

Bachelor's Thesis

Information Technology

2011

Kenneth Emeka Odoh

DESIGN AND IMPLEMENTATION OF A WEB- BASED AUCTION SYSTEM



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS (UAS) | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Networking

Date of completion of the thesis | 12th January 2012

Instructor: Ferm Tiina

Kenneth Emeka Odoh

DESIGN AND IMPLEMENTATION OF A WEB-BASED AUCTION SYSTEM

Electronic auction has been a popular means of goods distribution. The number of items sold through the internet auction sites have grown in the past few years. Evidently, this has become the medium of choice for customers.

This project entails the design and implementation of a web-based auction system for users to trade in goods. The system was implemented in the Django framework. On account that the trade over the Internet lacks any means of ascertaining the quality of goods, there is a need to implement a feedback system to rate the seller's credibility in order to increase customer confidence in a given business. The feedback system is based on the history of the customer's rating of the previous seller's transactions. As a result, the auction system has a built-in feedback system to enhance the credibility of the auction system.

The project was designed by using a modular approach to ensure maintainability. There is a number of engines that were implemented in order to enhance the functionality of the auction system. They include the following: commenting engine, search engine, business intelligence (user analytic and statistics), graph engine, advertisement engine and recommendation engine.

As a result of this thesis undertaking, a full-fledged system robust enough to handle small or medium-sized traffic has been developed to specification.

KEYWORDS:

(Django, auction)

FOREWORD

I would like to thank Mrs. Tiina Ferm for making the time out of her busy schedule to supervise this thesis. I would like to thank Mr. Anselm Ogbunugafor and Mr. Abdoulie Sidibeh for explaining some of the confusing concepts in web programming. This knowledge proved invaluable to the successful completion of the project. I would also like to thank my mother, Mrs. Felicia Odoh, for the support that she gave while preparing this work. Lest I forget, let me express my thanks to Mr. Joseph K. Muli for helping me create those very interesting and beautiful 2D graphic images that were used in the project.

Spring 2012

Kenneth Emeka Odoh

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	Core Software Used In Project	2
1.2	Thesis Overview	4
2	LITERATURE REVIEW	6
2.1	Feedback system	7
2.2	Cryptology	9
2.3	Web Security	9
2.4	Optimization	12
2.5	Web server	14
2.5.1	HTTP	16
3	DESIGN AND IMPLEMENTATION OF THE CORE ENGINE	17
3.1	Choice of Programming Language	17
3.2	Choice of Django	18
3.3	Use Cases and Multiple Views / Requests	19
3.4	Bidding Engine	22
3.4.1	Resolving of Bids	25
3.4.2	Django-Celery for scheduling	25
4	DESIGN AND IMPLEMENTATION OF OTHER ENGINES	27
4.1	Commenting Engine	27
4.2	Search Engine	28
4.3	Business Intelligence (User analytics and statistics)	29
4.4	Graph Engine	30
4.5	Advertisement Engine	31
4.6	Recommendation Engine	32
4.7	Notification	33
4.8	Database Design	36
4.9	Performance Optimization	37
4.10	Scalability	39
5	IMPLEMENTATION OF WEB SECURITY AND OPTIMIZATION	42
5.1	Web Security	42

5.2	Ways to Prevent Security Attacks	43
5.3	Safe Practices	44
5.4	Optimization	45
5.4.1	Search Engine Optimization	45
5.5	Saving of Data	46
5.6	Concurrency	48
5.7	Web Deployment	49
6	CONCLUSION	52
6.1	Result	52
6.2	Recommendations	53
	REFERENCES	55
	APPENDIX 1.0 - Bidding system	57
	APPENDIX 1.1 (views.py)	57
	APPENDIX 1.2 (models.py)	69
	APPENDIX 1.3 (url.py)	71
	APPENDIX 1.4 (extra.py)	74
	APPENDIX 1.5 (forms.py)	76
	APPENDIX 1.6 (admin.py)	80
	APPENDIX 1.7 (tasks.py)	80
	FIGURES	
	Figure 1.0 Process diagram of auction.	6
	Figure 3.0 Description of model –view- controller	20
	Figure 3.1 Use case showing how the roles of users	22
	Figure 3.2 Use case showing the possible actions for a customer	24
	Figure 3.3 Sequence diagram for bidding	24
	Figure 3.4 Bidding form for making new bids.	26
	Figure 4.0 Use case for searching and commenting	28

Figure 4.1 Use case for the administrator.	30
Figure 4.2. Scatter Diagram of number of online user against number of bids	31
Figure 4.3. Advertisement creation form.	32
Figure 4.4 Comment form	34
Figure 4.5 Home page of the auction system	35
Figure 4.6 Entity relationship diagram	36
Figure 4.6 Relational Model	37

TABLES

Table 1.1 Software used in the project	3
Table 3.0 List of the Engines used in the project	21

List of terms (in alphabetical order)

Amazon	This is a popular web-based retail store.
Auction	These are items that are displayed for bidding.
ACID	Atomicity, Consistency, Isolation and Durability
Bid	This is the amount that the customer is willing to pay for the item on display.
Bid queue	This is a list of bid objects that are arranged in order of time.
canonicalization	This is a technique used to prevent duplicate content in search engine optimization.
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart This is used to prevent spam and confirm a post operation.
copyleft	This is a term in the GPL license that forces the developer to make public his source code and promote learning in the open source community.

cron	This is a scheduler for making timed event.
CSS, HTML and javascript	These are popular technologies for web design.
CRC	Cyclic redundancy check This is used for error checking and to know about accidental changes to the data.
crawler	This is used by search engines to obtain data from resources spread across the Internet. This collated results are then indexed to make searching very easy.
Django	This is a Python-based open source web development framework which is modeled with the model view controller factory design pattern.
eBay	This is a popular web-based auction site.
E-Business, E-commerce	Electronic Business This is business / commerce that is carried out through the internet.
HTTP	Hypertext Transfer Protocol

Igbo	<p>This is a stateless protocol for transfer data across the Internet.</p> <p>This is a language that is spoken in south-eastern part of Nigeria.</p>
NOSQL	<p>not only SQL</p> <p>This is a highly scalable database.</p>
public key	<p>This is the cryptographic key that is freely available.</p>
private key	<p>This is the cryptographic key that is only known to the individual.</p>
Python	<p>This is a high-level programming language which is object-oriented in design but supports other programming paradigms.</p> <p>It has a comprehensive standard library and ensures code readability.</p>
RDBMS	<p>Relational database management system</p>
Scale	<p>This is the handling large user loads.</p>
statefulness	<p>This means the request and response do not have any relationship.</p>

SQL	Structured Query Language This language is used to query the database.
TradeMe	This is a popular web-based auction site.
UML	Unified Modeling Language This is a multipurpose modeling language that is used in object-oriented programming to abstract the higher level view of the system.
Web 2.0	This is a content-rich web application.

1 INTRODUCTION

As the world has become a place where many distant parts can be reached within the fraction of a second using the Internet, similarly there is a growing demand for sustained research into the field of e-commerce. The technological explosion in the 21st century has given rise to people seeking more convenient and cost-effective method of doing business. Gone are the days when people only deal with their neighbors. The world has become a global village where people now do business without any personal contact; only by the use of the web (Peters and Bodkin, 2007).

Electronic Auction has become the cornerstone of electronic commerce. This has witnessed unprecedented growth in the past few years. E-auction has become the method of choice for users to deal with goods. The Internet network has placed a lot of power in the hands of customers. They can easily compare prices of items across different stores to make an informed decision about the bid amount (Turban and King, 2003). Internet auction sites also enable users to act as either buyers or sellers while obeying the business rule. The seller creates the product for sale by auctioning whereas, the buyer bids on the item. On the other hand, the system resolves the auction after the validity period has expired. This work has demonstrated the foundation for building such auction web software.

However, the success of any e-commerce web application is dependent on its robust security features. E-auction systems guarantee confidentiality for the bids. This ensures that only the buyer knows the amount that he has bidden. On the other hand, the concept of Bid integrity prevents bidders from bid denial or alterations. There is a common form (type) of cheating in the web-based auction system known as the bid shilling. Moreover, the implementation of proprietary schemes is necessary to mitigate against these attacks.

E-commerce will continue to experience a lot of substantial growth as many devices become connected to the Internet e.g. mobile devices. Electronic

auction systems extend the life cycle of products and reduce demand from retail stores. Sellers tend to distribute second-hand goods (Cameron and Galloway, 2005; Nissanoff, 2006; Chu and Liao, 2007). There is a growing number of small scale enterprises that rely on eBay and TradeMe for selling their goods (Wood and Sute, 2004). The use of the electronic auction system for trading stock has become exceedingly popular (Herschlag and Zwick, 2002; Tang and Forster, 2007). In the US alone, the sales from the online auction site are five times greater than the sales from the world's largest online retail store, amazon (Chong, 2004). eBay has become a significant player in the industry (Black, 2007). eBay is responsible for 15 percent of all US e commerce, and annual sales on eBay were greater than the GDP of many countries (Black, 2007; Chong, 2004; Datamonitor, 2009). Thus, there is a positive outlook of growth in the electronic auction. This has encouraged the author to develop a web-based auction system as his Bachelor's thesis. This work describes the necessary details to consider when a designing web-based auction system.

The web auction system described in this thesis project is ideal for small and medium-sized traffic.

1.1 Core Software used in project

The following software were used in the project. These are described in Table 1.1.

Table 1.1 Software used in the project

Name	Description
Django 1.3	This is the current version of the Django web framework.
Python 2.7	This is the version of python used in this project.
MySQL 5.1	This is the version of relational database used in this project.
Apache 2.2	This web server is used for handling dynamic requests
NginX	This web server is used handling static media request.
Firebug 1.8.2	This is used for inspecting, editing and monitoring CSS,HTML and JavaScript pages.
mod_wsgi	This allows the Apache web server to host any Python application.
Ubuntu 11.04	A modern open source operating system.
Firefox 5.0.1	A modern open source web browser.
django-celery	This is a free asynchronous task queue/job queue based on distributed message passing.
ghetto	This is a queue framework and it works with django-celery.
django-picklefield	This is implementation of a pickled object field. This works with Django -celery and Ghettoq.
django-kombu	This works with Django-celery, Ghettoq and Django-picklefield. This serves as a message store.
Matplotlib	This is a python 2D plotting library.

1.2 Thesis Overview

This thesis report is split into different chapters. Each chapter will address specific aspects of the project. This is a summary of each chapter's content follows:

Chapter 1 - Introduction

This gives an overview of the thesis. This chapter explains the aim for the project and motivation for carrying out the project.

Chapter 2 - Literature Review

This chapter covers the fundamental concepts of web programming. Through this chapter, the readers can understand the challenges needed to implement security in a web auction system. The user will also be given some historical insight on the development of e-commerce and the evolution of electronic bidding. The reader will also see the similarities with existing auction system and draw parallel with conflicting design of the auction system put in context.

Chapter 3 - Design And Implementation Of The Core Engine

This chapter discusses the choices between programming language and web framework. This is a detailed description of the Django framework. The reader will understand the implementation details of building the bidding engine which involve making and resolving bids.

Chapter 4 – Design And Implementation Of Other Engines

This chapter describes all the details of the following. They include Commenting engine, Search engine, Business intelligence(user analytic and statistics), graph engine, advertisement engine and the recommendation engine. The reader will understand the notification mechanism in use and understand the database design and performance optimization. There is a discussion of scalability.

