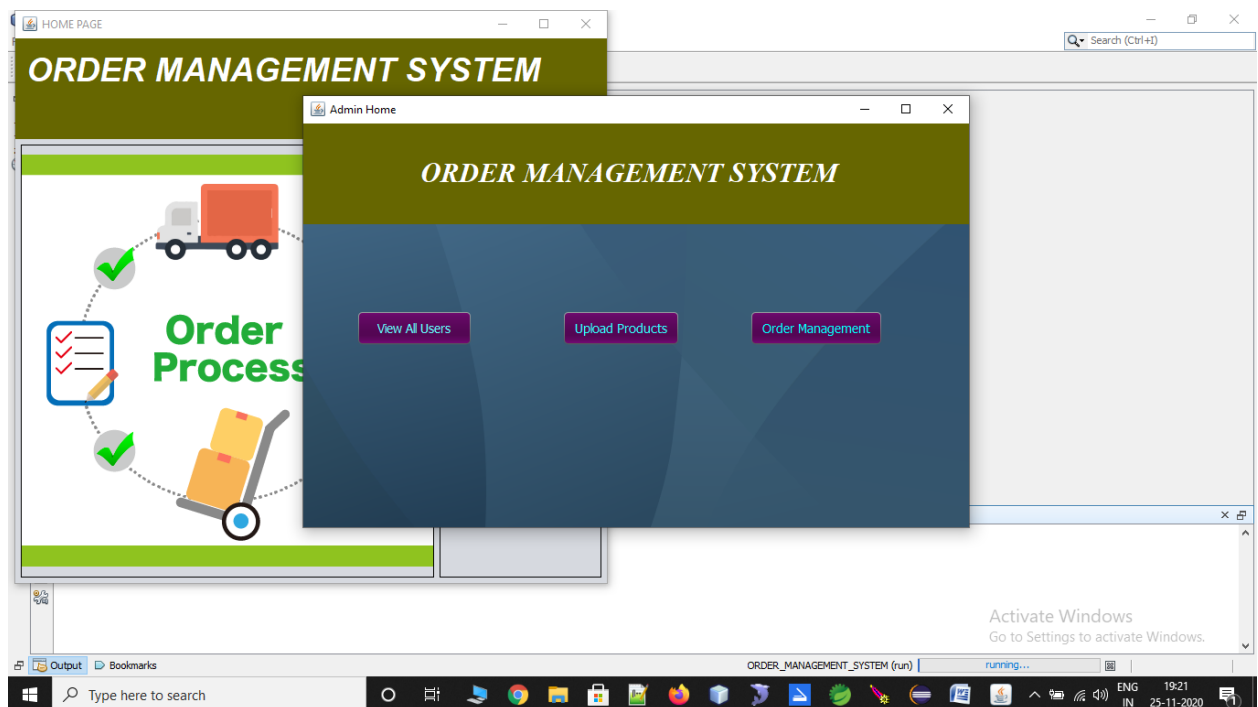
Admin login Screen



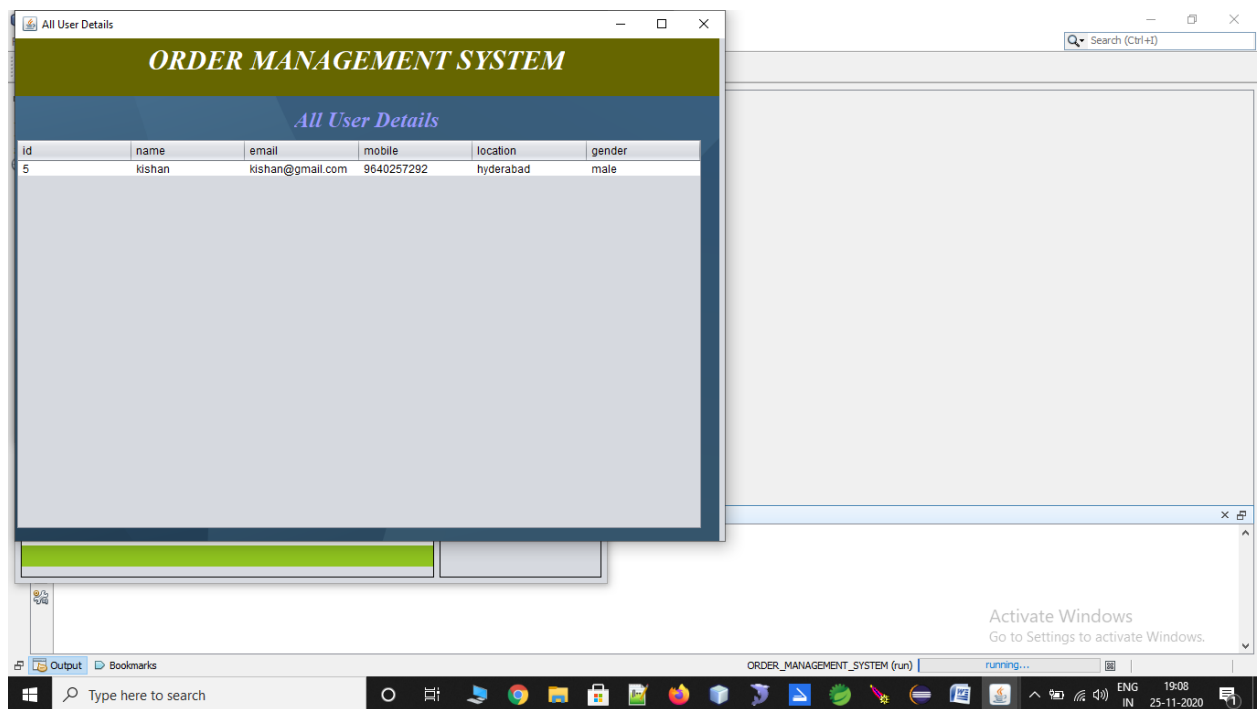
Login status message



