*A Major Project Report on*

# ONLINE AUCTION FRAUD DETECTION

*Submitted to*

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,**

**HYDERABAD**

*In partial fulfillment of the requirement for the award of degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

***By***

**SHAIK ABDUL SAHIL MAJEED (15RA1A0531)**

**NISHA KUMARI BIDLA (15RA1A0522)**

**S.SAI SHRAVYA (15RA1A0528)**

*Under the guidance of*

**Mrs. K. Suparna**

(Assistant Professor)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY**

**(Affiliatedto JNTUH, Ghanpur(V), Ghatkesar(M), Medchal(D)-500088)**

**2015-2019**

**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY**

**(Affiliated to JNTUH, Ghanpur(V), Ghatkesar(M), Medchal(D)-500088)**

## CERTIFICATE

This is to certify that the project work entitled **"ONLINE AUCTION FRAUD DETECTION" is submitted by Mr. Shaik Abdul Sahil Majeed (15RA1A0531), Ms. Nisha Kumari Bidla (15RA1A0522), Ms. S. Sai Shravya (15RA1A0528)** bonafied students of **Kommuri Pratap Reddy Institute of Technology** in a partial fulfillment of the requirement for the award of Bachelor of Technology in **Computer Science and Engineering** of the **Jawaharlal Nehru Technological University, Hyderabad** during the year 2015-19.

**Internal Examiner** **HOD**

**Mrs. K. Suparna**

(Asst. Professor)

**External Examiner**

## DECLARATION

We hereby declare that this project work entitled **"ONLINE AUCTION FRAUD DETECTION"** in partial fulfilment of requirements for the award of degree of **Computer Science and Engineering** is a bonafied work carried out by us during the academic year 2018-19.

We further declare that this project is a result of our effort and has not been submitted for the award of any degree by us to any institute.

By

**SHAIK ABDUL SAHIL MAJEED (15RA1A0531)**

**NISHA KUMARI BIDLA (15RA1A0522)**

**S.SAI SHRAVYA (15RA1A0528)**

# ACKNOWLEDGEMENT

It gives us immense pleasure to acknowledge with gratitude, the help and support extended throughout the project report from the following:

We will be very much grateful to almighty our **Parents** who have made us capable of carrying out our job.

We express our profound gratitude to our Principal **Dr. D.Eshwar** of **Kommuri Pratap Reddy Institute of Technology**, who has encouraged in completing our project report successfully.

We are grateful to **Mr. A. Prakash** who is our **Head of the Department**, **CSE** for his amiable ingenious and adept suggestions and pioneering guidance during the project report.

We express our deep sense of gratitude and thanks to **Internal guide, Mrs. K. Suparna** for her support during the project report.

We are also very thankful to our **Management**, **Staff Members** and all **Our Friends** for their valuable suggestions and timely guidance without which we would not have been completed it.

By

**SHAIK ABDUL SAHIL MAJEED (15RA1A0531)**

**NISHA KUMARI BIDLA (15RA1A0522)**

**S.SAI SHRAVYA (15RA1A0528)**

# ABSTRACT

**We consider the problem of building online machine-learned models for detecting auction frauds in e-commerce web sites. Since the emergence of the World Wide Web, online shopping and online auction have gained more and more popularity. While people are enjoying the benefits from online trading, criminals are also taking advantages to conduct fraudulent activities against honest parties to obtain illegal profit. Hence proactive fraud-detection moderation systems are commonly applied in practice to detect and prevent such illegal and fraud activities. Machine-learned models, especially those that are learned online, are able to catch frauds more efficiently and quickly than human-tuned rule-based systems. In this paper, we propose an online probit model framework which takes online feature selection, coefficient bounds from human knowledge and multiple instance learning into account simultaneously. By empirical experiments on a real-world online auction fraud detection data we show that this model can potentially detect more frauds and significantly reduce customer complaints compared to several baseline models and the human-tuned rule-based system.**

# INDEX

**CONTENTS** **Page No.**

# LIST OF FIGURES

# CHAPTER 1. INTRODUCTION

Since the emergence of the World Wide Web (WWW), electronic commerce, commonly known as e-commerce, has become more and more popular. Websites such as eBay and Amazon allow Internet users to buy and sell products and services online, which benefits everyone in terms of convenience and profitability. The traditional online shopping business model allows sellers to sell a product or service at a preset price, where buyers can choose to purchase if they find it to be a good deal. Online auction however is a different business model by which items are sold through price bidding. There is often a starting price and expiration time specified by the sellers. Once the auction starts, potential buyers bid against each other, and the winner gets the item with their highest winning bid.

Similar to any platform supporting financial transactions, online auction attracts criminals to commit fraud. The varying types of auction fraud are as follows. Products purchased by the buyer are not delivered by the seller. The delivered products do not match the descriptions that were posted by sellers. Malicious sellers may even post non-existing items with false description to deceive buyers, and request payments to be wired directly to them via bank-to-bank wire transfer. Furthermore, some criminals apply phishing techniques to steal high-rated seller’s accounts so that potential buyers can be easily deceived due to their good rating. Victims of fraud transactions usually lose their money and in most cases are not recoverable. As a result, the reputation of the online auction services is hurt significantly due to fraud crimes.

To provide some assurance against fraud, E-commerce sites often provide insurance to fraud victims to cover their loss up to a certain amount. To reduce the amount of such compensations and improve their online reputation, ecommerce providers often adopt the following approaches to control and prevent fraud. The identifies of registered users are validated through email, SMS, or phone verifications. A rating system where buyers provide feedbacks is commonly used in e-commerce sites so that fraudulent sellers can be caught immediately after the first wave of buyer complaints. In addition, proactive moderation systems are built to allow human experts to manually investigate suspicious sellers or buyers. Even though e-commerce sites spend a large budget to fight frauds with a moderation system, there are still many outstanding and challenging cases. Criminals and fraudulent sellers frequently change their accounts and IP addresses to avoid being caught. Also, it is usually infeasible for human experts to investigate every buyer and seller to determine if they are committing fraud, especially when the e-commerce site attracts a lot of traffic. The patterns of fraudulent sellers often change constantly to take advantage of temporal trends. For instance, fraudulent sellers tend to sell the “hottest” products at the time to attract more potential victims. Also, whenever they find a loophole in the fraud detection system, they will immediately leverage the weakness.

In this project, we consider the application of a proactive moderation system for fraud detection in a major Asian online auction site, where hundreds of thousands of new auction cases are created every day. Due to the limited expert resources, only 20%-40% of the cases can be reviewed and labeled. Therefore, it is necessary to develop an automatic pre-screening moderation system that only directs suspicious cases for expert inspection, and passes the rest as clean cases. The moderation system for this site extracts rule-based features to make decisions. The rules are created by experts to represent the suspiciousness of sellers on fraudulence, and the resulting features are often binary. For instance, we can create a binary feature (rule) from the ratings of sellers, i.e. the feature value is 1 if the rating of a seller is lower than a threshold (i.e. a new account without many previous buyers); otherwise it is 0. The final moderation decision is based on the fraud score of each case, which is the linear weighted sum of those features, where the weights can be set by either human experts or machine-learned models. By deploying such a moderation system, we are capable of selecting a subset of highly suspicious cases for further expert investigation while keeping their workload.

# CHAPTER 2. LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool, the programmers need lot of external support. This support can be obtained from senior programmers, books or from websites.

Our application is to detect online auction frauds for a major Asian site where hundreds of thousands of new auction cases are posted every day. Every new case is sent to the proactive anti-fraud moderation system for pre-screening to assess the risk of being fraud. The current system is featured by:

• Rule-based features: Human experts with years of experience created many rules to detect whether a user is fraud or not. An example of such rules is “blacklist”, i.e. whether the user has been detected or complained as fraud before. Each rule can be regarded as a binary feature that indicates the fraud likeliness.

• Linear scoring function: The existing system only supports linear models. Given a set of coefficients (weights) on features, the fraud score is computed as the weighted sum of the feature values.

• Selective labeling: If the fraud score is above a certain threshold, the case will enter a queue for further investigation by human experts. Once it is reviewed, the final result will be labeled as Boolean, i.e. fraud or clean. Cases with higher scores have higher priorities in the queue to be reviewed. The cases whose fraud score are below the threshold are determined as clean by the system without any human judgment.

• Fraud churn: Once one case is labeled as fraud by human experts, it is very likely that the seller is not trustable and may be also selling other frauds; hence all the items submitted by the same seller are labeled as fraud too. The fraudulent seller along with his/her cases will be removed from the website immediately once detected.

• User feedback: Buyers can file complaints to claim loss if they are recently deceived by fraudulent sellers.

Motivated by these specific attributes in the moderation system for fraud detection, in this section we describe our Bayesian online modeling framework with details of model fitting via Gibbs sampling. We start from introducing the online probit regression model in Section 2.1. In Section 2.2 we apply stochastic search variable selection (SSVS), a well-known technique in statistics literature, to the online probit regression framework so that the feature importance can dynamically evolve over time. Since it is important to use the expert knowledge, we describe how to bind the coefficients to be positive and finally combine our model with multiple instance learning.

In this project we build online models for the auction fraud moderation and detection system designed for a major Asian online auction website. By empirical experiments on a real-word online auction fraud detection data, we show that our proposed online probit model framework, which combines online feature selection, bounding coefficients from expert knowledge and multiple instance learning, can significantly improve over baselines and the human-tuned model. Note that this online modeling framework can be easily extended to many other applications, such as web spam detection, content optimization and so forth. Regarding to future work, one direction is to include the adjustment of the selection bias in the online model training process. It has been proven to be very effective for offline models. The main idea there is to assume all the unlabeled samples have response equal to 0 with a very small weight. Since the unlabeled samples are obtained from an effective moderation system, it is reasonable to assume that with high probabilities they are non-fraud. Another future work is to deploy the online models described in this paper to the real production systems.

# CHAPTER 3. SYSTEM REQUIREMENTS

This part of the chapter mainly focuses on software requirement specification, user requirement specification and hardware and software system configurations.

## 3.1 SOFTWARE REQUIREMENT SPECIFICATION

Software Requirements Specification plays an important role in creating quality software solutions. Specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation.

Requirements may be specified in a variety of ways. However, there are some guidelines worth following: -

- ➢ Representation format and content should be relevant to the problem
- ➢ Information contained within the specification should be nested
- ➢ Diagrams and other notational forms should be restricted in number and consistent in use.
- ➢ Representations should be revisable.

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to the software as a part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, and indication of performance requirements and design constraints, appropriate validation criteria and other data pertinent to requirements.

### 3.1.1An Outline of the Software Requirements Specification

A simplified outline can be given for the framework of the specifications. This is according to the IEEE Standards.

## 3.2 USER REQUIREMENT SPECIFICATION

The user interface helps the users upon the system in searching through the existing data and required services. The operational user interface also helps in adding new data as and when required. Also, the user can update/delete the data if wish to. There is no restriction or access rights for the user to access the system. The interface helps the users with all the transactional states like data insertion, data deletion and data updation.

## SYSTEM CONFIGURATION

### H/W System Configuration

| | | |
|---|---|---|
| Processor | - | Intel core i5 |
| Speed | - | 2.50 Ghz |
| RAM | - | 8 GB |
| Hard Disk | - | 100 GB |

### S/W System Configuration

| | |
|---|---|
| Operating System | : Windows10 |
| Application Server | : Tomcat 9.0 |
| Front End | : HTML, Java, JSP |
| Scripts | : JavaScript. |
| Server side Script | : Java Server Pages. |
| Database | : MySQL |
| Database Connectivity | : JDBC. |

# CHAPTER 4. SYSTEM ANALYSIS

This part of the chapter mainly focuses on the existing system with its advantages and proposed system with its advantages and disadvantages.

## 4.1 EXISTING SYSTEM

The traditional online shopping business model allows sellers to sell a product or service at a preset price, where buyers can choose to purchase if they find it to be a good deal. Online auction however is a different business model by which items are sold through price bidding. There is often a starting price and expiration time specified by the sellers. Once the auction starts, potential buyers bid against each other, and the winner gets the item with their highest winning bid.

**Disadvantages:**

- Product expiration time will be there.
- Product does not delivered to the customer because there is no fraud detection.
- Without registration sellers are interacted.

## 4.2 PROPOSED SYSTEM

We propose an online probit model framework which takes online feature selection, coefficient bounds from human knowledge and multiple instance learning into account simultaneously. By empirical experiments on a real-world online auction fraud detection data we show that this model can potentially detect more frauds and significantly reduce customer complaints compared to several baseline models and the human-tuned rule-based system. Human experts with years of experience created many rules to detect whether a user is fraud or not. If the fraud score is above a certain threshold, the case will enter a queue for further investigation by fraud or clean. Cases with higher scores have higher priorities in the queue to be reviewed. The cases whose fraud score are below the threshold are determined as clean by the system without any human judgment.

**Advantages:**

- There is no time limit or day limit.
- Time saving.
- Only authorized and verified customer can participate in auction.
- Fraud customer or seller gets detect in early stages and measure to prevent it.
- Authenticate legitimate users can buy the product online very efficiently and securely with the help of this system.
- The products for auction does not need any physical location.
- The bidder can participate in auction from anywhere at any time through online auction.

**Disadvantages:**

- Customer can view only the product picture and some details on the auction website, which may lead to lack of product genuineness.
- Purchased product may not be delivered.