Chapter 5 – Implementation Of Web Security And Optimization

This discusses the security vulnerability that is available in modern web development. This chapter discusses the strategies for mitigating against this kind of security attack. There is also a detailed analysis of search engine optimization by on the site level and search engine level. The reader will have a better understanding of concurrency. This chapter discusses web deployment on the Internet.

Chapter 6 – Conclusion

This chapter provides the goals of the projects. There is also a detailed description of possible improvement that should be implemented in order to achieve commercial success.

2 LITERATURE REVIEW

The electronic auction follows the pattern of traditional commerce. This is a resource distribution channel (Chakravarti et al., 2002). Every auction undergoes the following processes: price negotiation, payment and goods delivery.

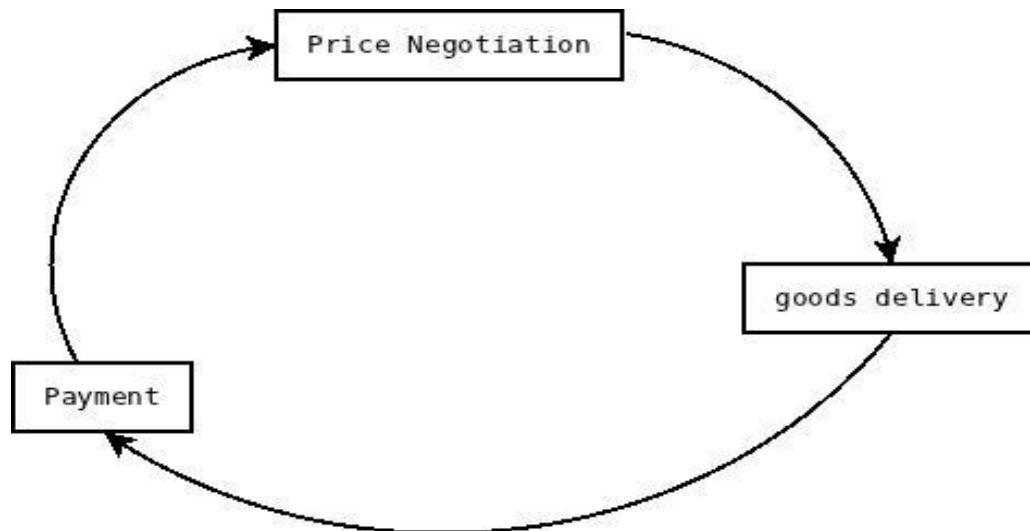


Figure 1.0 . Process diagram of auction.

We need trust to complete the process described in Figure 1.0. In a traditional market place, the number of personal contacts between the buyer and the seller increases mutual trust (Cabral and Hortacsu, 2005). There is an absence of real life contact between trading partners in online auctions. The buyers cannot inspect the goods to ascertain the quality (Rafaeli and Noy, 2002; Yen and Lu, 2008). There is a possibility that the seller may not even be available at the time a buyer makes a bid. This can pose a challenge to trust (Chakravarti et al., 2002; Cui et al., 2008). The parties involved in the business are anonymous. They are unlikely to have another transaction again in the future. However, the buyer gives the feedback after each transaction. Conversely, this can be used to form a history of previous transactions (Rauniar et al., 2009; Zhou, Dresner, and Windle, 2009).

The auction system should be designed so that the bidders can compete with each other in a fair manner guaranteeing that there is no undue advantage

given to any user. The auction's result must be free and fair. This should be available for anyone to prove. The buyer procures the product based the seller's description of the item. Similarly, these items are untested. This makes the customer depend only on the seller's description of the product (Melnik and Alm, 2002).

2.1 Feedback System

For any electronic auction site that has profit-making intentions, the business owners must take time and resources to increase customer's conviction on the security of the web application. The premise has led to auction system's developer to create a sort of trust by considering the previous sales made by the seller and use this information to create a feedback system. These feedback systems provide information that are useful for future buyers. This information allows the customer to make an informed decision about buying from the seller in question (Cabral and Hortacsu, 2005).

In 1995, eBay began operations in the United State. This company has grown to become an extremely influential player in the online auction business. eBay has built a state of the art reputation system which allows buyers and sellers to rate one another after any transaction. This system generates records that are available to all customers. The reputation system provides a yardstick for measuring the trustworthiness of the seller. It is customary for the seller not to ship items until they have received payment. There has been some research on the impact of seller's reputation on the selling price. The results are from the auction of the 1999 mint condition \$5 U.S gold coin. These show that bidders are willing to bid higher for these items from the reputable sellers (Melnik and Alm, 2002).

A similar research corroborates the evidence suggested by the former. The researchers in this survey considered the effect of bidders' and sellers' reputation of Intel Pentium III 500 Mhz auctions on eBay. The conclusion from the experiment was that bidder reputation does not have an effect on the

auctions but the seller's reputation had considerable impact on the auction price (Houser and Wooders, 2005). eBay calculates the trustworthiness of users in two ways. They include:

- The percentage of positive and negative buyer's rating.
- The difference between positive and negative feedback of the buyer. The seller's account contains all the reputation variables and these are available to the public (Melnail and Alm, 2002; Cabral and Hortacsu, 2005; Resnick et al, 2005; Tobias et al, 2009).

However, eBay has implemented a mechanism for the seller to retaliate against negative feedback from buyers (Tobias et al, 2009). This creates a rather difficult situation for the customer as they do not tend to provide exact feedback on the quality of the seller's transaction for fear of retaliation. There are some changes that eBay implemented to prevent bid fraud. This scheme became operational as early as February 2008. If there are more than one transactions, then the trading partner within a week, then the buyer gives the seller a scheduled feedback. The total feedback score of the seller increases by one if the buyer gives a positive feedback. On the contrary, there is a point deduction from the feedback score if the buyer's feedback is negative (Tobias et al, 2009).

As of May 2008, eBay improved the reputation scheme to enforce that only buyers can rate the seller only after they have had a transaction. This will reduce the possibility of retaliation from other sellers. This will allow the buyers to give honest feedback (Tobias et al, 2009). The reputation engine allows the suspension of a member's account if a member violates the terms of eBay. The system ensures that sellers cannot give negative feedback about the buyers, but can report violation of the buyer. The system allows the seller to prevent the buyer who has flouted the terms of the eBay from bidding on future auctions (Tobias et al, 2009). Feedback rating shows an indication of the seller's behaviour. This measures the trustworthiness of the sellers. Similarly, bidders tend to pay a lot to buy goods for sellers of high repute.

Bid shilling is a serious problem that seriously undermines the trustworthiness of the auction system. This is how it works. A user creates multiple accounts in the web auction system. The user makes a bid on the item just above the minimum price level. He then logs into one of his phoney account and make an outrageous (unrealistic bid price) bid on the item. This will scare other potential buyers from bidding on the item. The phoney account of the same user with wins the auction. After winning, he has to wait and refuse to pay for the item. He waits until the paying period is over. The next user in the bid queue becomes the winner. The same user has intentionally made the selling price unusually low. This practice should be discouraged by some proprietary scheme.

2.2 Cryptology

There is a risk to the web auction system due to the security weaknesses inherent in the Internet which is an open environment. The viable solution for the electronic sealed-bid auction is the implementation of the cryptology algorithm to secure the transaction. Auctions are a proven means of price negotiation and resource distribution.

The field of cryptology is outside the scope of the thesis. This project makes use of asymmetric cryptology that ensures the privacy of bids. This scheme ensures that the views which contain the bids, user details and other bid sensitive views are encrypted to ensure that no one can tamper with the bidder's information. The bidder's information should be encrypted to prevent tampering.

2.3 Web Security

There is a number of known threats that affect web applications. This situation is dynamic as these attacks become more sophisticated. There is also a need for a robust response to these security threats. The rule of thumb is to reduce the surface of attack by reducing the possibilities of a successful security intrusion. There are some popular security attacks in the past few years.

Security Attacks

There is a number of popular security attacks. They include the following:

- Cross-Site Scripting Attacks.
- SQL Injection.

Cross-Site Scripting Attacks

In cross-site scripting attacks (or XSS attacks), the attacker attacks the customers on a site by the use of some malicious JavaScript code to bypass client-side to gain access to user's cookies and session data. The attacker looks for the part of the website where user's input is rendered back to the user as part of the web page. These attacks exploit the feature of the web browser which receives a page and renders it to the user. The browser has no way to distinguish genuine users from malicious users. The browser just renders even the malicious script. Django has a default behavior of escaping all the HTML tags in the template before rendering the item on the page. This makes the malicious codes just plain text. There is a similar but different kind of attack known as cross-site request forgery. This exploits the trust that the web site has for the user. The attacker includes a link in the page that the user has been authenticated, For example, if a user, Samuel is in a chat forum where another user, John, has sent a message. Imagine that Fred has created a malicious HTML image element in place of normal image file e.g.

```
<img src = http://bank.myexample.com/withdrawcash?account=john&amount=4500&for=Fred />
```

This becomes an issue when Samuel's bank data are saved in cookies. This can be exploited if the cookies are active, as any attempt to load the images on the chat forum would automatically make a withdrawal from Samuel's account without his approval. The attack is possible because there is no way for the server to differentiate genuine request from a bogus or malicious request.

SQL Injection

This is one of the most common web vulnerability that has plagued web development to date. The attacker attempts to steal, manipulate or destroy important data by exploiting features in the SQL language. This happens when raw SQL queries are used inside the web application for user submitted data. These user-input is passed into the where clause of the SQL statement. The user can enter “item” in the user textbox. This will result in this SQL statement being sent to the database.

```
SELECT * FROM products WHERE name LIKE '%item%';
```

However, the situation becomes very problematic if the user enters item%'; DROP TABLE products in the user textbox. This SQL statement generated from the input is equivalent to

```
SELECT * FROM products WHERE name LIKE 'item%'; DROP TABLE products;
```

There are some recommendations to minimize the security threats in web application. They include:

- Authentication: which ensures that we know the user who is making a request.
- Authorization: which ensures that the user has the right to get a response for a specific request.
- Encryption: which ensures that the data is encrypted so there can't be any successful eavesdropping.
- Avoid privilege escalation: which ensures that file permissions on the system are appropriate. Otherwise, it can become a loophole to launch a number of security attacks.
- Mitigation strategies: which ensures that there are a mechanism in place to reduce the impact of a number of successful attack .This is a containment strategy.
- Audit Logs
 - There is a need for web server to create a log of all the activities.

- This contains the timestamps, client IP address, request, execution trace, execution times.
- Logs are usually created in simple text files to avoid overheads.
- They detect ongoing brute force or password guessing attacks.
- They detect security breaches and their impact (after they have happened)
- They can be used to debug the application
- They create usage statistics to profile customers, improve performance, etc.
- Log files can contain sensitive information and become a security risk.

(Alchin, 2009)

2.4 Optimization

However, there is plenty of room for optimization in web application. It is imperative for any web based auction system to rank well in search engines. The aim of optimization is the removal of all bottlenecks that may affect the user's experience.

Search Engine Optimization

There is a need for every web application to be easily searched from the Internet. Search engine optimization is the process of making a site friendly for search engines (Google or Yahoo). The sites need to be designed so that spiders can crawl, index our web pages and add them to their Search Engine Results Pages (SERPs). These optimizations are a set of rules that web masters can follow to appear on the top of the site. There is a penalty for duplicate content; every item must have a unique URL. Search rewards good practices and index your web pages at the top of the search result. However, search engines strictly penalize web masters who duplicate the content of the web pages. This is the process by which the search engines make a decision

about the host name in use is known as canonization. The search engine would index the presumably best host name. However, this does not solve the "duplicate content problem". We could canonicalize the site by ourselves and leave nothing to chance. We can explicitly decide the host name that will be indexed by the crawler. This will prevent the search engine from indexing the host name that we do not want to use. In this way we can avoid the problem of duplicate content.