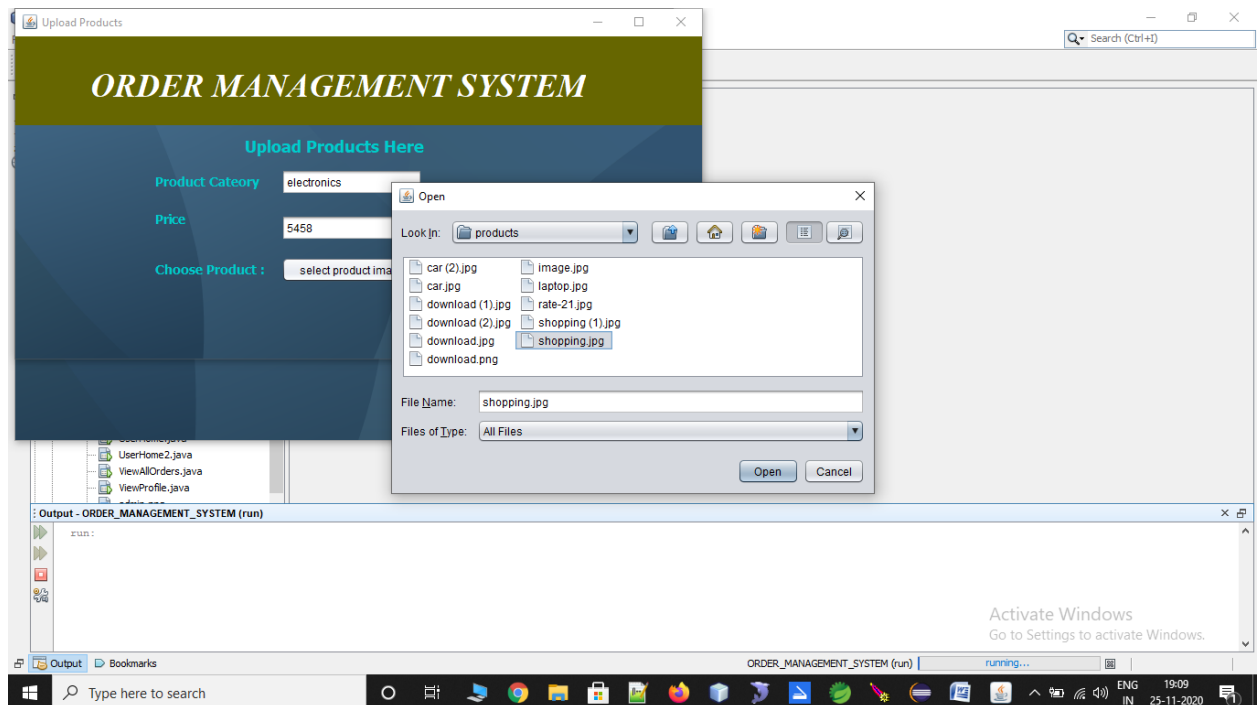
Admin Home page



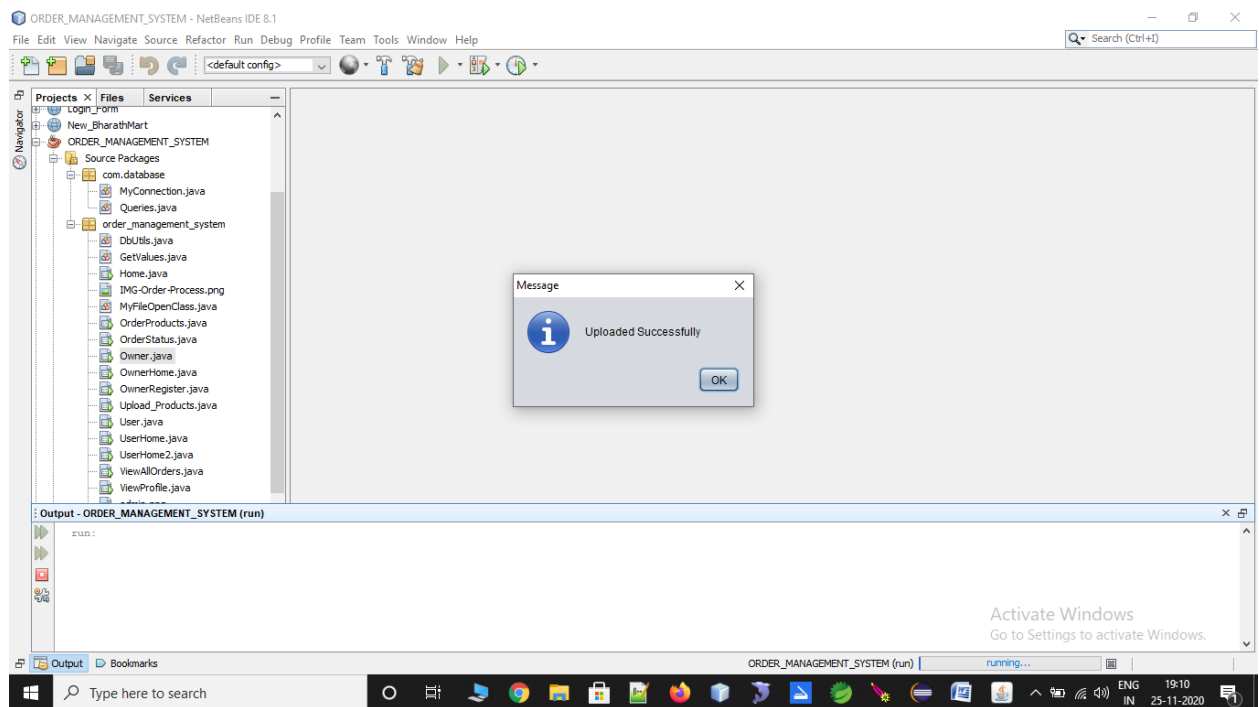
All users Details