# CHAPTER 5. SYSTEM STUDY

## FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

## 5.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified.

Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## 5.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## 5.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# CHAPTER 6. SYSTEM DESIGN

This part of the chapter mainly focuses on input design, output design, system design, sub system design, data flow diagrams and its types, UML conceptual model, UML diagrams.

## 6.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

### 6.1.1 OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

## 6.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output.

- Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- Select methods for presenting information.
- Create document, report, or other formats that contain information produced by the system.

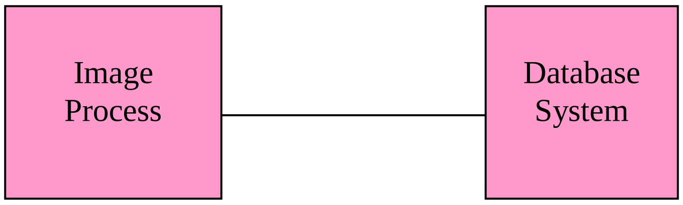
The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

## 6.3 HIGH LEVEL DESIGN

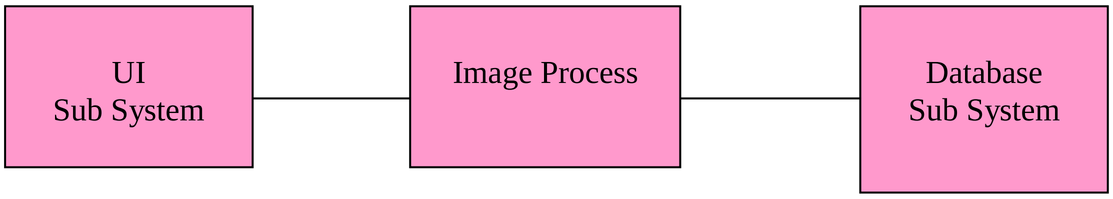
### 6.3.1 SYSTEM DESIGN

A System design is an architecture which identifies different working (possible) systems interacting with our main working system.



### 6.3.2 SUB SYSTEM DESIGN

When a system is large and too complex to understand we divide the core system into parts called Sub Systems.



### 6.3.3 DATA FLOW DIAGRAMS

Data flow diagram is a structure analysis tool that is used for graphical representation of Data processes through any organization. The data flow approach emphasis on the logic underlying the system, by using combination of only 4 symbols. It follows a top down approach. A full description of a system actually consists of set of DFD s, which comprises of various levels. And initial over view model is exploded lower level diagrams that show additional feature of the system. Further each process can be broken down into a more detailed DFD. This occurs repeatedly until sufficient details are described. The top-level diagram is often called a "*context diagram*". It contains a single process, but it plays a very important role in studying the current system. The context diagram defines the system that will be studied in the sense that it determines the boundaries. Anything that is not inside the process identified in the context diagram will not be part of the system study. It represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows respectively.

### 6.3.4 TYPES OF DATA FLOW DIAGRAMS

Data Flow Diagrams are of two types as follows:

1. Physical DFD
2. Logical DFD

**PHYSICAL DFD**

Structured analysis states that the current system should be first understand correctly. The physical DFD is the model of the current system and is used to ensure that the current system has been clearly understood. Physical DFDs shows actual devices, departments, and people etc., involved in the current system

**LOGICAL DFD**

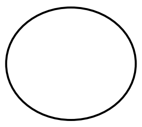
Logical DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts.

**Basic Notation**

The Basic Notation used to create a DFD's are as follows:

**Dataflow:** Data move in a specific direction from an origin to a destination.

**Process:** People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



**Source:** External sources or destination of data, which may be People, programs, organizations or other entities.

**Data Store:** Here data are stored or referenced by a process in the System

## 6.4 UML EXPLANATION

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

These are the artifacts of a software-intensive system.

### 6.4.1 A Conceptual Model OF UML

The three major elements of UML are

- The UML's basic building blocks
- The rules that dictate how those building blocks may be put together.
- Some common mechanisms that apply throughout the UML.

Basic building blocks of the UML. The vocabulary of UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model;

Relationships tie these things together;

Diagrams group the interesting collection of things.

Things in UML: There are four kind of things in the UML

- Structural things
- Behavioral things.
- Grouping things
- An notational things

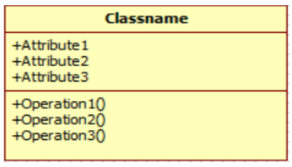
These things are the basic object-oriented building blocks of the UML.They are used to write well-formed models.

### 6.4.2 Structural Things

Structural things are the nouns of the UML models. These are mostly static parts of the model, representing elements that are either conceptual or physical. In all, there are seven kinds of Structural things.

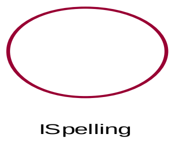
#### Class

A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically a class is rendered as a rectangle, usually including its name, attributes and operations, as shown below.



#### Interface

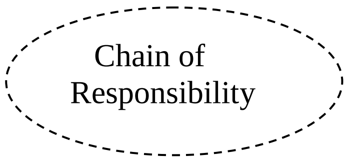
An interface is a collection of operations that specify a service of a class or component. An interface describes the externally visible behavior of that element



Graphically the interface is rendered as a circle together with its name.

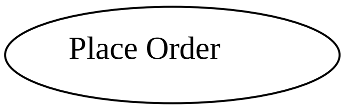
#### Collaboration

Collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. Graphically, collaboration is rendered as an ellipse with dashed lines, usually including only its name as shown below.



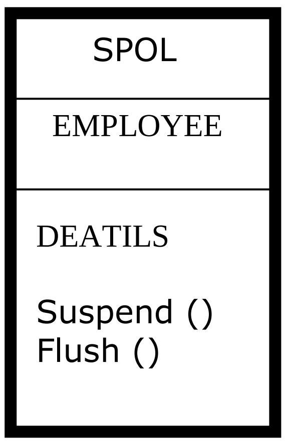
**Use Case**

Use case is a description of a set of sequence of actions that a system performs that yields an observable result of value to a particular thing in a model. Graphically, Use Case is rendered as an ellipse with dashed lines, usually including only its name as shown below.



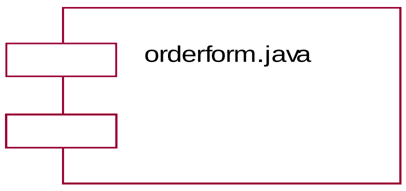
**Active Class**

An active class is a class whose objects own one or more processes or threads and therefore can initiate control activity. Graphically, an active class is rendered just like a class, but with heavy lines usually including its name, attributes and operations as shown below.



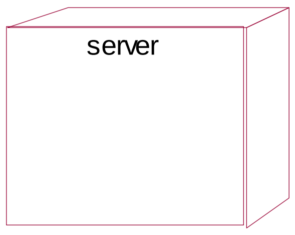
**Component**

Component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Graphically, a component is rendered as a rectangle with tabs, usually including only its name, as shown below.



**Node**

A Node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and often, processing capability. Graphically, a node is rendered as a cube, usually including only its name, as shown below.

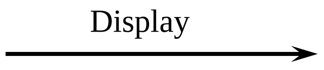


### 6.4.3 Behavioral Things

Behavioral Things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space.

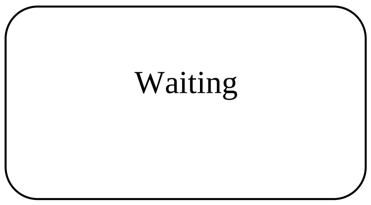
**Interaction**

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. Graphically, a message is rendered as a direct line, almost always including the name if its operation, as shown below.



**State Machine**

A state machine is a behavior that specifies the sequence of states an object are an interaction goes through during its lifetime on response to events, together with its responses to those events. Graphically, a state is rendered as a rounded rectangle usually including its name and its sub-states, if any, as shown below.

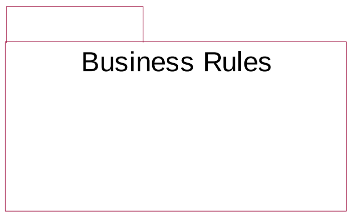


### 6.4.4 Grouping Things

Grouping things are the organizational parts of the UML models. These are the boxes into which a model can be decomposed.

**Package**

A package is a general-purpose mechanism for organizing elements into groups.

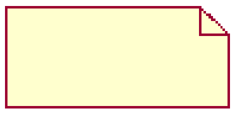


### 6.4.5 Annotational Things

Annotational things are the explanatory parts of the UML models.

**Note**

A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements. Graphically a note is rendered as a rectangle with dog-eared corner together, with a textual or graphical comment, as shown below.



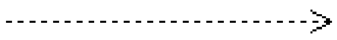
### 6.4.6 RELATIONSHIPS IN THE UML

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

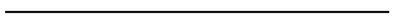
**DEPENDENCY:**

This is relationship between two classes whenever one class is completely dependent on the other class. Graphically the dashed line represents it with arrow pointing to the class that it is being depended on.



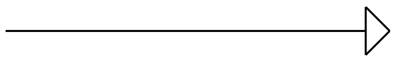
**ASSOCIATION:**

Association is a relationship between instances of the two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes. Graphically it is represented by line as shown.



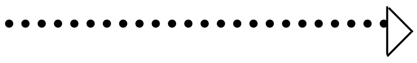
**GENERALIZATION:**

An inheritance is a link indicating one class is a super class of the other. A generalization has a triangle pointing to the super class. Graphically it is represented by line with a triangle at end as shown.



**REALIZATION:**

A realization relationship is a relationship between two model elements, inwhich one model element (the client) realizes the behavior that the other model element specifies. The graphical representation is show as a dotted arrow with empty head.



## 6.5 UML DIAGRAMS

Diagrams play a very important role in the UML. There are nine kind of modeling diagrams as follows:

- Class Diagram
- Use Case Diagram
- Object Diagram
- Sequence Diagram
- Collaboration Diagram
- State Chart Diagram
- Activity Diagram
- Component Diagram and Deployment Diagram

### 6.5.1 Class Diagram

Class diagrams are the most common diagrams found in modeling object-oriented systems. A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Graphically, a class diagram is a collection of vertices and arcs.

**Contents:**

Class Diagrams commonly contain the following things:

- Classes
- Interfaces
- Collaborations
- Dependency
- Generalization
- Association Relationships

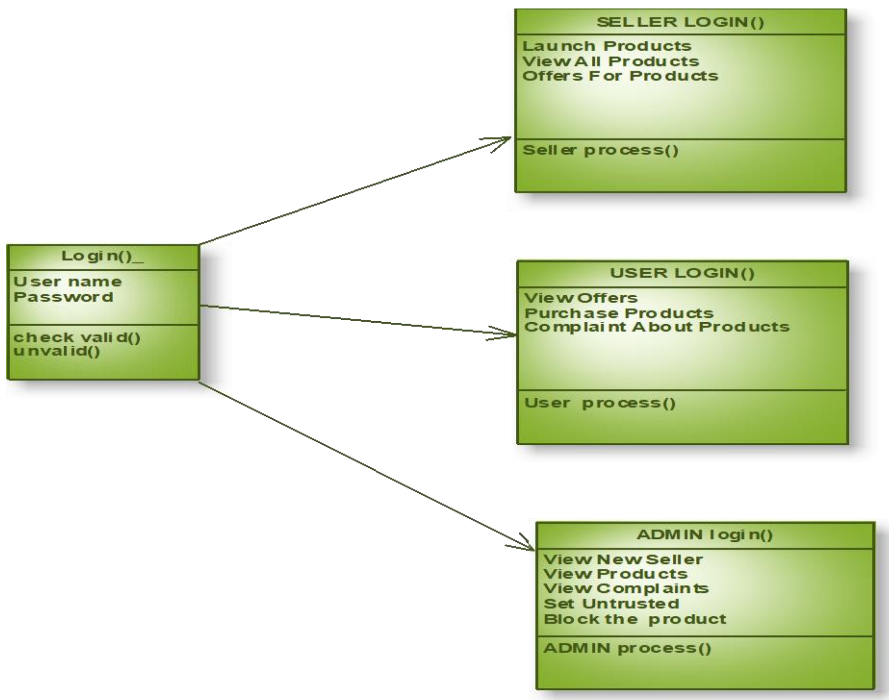


**Fig 6.1 Class Diagram**

### 6.5.2 Use Case Diagram

Use Case diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of systems (activity diagrams, sequence diagrams, state chart diagrams and collaboration diagrams are the four other kinds of diagrams in the UML for modeling the dynamic aspects of systems). Use Case diagrams are central to modeling the behavior of the system, a sub-system, or a class. Each one shows a set of use cases and actors and relationships.

**Common Properties**

A Use Case diagram is just a special kind of diagram and shares the same common properties, as do all other diagrams- a name and graphical contents that are a projection into the model. What distinguishes a use case diagram from all other kinds of diagrams is its particular content.

**Contents**

Use Case diagrams commonly contain:

Use Cases

Actors

Dependency, generalization, and association relationships

Like all other diagrams, use case diagrams may contain notes and constraints. Use Case diagrams may also contain packages, which are used to group elements of your model into larger chunks. Occasionally, you will want to place instances of use cases in your diagrams, as well, especially when you want to visualize a specific executing system.