Concurrency

Concurrency can improve performance as most of the computer's processing power is put to use. It can also lead to performance bottleneck if not properly implemented. Concurrency problems can occur in two major forms which are race condition and deadlocks.

How to prevent Race Condition

Race condition is possible in multi-threaded application. There is a need for mechanisms to handle these potentially dangerous situations. This can be prevented by the use of locks or by setting the isolation level of the database to serialize. The use of the highest isolation level in the database also undermines scalability. One protocol used to prevent race condition is Strict Two-phase Locking.

Strict Two-phase Locking (Strict 2PL) Protocol

This is a concurrency control method that is used to ensure reliable concurrency. This uses locks to block other transaction from accessing a shared resource during the lifetime of the process (Weikum and Vossen, 2001). This uses a lock to restrict the access of the process to a shared resource. Each process must obtain a S(shared) lock when reading or writing on object. At the end of a transaction, the appropriate lock is released. If a process holds an X lock on an object, no other process can get a lock (S or X) on that object. Strict 2PL allows only serializable schedules. Database locking can be performed at the following. They include:

- Table level
- Row level

Deadlocks

Deadlock occurs when two processes are preventing each other from running when they are trying to acquire a shared resource. This is very common when locks are used. Locking multiple resources can lead to deadlock.

T1: LOCK TABLE A; LOCK TABLE B;

T2: LOCK TABLE B; LOCK TABLE A;

These are the conditions. They include:

- Mutual Exclusion
- Hold and Wait
- No preemption
- Circular Wait

Mutual Exclusion: Only one process can hold an exclusive lock at a time.

Hold and Wait: There is at least one process that holds a lock, while waiting for the process to release a lock. It should be known that only the process holding the lock can release it.

Circular Wait: There is a set $p_1 \dots p_n$ such that p_1 is waiting for a lock held by $p_2 \dots p_n$ is waiting for a lock held by p_1 .

One strategy for avoiding deadlocks is that all processes always acquire locks in a given order. This eliminates the circular wait condition.

2.5 Web server

This is the operating system that performs the following operations. They include:

- Waiting for an incoming TCP connection
- Read HTTP request

- Interpret request
- Locate or generate a resource
- Format and send HTTP response

Every useful web server must have the following characteristics. They include:

- Reliability
- Security
- Availability
- Scalability
- Usability

There are two classes of server. They include:

- Stateful server
- Stateless server

The Stateless server supports stateless protocol like HTTP. These are more fault-tolerant as failure in the client does not propagate to the server e.g. Apache, NginX etc. The stateless server uses less memory footprint so it lends itself easily to scalability. However, not every application requires statefulness.

Moreover, there are several use cases where statefulness is required. There are some clever ways to maintain states across the session. In this situation, we create a session which is a group of HTTP request and response performed by the same user in a short period of time. The methods include:

- Cookies
- Hidden field
- URL

The building block of the web includes:

- Markup language for hypertext documents e.g. HTML
- Uniform scheme for addressing resources over the network (.e.g. URL, URI)
- Protocol for transport message (e.g. HTTP)

2.5.1 HTTP

HTTP is a protocol that uses client-server architecture. This is stateless and uses the request/response paradigm. It uses error codes to identify the cause of the error. The HTTP transaction consists of request from the client to a server. It has a number of request method they include GET, POST, HEAD etc.

- GET

It should be free of side-effects. A GET request should be idempotent.

- POST

This has a side effect. There is a need for a "captcha" to make the user aware of the action that he is about to perform.

A good database must have the following properties to be able to handle transactions (ACID) (Philip et al, 1987). They include:

- Atomicity

The result is either committed or rolled back. There is no transition state.

- Consistency

There are no invalid transactions. If any transaction breaks integrity rules, then it is rolled back.

- Isolation

The result of a transaction is invisible to other transaction until they are complete.

- Durability

Once completed, the result of a transaction is permanent.

3 DESIGN AND IMPLEMENTATION OF THE CORE ENGINE

The building blocks of any software are their underlying algorithms. These algorithms need to be implemented in any programming language to be used for problem solving. There is a number of programming languages where we can implement the web-based auction algorithms. It is well known that all programming languages can solve problems but some languages are more intuitive than others. The core of this project is the bidding engine. The success of the project depends on this engine. If it does not work properly, then the project is a failure.

3.1 Choice of Programming Language

The Python language was chosen for the following reasons. They include:

- Conciseness

Programs that are written in the Python language are smaller in size compared with programs with other languages. This makes the implementation of algorithm easier.

- Clear Syntax

The Python language has a remarkably clear syntax. Conversely, the language syntax does not get affect the programmer's productivity. On the other hand, this programming language is self-documenting.

- Easily extensible

The Python language has a rich set of algorithms implemented in the standard library. There is a number of readily available third-party software to make programming more fun.

- Interactive

The Python language comes with an interactive interpreter. This allows the programmer to test the code snippet before using them in production.

- Multiparadigm

There are multiple styles such as object-oriented, procedural, and functional styles of programming. This allows the programmer to think in the manner that is most suitable for the problem.

- Multiplatform and free

Python works across many platforms. There are fewer operating system specific quirks. It is an open source programming language.

3.2 Choice of Django

Django was chosen for the following reason. They include:

- The availability of Django admin interface

Django comes with a built-in admin interface. This was designed with usability in mind. This allows the programmer to manage records with minimal coding. This also has a robust security feature. The availability of a built-in interface greatly reduces the learning curve.

- Simple URL management

This manages URLs at the application level saving us the hassle of managing URL by modifying the apache conf file. It also generates beautiful URLs which are search-engine friendly.

- High speed

This is a language that has a large user base. This is an interpreted language that has been heavily optimized to ensure quick program execution.

- Django is open source

Django is open source. It is free for any purpose. However, it is shipped with a less restrictive license which gives the user unlimited conditions on how to use the software.

Django is a web framework that was designed using the Model-View-Controller pattern. The models interact with the database backend by means of an object relational mapper in the framework. This controller handles the logic and manages the request and response, the views are the means by which the users interact with the web application.

There are some other popular python-based web framework. They include:

- Grok
- Pylons
- TurboGears
- web2py
- Zope

3.3 Use Cases and Multiple Views / Requests

Use cases can be implemented using multiple views and requests. The database can change between a use case requests. This occurs if there is a single process. We should leave the transactions open or object locked between requests.

There is a scheme to handle in-deterministic request which is called optimistic locking strategy. The scheme is described as follow:

- Do not lock resources
- Implement a mechanism to detect if the objects have been modified.
- Abort or notify the user in that case

- Django does not have direct support for optimistic locking. This can be implemented as specific hooks.

In this basic architecture, the views have 3 functions. They include:

- Validate input
- Process application logic
- Select and render output

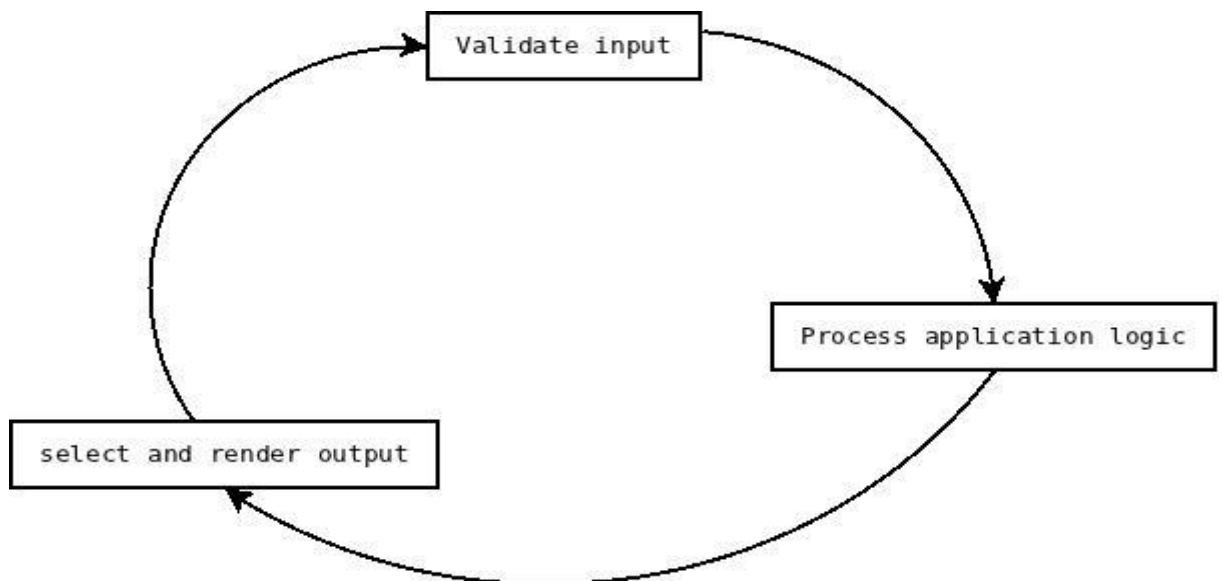


Figure 3.0 Description of model –view- controller (McGaw, 2009)

The basis for the thesis work is the design and implementation of an e-auction based system. This system is designed to enable users to create and participate in auctions by created by a community of users interested in buying or selling the most diverse items. (Belk et al., 1988; Cameron and Galloway, 2005; Lastovicka and Fernandez, 2005). This web system also provides users with the tools needed to communicate in an efficient manner(Turban and King, 2003). The site will be called 'kpoba' which means in my mother tongue (Igbo) to auction or display. The auction site is different from a normal web store. This is because the items are not being sold by the administrator. They are created by a community of users who wish to sell their item. The items are sold by a

method of bidding. In a web shop, there could be possibility for after-sale service which is rare in a web auction based system.

There are several features that are optional in an e-auction site. The availability of the buyout price. This is common in a rather unstable market where there is much price fluctuation. There is information about the frequency of items with buyout prices in Yahoo and eBay taken on the March 27, 2002, where a total of 1,248 were sampled from Yahoo and 842 of these items have the buyout price (Reynolds and Wooder, 2003). This is becoming a prevalent trend in modern auction sites. Due to its optional nature, the use of the buyout price was not implemented in the project.

The project contains the following subsystems.

Table 3.0 List of the Engines used in the project.

Type of System	Engine
Core System	<ul style="list-style-type: none"> • Bidding engine
Sub-Systems	<ul style="list-style-type: none"> • Commenting engine • Search engine • Business Intelligence (User analytics and statistics) engine • Graph engine • Advertisement engine • Recommendation Engine

The project objectives can be met by achieving all the goals from the different sub-systems.