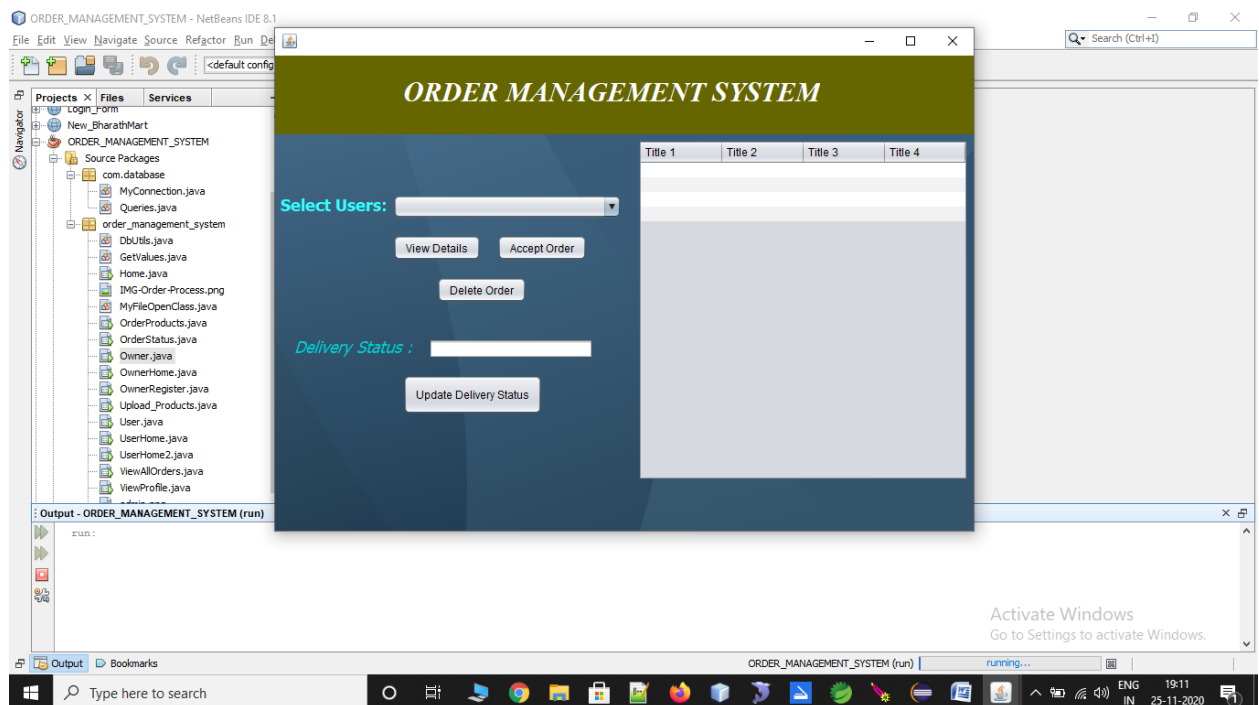
Upload products



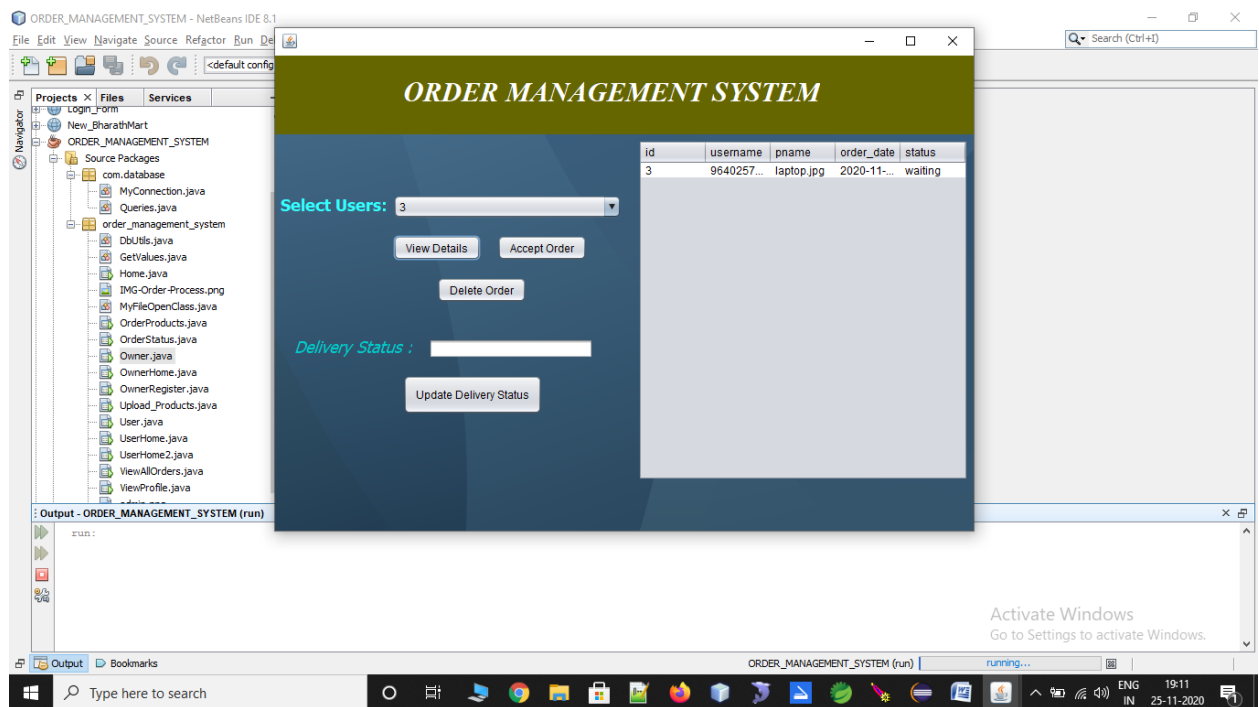
Upload products status:



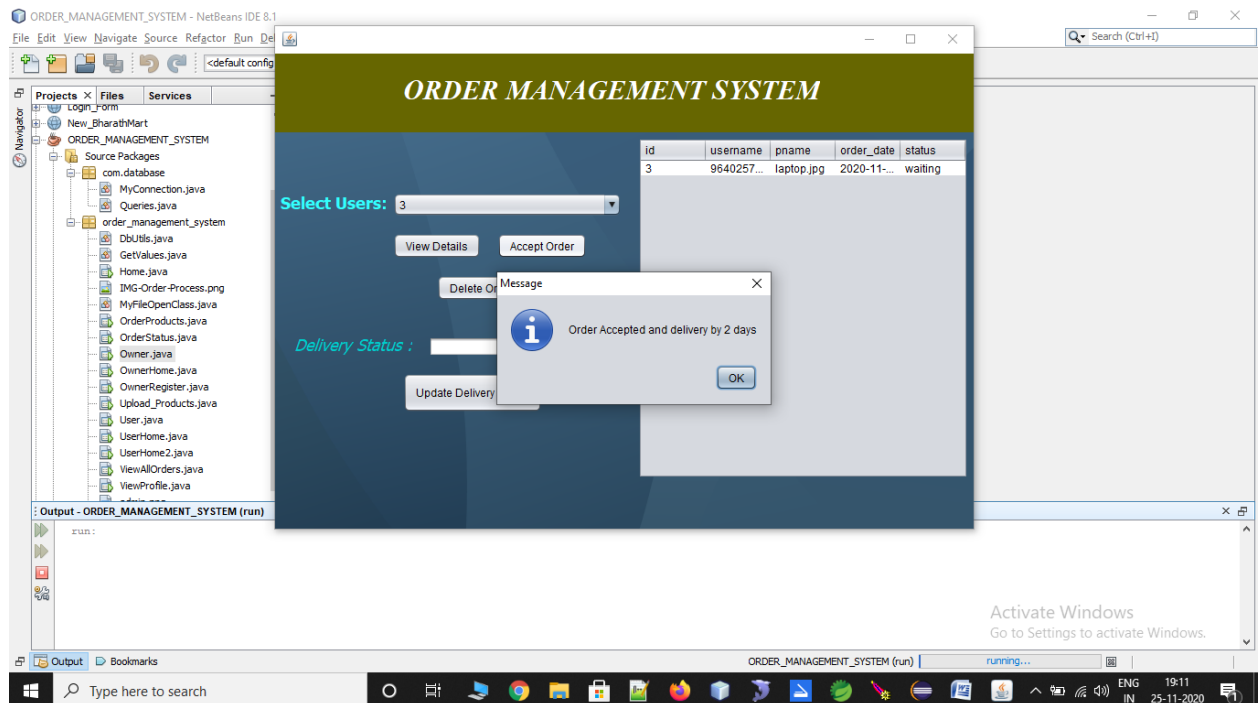
Order management screen



Order details



Order accepted status



User register

The screenshot shows a web browser window titled "USER REGISTER" displaying the "ORDER MANAGEMENT SYSTEM" header. Below the header is a "USER REGISTRATION" form. The form contains the following fields and controls:

- Full Name:
- Email:
- Mobile :
- Location :
- Gender :
- UserName :
- Password :
- Register button (yellow)
- Reset button (grey)
- Already Have An Account ? [Login](#)

The background shows a Microsoft Word document titled "SHOTS.docx" with a ribbon menu and a taskbar at the bottom with various application icons and a system clock showing 19:22 on 25-11-2020.

