ONLINE AUCTION AND RENTING WEBSITE

A PROJECT REPORT

Submitted by

AJAY KUMAR - 161B015

SHIKHAR TIWARI - 161B213

VINEET SINGH - 161B266

Under the Guidance of: MR. AMIT KELOTRA



JUL - DEC 2018

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,

A-B ROAD, RAGHOGARH, DT. GUNA - 473226, M.P., INDIA

**Declaration by student**

I hereby declare that the work reported in the B. Tech. project entitled as “Online Auction and Renting System”, in partial fulfillment for the award of degree of B. Tech submitted at Jaypee University of Engineering and Technology, Guna, as per best of my knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation I will solely be responsible.

**Date:**  **Signature of the student**

**(Ajay Kumar)**

**(Vineet Kumar Singh)**

**(Shikhar Tiwari)**

**CERIFICATE**

This is to certify that the work titled “**ONLINE AUCTION AND RENTING WEBSITE**” submitted by “ **AJAY KUMAR, SHIKHAR TIWARI, VINEET SINGH**” in partial fulfillment for the award of degree of B. Tech of Jaypee University of Engineering & Technology, Guna has been carried out under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property right and copyright. Also, this work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern student will solely be responsible.

**Signature of Supervisor**

**(Mr. Amit Kelotra)**

**Assistant Professor**

**Date:**

**ACKNOWLEDGEMENT**

It has been great pleasure for us to develop a Web Application for Online Auction and Renting System. We have gathered sufficient knowledge and experience during this project. Most important we would like to thank our honorable teacher, senior head Dr. Ajay kumar, Dept. of CSE at Jaypee University, Guna who guided us to proper analysis of the system and helped to develop an elegant and efficient system. It was a great pleasure to study and work with many gifted people who influenced us in many ways. First of all, we would like to thank our project supervisor Mr. Amit Kelotra. He encouraged us to seek out the clearest and deepest description of theoretical ideas as well as practical implementations. We are very grateful to him for his continuous support, advice and guidance. Finally we would like to convey our special thanks to our parents and friends whom have always given us tremendous Support.