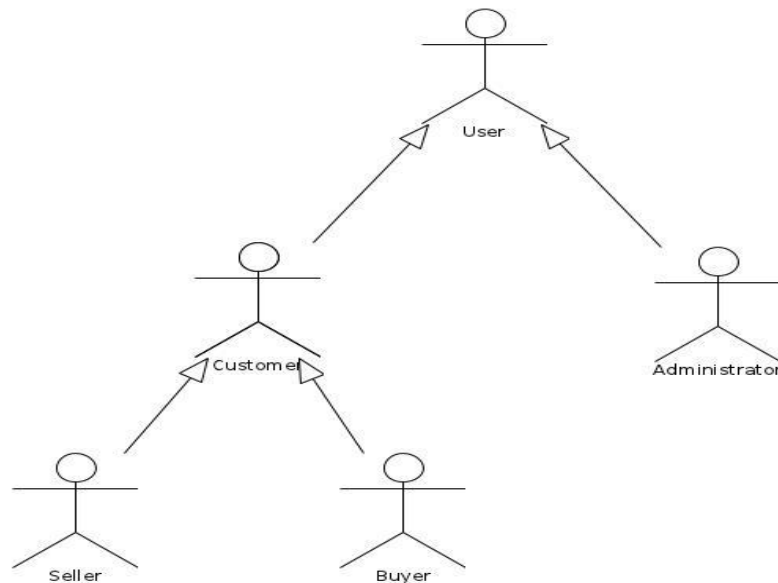


Figure 3.1 Use case showing how the roles of users

3.4 Bidding Engine

There are two types of auction. They include:

- Single Attribute
- Multi Attribute

Single Attribute: This uses only one parameter to make bids on the object. This is useful for our web auction system as only the price is needed to make an auction.

Multi Attribute: This uses a number of the parameters needed to make a bid on a subject. This is useful for cases like bidding for a contract (Sunderam and Parkes 2002). In this example, project manager needs consider the cost of the

project , the skill of the contractor, the quality of material to be used and other miscellaneous information. These are needed to make an informed decision (Koppius 2002; Teich, Wallenius, Wallenius and Koppius 2003). The auction (item) is created by the user interact with the auction manager to make a new auction. This process involves the filling out of a form with all the necessary information about the item. The seller when creating the auction must clearly specific the required information on the auction. The bid manager enforces the bidding rules and accepts only bids that meet these criteria. The bid manager has to check the database to see if the bid met the minimum requirement. If these criteria are met, then the new bid is stored in the database otherwise, it is rejected. Clear manager is needed for clearing auctions. This is used to remove expired bids from public view.

The implementation details can be expressed as the following objectives below:

- Implement a scheme for anonymous users to be able browse and search for auctions.
- Devise a means for anonymous user to create a user account so that their data can be saved in the system
- Develop a means for the registered user to be able to edit user account information
- Implement a scheme for registered users to be able to create a new auction
- Allow the registered user to bid on auctions.
- Create a means for resolving auctions according to business rules.

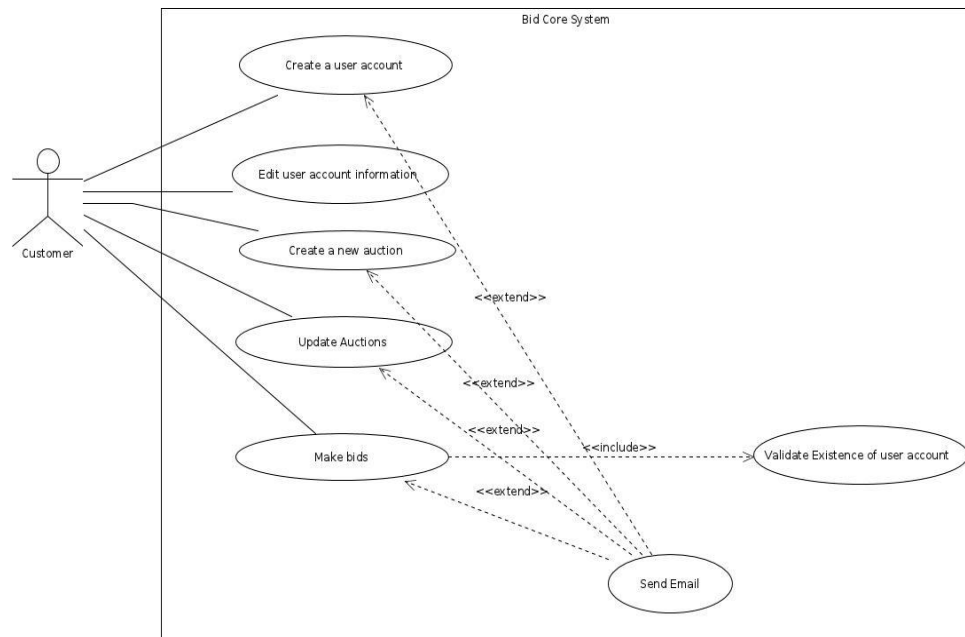


Figure 3.2 Use case showing the possible actions for a customer.

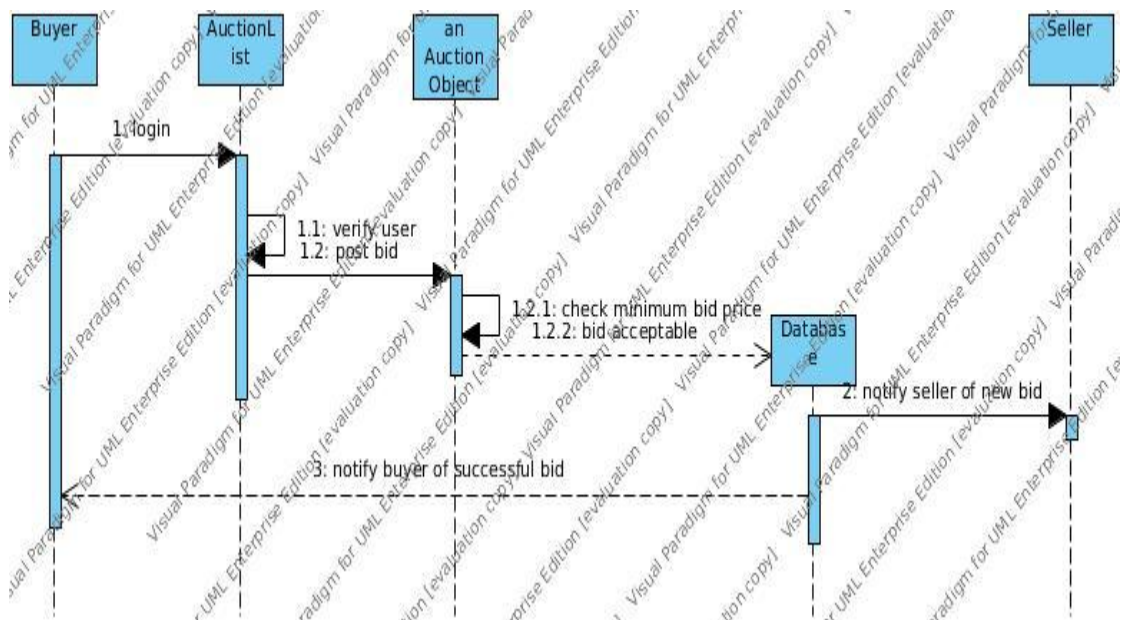


Figure 3.3 Sequence diagram for bidding

3.4.1 Resolving of Bids

Once the end date of the item is reached, the cron scheduler runs in the background and checks the first item on the bid list. The owner of this item is identified and an email is sent to the winner. This email contains a congratulatory message and a link for giving customer review. This review is attached to the user and the user assigns a rating of the credibility of the seller on that transaction. This credibility is shown on the page where the user is about to make the bid. This gives a customer rating of the last ten items sold by the seller. This will help other buyers in the future to be able to judge the seller as credible or dubious.

3.4.2 Django-Celery for Scheduling

The cron job is the native way of making scheduled events in Unix. This works very well for most cases. However, this can automate the task by writing events in a scripting language like bash, Tcl etc. However, Django-celery is a real-time asynchronous scheduler that can be used to automate the task. It has many advantages over cron because it understands Python programming language. This project made use of Django-celery to meet all its automation needs.

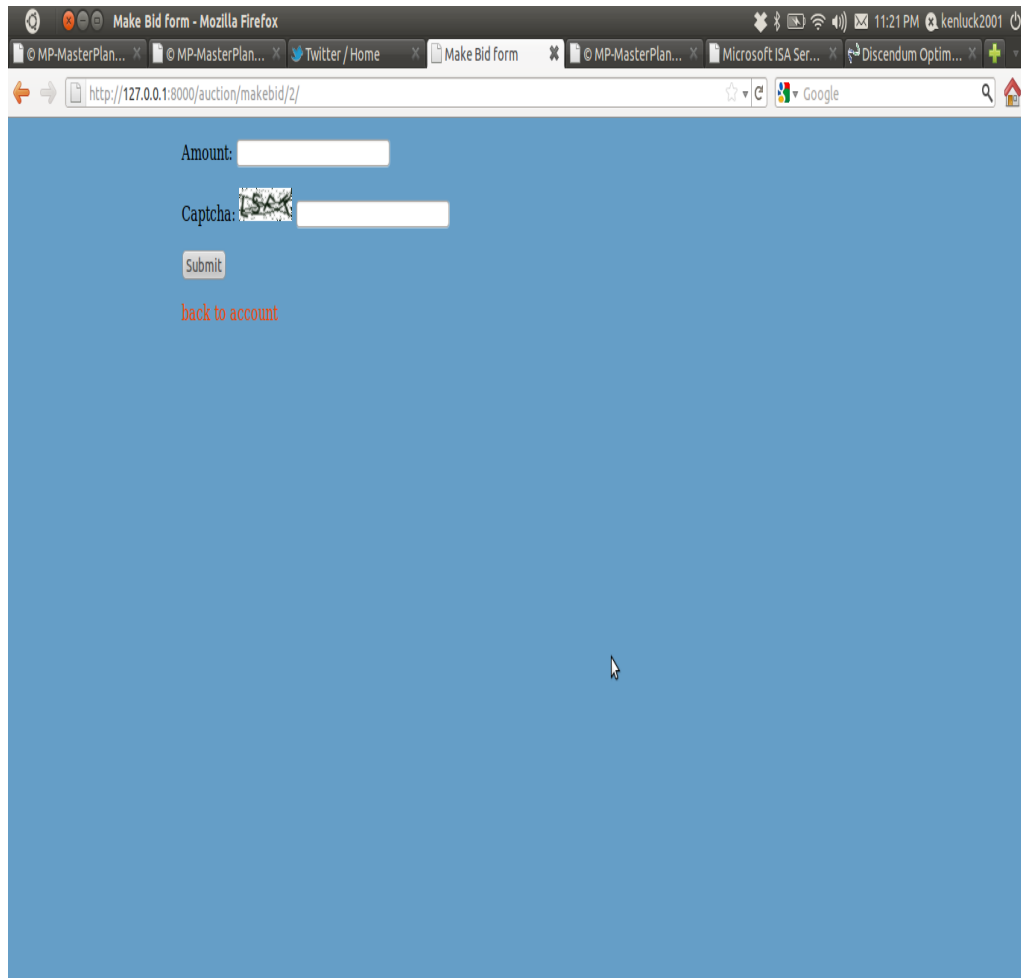
A screenshot of a Mozilla Firefox browser window displaying a bidding form. The browser's address bar shows the URL 'http://127.0.0.1:8000/auction/makebid/2/'. The form is set against a solid blue background and contains the following elements: a text input field labeled 'Amount:', a CAPTCHA image with the text '52X' and an adjacent text input field labeled 'Captcha:', a 'Submit' button, and a red text link labeled 'back to account'. The browser's tab bar shows several open tabs, including 'MP-MasterPlan...', 'Twitter / Home', 'Make Bid form', and 'Microsoft ISA Ser...'. The system tray at the top right indicates the time as 11:21 PM and the user as 'kenluck2001'.

Figure 3.4 Bidding form for making new bids.

The source codes of the bidding system can be found in the following appendices. They include:

- APPENDIX 1.0 - Bidding system
- APPENDIX 1.1 (views.py)
- APPENDIX 1.2 (models.py)
- APPENDIX 1.3 (url.py)
- APPENDIX 1.4 (extra.py)
- APPENDIX 1.5 (forms.py)
- APPENDIX 1.6 (admin.py)
- APPENDIX 1.7 (tasks.py)

4 DESIGN AND IMPLEMENTATION OF OTHER ENGINES

There are a number of engines that needed to work with the bidding systems to create a working auction system.