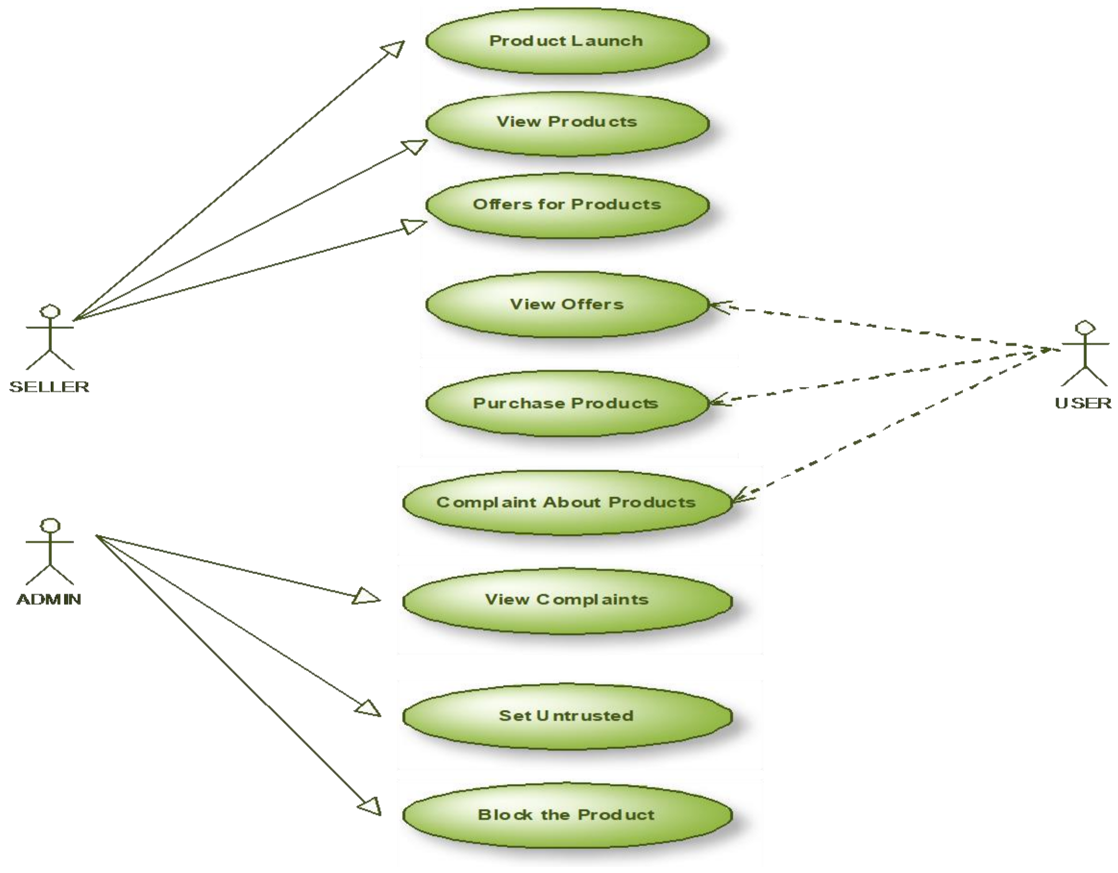


Fig 6.2 Use Case Diagram

### 6.5.3 Interaction Diagram

An Interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them. Interaction diagrams are used for modeling the dynamic aspects of the system.

A sequence diagram is an interaction diagram that emphasizes the time ordering of the messages. Graphically, a sequence diagram is a table that shows objects arranged along the X-axis and messages, ordered in increasing time, along the Y-axis and messages, ordered in increasing time, along the Y-axis.

**Contents**

Interaction diagrams commonly contain:

Objects

Links

Messages

Like all other diagrams, interaction diagrams may contain notes and constraints.

### 6.5.4 Sequence Diagram

A sequence diagram is an interaction diagram that emphasizes the time ordering of the messages. Graphically, a sequence diagram is a table that shows objects arranged along the X-axis and messages, ordered in increasing time, along theY-axis. Typically, you place the object that initiates the interaction at the left, and increasingly more sub-routine objects to the right. Next, you place the messages that these objects send and receive along the Y-axis, in order of increasing time from top to the bottom Sequence diagrams have two interesting features:

- There is the object lifeline. An object lifeline is the vertical dashed line that represents the existence of an object over a period of time. Most objects that appear in the interaction diagrams will be in existence for the duration of the interaction, so these objects are all aligned at the top of the diagram, with their lifelines drawn from the top of the diagram to the bottom.
- There is a focus of the control. The focus of control is tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through the subordinate procedure. The top of the rectangle is aligning with the action; the bottom is aligned with its completion.

**Contents**

Sequence diagrams commonly contains:

Objects

Object Life Line

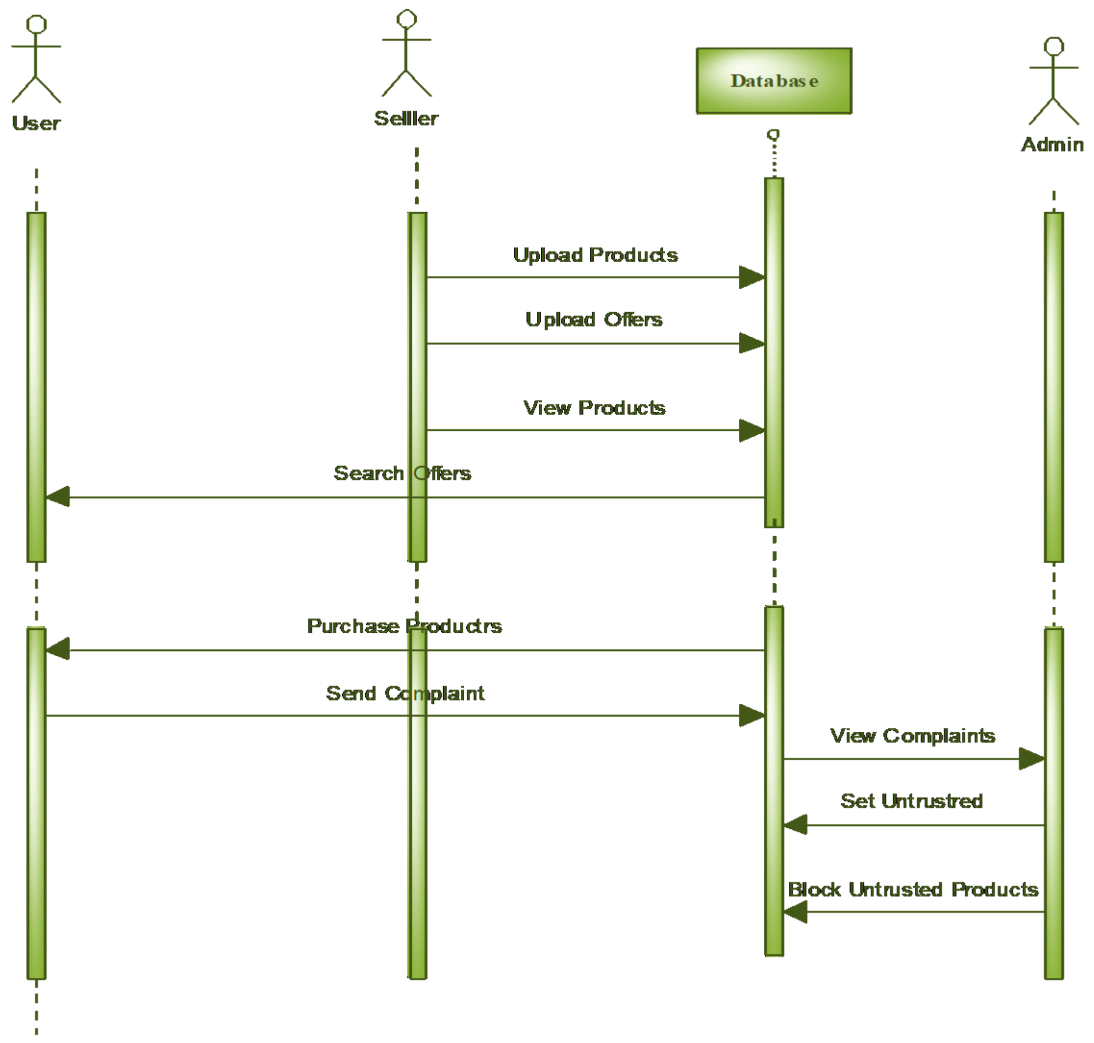


Fig 6.3 Sequence Diagram

## 6.5.5 Activity Diagram

An Activity Diagram is essentially a flow chart showing flow of control from activity to activity. They are used to model the dynamic aspects of as system. They can also be used to model the flow of an object as it moves from state to state at different points in the flow of control. An activity is an ongoing non-atomic execution with in a State machine. Activities ultimately result in some action, which is made up of executable atomic computations that result in a change of state of distinguishes a use case diagram from all other kinds of diagrams is its particular content.

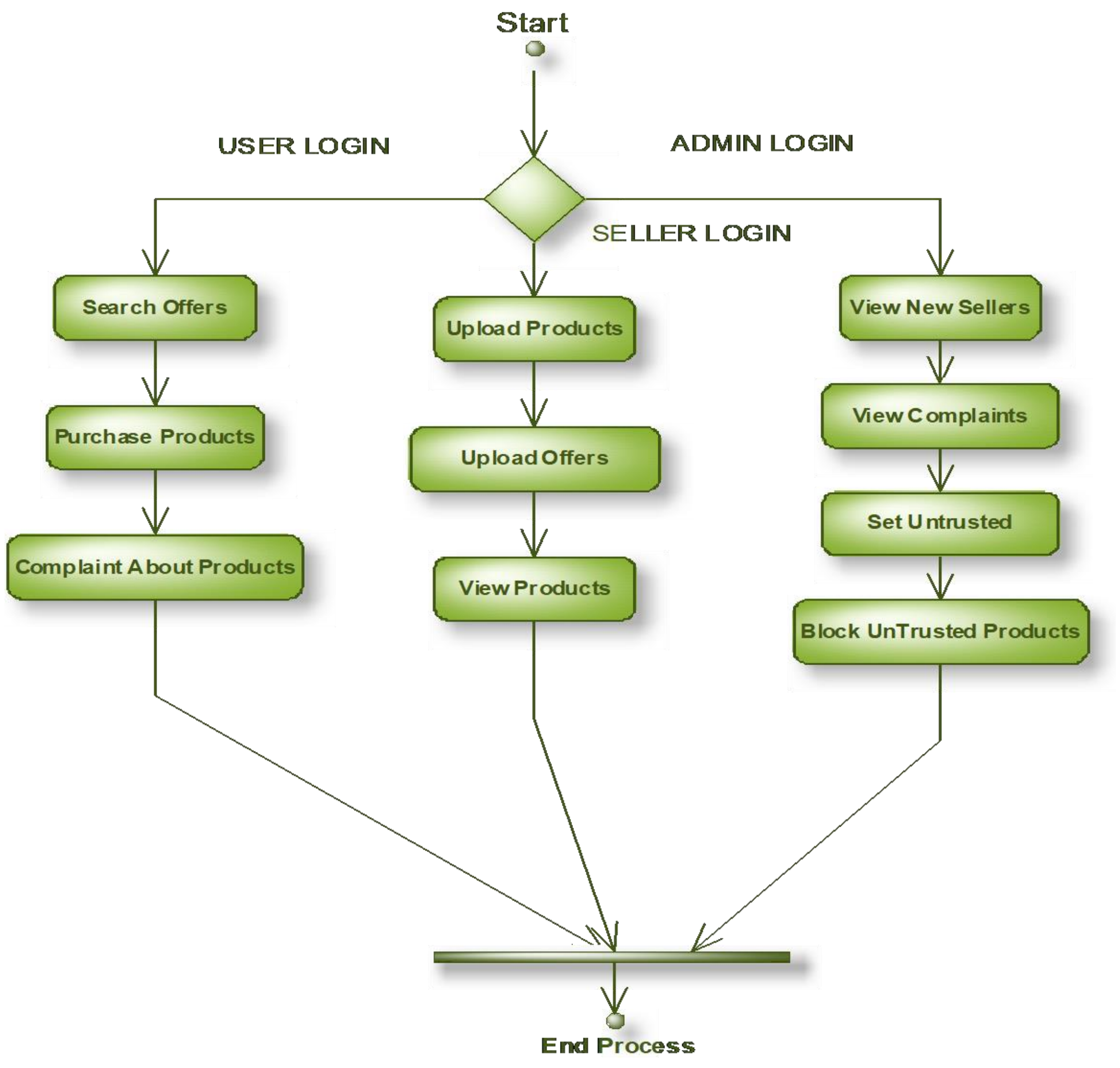


**Fig 6.4 Activity Diagram**

# CHAPTER 7. CODING AND IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

**Modules:**

• **Rule-based features:** Human experts with years of experience created many rules to detect whether a user is fraud or not. An example of such rules is “blacklist”, i.e. whether the user has been detected or complained as fraud before. Each rule can be regarded as a binary feature that indicates the fraud likeliness.

• **Selective labeling:** If the fraud score is above a certain threshold, the case will enter a queue for further investigation by human experts. Once it is reviewed, the final result will be labeled as Boolean, i.e. fraud or clean. Cases with higher scores have higher priorities in the queue to be reviewed. The cases whose fraud score are below the threshold are determined as clean by the system without any human judgment.