**Signature of the student**

**(Ajay Kumar)**

**(Vineet Kumar Singh)**

**(Shikhar Tiwari)**

**Date:**

**LIST OF FIGURES**

Figure 2.1. CSS styles …………………………………………………………………………… 7

Figure 2.2. Buttons ……………………………………………………………………………… 10

Figure 2.3. Form ………………………………………………………………………………... 11

Figure2.4. Navbars ………………………………………………………………………………11

Figure2.5. Cards …………………………………………………………………………………12

Figure 2.6. DOM Structure ………………………………………………………………………17

Figure 2.7. Concepts …………………………………………………………………………… 23

Figure 3.1. Use case diagram …………………………………………………………………… 26

Figure 3.2. DFD Symbols ………………………………………………………………………. 28

Figure 3.3. Level 0 ………………………………………………………………………………29

Figure 3.4. Level 1 ………………………………………………………………………………29

Figure 3.5. Class Diagram ………………………………………………………………………30

TABLES OF CONTENT

FRONT PAGE ……………………………………………………………………………… 1

DECLARATION BY STUDENT…………………………………………………………... 2

CERTIFICATE……………………………………………………………………………… 3

ACKNOWLEDGEMENT………………………………………...………………………… 4

LIST OF FIGURES ……………………………………………………………………………….. 5

**CHAPTER 1- INTRODUCTION** …………………….……………………………………….…. 8

1.1: NUMBER OF MODULE ……………….…………………………………….……. 9

1.2: DESCRIPTION OF MODULES ………….………………………….……….……. 9

1.2.1: Admin module ……………………...…………..…………………..…….... 9

1.2.2: Seller module ………………………………………………………………. 9

1.2.3: Buyer module ………………………………………………………………. 9

1.2.4: Visitor module …………………..……………………………..…………... 10

1.2.5: Security and Authentication ……..…………………………..…………… 10

**CHAPTER 2- TECHNOLOGY DESCRIPTION** ……………………………..……..…………. 11

2.1- HTML …………………………………………………..……………..……………… 11

2.1.1- HTML Elements ……………..………………..………………………….. 12

2.1.2- HTML Attributes ……………..……………………..……………………. 12

2.2- CASCADING STYLING SHEETS (CSS) …..………………..………………….. 12

2.2.1- Advantages of CSS …………………..…………………..………………. 12

2.2.2- SELECTORS ……………………………………..………..……………... 13

2.3- BOOTSTRAP ……………………………..…………………..…………………….. 15

2.3.1- Why to use bootstrap? …………………….…………………………..…. 15

2.3.2- What Bootstrap Package includes? ………..……………….…………... 15

2.3.3- Bootstrap components examples ……………………….……..………... 16

2.4- JAVASCRIPT ……………………………………………………………....……...... 18

2.4.1- What is JavaScript? …………………………………………..…..………. 18

2.4.2- Client-Side JavaScript …………………………………………....………19

2.4.3- Advantages of JavaScript …………………………..…………...……….. 19

2.4.4- JavaScript Development Tools ……………………….………………… 19

2.5- JQUERY ……………………………………………………………….……………. 20

2.5.1- What is jQuery? ………………………………………….………………. 20

2.5.2- How to use jQuery? ………………………………………….………….. 20

2.6- DOCUMENT OBJECT MODEL OR DOM ……………………………………. 21

2.6.1- DOM Compatibility …………………………………………………….. 22

2.7- NODE.JS …………………………………………………………………………… 23

2.7.1- What is Node.js? ………………………………………………………... 23

2.7.2- Features of Node.js ……………………………………………………... 24

2.7.3- Concepts …………………………………………………………………. 25

**CHAPTER 3- SYSTEM ENTITY RELATION** ……………………………….……….….…… 26

3.1- WHAT IS A DATA FLOW DIAGRAM (DFD) ………………….…….…….….. 26

3.1.1- Symbols and Notations used in DFD’s …………………….….…….….. 26

3.1.2- DFD Rules and Tips ………………………………………………...……. 26

3.1.3- DFD levels and Layers ……………………………………………...……. 27

3.2- DATA FLOW DAIGRAM …………………………………………………...……. 28

3.2.1- What is use case diagram? …………………………………………...…... 28

3.2.2- When to apply use case diagrams …………………………………..…… 29

3.2.3- Identifying Actors and Use Cases …………………………………..…… 30

3.2.4- DFD levels and layers ……………………………………………….............. 30

3.3- UML Class DIAGRAM ………………………………………………………..…… 32

**CHAPTER 4- FUTURE ENHANCEMENT AND CONCLUSION** …...................................... 33

4.1- IMPLEMENTED FUNCTIONALITY …………………………………………… 33

4.2- KNOWLEDGE ACQUIRED ……………………………………………………… 34

4.3- FUTURE WORK …………………………………………………………………… 34

REFERENCES……………………………………………………………………………….35

**CHAPTER 1**

**INTRODUCTION**

The purpose of this project is to build an “online auction and renting system”, a place for buyers and sellers to come together and trade almost anything.An auction is a public scale in which goods are sold to the highest bidder and renting is the process in which buyer pay someone for the use of any goods or property. Auction involves bidding.

Online Bidding has become more wide spread in all sorts of industrial usage. It not only includes the product or goods to be sold, it also has services which can be provided. Due to their low cost this expansion made the system to grow.

There are several different types of bidding and certain rules exist for each bidding. There are variations for bidding which may include minimum price limit, maximum price limit and time limitations. Depending upon the bidding method bidder can participate remotely or in person. Remote auction and renting include participating through telephone or internet.

In fact, the system consists in a web-portal where registered users can propose new auctions, place bids in order to buy the items on auction, send messages to other users and receive automatically news via e-mail. Therefore in order to participate in an online auction and renting process, the user need to register on our site (user cannot use same e-mail again and again for registration).

Auctions have a name, a description, possibly a photo (of the related item) uploaded by users and an end period: users cannot place bids when the auction interval (start - end period) ends, but, in case there were no offers for an item, there is the possibility to extend the interval.

Online bidding has become a standard method for procurement process. Bidders can be maintained in a single database according to the preference, and they can be monitored. User’s data can be maintained in a confidential way for validity and integrity of contractual documentation. Neat reporting reduces paperwork, postage, photocopying and time beneficial. Multiple bidders can be communicated with a great ease. This system allows multiple bids by single users. Online bidding is based upon lowest or the highest price which is initiated but not the best value for the product. Although there is a chance to fix the criteria against the fact expected to have desired value by the seller.

* 1. **No. of module**

The system after careful analysis has been identified to be presented with the following modules:

* Admin Module.
* Seller Module.
* Buyer Module.
* Visitor Module.
* Security and authentication.
  1. **Description of modules**

The following sections include the descriptions for modules.

1.2.1 Admin Module

This module provides the complete information related to products for sale and the buyers can bid for the products and can own them. All this has to be provided and maintained by the admin because the complete auction process is to be kept under control till the product sale gets confirmed.

1.2.2 Seller Module

Sellers want a place where seller can sale their products at a higher price and get maximum benefit out of that. This is the place where seller can display all his products and sell them.