4.1 Commenting Engine

Commenting can help the buyer know more about the product that is displayed for sale. The buyer tends to undergo some discussion on the commenting engine with the buyer and every other bidder on the item. This is a way of implementing integrity as people could raise alarm if the seller is dubious and warn everyone in the commenting engine. This knowledge sharing in the commenting section is useful for the community. These comments also help the rapid indexing of the web page. Comments are always quickly indexed in the search engine. The web master can leverage on these premises to increase the ranking of the web page in the search engine. The search engine tends to re-index pages that have new content. The comments make the web page have continuous new content and thereby drives the search result. This feature of commenting has been exploited by spammers. They tend to make unsolicited comments on the web page. This can be avoided by the use of "CAPTCHA". The acronym "CAPTCHA" stands for "Completely Automated Public Turing test to tell Computers and Humans Apart". This was used to help to differentiate real user from spamming software. This is because only a human user can pass the Turing test. "CAPTCHA" were extensively used in this project to avoid spam and to allow the user to confirm a post operation.

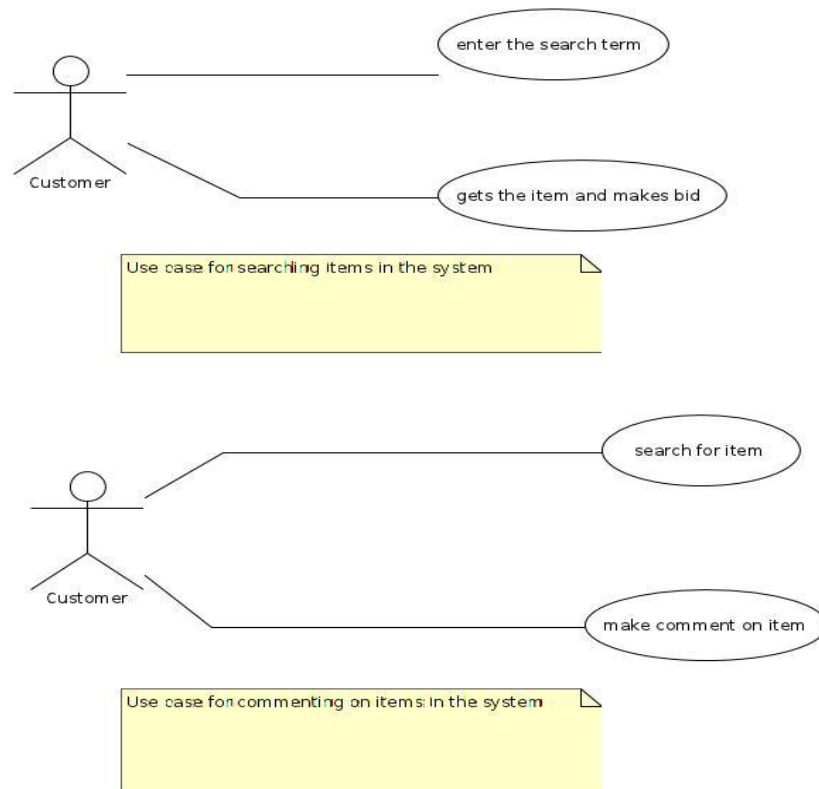


Figure 4.0 Use case for searching and commenting

4.2 Search Engine

The profitability of an E-commerce venture that interacts with users is dependent on how easily the buyers can easily see the items on sale. The users need to search for items. In the case of an auction site, if the user can not find the item, then it will reach the expiry date and cleared from the system before anyone could bid on it. This calls for some clever implementation of the search algorithm that gives the user the most relevant result with the least search terms. This is discussed extensively under search engine optimization.

In addition, the users are able to find items on the site. They should also be able to find the site on the internet. The implementation details can be expressed as below.

- Design a scheme for user to be able to see and bid on available items. This should be designed with great consideration as it is imperative that user should be able to see the items that they are to bid on. Otherwise, the auctions expire before the users can bid on them.

4.3 Business Intelligence (User analytics and statistics)

There is data everywhere but less information. We need a mechanism for gathering useful business data that we can use to make informed business decision. These are needed to give the manager a clear view of the market conditions. This makes the company very able to make accurate decision in the rather volatile market.

The implementation details can be expressed below.

- Implement a system for tracking user behavior, monitoring market conditions and analyzing statistical information about bids. This helps the user to understand the customer's habit and use them for business decision. This kind of information is useful for manager to know at a glance how the business is performing.

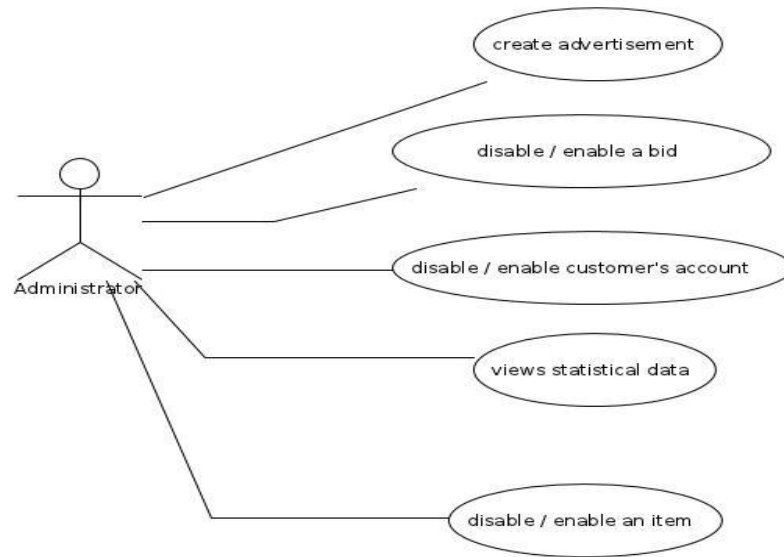


Figure 4.1 Use case for the administrator.

4.4 Graph Engine

There is a need to represent the business analytic data in a graphical form. This because a picture speaks more than a thousand words. The project made use of Matplotlib to make 2D plot of the information gathered by the business intelligence engine. This gives a pictorial view of the current market condition.

The implementation details can be expressed below.

- Implement a way for showing 2D-plots of the statistical data.

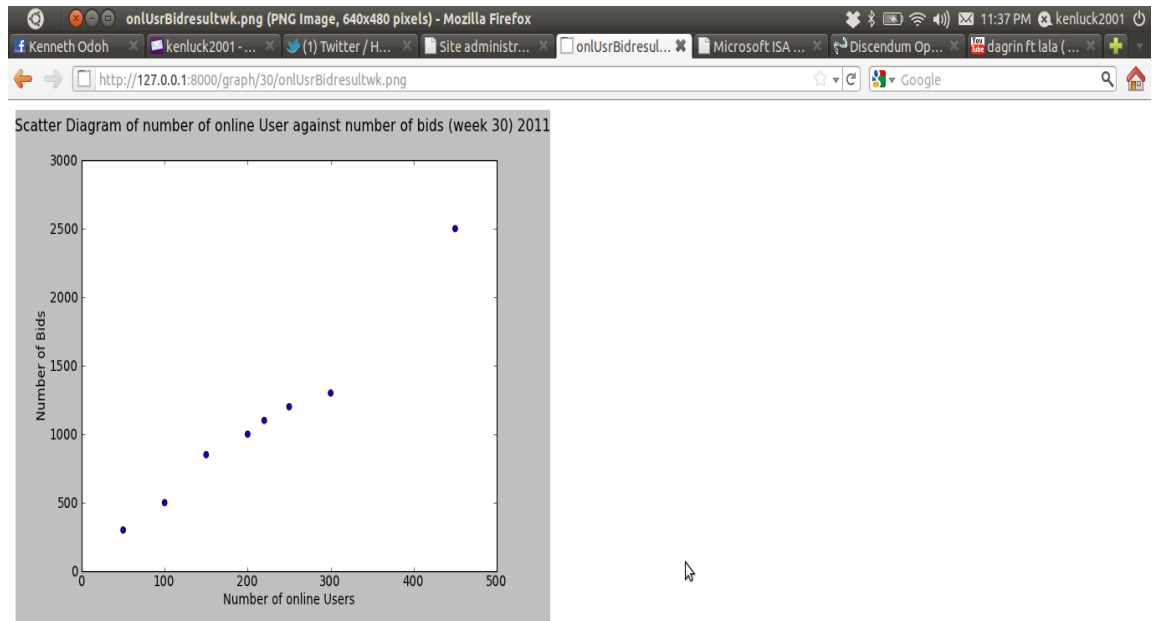


Figure 4.2. Scatter Diagram of number of online user against number of bids

4.5 Advertisement Engine

The origins of internet advertising can be traced to October 27, 1994 when Hotwired (<http://www.hotwired.com>) when the online version of Wired magazine allowed banner advertisements on its web pages (Kaye and Medoff, 2000). The business of the online advertisement has grown exponentially in the past few years. Smart banners have come into existence. They harvest user data to make a personalized advertisement (Kaye and Medoff, 2000). This is a disruptive technology that tends to overtake traditional advertisement channels like the television, radio and print media. Banner advertisements should be carefully crafted to rapt the attention of the web surfer. A banner advertisement was incorporated as a module in the web auction site so that it could provide

alternative income for the administrator.

The screenshot shows a web browser window with the title 'Advertisement Creation Form - Mozilla Firefox'. The address bar displays 'http://127.0.0.1:8000/advert/advertform/'. The form itself is set against a solid blue background and contains the following elements from top to bottom: a label 'Name of Advert:' followed by a white text input field; a large, empty white rectangular box; a label 'Description:' followed by a white text area; a label 'Advertiser's website:' followed by a white text input field; a label 'Photo:' followed by a white text input field and a 'Browse...' button; a 'Submit' button; and a red text link 'back to admin'.

Figure 4.3. Advertisement creation form.

4.6 Recommendation Engine

The recommendation engine was built using the principle of collective intelligence. This is the gathering of statistical data from a group of users, analyzing it and making the statistical conclusion about the group which no individual member would have known by themselves. This takes the form of census and survey. However, in the web, the gathering of data is a lot easier as we can collect this data anonymously and tend to analyze them in order to make conclusions about the user.

The inferences can be made by using

- Item-Based Filtering
- User-Based Filtering

Item-based filtering

This predicts user interest on an item based on similarities between items.

User-based filtering

This predicts the user's interest on an item based on rating from other similar user's profile. The recommendation engine is based on this kind of filtering as it depends on the items with the largest bids are displayed so other users can bid on them. The assumption is that the item is a prized item and we would likely fetch the largest bid price for such item as it would subsequently increase the commission of the administrator (Segaran, 2007).

4.7 Notification

This is concerned with the information that is sent from the system to the user. Information should be sent to the user in the following scenarios. They include:

- Information is sent auctions are created or updated .
- Information is sent when bids are made.
- Information is sent for activating account.
- Information is sent when bids are resolved.

Information sending can be any of the following. They include:

- Push technology
- Pull technology

Push technology: Changes in the server is relayed by pushing the new information to the client. This overloads the server with deliver messages that may not be relevant at the time. Email notification is a push based technology and was used extensively in this project.

Pull technology: Changes in the server are only received by the clients when they request it. The pull technology only requests for information when needed and hereby reduces the server workload.