• **Fraud churn:** Once one case is labeled as fraud by human experts, it is very likely that the seller is not trustable and may be also selling other frauds; hence all the items submitted by the same seller are labeled as fraud too. The fraudulent seller along with his/her cases will be removed from the website immediately once detected.

•**User Complaint:** Buyers can file complaints to claim loss if they are recently deceived by fraudulent sellers. The Administrator view the various type of complaints and the percentage of various type complaints. The complaints values of a products increase some threshold value the administrator set the trust ability of the product as Untrusted or banded. If the products set as banned, the user cannot view the products in the website.

## 7.1 SOFTWARE ENVIRONMENT

### 7.1.1 Java Technology

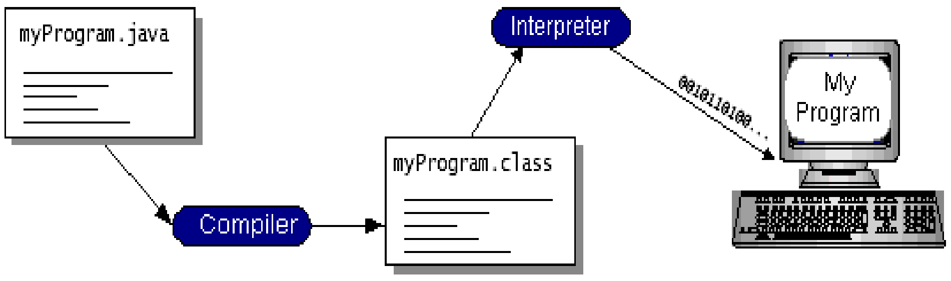
Java technology is both a programming language and a platform.

The Java Programming Language

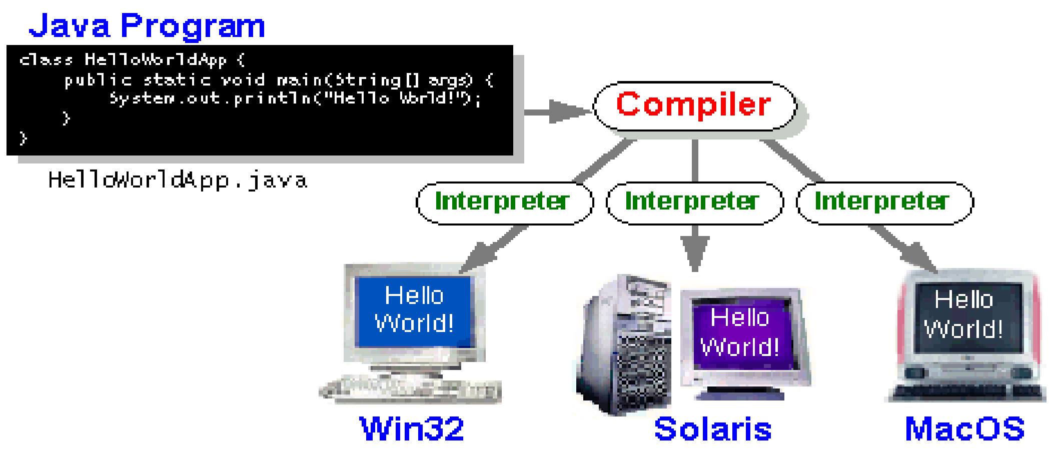
The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic and Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The Java Platform

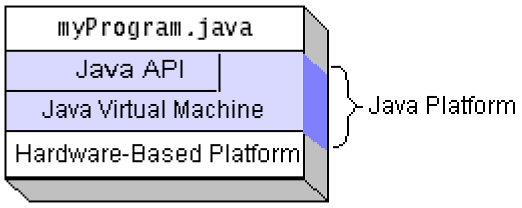
A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.
The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

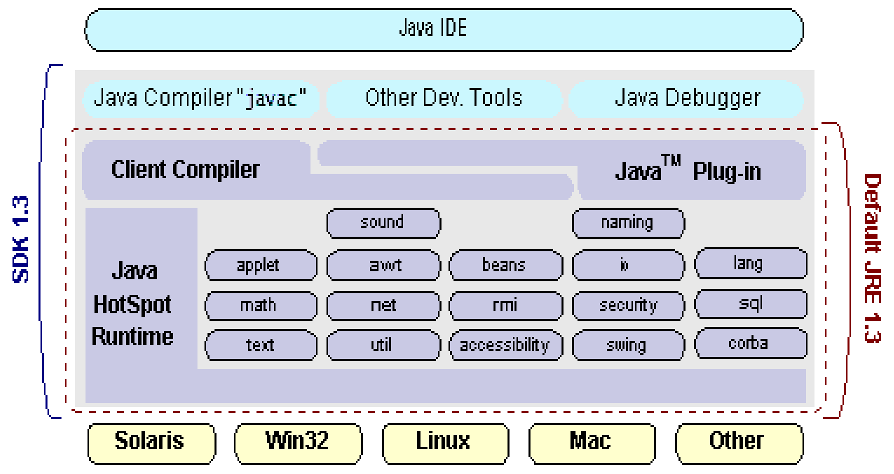
**What Can Java Technology Do?**

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser. However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs. An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



**How Will Java Technology Change My Life?**

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written

in other languages. The 100% Pure Java Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.

- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

## ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE.

There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources. From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer. The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

## JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution. JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after. The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

**JDBC Goals**

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java. The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. **SQL Level API**

   The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC's complexities from the end user.

2. **SQL Conformance**

   SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. **JDBC must be implemental on top of common database interfaces**

The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. **Provide a Java interface that is consistent with the rest of the Java system**

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. **Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

7. **Keep the common cases simple**

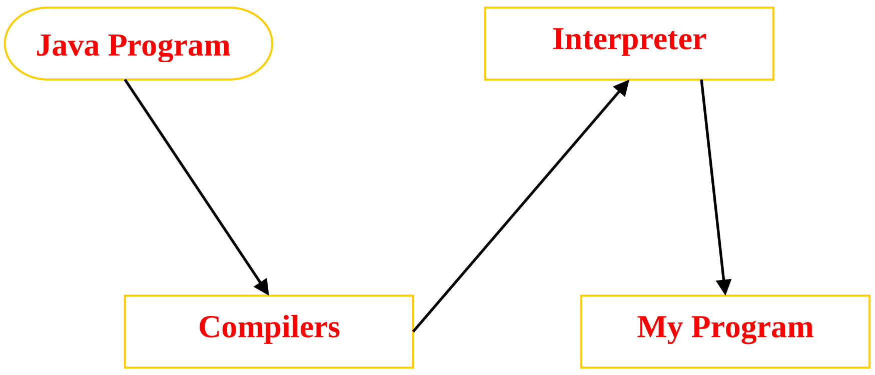
Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible. Finally we decided to proceed the implementation using Java Networking

**Java has two things: a programming language and a platform.**

Java is a high-level programming language that is all of the following

| | |
|---|---|
| Simple | Architecture-neutral |
| Object-oriented | Portable |
| Distributed | High-performance |
| Interpreted | Multithreaded |
| Robust | Dynamic |

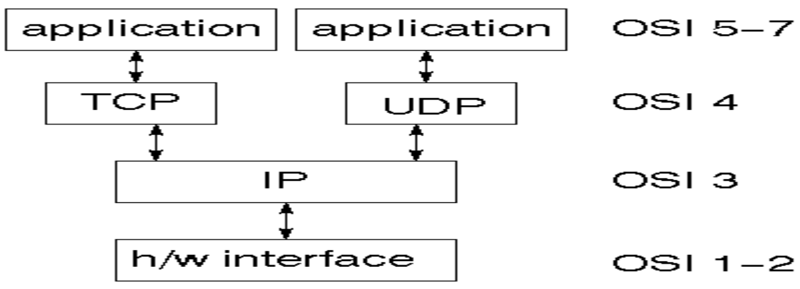
Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

## Networking

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

## IP datagrams

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

## UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

## TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

## Internet Address

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32-bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

## Network Address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16-bit network addressing. Class C uses 24-bit network addressing and class D uses all 32.

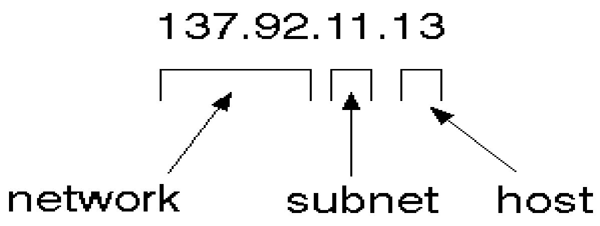
## Subnet Address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

## Host Address

8 bits are finally used for host addresses within our subnet. This place a limit of 256 machines that can be on the subnet.

## Total Address



The 32-bit address is usually written as 4 integers separated by dots.

## Port Address

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

## Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

## JFreeChart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes: A consistent and well-documented API, supporting a wide range of chart types; A flexible design that is easy to extend, and targets both server-side and client-side applications; Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG); JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public License (LGPL), which permits use in proprietary applications**.**

1. **Map Visualizations**

   Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world.

2. **Time Series Chart Interactivity**

   Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. **Dashboards**

   There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

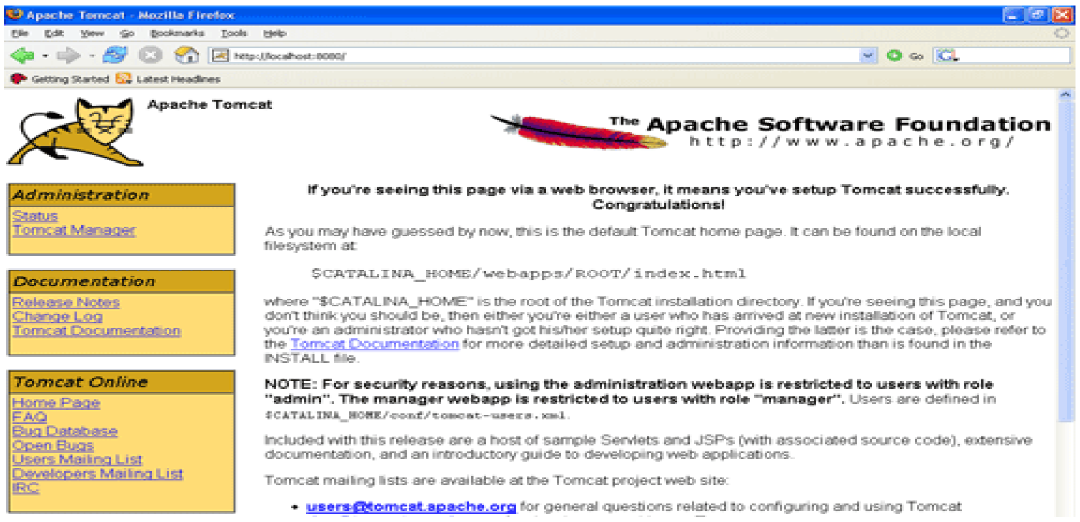
4. **Property Editors**

   The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or re-implement) this mechanism to provide greater end-user control over the appearance of the charts.

5. **Tomcat 6.0 web server**

   Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Web Servers like

Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs WebLogic, is one of the popular application server).To develop a web application with JSP/servlet install any web server like JRun, Tomcat etc to run your application.



**If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!**

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then either you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

**NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager".** Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- users@tomcat.apache.org for general questions related to configuring and using Tomcat