1.2.3 Buyer Module

The people always want different things to purchase but in the local market they can have local products only. But in this application buyer can buy any product from any part of the world at a very best competitive price and own the product

.

1.2.4 Visitor Module

Visitor is nothing but all the people who visits this application online. They can know the information of all the products, which are for sale under this application.

1.2.5 Security and Authentication

The security and authentication is as follows:

* Login as buyer or seller or administrator.
* Change password.
* Forgot Password.
* Registration for buyer / seller.

**CHAPTER 2**

**TECHNOLOGY DESCRIPTION**

* 1. **HTML**

HTML stands for Hyper**t**ext Markup Language, and it is the most widely used language to write Web Pages.

* Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
* As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

## HTML Elements

An HTML element usually consists of a **start** tag and **end** tag, with the content inserted in between:

<tagname> Content goes here </tagname>

The HTML **element** is everything from the start tag to the end tag, e.g.:

<p> This is my paragraph.</p>

## 2.1.2 HTML Attributes

* All HTML elements can have attributes
* Attributes provide additional information about an element
* Attributes are always specified in the start tag
* Attributes usually come in name/value pairs like: name="value"

**2.2 CASCADING STYLE SHEETS (CSS)**

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

## 2.2.1 Advantages of CSS

* CSS saves time − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
* Pages load faster − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
* Easy maintenance − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
* Superior styles to HTML − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts −

Selector − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

Property − A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

Value− Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule Syntax as follows:

selector {property: value}

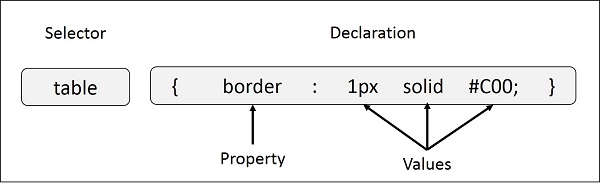


Figure 2.1. CSS styles

## 2.2.2 SELECTORS

## a) The Type Selectors:

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings −

h1 {color: #36CFFF;}

## b) The Universal Selectors:

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type −

\* {color: #000000; }

This rule renders the content of every element in our document in black.

## c) The Descendant Selectors:

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to <em> element only when it lies inside <ul> tag.

ul em {color: #000000;}

## d) The Class Selectors:

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

. black {color: #000000;}

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example −

h1. black {color: #000000;}

This rule renders the content in black for only <h1> elements with class attribute set to black.

You can apply more than one class selectors to given element. Consider the following example −

<p class = "center bold">

This para will be styled by the classes center and bold. </p>

## e) The ID Selectors:

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

#black {color: #000000; }

**2.3 BOOTSTRAP**

Bootstrap is the most popular front end framework in the recent time. It is sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript. This tutorial will teach you the basics of Bootstrap Framework using which you can create web projects with ease.

## 2.3.1 Why Use Bootstrap?

* Mobile first approach − Bootstrap 3, framework consists of Mobile first styles throughout the entire library instead them of in separate files.
* Browser Support − It is supported by all popular browsers.
* Easy to get started − With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has a good documentation.
* Responsive design − Bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles.
* Provides a clean and uniform solution for building an interface for developers.
* It contains beautiful and functional built-in components which are easy to customize.
* It also provides web based customization.
* And best of all it is an open source.

## 2.3.2 What Bootstrap Package Includes?

* Scaffolding − Bootstrap provides a basic structure with Grid System, link styles, and background. This is covered in detail in the section Bootstrap Basic Structure
* CSS − Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system. This is covered in detail in the section Bootstrap with CSS.
* Components − Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more. This is covered in detail in the section Layout Components.
* JavaScript Plugins − Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one. This is covered in details in the section Bootstrap Plugins.
* Customize − You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

2.3.3 Bootstrap components Examples:

Buttons**:**

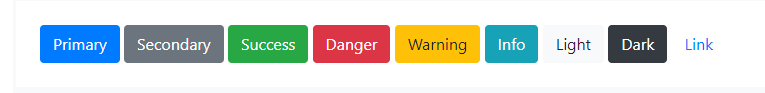
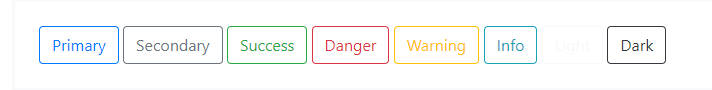
Use Bootstrap’s custom button styles for actions in forms, dialogs, and more with support for multiple sizes, and states.