However, the web auction system was split into several sub-systems to ensure that there is modularity. This will make the system much easier to maintain and add new features. This will improve code reuse as some of the modules can be fitted into another similar project will less modification. The most challenging part of the project is to integrate into the sub-systems to work seemingly as a whole with the expected results.

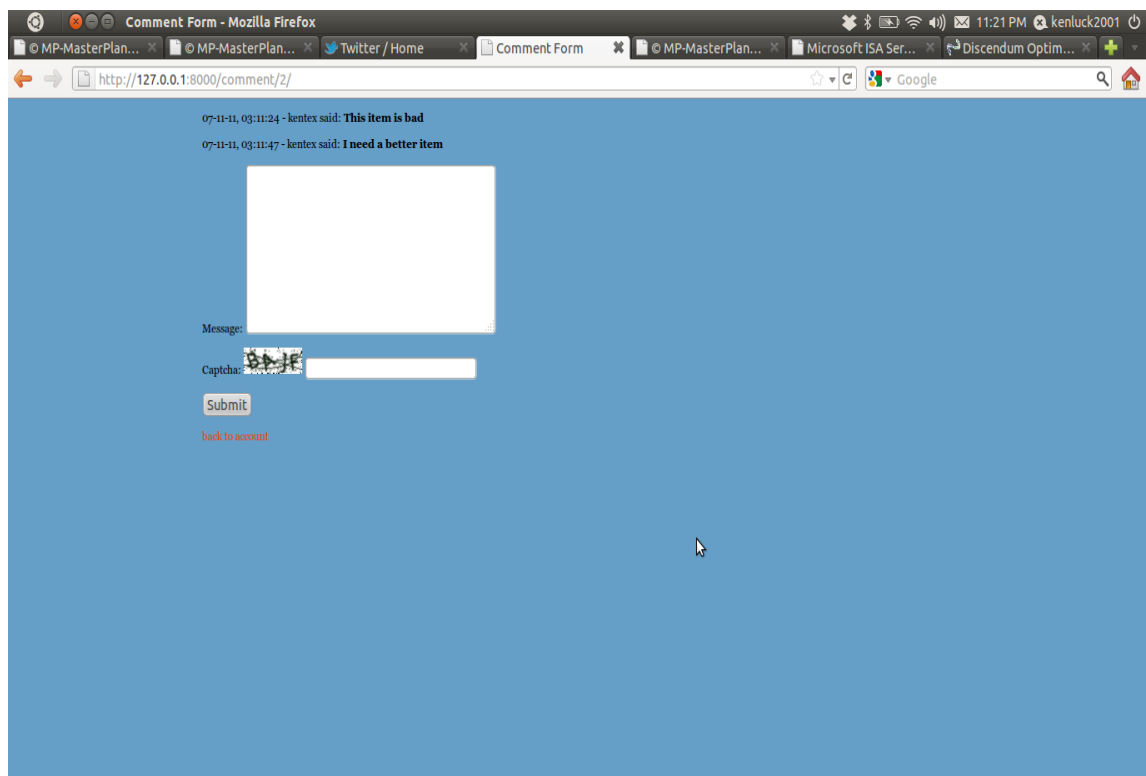


Figure 4.4 Comment form

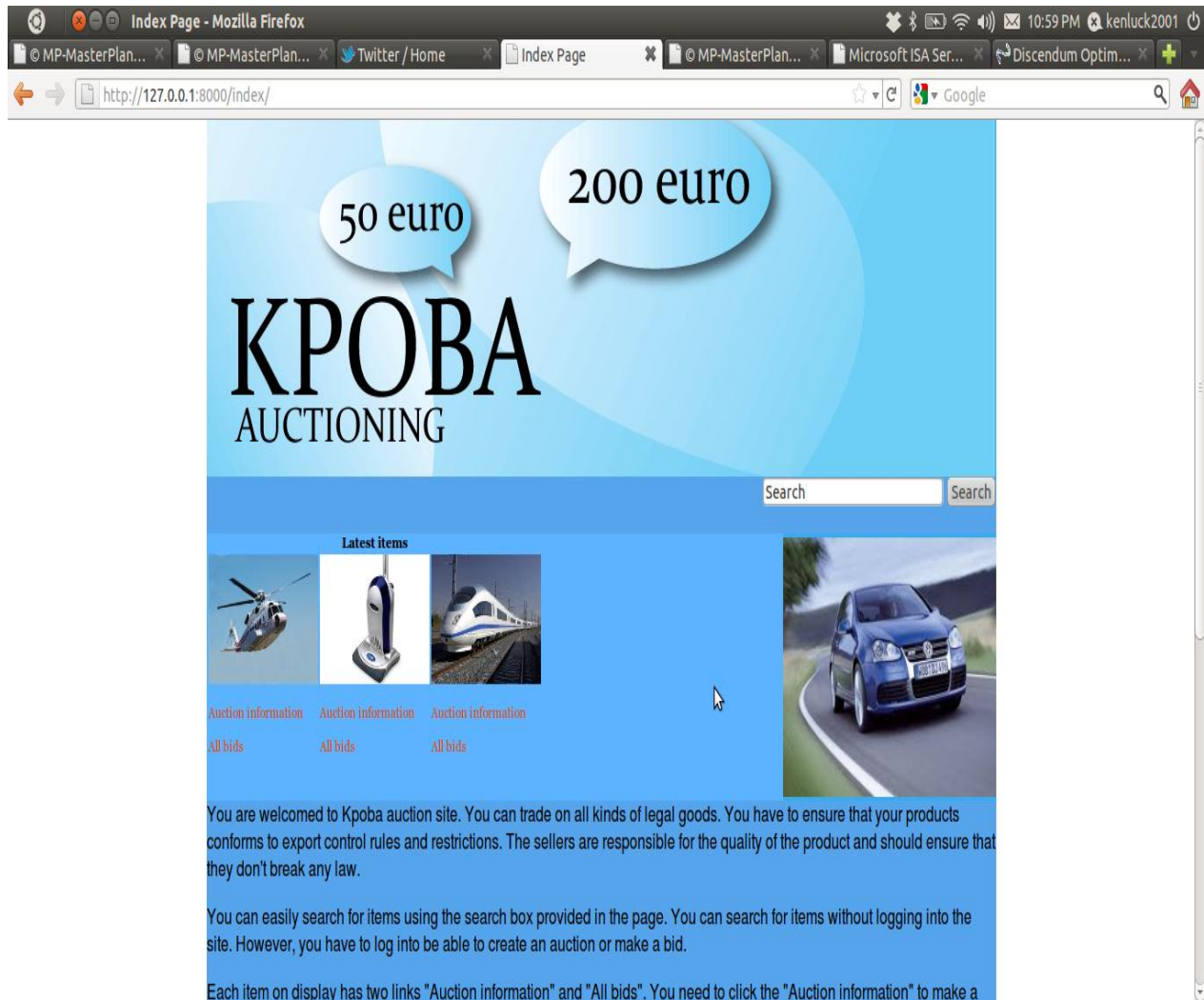


Figure 4.5 Home page of the auction system

4.8 Database Design

This is the blue print of the system. The performance of the system is dependent on a good database design. These help us solve many problems that could cause bottlenecks in the system (Elmasri and Navathe, 2004).

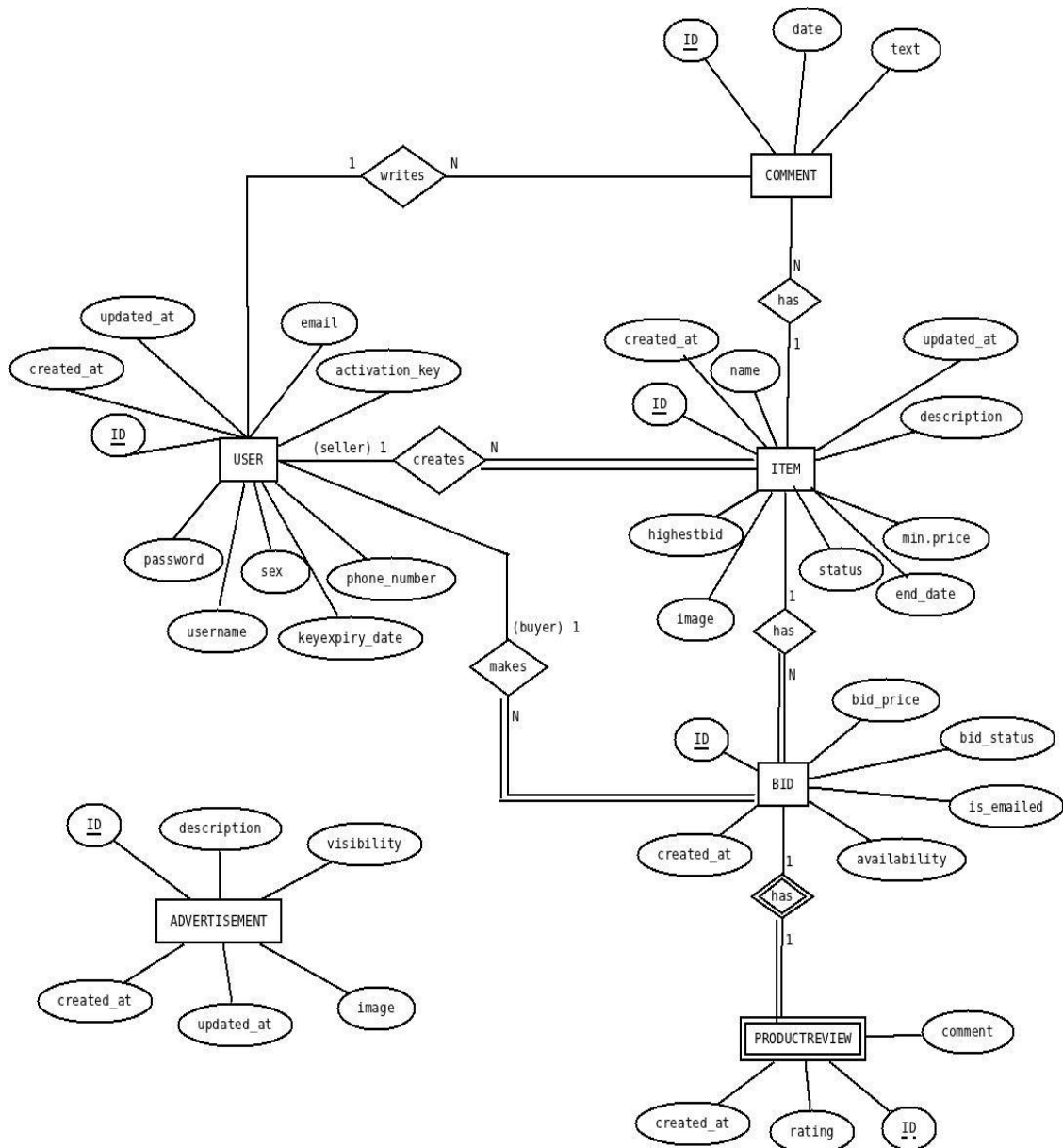


Figure 4.6 Entity relationship diagram.