## SAMPLE CODE

### ADMIN_HOME

```
<%@ page  import="java.sql.*" import="databaseconnection.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<title>Auction Fraud</title>
<link href="style.css" rel="stylesheet" media="all" type="text/css" />
</head>
<body>
<div id="wrapper">
<div id="container">
```

```
<div id="header">
<div id="logo"><br>
<br>
<br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
strong><font color="#FFFFFF" size="+2" face="Georgia, Times New Roman,
Times, serif">
Online Modeling of Proactive Moderation System for <br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
Auction Fraud Detection</font></strong></div>
</div>
<!-- /header -->
<div id="navbar">
<ul>
<li><a href="admin_home.jsp" class="active">Home</a></li>
<li><a href="admin_seller.jsp">sellers</a></li>
<li><a href="index.html">Logout</a></li>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
<strong><font color="#FFFFFF" size="+1">Welcome:</font><font
color="#FF0000" size="+1">ADMIN</font></strong>
<!--<li><a href="#">Admin</a></li>
<li><a href="#">Link</a></li>
<li><a href="#">Link</a></li>
<li><a href="#">Link</a></li>-->
</ul>
</div>
<!-- /navbar -->
<div id="main">
```

```
<div id="intro">
<!--<div id="sellerpic">
</div>-->
<!-- /jakepic -->
<div id="text"></div>
<!-- /text -->
<table height="350" align="center" width="700">
<trbgcolor="#CC3300">
<td width="610" bgcolor="#FBF7E1"
valign="top"><br><br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; &nbsp;&nbsp;&nbsp;&nbsp;
<strong><font color="#FF3300" size="+1" face="Georgia, Times New Roman,
Times, serif">All
        Products</font></strong><br><br><br><form name="f"
action="off_search.jsp" method="post" onsubmit="return valid()">
<table bgcolor="#FFFFFF" width="700" border="0">
<tr>
<td colspan="2" align="center"><font size="2"><b>
<%
                    String message=request.getParameter("message");
if(message!=null &&message.equalsIgnoreCase("fail"))
                              {
out.println("<font color='red'
size='+1'><blink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;Offer Not available !</blink></font>");
                              }

             else if(message!=null &&message.equalsIgnoreCase("success"))
```

```
                                        {
out.println("<font color='red'
size='+1'><blink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;Material Purchased !</blink></font>");
                                        }
                                    %>
</b></font></td>
</tr>
<trbgcolor="#E4E4F1">
<td align="center"><font color="#110022"><strong>Product
Id</strong></font></td>
<td align="center"><font color="#110022"><strong>Company
          Name</strong></font></td>
<td align="center"><font color="#110022"><strong>Product
Name</strong></font></td>
<td align="center"><font color="#110022"><strong>Product
Image</strong></font></td>
<td align="center"><font
color="#110022"><strong>Status</strong></font></td>
<td align="center"><font
color="#110022"><strong>Trustability</strong></font></td>
<td align="center"><font
color="#110022"><strong>Details</strong></font></td>
</tr>
<%
                String pname=null,sta=null,pid=null,cname=null,ac=null;
ResultSetrs=null;
try
{
        Connection con = databasecon.getconnection();
        Statement st = con.createStatement();
  String qry="select * from products";
```

```
                rs =st.executeQuery(qry);
        while(rs.next())
        {
        pid=rs.getString("pid");
        cname=rs.getString("comname");
        pname=rs.getString("proname");

        sta =rs.getString("status");
        ac =rs.getString("adminact");
%>
<trbgcolor="#FFFFCC">
<td align="center"><strong><font color="#6300C6"><%=pid%>
</font></strong></td>
<td align="center"><strong><font color="#6300C6"><%=cname%>
</font></strong></td>
<td align="center"><strong><font color="#6300C6"><%=pname%>
</font></strong></td>
<td align="center"><strong><font
color="#FF3300"><imgsrc="image.jsp?product_name=<%=pid%>" height="50"
width="50" >
</font></strong></td>
<td align="center"><strong><font color="#6300C6"><%=sta%>
</font></strong></td>
<td align="center"><strong><font color="#6300C6"><%=ac%>
</font></strong></td>
<td align="center"><a href="admin_trust.jsp?<%=pid%>"><strong><font
color="#CC0000">View</font></strong></a></td>
</tr>
<%
}
  }
catch(Exception e1)
```

```
            {
        out.println(e1.getMessage());
            }
 %>
</table>
</form></td>
</tr>
</table>
</div>
<!-- /intro -->
<div id="columns-wrapper">
<!-- /form-intro -->
<!-- /newsletter -->
<!-- /right -->
<!-- /left -->
</div>
<!-- /columns-wrapper -->
</div>
<!-- /main -->
<div id="footer">
<div id="footer-right">&nbsp;</div>
<div id="footer-left">&nbsp;</div>
<br>
<br>
<strong><font color="#33CC66">WWW.online.com</font></strong></div>
<!-- /footer -->
</div>
<!-- /container -->
</div>
<!-- /wrapper -->
</body>
</html>
```

**USER_HOME**

```
<%@ page  import="java.sql.*" import="databaseconnection.*"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<title>Auction Fraud</title>
<link href="style.css" rel="stylesheet" media="all" type="text/css" />
</head>
<body>
<%
String name=(String)session.getAttribute("name");
%>
<div id="wrapper">
<div id="container">
<div id="header">
<div id="logo"><br>
<br>
<br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
strong><font color="#FFFFFF" size="+2" face="Georgia, Times New Roman,
Times, serif">
	Online Modeling of Proactive Moderation System for <br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Auction Fraud
Detection</font></strong></div>
</div>
```

```
<!-- /header -->
<div id="navbar">
<ul>
<li><a href="#" class="active">Home</a></li>
<li><a href="my_products.jsp">My Products</a></li>
<li><a href="index.html">Logout</a></li>
<!--<li><a href="#">Admin</a></li>
<li><a href="#">Link</a></li>
<li><a href="#">Link</a></li>
<li><a href="#">Link</a></li>-->
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;<font color="#333366" face="Georgia, Times New
Roman, Times, serif" size="+1"><strong>welcome:</strong></font>
&nbsp;<font color="#FF0000" face="Georgia, Times New Roman, Times, serif"
size="+1"><strong><%=name%></strong></font>
</ul>
</div>
<!-- /navbar -->
<div id="main">
<div id="intro">
<!--<div id="sellerpic">
</div>-->
<!-- /jakepic -->
<div id="text"></div>
<!-- /text -->
<table height="350" align="center" width="700">
<trbgcolor="#CC3300">
```

```
<td width="610" bgcolor="#FBF7E1" align="center"><strong><font
color="#FF0000" size="+1" face="Georgia, Times New Roman, Times,
serif"><em>Search
offers</em></font></strong><br><br><form name="f" action="off_search.jsp"
method="post" onsubmit="return valid()">
<table bgcolor="#FFFFFF" width="700" border="0">
<tr>
<td colspan="2" align="center"><font size="2"><b>
<%
                                        String
message=request.getParameter("message");
if(message!=null &&message.equalsIgnoreCase("fail"))
                                        {
out.println("<font color='red'
size='+1'><blink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;Offer Not available !</blink></font>");
                                        }

                  else if(message!=null &&message.equalsIgnoreCase("success"))
                                        {
out.println("<font color='red'
size='+1'><blink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;Material Purchased !</blink></font>");
                                        }
                                  %>
</b></font></td>
</tr>
<trbgcolor="#E4E4F1">
<td align="center"><font color="#110022"><strong>Product
Id</strong></font></td>
<td align="center"><font color="#110022"><strong>Company
                Name</strong></font></td>
```

```
<td align="center"><font color="#110022"><strong>Product
Name</strong></font></td>
<td align="center"><font color="#110022"><strong>Product
Image</strong></font></td>
<td align="center"><font
color="#110022"><strong>Status</strong></font></td>
<td align="center"><font
color="#110022"><strong>Offers</strong></font></td>
</tr>
<%
                        String ba="banded";
                        String pname=null,sta=null,pid=null,cname=null;
ResultSetrs=null;
try
{
        Connection con = databasecon.getconnection();
        Statement st = con.createStatement();

   String qry="select * from products where adminact!='"+ba+"'";
                rs =st.executeQuery(qry);
        while(rs.next())
        {
        pid=rs.getString("pid");
        cname=rs.getString("comname");
        pname=rs.getString("proname");
        sta =rs.getString("status");
%>
<trbgcolor="#FFFFCC">
<td align="center"><strong><font color="#6300C6"><%=pid%>
</font></strong></td>
<td align="center"><strong><font color="#6300C6"><%=cname%>
</font></strong></td>
<td align="center"><strong><font color="#6300C6"><%=pname%>
</font></strong></td>
<td align="center"><strong><font
color="#FF3300"><imgsrc="image.jsp?product_name=<%=pid%>" height="50"
width="50" >
</font></strong></td>
<td align="center"><strong><font color="#6300C6"><%=sta%>
</font></strong></td>
<td align="center"><a href="off_search.jsp?<%=pid%>"><strong><font
color="#CC0000">Offers</font></strong></a></td>
</tr>
<%
}
          }
        catch(Exception e1)
          {
        out.println(e1.getMessage());
          }
 %>
</table>
</form></td>
</tr>
</table>
</div>
<!-- /intro -->
<div id="columns-wrapper">
<!-- /form-intro -->
<!-- /newsletter -->
<!-- /right -->
<!-- /left -->
</div>
<!-- /columns-wrapper -->
```

```
</div>
<!-- /main -->
<div id="footer">
<div id="footer-right">&nbsp;</div>
<div id="footer-left">&nbsp;</div>
<br>
<br>
<strong><font color="#33CC66">WWW.online.com</font></strong></div>
<!-- /footer -->
</div>
<!-- /container -->
</div>
<!-- /wrapper -->
</body>
</html>
```

**SELLER_HOME**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<title>Auction Fraud</title>
<script type="text/javascript">
function valid()
{
var x = document.f.comname.value;
var a = document.f.proname.value;
var b = document.f.wardate.value;
var c = document.f.proimage.value;
var d = document.f.prorate.value;
```

```
if(x=="")
{
alert("enter Company Name");
document.f.comname.focus();
return false;
}

if(a=="")
{
alert("enter Product Name");
document.f.proname.focus();
return false;
}
if(b=="")
{
alert("enter Warrenty days");
document.f.wardate.focus();
return false;
}
if(isNaN(b))
{
alert("enter valid Warrenty days");
document.f.wardate.focus();
return false;
}
if(c=="")
{
alert("browse Product image");
document.f.proimage.focus();
return false;
}
```

```
if(d=="")
{
alert("enter Product Rate");
document.f.prorate.focus();
return false;
}
if(isNaN(d))
{
alert("enter Product Rate in Digits");
document.f.prorate.focus();
return false;
}
}
</script>
<link href="style.css" rel="stylesheet" media="all" type="text/css" />
</head>
<body>
<%
String sname=(String)session.getAttribute("sname");
String scname=(String)session.getAttribute("scname");
%>
<div id="wrapper">
<div id="container">
<div id="header">
<div id="logo"><br>
<br>
<br>
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<
strong><font color="#FFFFFF" size="+2" face="Georgia, Times New Roman,
Times, serif">
      Online Modeling of Proactive Moderation System for <br>
```

```
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Auction Fraud
Detection</font></strong></div>
</div>
<!-- /header -->
<div id="navbar">
<ul>
<li><a href="#" class="active">Products</a></li>
<li><a href="re_offers.jsp">Offers</a></li>
<li><a href="index.html">Logout</a></li>
<!--<li><a href="#">Admin</a></li>
<li><a href="#">Link</a></li>
<li><a href="#">Link</a></li>
<li><a href="#">Link</a></li>-->
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<font size="+1"
face="Georgia, Times New Roman, Times, serif"
color="#00FFCC">Welcome:&nbsp;</font><font size="+1"
color="#FF0000"><%=sname%></font>
</ul>
</div>
<!-- /navbar -->
<div id="main">
<div id="intro">
<!--<div id="sellerpic">
</div>-->
<!-- /jakepic -->
<div id="text"></div>
<!-- /text -->
<table height="350" align="center" width="700">
<trbgcolor="#CC3300">
```

```
<td width="410" bgcolor="#FBF7E1"><fieldset>
<legend><strong><em><font color="#FF0000" size="+2" face="Courier
New">New Products</font></em></strong></legend>
<form name="f" action="re_pro_insert.jsp" method="post" onsubmit="return
valid()">
<table cellpadding="10" cellspacing="10">
<tr>
<td colspan="2" align="center"><font size="2"><b>
<%
                                        String
message=request.getParameter("message");
if(message!=null &&message.equalsIgnoreCase("success"))
                                        {
out.println("<fieldset><font color='red'><blink>Product Successfully Registered
!</blink></font></fieldset>");
                                        }
                                  %>
</b></font></td>
</tr>
<tr>
<td width="161" align="center"><font color="#333366" size="+1"
face="Georgia, Times New Roman, Times, serif"><strong>Company
                Name:</strong></font></td>
<td width="162" align="center"><input type="text" name="comname"
value="<%=scname%>" id="b"></td>
</tr>
<tr>
<td align="center"><font color="#333366" size="+1" face="Georgia, Times New
Roman, Times, serif"><strong>Product
                Name:</strong></font></td>
<td align="center"><input type="text" name="proname" id="b"></td>
</tr>
```

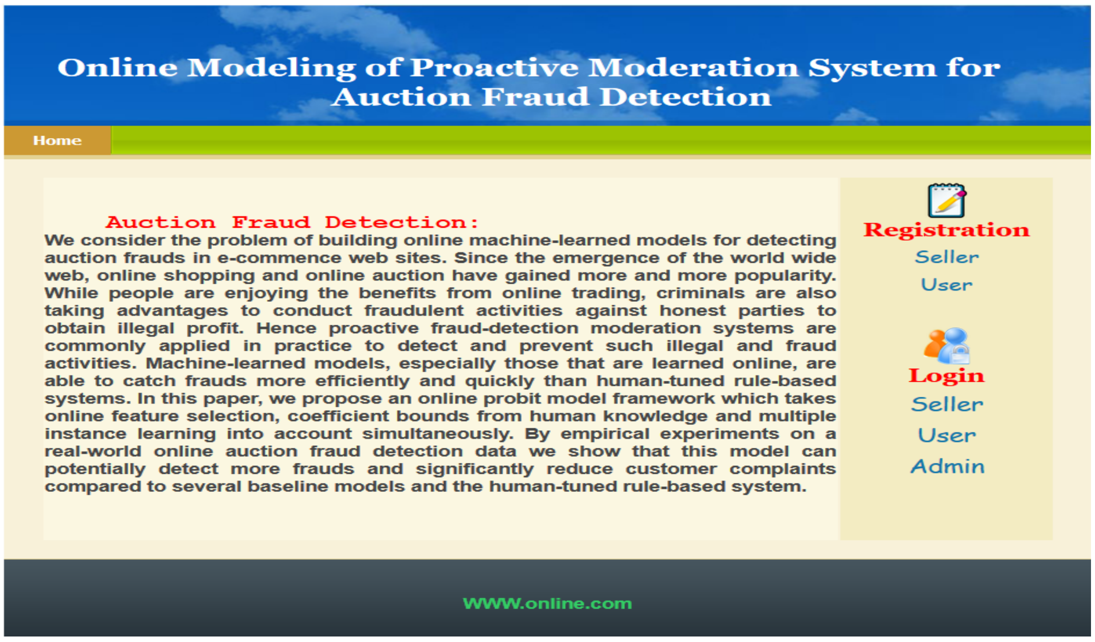
```
<tr>
<td align="center"><font color="#333366" size="+1" face="Georgia, Times New
Roman, Times, serif"><strong>Warranty
                Days:</strong></font></td>
<td align="center"><input type="text" name="wardate" id="b"></td>
</tr>
<tr>
<td align="center"><font color="#333366" size="+1" face="Georgia, Times New
Roman, Times, serif"><strong>Product
                Image:</strong></font></td>
<td align="center"><input type="file" name="proimage" size="8" id="b"></td>
</tr>
<tr>
<td align="center"><font color="#333366" size="+1" face="Georgia, Times New
Roman, Times, serif"><strong>Product
                Rate:</strong></font></td>
<td align="center"><input type="text" name="prorate" id="b"></td>
</tr>
<tr>
<td><input type="reset" name="re" value="clear" class="btn"></td>
<td><input type="submit" name="sub" value="submit" class="btn"></td>
</tr>
</table>
</form>
</fieldset></td>
<td width="200" bgcolor="#FBF7E1" align="center"><imgsrc="product/tv.png"
width="100" height="100">
                          <br><br><br><imgsrc="product/fridge.png" width="100"
height="200"></td>