Figure 2.2. Buttons

# Forms:

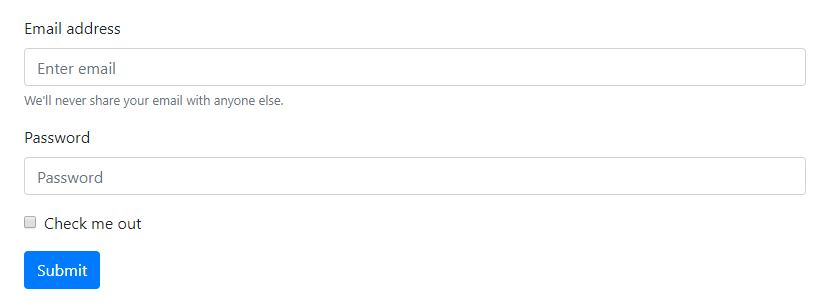
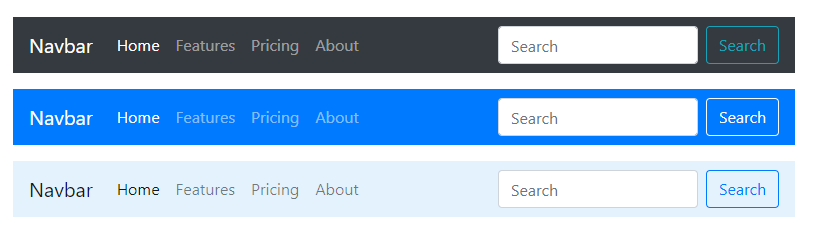


Figure 2.3. Form

# Navbar:



# Cards:

A **card** is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options. If you’re familiar with Bootstrap 3, cards replace our old panels, wells, and thumbnails. Similar functionality to those components is available as modifier classes for cards.

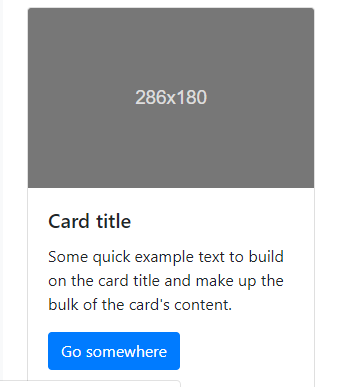


figure 2.5. Cards

**2.4 JAVASCRIPT**

## 2.4.1 What is JavaScript?

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

The [ECMA-262 Specification](http://www.ecma-international.org/publications/index.html) defined a standard version of the core JavaScript language.

* JavaScript is a lightweight, interpreted programming language.
* Designed for creating network-centric applications.
* Complementary to and integrated with Java.
* Complementary to and integrated with HTML.
* Open and cross-platform

## 2.4.2 Client-side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

## 2.4.3 Advantages of JavaScript

* The merits of using JavaScript are −
* Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.
* Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

## 2.4.4 JavaScript Development Tools

One of major strengths of JavaScript is that it does not require expensive development tools. You can start with a simple text editor such as Notepad. Since it is an interpreted language inside the context of a web browser, you don't even need to buy a compiler.

To make our life simpler, various vendors have come up with very nice JavaScript editing tools. Some of them are listed here −

* Microsoft FrontPage − Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
* Macromedia Dreamweaver MX − Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.

**2.5 JQUERY**

## 2.5.1 What is jQuery?

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto: Write less, do more. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery −

* DOM manipulation − The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.
* Event handling − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
* AJAX Support − The jQuery helps you a lot to develop a responsive and feature rich site using AJAX technology.

## 2.5.2 How to use jQuery?

There are two ways to use jQuery.

* Local Installation − You can download jQuery library on your local machine and include it in your HTML code.
* CDN Based Version − you can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

**2.6 DOCUMENT OBJECT MODEL OR DOM**

Every web page resides inside a browser window which can be considered as an object.

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.

The way a document content is accessed and modified is called the Document Object Model, or DOM. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

* Window object − Top of the hierarchy. It is the outmost element of the object hierarchy.
* Document object − Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
* Form object − Everything enclosed in the <form>...</form> tags sets the form object.
* Form control elements − The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

Here is a simple hierarchy of a few important objects –



Figure 2.6. DOM Structure

There are several DOMs in existence. The following sections explain each of these DOMs in detail and describe how you can use them to access and modify document content.

* [The Legacy DOM](https://www.tutorialspoint.com/javascript/javascript_legacy_dom.htm) − This is the model which was introduced in early versions of JavaScript language. It is well supported by all browsers, but allows access only to certain key portions of documents, such as forms, form elements, and images.
* [The W3C DOM](https://www.tutorialspoint.com/javascript/javascript_w3c_dom.htm) − This document object model allows access and modification of all document content and is standardized by the World Wide Web Consortium (W3C). This model is supported by almost all the modern browsers.
* [The IE4 DOM](https://www.tutorialspoint.com/javascript/javascript_ie4_dom.htm) − This document object model was introduced in Version 4 of Microsoft's Internet Explorer browser. IE 5 and later versions include support for most basic W3C DOM features.