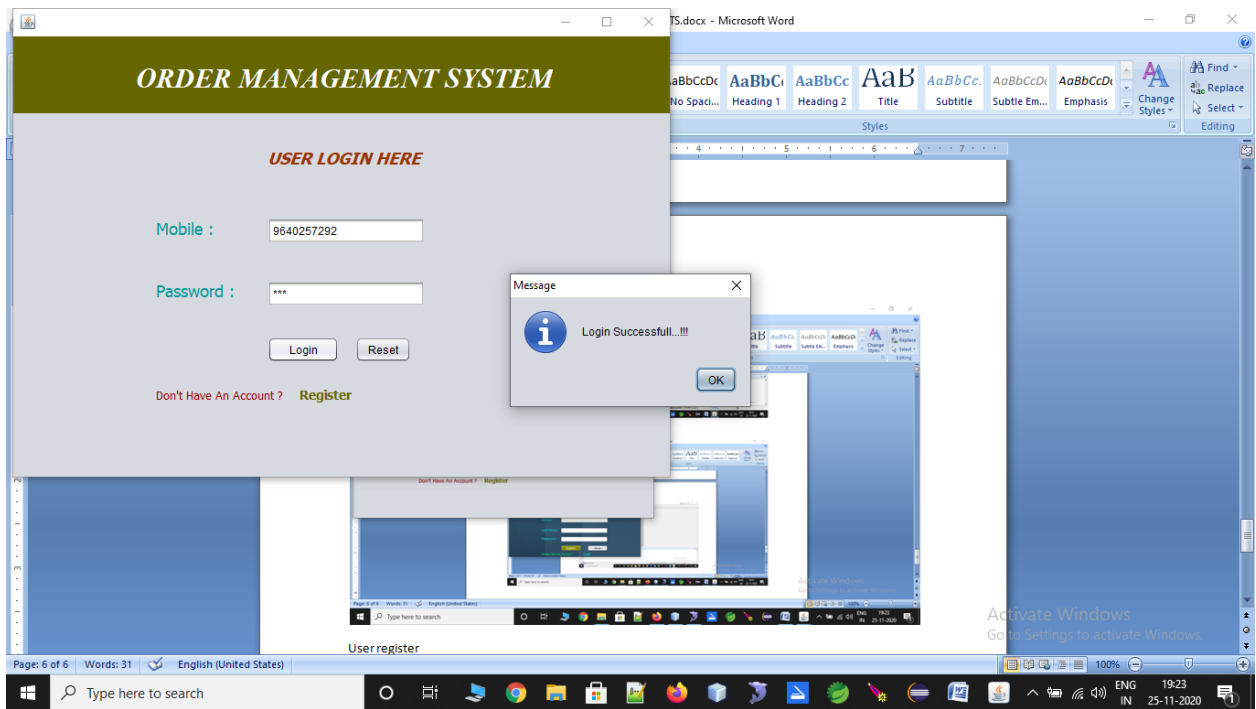
User login:

The screenshot shows a web browser window titled "USER LOGIN" displaying the "ORDER MANAGEMENT SYSTEM" header. Below the header is a "USER LOGIN HERE" form. The form contains the following fields and controls:

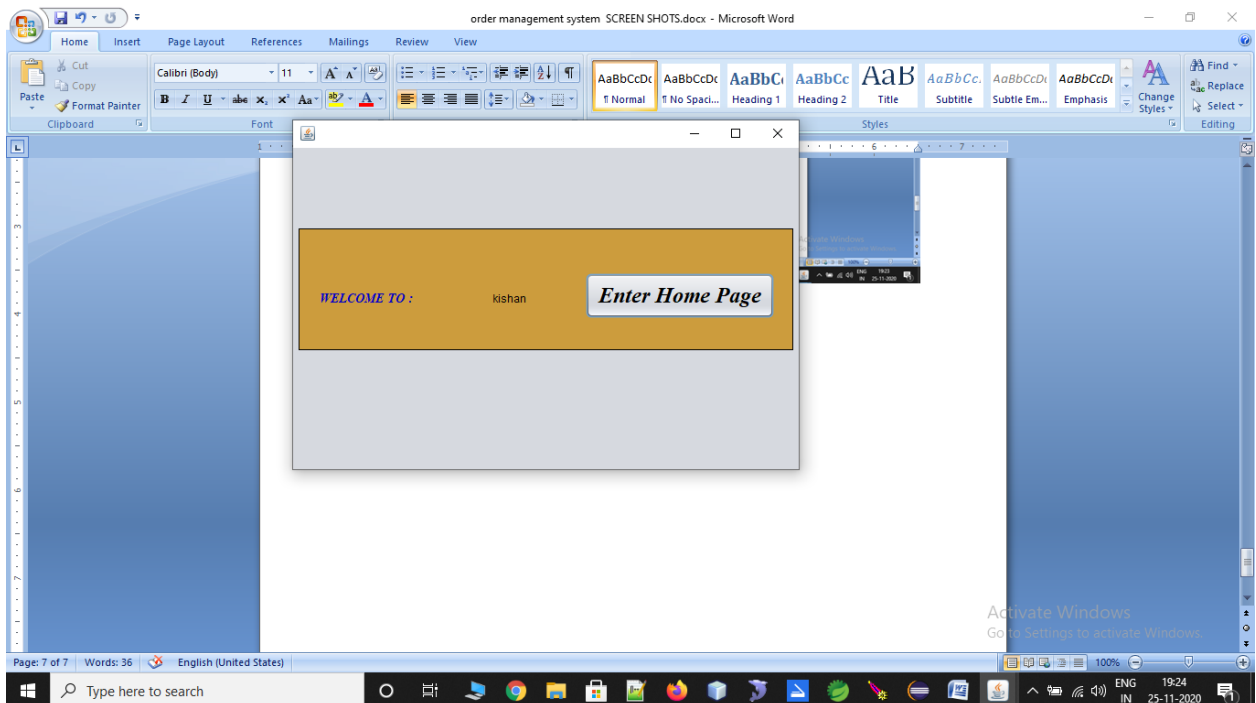
- Mobile :
- Password :
- Login button (grey)
- Reset button (grey)
- Don't Have An Account ? [Register](#)

The background shows a Microsoft Word document titled "SHOTS.docx" with a ribbon menu and a taskbar at the bottom with various application icons and a system clock showing 19:23 on 25-11-2020.