</tr>
</table>
```

```
</div>
<!-- /intro -->
<div id="columns-wrapper">
<!-- /form-intro -->
<!-- /newsletter -->
<!-- /right -->
<!-- /left -->
</div>
<!-- /columns-wrapper -->
</div>
<!-- /main -->
<div id="footer">
<div id="footer-right">&nbsp;</div>
<div id="footer-left">&nbsp;</div>
<br>
<br>
<strong><font color="#33CC66">WWW.online.com</font></strong></div>
<!-- /footer -->
</div>
<!-- /container -->
</div>
<!-- /wrapper -->
</body>
</html>
```

## OUTPUT SCREENS

### Home Page



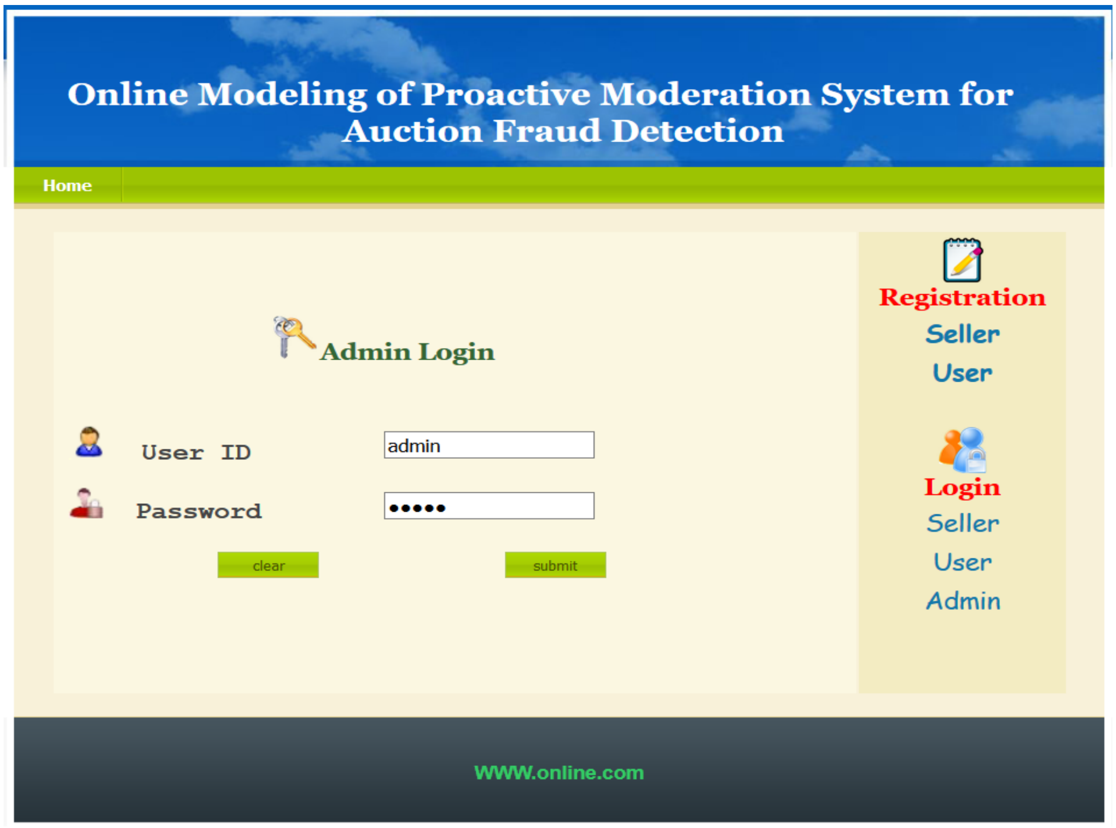
Online Modeling of Proactive Moderation System for Auction Fraud Detection

Auction Fraud Detection:

We consider the problem of building online machine-learned models for detecting auction frauds in e-commence web sites. Since the emergence of the world wide web, online shopping and online auction have gained more and more popularity. While people are enjoying the benefits from online trading, criminals are also taking advantages to conduct fraudulent activities against honest parties to obtain illegal profit. Hence proactive fraud-detection moderation systems are commonly applied in practice to detect and prevent such illegal and fraud activities. Machine-learned models, especially those that are learned online, are able to catch frauds more efficiently and quickly than human-tuned rule-based systems. In this paper, we propose an online probit model framework which takes online feature selection, coefficient bounds from human knowledge and multiple instance learning into account simultaneously. By empirical experiments on a real-world online auction fraud detection data we show that this model can potentially detect more frauds and significantly reduce customer complaints compared to several baseline models and the human-tuned rule-based system.

Fig.7.1 Screen for Home Page

### Admin Login

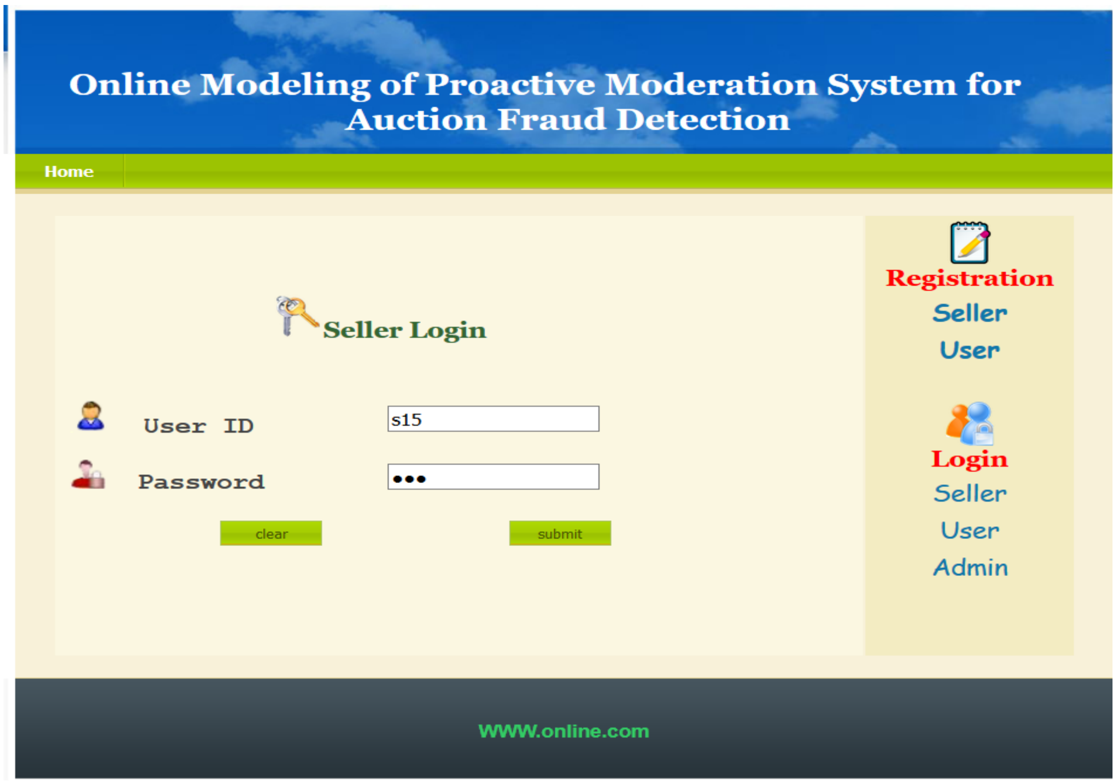


Online Modeling of Proactive Moderation System for Auction Fraud Detection

| | |
|---|---|
| User ID | admin |
| Password | ••••• |

Fig.7.2 Screen for Admin Login

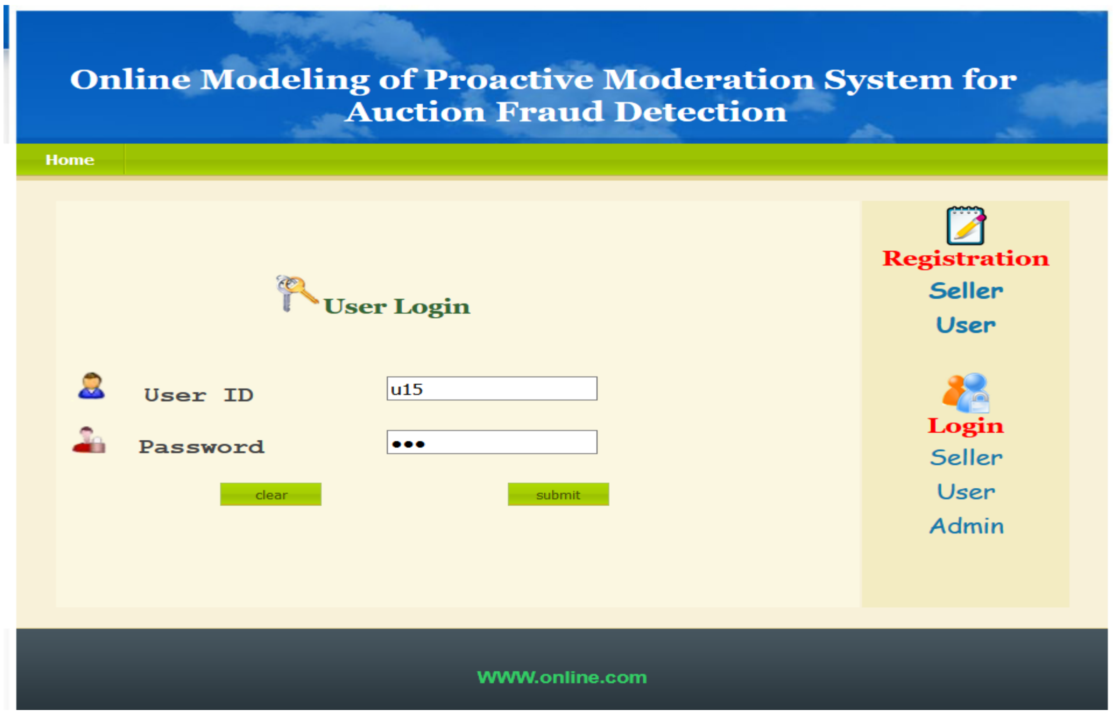
## Seller Login



User ID: s15
Password: •••

Fig.7.3 Screen for Seller Login

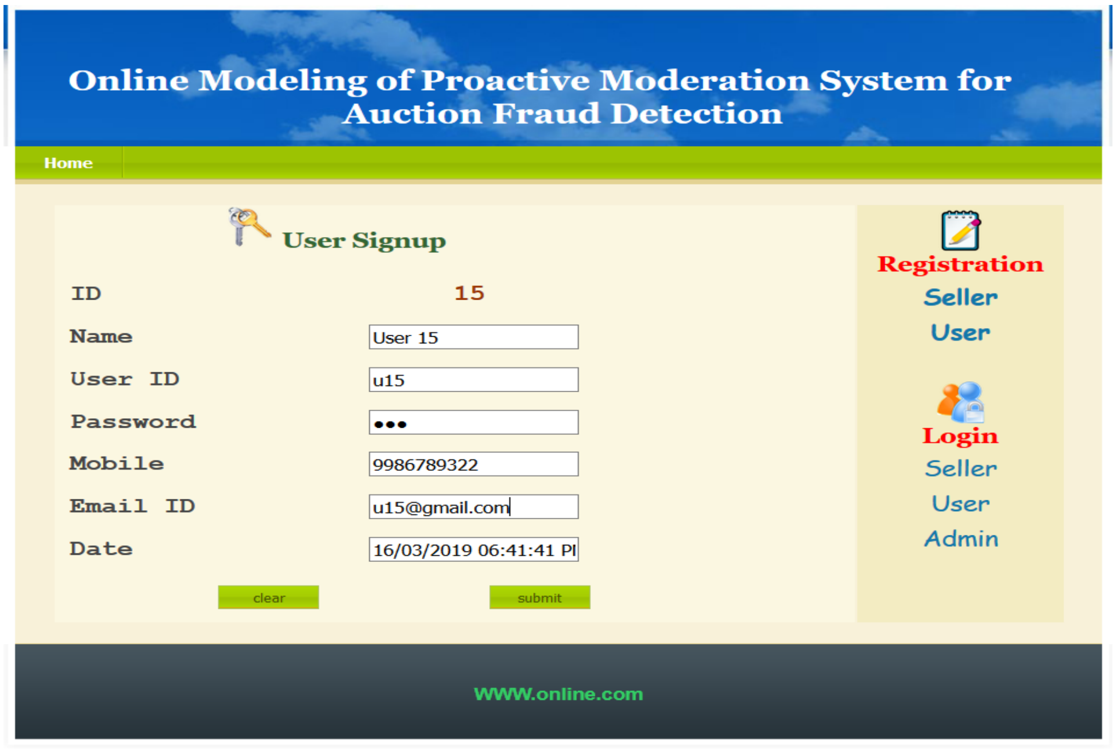
## User Login



User ID: u15
Password: •••

Fig.7.4 Screen for User Login

## User Signup

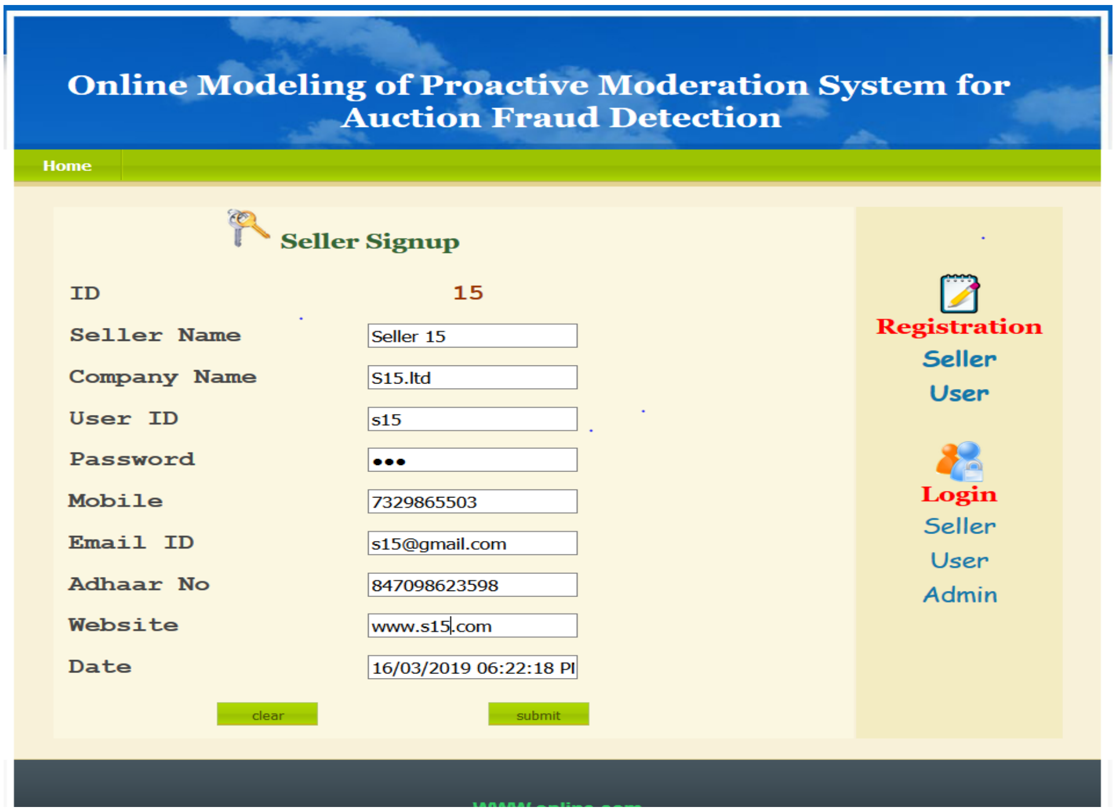

Online Modeling of Proactive Moderation System for Auction Fraud Detection
Home
Registration
Seller
User
Login
Seller
User
Admin
clear
submit
WWW.online.com

User Signup

| | |
|---|---|
| ID | 15 |
| Name | User 15 |
| User ID | u15 |
| Password | ●●● |
| Mobile | 9986789322 |
| Email ID | u15@gmail.com |
| Date | 16/03/2019 06:41:41 PI |

Fig.7.5 Screen for User Signup

## Seller Signup

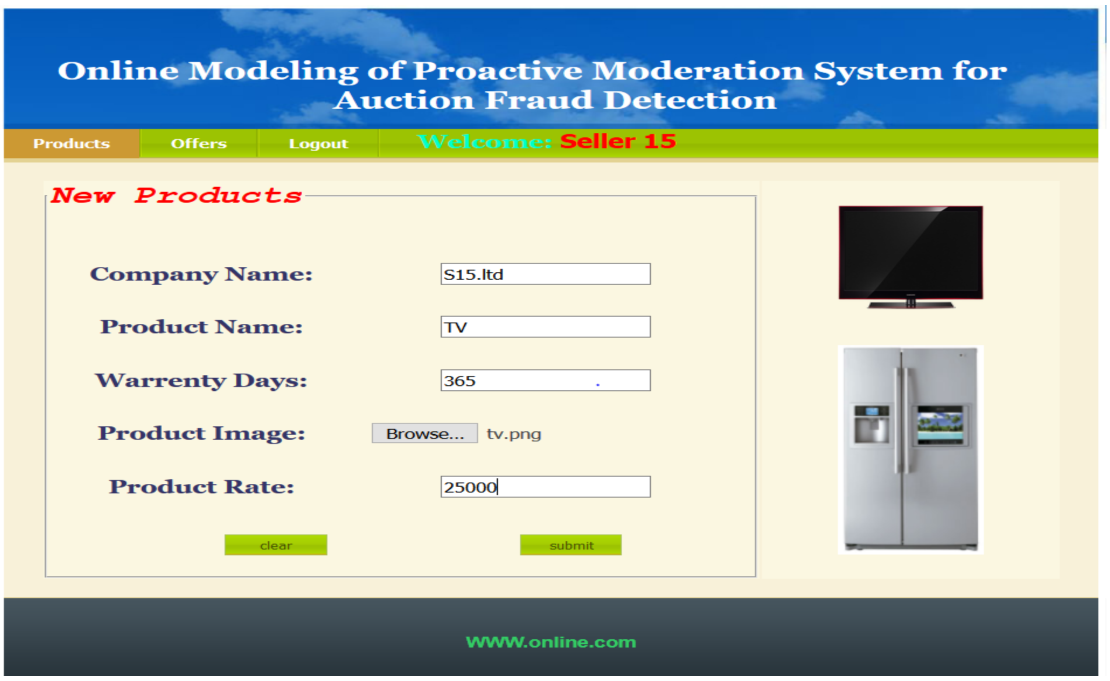

Online Modeling of Proactive Moderation System for Auction Fraud Detection
Home
Registration
Seller
User
Login
Seller
User
Admin
clear
submit

Seller Signup

| | |
|---|---|
| ID | 15 |
| Seller Name | Seller 15 |
| Company Name | S15.ltd |
| User ID | s15 |
| Password | ●●● |
| Mobile | 7329865503 |
| Email ID | s15@gmail.com |
| Adhaar No | 847098623598 |
| Website | www.s15.com |
| Date | 16/03/2019 06:22:18 PI |

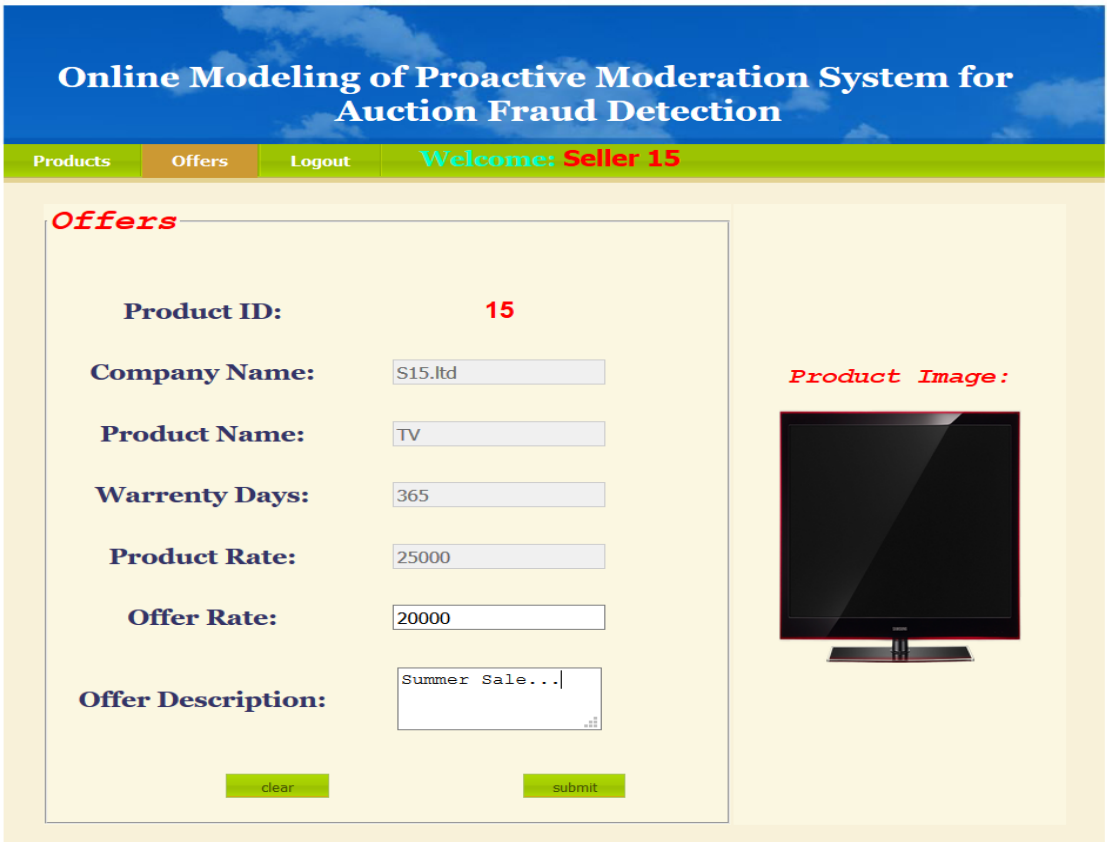
Fig.7.6 Screen for Seller Signup

## Add product and offers



| | |
|---|---|
| Company Name: | S15.ltd |
| Product Name: | TV |
| Warrenty Days: | 365 |
| Product Image: | tv.png |
| Product Rate: | 25000 |

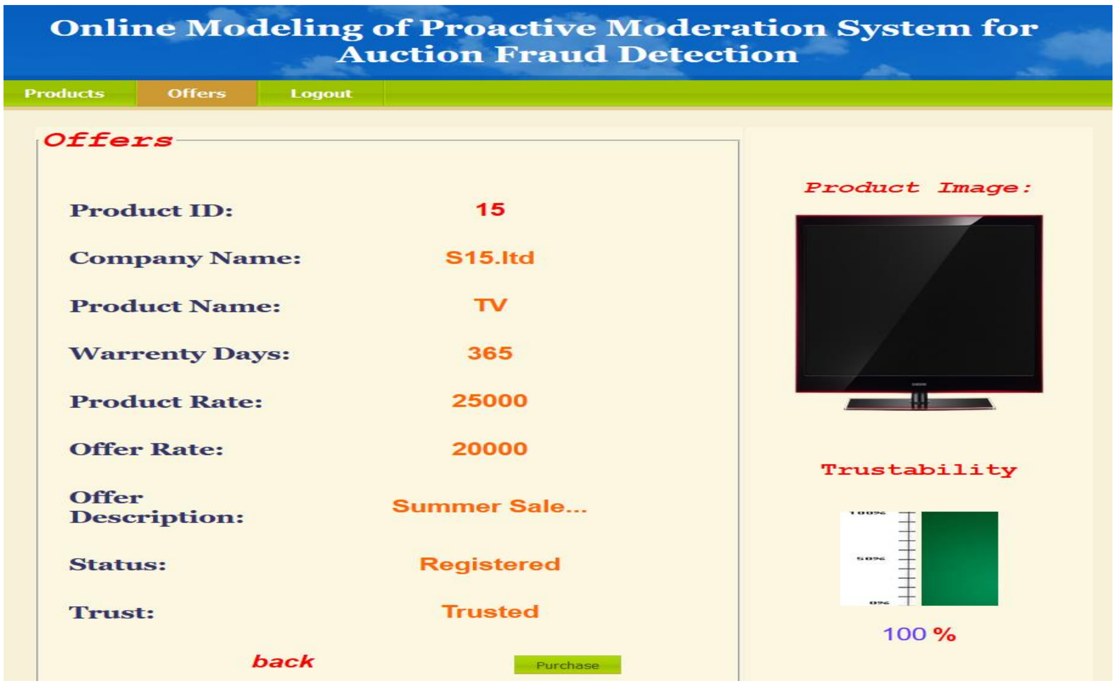