## 2.6.1 DOM compatibility

If you want to write a script with the flexibility to use either W3C DOM or IE 4 DOM depending on their availability, then you can use a capability-testing approach that first checks for the existence of a method or property to determine whether the browser has the capability you desire. For example −

if (document.getElementById) {

// If the method exists, use it

}

else if (document.all) {

// If the all[] array exists, use it

}

else {

// Otherwise use the legacy DOM

}

**2.7 NODE.JS**

## 2.7.1 What is Node.js?

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. The definition of Node.js as supplied by its [official documentation](https://nodejs.org/) is as follows −

Node.js is a platform built on [Chrome's JavaScript runtime](https://code.google.com/p/v8/) for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

## 2.7.2 Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

* Asynchronous and Event Driven − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* Very Fast − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* Single Threaded but Highly Scalable − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* No Buffering − Node.js applications never buffer any data. These applications simply output the data in chunks.
* License − <https://raw.githubusercontent.com/joyent/node/v0.12.0/LICENSE>

## 2.7.3Concepts

The following diagram depicts some important parts of Node.js which we will discuss in detail in the subsequent chapters.



Figure 2.7. Concepts

**CHAPTER 3**

**FUNCTIONAL REQUIREMENT AND SYSTEM DESIGN**

The first step of the project is to find the functional requirement of the on line auction portal with the help of techniques like Use Cases and User Stories. After having found the functional requirements, the project goes on with the system design using the following techniques: the UML class diagram, the EER diagram for the database design and the page flow diagram.

**3.1 USE CASE DIAGRAM**

3.1.1 What is use case?

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

1. Scenarios in which your system or application interacts with people, organizations, or external systems
2. Goals that your system or application helps those entities (known as actors) achieve
3. The scope of your system

## 3.1.2 When to apply use case diagrams?

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

1. Representing the goals of system-user interactions
2. Defining and organizing functional requirements in a system
3. Specifying the context and requirements of a system
4. Modeling the basic flow of events in a use case

3.1.3 Identifying Actors and Use cases:

**Actor –**

1. User
2. Administrator

**Use cases for user** –

1. Login
2. Register
3. Search
4. post item for auction and rent
5. check item user try to buy/sell
6. place bid

**Use cases for Administrator** –

1. Login
2. manage item
3. manage user
4. manage categories

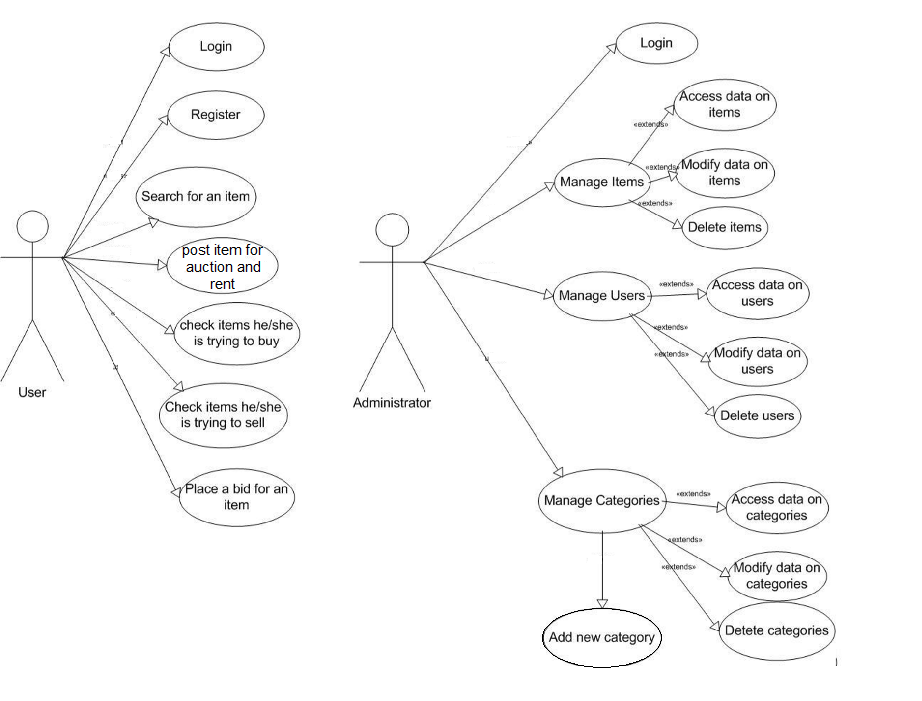


Figure 3.1: Use case diagram

**3.2 DATA FLOW DIAGRAM**

## 3.2.1 What is a data flow diagram (DFD)?

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

## 3.2.2 Symbols and Notations Used in DFDs

Two common systems of symbols are named after their creators:

* Yourdon and Coad
* Yourdon and DeMarco
* Gane and Sarson

Using any convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

1. **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
2. **Process:**any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”
3. **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”
4. **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

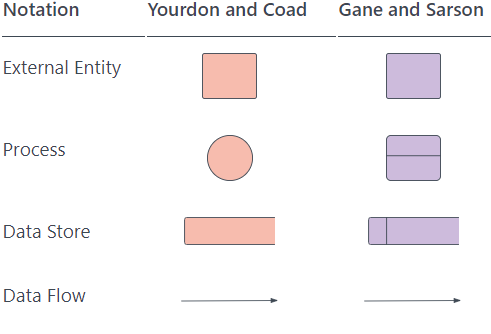


Figure 3.2. DFD Symbols

## 3.2.3 DFD rules and tips

* Each process should have at least one input and an output.
* Each data store should have at least one data flow in and one data flow out.
* Data stored in a system must go through a process.
* All processes in a DFD go to another process or a data store.

## 3.2.4 DFD levels and layers

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece.  DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

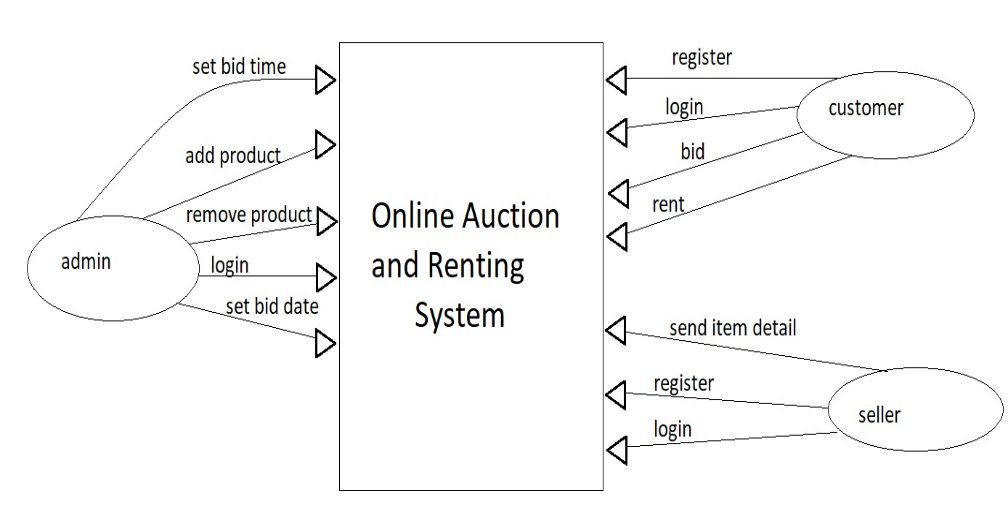
* DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