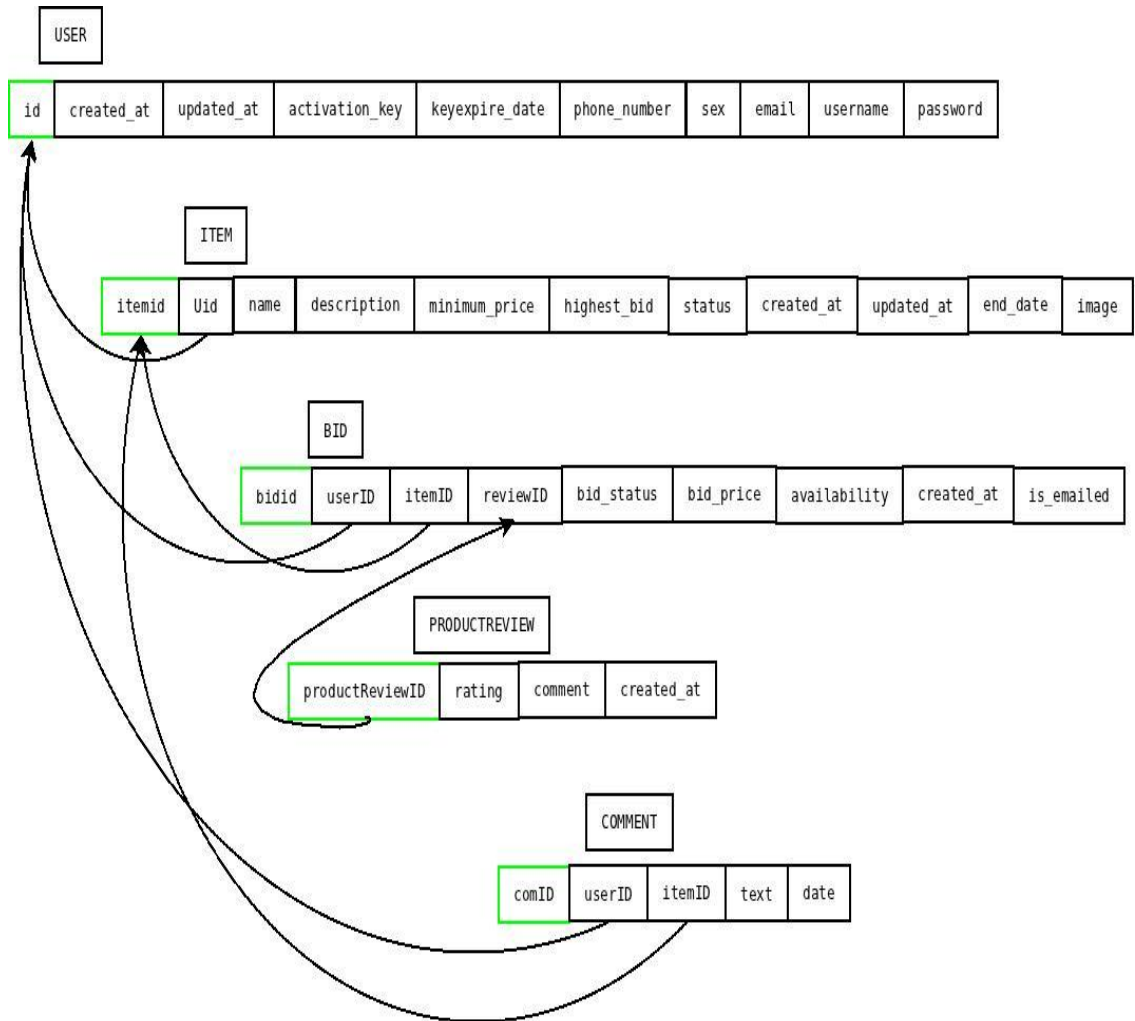


Figure 4.6 Relational Model.

4.9 Performance Optimization

The performance of the system can be improved by the following procedures.

- Database Optimization

These are factors that the programmer should consider when optimizing the database relations. They include:

- Ensure that small relations are not indexed.
- Ensure that the primary key is index manually, if it is not done automatically by the DBMS.

- Ensure that if the foreign key is frequently accessed , then add secondary index to the foreign key.
 - Ensure that secondary index is added to any attribute that is heavily used as a secondary key.
 - Ensure that secondary index is added on attributes that are involved in: selection or join criteria; ORDER BY; GROUP BY; and other operations involving sorting (such as UNION or DISTINCT).
 - Ensure that attribute or relation that are frequently are not indexed.
 - Ensure that attribute that querying will retrieve a significant proportion of the tuples in the relation should not be indexed.
 - Attributes should not contain long character strings.
- Delete all old data (expired session data).
 - Ensure that more data is stored on the client.
 - The separation of content from presentation is a vital step to take to split the roles of programmers and web designers. The web designed should not have to focus on understanding the implementation details of the application. This is necessary for division of labor.
 - Ensure that the template should have minimal business logic. The logic should be implemented in the views. This will make the application easy to maintain.
 - Template caching / caching with Memcached

This can be classified as

- Server-side caching
- Client-side caching

Server-side caching

Caching is a way of reducing server load. There is a working solution that can be used for caching in Django-based application. Memcached is a caching server that reduces the number of database hits. Care must be taken to handle stale data appropriately.

Client-side caching

There is a possibility to cache data on the template. This is a feature that should be used with caution and works well if the cache time is set appropriately.

4.10 Scalability

Scaling is dependent on the hardware and software configuration to handle unplanned load. The web site may be designed for handling loads of page views, bidding events and resolving of auctions. The aim of scalability is to serve many users at the same time. In some cases, the best way to scale an application is to redesign the database schema in order to avoid expensive join operations.

There are two parameters that are considered very useful in scalability. They include:

- Response time: This is the time taken for a request to be processed and return a response.
- Throughput: This is the number of completed requests per second.

Web application programmers are more concerned with high throughput. The measure of scalability is the ability to handle many requests at a time and not at handling one request at an amazing speed. There are two ways to scale an application.

- Vertical scaling

- Horizontal scaling

Vertical scaling

This is the easiest approach. This involves the increasing in storage spaces on the infrastructures supporting the web application. The administrator keeps increasing the hardware with the impression that hardware is cheap. Scaling an application this way can result in financial disaster for the enterprise as they can run out of memory space. The business requirements can quickly exceed the limits of the technology available for the hardware.

Horizontal Scaling

A better system is to scale horizontally. The loads are spread across many machines. This is similar to the concept of division of labor. As a matter of fact, most large web applications use horizontal integration as a means of keeping their architectures scalable. This is the approach adopted by high technology companies like Facebook, Google, eBay and Amazon. Django is built to horizontally scale the work load. This goal was achieved in Django by ensuring that the framework is modular and loosely coupled. This is a cheaper approach as we can use a number of cheaper hardware instead of one extremely expensive hardware.

Moreover, the web administrator should be conscious of the “Slashdot effect”. This is a case where there is the sudden increase of traffic to the website because it was advertised in a larger forum and people just want to access the site. This effect happens momentarily and the goal of scalability should be for the ideal number of users.

Nevertheless, there is also a classical approach to reduce the load on the web server and thereby scale the application. We could use an HTTP proxy server

which is an intermediary between two processes communicating using the HTTP protocol. It connects HTTP request to their actual target.

It blocks the following. They include:

- Blocks malicious or inappropriate sites
- Cache content local
- Log traffic
- Load balancing

(Shklar and Rosen, 2003 ; McGaw, 2009)

5 IMPLEMENTATION OF WEB SECURITY AND OPTIMIZATION

There are ways to improve the security of the application. However, there are lots of features that can be added to the project to improve security. On the other hand, optimization can take the form of enhancing scalability and concurrency.

5.1 Web Security

The security of the application is the aim of every developer. There are some good practices on how to prevent some security attack in our application.

Cross-Site Scripting Attack

The best solution is to add a hidden input with a unique value to every form. When the form is processed, we can confirm that this page comes from our site. This means that we have to assign a long ID for the site which must match the hidden input in every form submitted to the site. Django comes with a built-in solution to mitigate the effect of cross site forgery. This is implemented on a middleware. This adds a hidden input field with a token value. The validation token is obtained from the user's session ID and SECRET_KEY value of the project. The validation token is added as a secret input field in the form. Every form that is processed is checked to see if its hidden input matches the expected validation token. If it does not match, then an HTTP 403 error page is relayed to the users. This behavior of Django was used to improve the security of our web application.

SQL Injection

This results in the deletion of the products table in the database. Django escapes all special characters in its ORM so these special characters are just a normal string in the where clause and this prevents an SQL injection. This is a minimalistic description of how Django prevents SQL injection.

5.2 Ways to Prevent Security Attacks

There is a number of ways to prevent security attacks. They include the following:

- Permissions
- Protection Against External Attacks
- Debug Mode

Permissions

Every web application should provide a different level of access to different classes of users. Django has built-in permission features to provide users with a varying permissions level. Permissions could be applied to an individual or a group of users. This is implemented in Django using boolean fields. They include: `is_superuser`, `is_staff` and `is_active`.

Protecting Against External Attacks

The input from the user should never be trusted. There should always be a form of validation. However, there are two kinds of validation. They include:

- Client-side validation
- Server-side validation

Client-side validation must be done to reduce the load on the server. However, if validation is done only on the client, then the user could subvert the security. There is a need for server-side validation to be done in addition to client-side validation. Every form which the user fills in should be treated as suspicious and must undergo both client and server-side validation. It does suffer from bandwidth wastage due to the sending back and forth session data from client to server. The user cannot easily change the client.

Debug Mode

There is a need to change the DEBUG flag from true to false in production. Failure to do this will result in the web application displaying error messages with some source code to the user. This can reveal the type of web framework that the website is using. This is enough information to make a successful attack. Django provides an easy way of setting this flag in the settings.py file.

5.3 Safe Practices

These are practices that can compromise the security of the application. They include the following:

- Storing Customer Passwords
- Scanning Incoming Files for Viruses

Storing Customer Passwords

The password is very sensitive to the web application security. This is supposed to be confidential and only accessible to the user. However, every web application needs to store passwords in one way or another for authentication. It is a serious security issue to store passwords as plain text. If we use encryption the password using the symmetric cryptography method, then we can use the same key to encrypt and decrypt. The saving of these cryptographic keys is also a security threat. The ideal solution would be to use a hash function which is a one-way function as we only need to encrypt the user's password without any need to decrypt. The Hash function takes a string (e.g password) and generates a unique string which is a representation of the string. This creates a mapping of the real string to a smaller unique string. The success of the scheme is guaranteed if the hashing algorithm is collision-resistant. Django uses the state of the art hash algorithm SHA-1 to generate hash values (message digest) for the password. This makes the use of the brute-force attack on the password very difficult because the password is "salted" by a developer password that is almost impossible to be guessed by the hacker. This user password input is converted to its hash value and compared with the hash value

stored in our database .If there is a match, then the user is authenticated. The default behavior of Django framework was used in the application.

Scanning Incoming Files for Viruses

There is a couple of UNIX viruses in the Internet. However, with good file permission, the web application can be saved from any infection. There could be need for using anti-virus solution where there will be a need to make cross-platform applications. We could scan all the files that the user wants to upload to our server. There are a couple of excellent open source virus scanning applications e.g. ClamAV, which was designed for use in servers. These provide an easy-to-use interface for scanning any incoming file. If a virus is found, the offending file can be quarantined before causing harm. This project did not use any anti-virus scanning due to the extra overhead in production. However, it made use of good file permissions. This will increase the security of the application.

5.4 Optimization

Nowadays, people cannot afford to waste time looking for the address of our site from a search engine. There is an abundance of competitors in the market, so the next auction site may just a click away.

5.4.1 Search Engine Optimization

There are some actions that can be taken to improve search engine optimization to rank the web page better in the page ranking algorithm of a search engine. They include:

- Generating a Keyword List
- Submitting Your URL
- Creating a robots.txt File

Generating a Keyword List

There is the need to create a keyword list for all the content on the web site. The administrator needs to ensure that keywords are related to the items that are displayed for bidding. The choice of keywords should be considered carefully to avoid copyright issues.