Fig.7.7 Screen for Add Products


Online Modeling of Proactive Moderation System for Auction Fraud Detection
Products
Offers
Logout
Welcome: Seller 15
Offers
Product Image:
clear
submit

| | |
|---|---|
| Product ID: | 15 |
| Company Name: | S15.ltd |
| Product Name: | TV |
| Warrenty Days: | 365 |
| Product Rate: | 25000 |
| Offer Rate: | 20000 |
| Offer Description: | Summer Sale... |

Fig.7.8 Screen for Add Offers

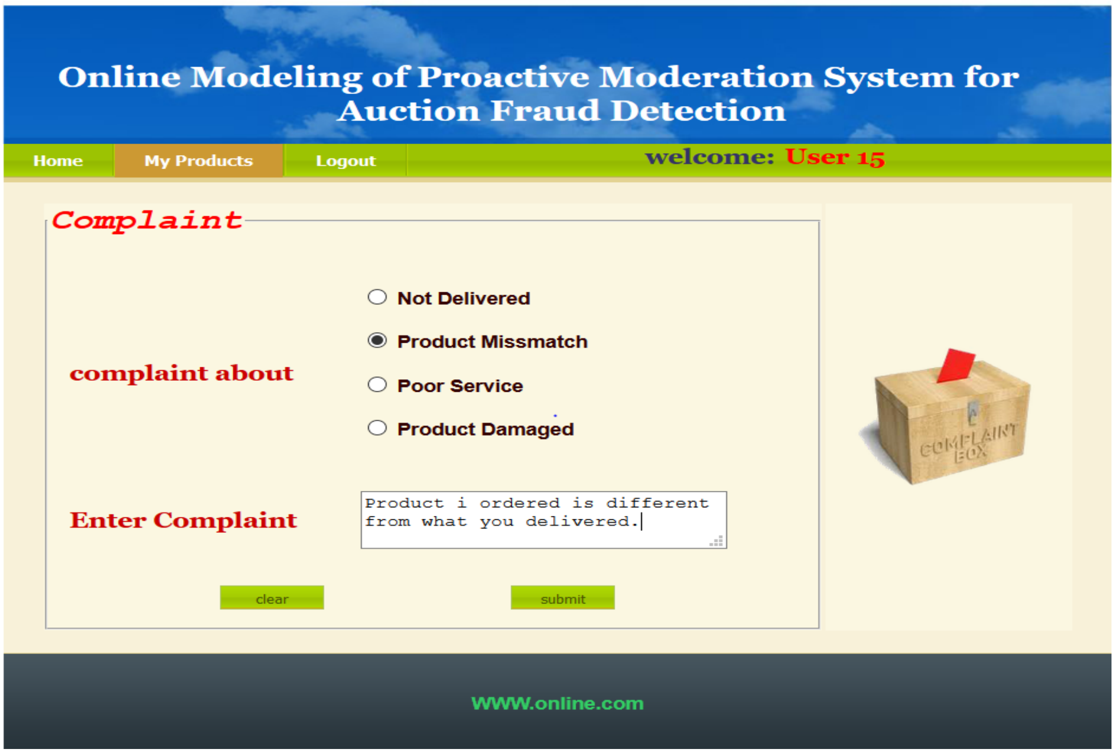
## User Purchasing a Product


Online Modeling of Proactive Moderation System for Auction Fraud Detection
Products
Offers
Logout
Offers
Product Image:
Trustability
100 %
back
Purchase

| | |
|---|---|
| Product ID: | 15 |
| Company Name: | S15.ltd |
| Product Name: | TV |
| Warrenty Days: | 365 |
| Product Rate: | 25000 |
| Offer Rate: | 20000 |
| Offer Description: | Summer Sale... |
| Status: | Registered |
| Trust: | Trusted |

Fig.7.9 Screen for Purchasing a Product

## User Registering a Complaint

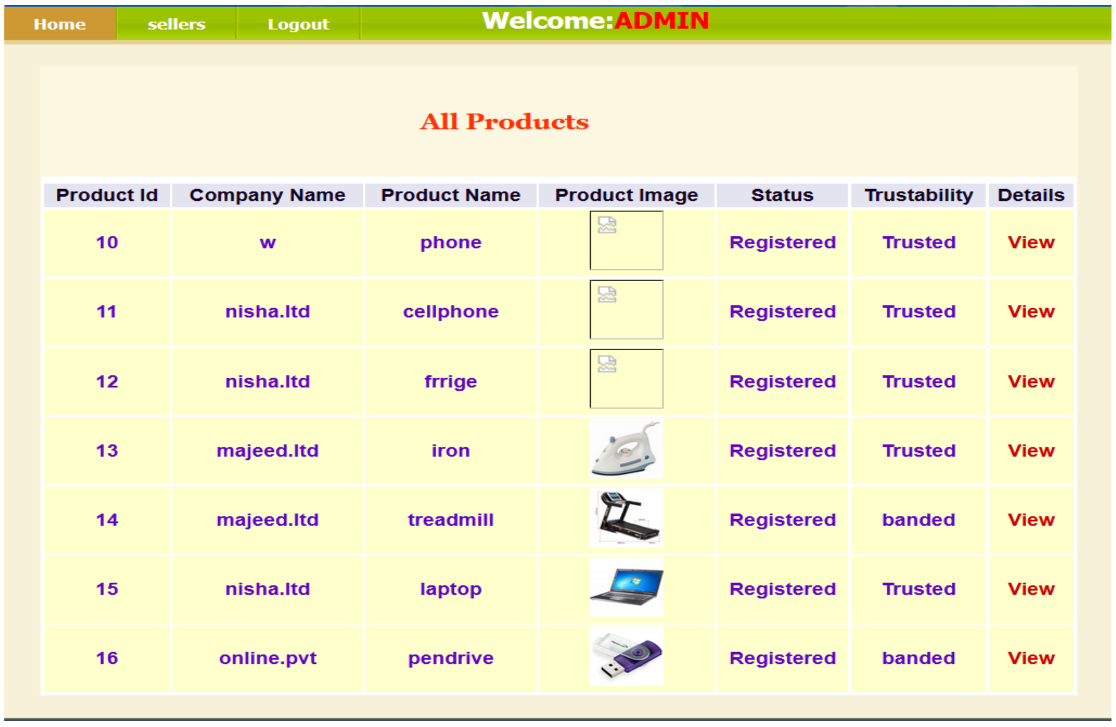

Online Modeling of Proactive Moderation System for Auction Fraud Detection
Home
My Products
Logout
welcome: User 15
Complaint
clear
submit
WWW.online.com

complaint about: ( ) Not Delivered [✓ Product Missmatch] ( ) Poor Service ( ) Product Damaged

Enter Complaint: Product i ordered is different from what you delivered.

Fig.7.10 Screen for User Registering a complaint

## Admin Panel

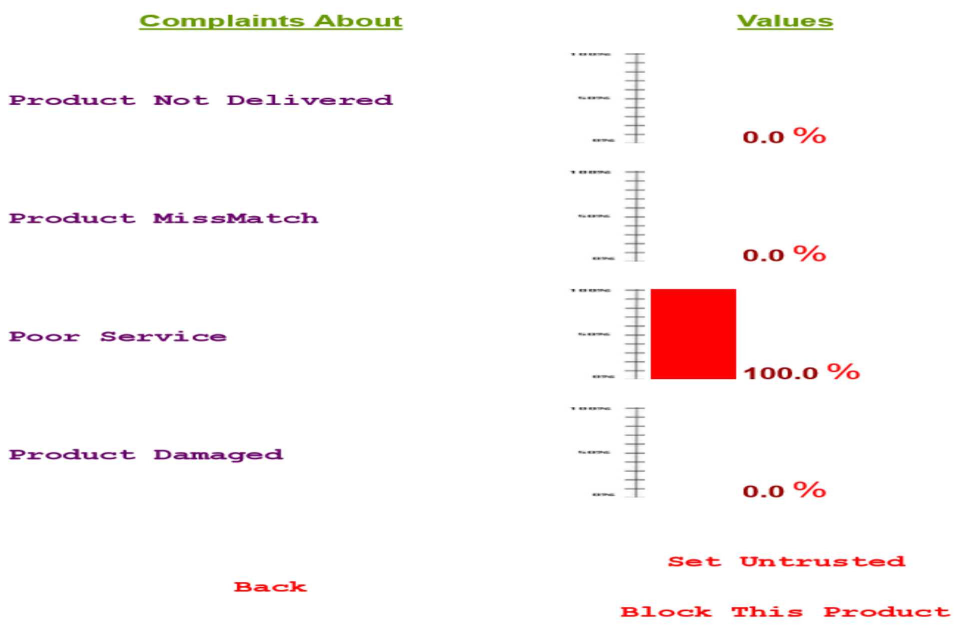


All Products

| Product Id | Company Name | Product Name | Product Image | Status | Trustability | Details |
|---|---|---|---|---|---|---|
| 10 | w | phone | | Registered | Trusted | View |
| 11 | nisha.ltd | cellphone | | Registered | Trusted | View |
| 12 | nisha.ltd | frrige | | Registered | Trusted | View |
| 13 | majeed.ltd | iron | | Registered | Trusted | View |
| 14 | majeed.ltd | treadmill | | Registered | banded | View |
| 15 | nisha.ltd | laptop | | Registered | Trusted | View |
| 16 | online.pvt | pendrive | | Registered | banded | View |

Fig.7.11 Screen for Admin Panel

## Admin Blocking a Product



| Complaints About | Values |
|---|---|
| Product Not Delivered | 0.0 % |
| Product MissMatch | 0.0 % |
| Poor Service | 100.0 % |
| Product Damaged | 0.0 % |

Fig.7.12 Screen for Admin Blocking a Product

# CHAPTER 8. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## 8.1 TYPES OF TESTING

### 8.1.1 Unit Testing

Unit test involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### 8.1.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### 8.1.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

| | |
|---|---|
| Invalid Input | : identified classes of invalid input must be rejected. |
| Functions | : identified functions must be exercised. |
| Output | : identified classes of application outputs must be exercised. |

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### 8.1.4 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### 8.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

### 8.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box.

You cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