Figure 3.3. Level 0

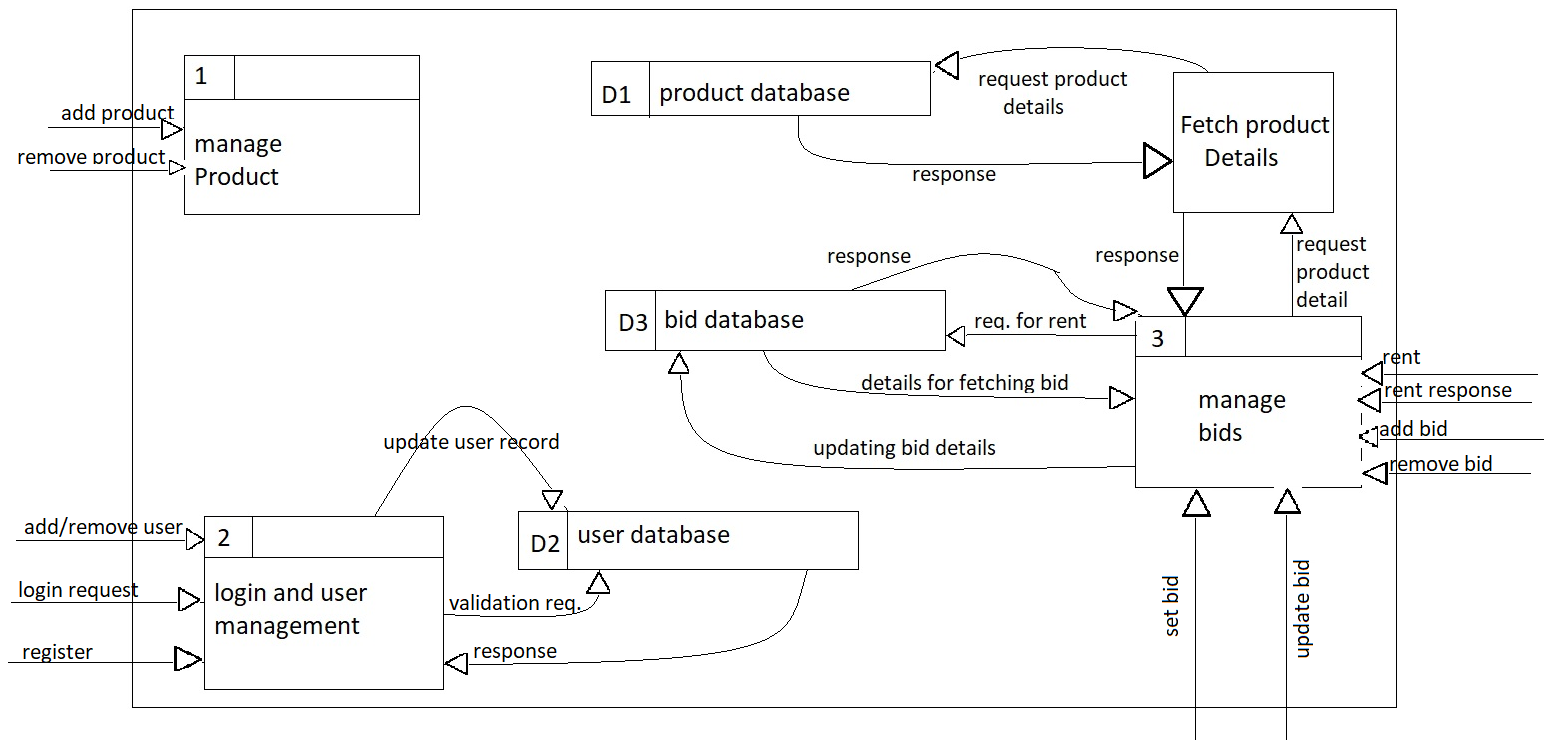
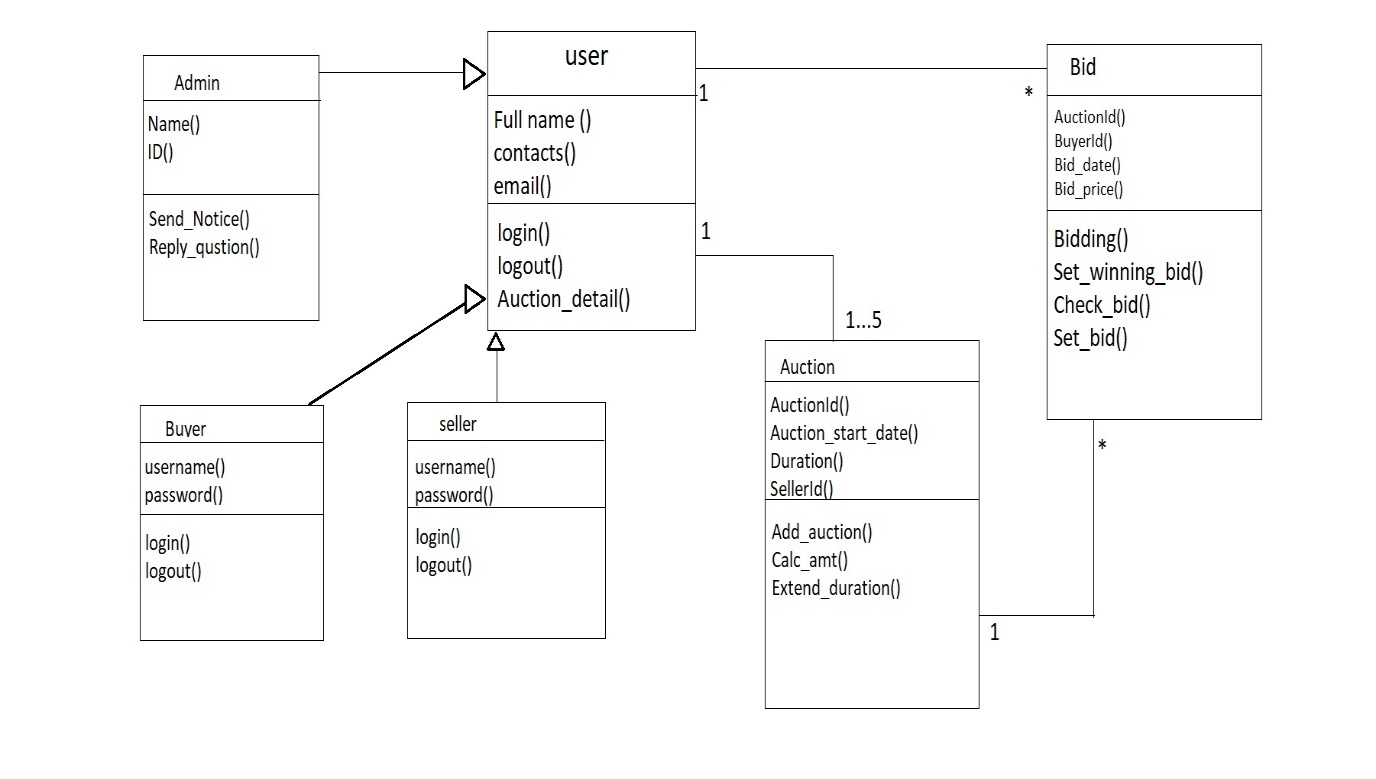
* DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram.