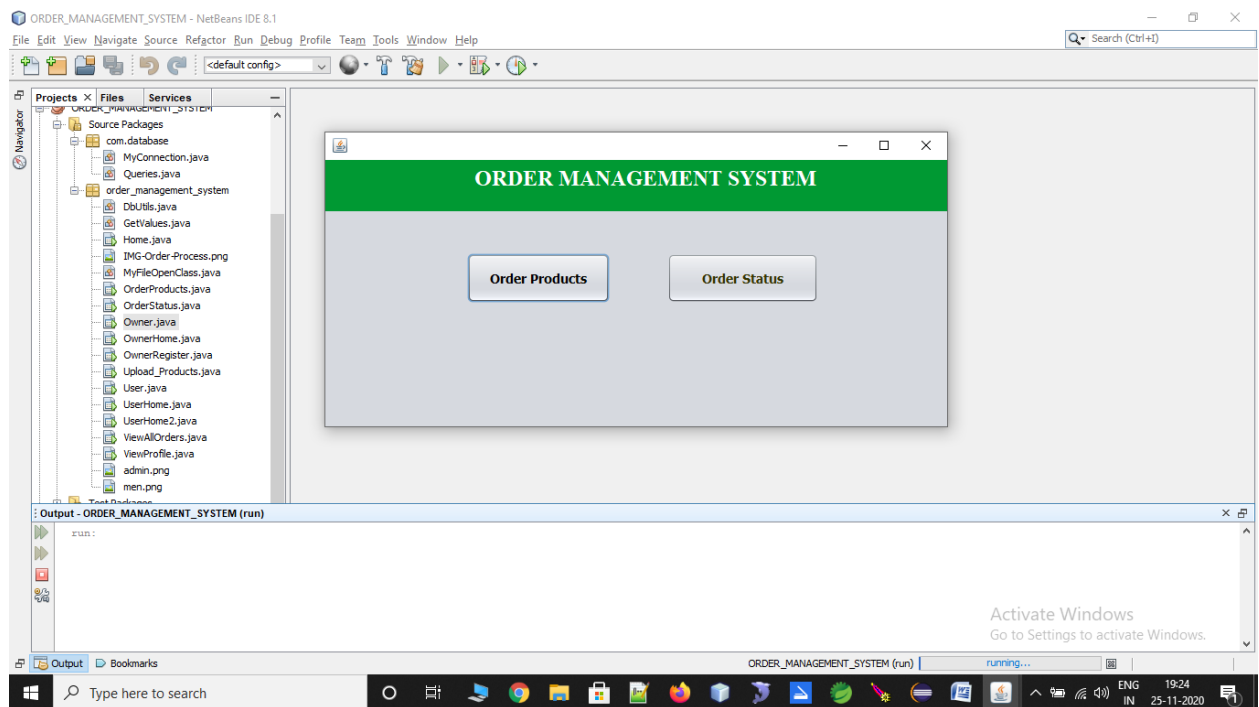
Login status



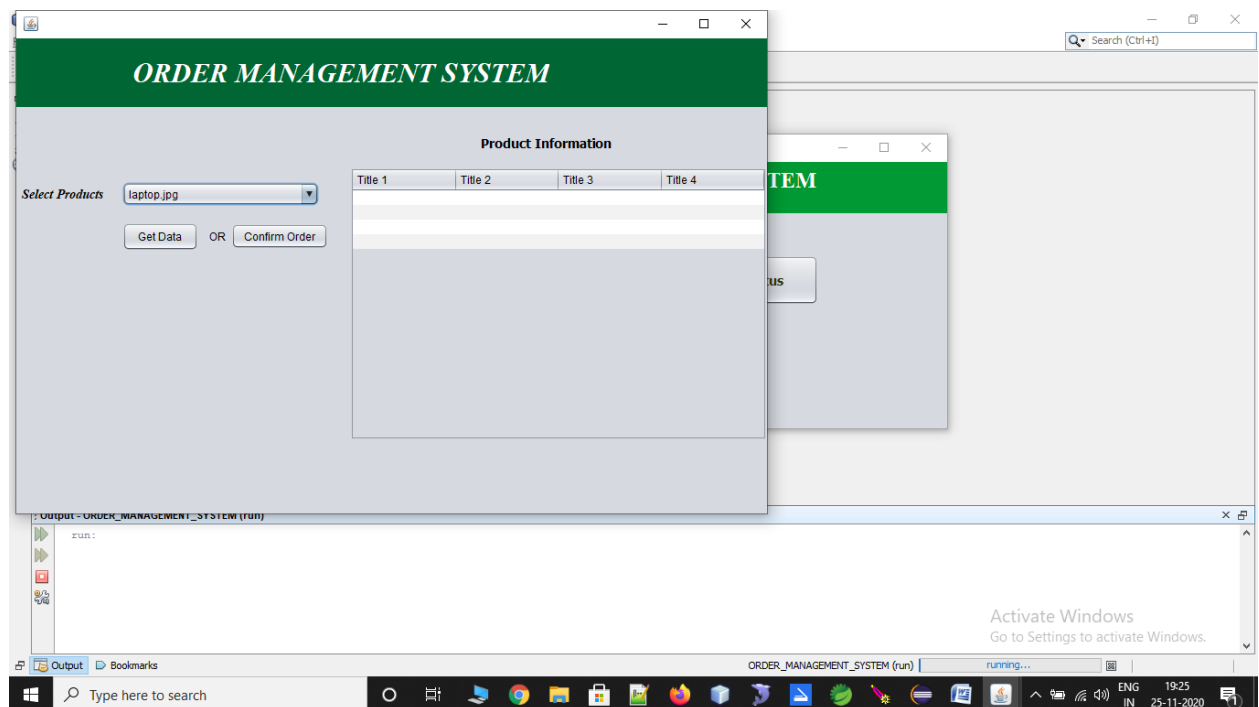
user home screen



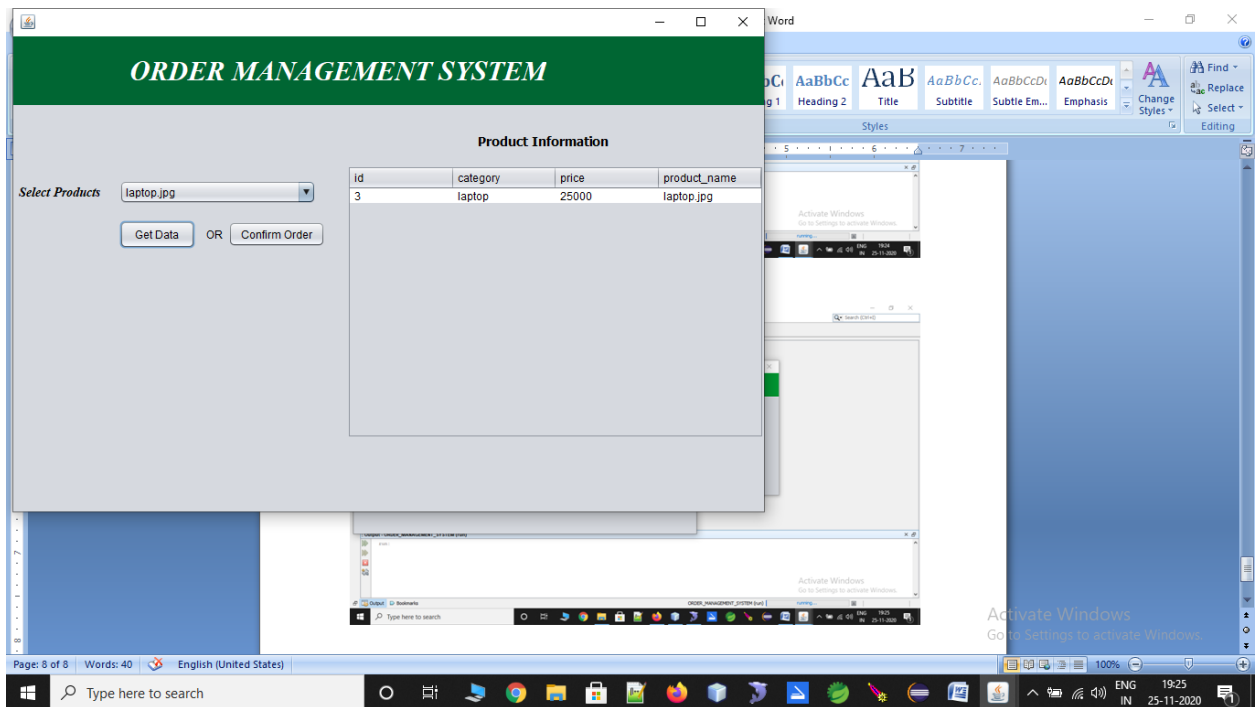
User menu



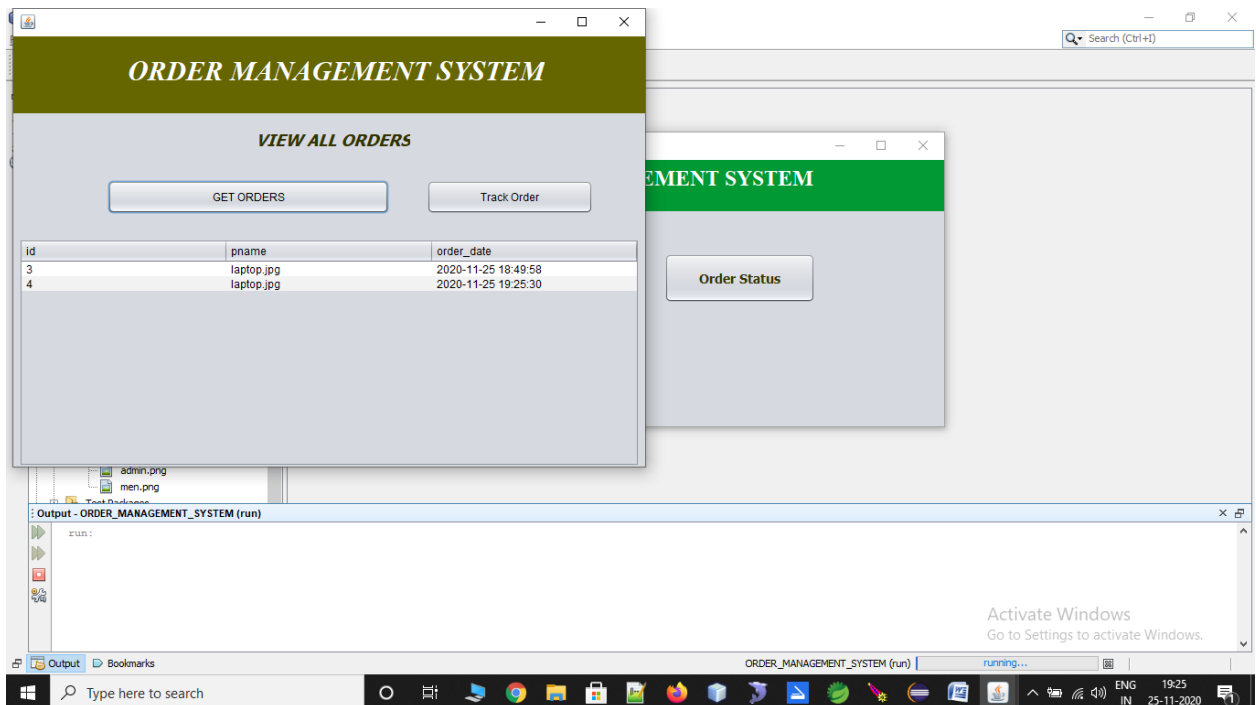
Order products



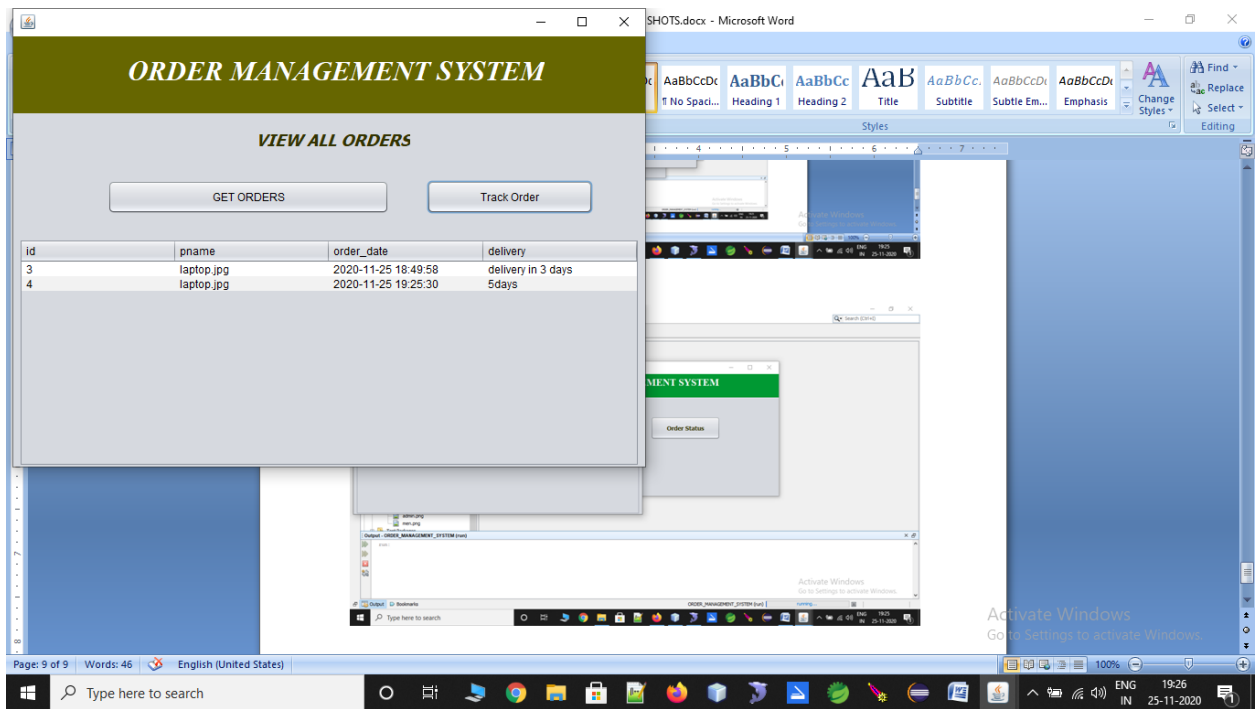
Get selected products name



All orders



Oder track of delivery



CHAPTER-11

CONCLUSION

The objective of the project is to build a Product which will be a plugin for the retailers. It can help to manage for both sales and inventory management. The Order module has been built and is ready to use with some technical issues. Due to these issues in the order module, the sales module has difficulties.

11.1 FUTURE ENHANCEMENT

Due to the issues with the chosen technology and the development process, the development team has decided to include hibernate for the better performance.

CHAPTER-12

REFERENCES/BIBLIOGRAPHY

1. Toni Stojanovski and Jordan Vrtanoski,” Order Handling in Convergent Environments”, 2011 2nd International Conference on eEducation, e-Business, e-Management and E-Learning (IC4E 2011).
2. Rockstrom and Zdebel. A network strategy for survival. Communications Magazine, IEEE (1998) vol. 36 (1) pp. 36 – 40.
3. Aloundeth Oupraxay and Mudasser Wyne,”Android Based Mobile Order Management System”
4. Iandolo. Convergence of wireless services. Wireless and Optical Communications, 2005. 14th Annual WOC 2005. International Conference on (2005)
5. Prashant K and Manohar Jawahar,”Emerging needs and challenges in BuySide Order Management Systems”
6. Pricing and Ordering strategies of E-Retailers in Electronic Business environment. By XIE Ming and CHEN Jian
7. Noll. Technical opinion: does data traffic exceed voice traffic?.Communications of the ACM (1999) vol. 42 (6).
8. Zhen Liu et al. Towards a Parlay-grid communication model for NGN service convergence. Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE (2005) vol. 1