### 8.1.7 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

**Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

### 8.1.8 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

### 8.1.9 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# CHAPTER 9. CONCLUSION

We build online models for the auction fraud moderation and etection system designed for a major Asian online auction website. By empirical experiments on a real-world online auction fraud detection data, we show that our proposed online probit model framework, which combines online feature selection, bounding coefficients from expert knowledge and multiple instance learning, can significantly improve over baselines and the human-tuned model. Note that this online modeling framework can be easily extended to many other applications, such as web spam detection, content optimization and so forth. Regarding to future work, one direction is to include the adjustment of the selection bias in the online model training process. It has been proven to be very effective for offline models in [38]. The main idea there is to assume all the unlabeled samples have response equal to 0 with a very small weight. Since the unlabeled samples are obtained from an effective moderation system, it is reasonable to assume that with high probabilities they are non-fraud. Another future work is to deploy the online models described in this paper to the real production system, and also other applications.

# CHAPTER 10. BIBLIOGRAPHY

[1] JAVA FUNDAMENTALS-A Comprehensive Introduction, Herbert Scheldt and Dale Skrien, TMH.

[2] JAVA FOR PROGRAMMERS, P.J.Deitel and H.M.Deitel, Pearson Education (OR) - JAVA: How to Program P.J.Deitel and H.M.Deitel, PHI.

[3] DATABASE MANAGEMENT SYSTEMS, Raghu Ramakrishnan, Johannes Gehrke, TMH, 3$^{rd}$ edition, 2003.

[4] SOFTWARE ENGINEERING, Ian Sommerville, 7$^{th}$ edition, Pearson Education.

## SITES REFERRED:

http://java.sun.com

http://www.sourcefordgde.com

http://www.networkcomputing.com/

http://www.roseindia.com/

http://www.java2s.com/

## REFERENCES

[1] D. Agarwal, B. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *Proceedings of the 18th international conference on WorldWide Web*, pages 21–30. ACM, 2009.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. *Advances in neural information processingsystems*, pages 577–584, 2003.

[3] C. Bliss. The calculation of the dosage-mortality curve. *Annals of Applied Biology*, 22(1):134–167, 1935.

[4] A. Borodin and R. El-Yaniv. *Online computation andcompetitive analysis*, volume 53. Cambridge University Press New York, 1998.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] R. Brent. *Algorithms for minimization without derivatives*. Dover Pubns, 2002.

[7] D. Chau and C. Faloutsos. Fraud detection in electronic auction. In *European Web Mining Forum (EWMF 2005)*, page 87.

[8] H. Chipman, E. George, and R. McCulloch. Bart: Bayesian additive regression trees. *The Annals ofApplied Statistics*, 4(1):266–298, 2010.

[9] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng. Unbiased online active learning in data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery anddata mining*, pages 195–203. ACM, 2011.

[10] C. Chua and J. Wareham. Fighting internet auction fraud: An assessment and proposal. *Computer*, 37(10):31–37, 2004.

[11] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Transactionson Pattern Analysis and Machine Intelligence*, pages 1631–1643, 2005.

[12] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge university press, 2006.

[13] T. Dietterich, R. Lathrop, and T. Lozano-P´erez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[14] Federal Trade Commission. Internet auctions: A guide for buyers and sellers. http://www.ftc.gov/bcp/conline/pubs/online/auctions.htm, 2004.

[15] J. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.