Figure 3.4. level 1

You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes. Level 1

**3.3 UML Class Diagram**

Class diagrams are one of the most useful types of diagrams in UML as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects. The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.



**5 Conclusion and Future Enhancement**

Under this we concluded about what we have achieved at the end of our project. Here it will be explained what we have realized with respect to the requirements, the problems found during the

Figure 3.5. Class Diagram

**CHAPTER 4**

**FUTURE ENHANCEMENT AND CONCLUSION**

Under this we conclude what we have achieved at the end of this project. Here it will be explained what we have realized with respect to the requirements, the problems found during the development of the system, what we have learned from the realization of the project and the possible improvements to the online auction and renting systems.

**4.1 IMPLEMENTED FUNCTIONALITY**

User side requirements:

* Home page
* Registration
* Login
* Item page
* Bid
* Sell

Administration side requirements:

* Administrator login
* Manage Item
* Manage users
* Manage categories

As mentioned above, all requirements have been implemented except multiuser simultaneous biding system. We did not implement these two requirements for time constraints and lack of knowledge about the used languages.

**4.2 KNOWLEDGE ACQUIRED**

During the development we learned how to leverage our knowledge of HTML, CSS, Javascript and JQuery to implement an application that is interactive and stable at the same time by using the properties of the used technologies.

We used different backend and frontend technologies to implement the application, the frontend technology enabled us to implement an interactive web application that can handle many users at the same time because as the users grow on the system we needed to understand how can we balance the load on the system, how efficiently we can manage multiple users logging in to the system at the same time and interactive with the system, that was a challenge for us and we learned a lot implement the services to our system.

**4.3 FUTURE WORK**

The online auction and renting portal works very well in all of its implemented functionality. However, some future works can be done on the existing system:

* Add simultaneous multiuser bidding feature to the project
* Add search field that provide ease to user to find their product
* Make portal more interactive to the user

**REFERENCES**

1. E. FREEMAN AND E. FREEMAN, Head First HTML with CSS & XHTML, O’Reily Media, Sebastopol, California, 2005.
2. GAMPASOFT, Quality assurance, Gampasoft.com, http://www.gampasoft.com/

quality.html, accessed May 2011, n.d.

1. N. RICHARDS AND S. GRIFFITH, JBoss–A Developer’s Notebook, O’Reilly Media,

Sebastopol, California, 2005.

1. D. THAU, The Book of JavaScript 2nd Edition: A Practical Guide to Interactive Web

Pages, No Starch Press, San Francisco, California, 2006.

1. J. TREVATHAN and W. READ, Detecting collusive shill bidding, in Proceedings of the

Fourth International Conference on Information Technology: New Generations, Las

Vegas, Nevada, 2007.

1. W.WANG, H. ZOLTÁN, AND A. B.WHINSTON**,** Shill bidding in multi-round online

Auctions, in Proceedings of the 35th Hawaii International Conference on System

Sciences, Big Island, Hawaii, 2002.

1. Cascading Style Sheets. Read 1.06.2016. <http://www.w3schools.com/css/css_intro.asp>
2. Hypertext Mark-up Language, Introduction. Read 29.05.2016. <http://www.w3schools.com/html/html_intro.asp>
3. JavaScript programming language, JS tutorial. Read 03.06.2016. <http://www.w3schools.com/js/>
4. Web development. Read <https://www.udemy.com/the-web-developer-bootcamp>
5. Sendall and A. Strohmeier: “From Use Cases to System Operation Specifications”. In S. Kent and A. Evans (Eds.), UML’2000 - The Unified Modeling Language: Advancing the Standard, Third International Conference, York, UK, October 2-6, 2000, pp. 1–15, Lecture Notes in Computer Science **1939**, Springer Verlag, 2000.