Submitting Your URL

There is a need to submit your URL to major search engines. There is a high probability that a search engine crawler will hit your web site and index your page in their search result. There is no need to leave the linking of the site to a search engine by chance. We can explicitly add our new URL to the <http://www.google.com/addurl/>

Creating a robots.txt File

This is a file that contains instructions that a spider should use when crawling the web site. Every web crawler is supposed to check for this file at the root of your project. This can be used to prevent duplicate content and make the search engine crawler avoid the duplicated content by not specifying in the robots.txt file. The use of robots.txt is just a request and there is no obligation that the web crawler obeys the rules written in the file. Most search engines are bound by law to obey the robots.txt file but aggressive web crawlers can use it to steal sensitive material from the web site.

To prevent issues from the violation of robots.txt terms, sensitive documents should require user login to view such file (McGaw, 2009).

5.5 Saving of Data

There are two primary media:

- File system
- Database

File system: This is used for saving large files like images. This can have fast searching time if we know the path. It has some disadvantages such as security risk due to wrong file permissions; there is not much support for concurrency and locking, so it cannot be used to support transactions.

Database: This abstracts the physical storing of data on the file system. There is a possibility for transaction support. We could easily support query optimization, back up and replication. However, this is not suitable for storing large objects such as images.

However, we can access the database by either manual or automation methods. They include:

Manual methods

- The use of Command-line tools
- The use of Dedicated visual tools

Automation methods

- The use of Dedicated drivers specific for a RDBMS
- The use of Standardized database connection libraries
- The use of Object-Relational mapping libraries

Django uses an object relationship mapper to interface with the database. This was utilized in the thesis.

There is a number of relational databases that are freely available in the market. Some of them are proprietary, while others are open source.

- SQLite

This has Zero-configuration for setting up. It is suitable for the embedded system database system. This has a smooth learning curve for beginners. It is not suitable for enterprise application.

- MySQL

This is very popular and portable with an open source license. It is lightweight and fast for small loads but it does not necessarily scale well and therefore is not necessarily the best option for all applications. This is suitable for our application due to its ease of use and flexibility.

- PostgreSQL

This is a highly reliable RDBMS and has good documentation. It is licensed as an open source with (BSD license). It is heavyweight and has supports for many advanced features. This can be complex to learn for beginners.

- Oracle

This is the most successful commercial RDBMS. It has many features and scales well. There is also commercial support. It is suitable for enterprise applications.

5.6 Concurrency

Concurrency is the running of several processes at a time. The use of multiple threads means that we have a concurrent system. The race condition is caused by many threads/processes trying to update the same information simultaneously. This occurs due to the simultaneous update of the shared resource. This is difficult to detect by informal testing. The application submits transaction, and we can think of each transaction as executing by itself. The database must begin and end transactions in a consistent state. A transaction might commit after completing all its actions, or it could abort (or be aborted by the DBMS) after executing some actions. The most important property of a RDBMS is atomicity. This means that a process is always executing all its actions in one step, or not executing any actions at all. DBMS logs all actions so

it can undo the actions of aborted transactions. This is needed to commit or revoke the transaction.

The Isolation property of the transaction in databases can be used to manage concurrency at a database level without the use of locks. They are four levels that can be used in appropriate situation. They include:

- Serializable
- Repeatable reads
- Read committed
- uncommitted

The project was not designed to handle very large loads. Therefore, it does not offer support for transactions.

5.7 Web Deployment

The Django development server is not resilient enough for production. We have to use an industrial-strength web server. They are two popular servers that were used in this project.

They include:

- Apache
- NginX

Apache

This is an open source and cross-platform HTTP server. It uses threads to handle requests and serves over 60% of all web sites. It is a stateless server. It has the following characteristics. They include:

- Site configuration using virtual web server.

- security
- Efficient integration of interpreter, e.g., Perl ,PHP, Python
- Gateway to other application server

This was used for handling dynamic content (Django pages) in our project. The static content is handled by any server. This division of work aids scalability.

NginX

This is an open sourced multiplatform HTTP server. It was designed for high performance, a low memory footprint and ability to handle heavy concurrent operation (Kleinman, 2010). This uses the asynchronous event-driven approach for addressing requests. This was used for handling static content (e.g. images) in our project. The static content is handled by any server. This division of work aids scalability.

Moreover, there is a need for the view to require security. The web application prevents any form of eavesdropping by the using SSL. This uses a public key cryptographic scheme to avoid problems with key sharing. We need to use HTTPS to communicate over port 443, instead of HTTP that uses a default port of 80. We encrypt using two keys. They include:

- public key
- private key

The public key is freely available and distributed to the client's browser while the private key is saved in the server. Both the public key and the private key are mathematically related. Messages are encrypted on the server using the private key. When the cipher-text arrives at the client, it is decrypted using the public key saved on the client's browser. Any conversation between the client and server is encrypted using the public key. This scheme helps us to identify the entity's identity. This scheme is susceptible to the man-in-the-middle attack because we may download the malicious certificate and this goes against the reasoning for using public key cryptography scheme. This problem is solved by

using a certifying authority (CA). The CA certifies the ownership of a public key. This makes the certificate trustworthy to the web browser. This stores the information about the organization that has the public key. The full implementation belongs to the field of public key infrastructure. This project makes use of the certificate that has been generated on a Unix computer. It did not make use of a CA.

6. CONCLUSION

The overall system been made to specification. Let us describe the results and discuss on ways to improve the work.

6.1 Results

The integration of all the sub-systems created an interesting web auction site that has the features of a web 2.0 application. The whole project achieved its goal. Users acting as sellers are able to create items, while other users acting as buyers can bid on the item. This auction website has a robust system for the handling of bids. This system has a goal of only accepting bids that are higher than the previous bids. The highest bidder is assured to win the bid and the system works this way in practice. The core system (bidding system) of the application is working flawlessly as users can bid and sell their item without hassle. Nevertheless, the other sub-systems are working according to specification.

The project is working seamlessly and the specification was met. Moreover, irrespective of the use of third-party libraries, the development process is still a complex task because some of the libraries are lacking in documentation. For example, Matplotlib that was used to plot the graph needed in the application has a plethora of examples for desktop application but has little or no samples plotting graphs in a web application. This situation would make the programmer to try different syntax while paying close attention to the generated error messages. This is an extremely strenuous task as it does not make use of any standard operating procedure. The next example is the Django celery library for scheduled events. This has a solid documentation but has little description of how to integrate with the Django project. I began asking help from programming fora before I discovered that all the periodic events must be contained in the task.py file saved at the root of the project. There were some other challenges which I met when using the anything slider library for making advertisements. I had to refactor the code but using firebug to search for the features which should not be part of the advertisement system. This is a highly-consuming

operation as sliders are meant for sliding static pictures but here we have to use to pictures objects saved in the database ORM.

However, the price paid for using open source libraries is inversely proportional to the amount of programming efforts used to make the libraries work. Moreover, it is still a cost-effective means as it provides the possibility of tweaking the source codes to suit your specific needs. There is also the problem of undocumented features of the library as this could pose a security risk.

There is also a security threat that is possible when using open source software as the compiled binary may be compromised. However, the best safeguard against this kind of attack is to compile from source after reading through every line of code. We can also check the CRC check on the downloaded binary or use Debian package manager to download from a reputable repository. This guarantees that the packets are not compromised using the trusted keys of the operating system.

This project shares lot of similarities with eBay. However, it was not meant to be an eBay clone. This is the design and implementation of a usable auction system. The algorithms for eBay are proprietary so any attempt to create similar features of eBay is a momentous feat. This involves recreating all the needed algorithms from scratch.

6.2 Recommendations

The system is ideal for small and medium loads. The major optimization used in this project is the indexing of all join operations. This considerably increases the performance of the system. The system can have increased scalability when we add more redundancy to the primary key – foreign key relationship. This is achieved by replacing the one-to-many relationship or one-to-one relationship with many-to-many relationship.

However, another improvement needs to be done on the advertisement engine. The whole advertisement object from the database should be cached on the

client and only updated periodically. This will considerably reduce the server load. There is also a need to avoid all expensive join operations. Instead, we should prepare a carefully crafted sub query which does not include joins. This will considerably increase the performance of the system.

There are serious challenges posed by the use of open source libraries. The legal ramification of the licenses used should be studied carefully to prevent lengthy patent litigation, for example, if the programmer includes any code with GPLv3 license in the application. This means that the derived source codes must be available for free. We know the fact that most software has profit-making intentions. This will seriously affect the business. This kind of license removes programmer's copyright from the software project and leaves him with a copyleft on his source code. This is really absurd. This was seriously considered during the development process to make sure that the author has full copyright over his work.

There are security vulnerabilities that will be discovered during the life cycle of the third party libraries using the project after they have been released to the community. It is the responsibility of the programmer to sign up to discussion fora, periodicals and social network for the announcement of bugs, bug fixes and current best practices. This also provides update on bug fixes, upgrades, patches and how to handle the problem.

The data that are collected from the user analytic and business intelligence can be saved and used for training a neural network if the company decides to operate an automatic market prediction tool. This will provide ready made data that is required for training such neural network or genetic programming solution.

Moreover, if there is a need to handle large traffic, then the architecture should be redesigned from scratch. This would involve replacing the database backend use NOSQL database, such as couchDB which is a structured database that was designed to horizontally scale.

REFERENCES

- Alchin, M. (2009). *Pro Django* (pp. 1-279). USA: Apress.
- Belk , R. W., Wallendorf, M., Sherry, J., Holbrook, M., & Roberts, S. (1988). Collectors and collecting. *Advances in Consumer Research*.15, 548-553.
- Belk, R. W., Sherry, J. F., Jr., & Wallendorf, M. (1988). A naturalistic inquiry into buyer and seller behavior at a swap meet. *Journal of Consumer Research*.14, 449-470.
- Black, G. S. (2007). Consumer demographics and geographics: Determinants of retail success for online auctions. *Journal of Targeting, Measurement and Analysis for Marketing*.15(2), 93-102.
- Cabral, L., & Hortacsu, A. (2004). *The dynamics of seller reputation: Theory and evidence from eBay*. Chicago: University of Chicago.
- Cameron, D. D., & Galloway, A. (2005). Consumer motivations and concerns in online auctions: An exploratory study. *International Journal of Consumer Studies*.29(3), 181-192.
- Chakravarti, D., Greenleaf, E., Sinha, A., Cheema, A., Cox, J. C., Friedman, D., et al.(2002). Auctions: Research opportunities in marketing. *Marketing Letters*, 13(3), 281-296.
- Chu, H., & Liao, S. (2007). Exploring consumer resale behavior in C2C online auctions: Taxonomy and influences on consumer decisions. *Journal of Academy of Marketing Science Review* .11(3), 1-25.
- Chong, B. (2004). How buyer experience in online auctions affects the dimensionality of trust in sellers: An unexpected finding. Paper presented at the Twenty-Fifth International Conference on Information Systems.
- Cui, X., Lai, V. S., & Liu, C. K. W. (2008). Research on consumer behaviour in online auctions: Insights from a critical literature review. *Electronic Markets*, 18(4), 345-361.
- Datamonitor. (2009). *eBay Inc - Company Profile*. New York: Datamonitor USA.
- Elmasri, R., & Navathe , S. B. (2003). *Fundamentals of database systems* (4th edition). USA: Addison Wesley.
- Herschlag, M., & Zwick, R. (2002). Internet auctions - Popular and professional literature review. *Quarterly Journal of Electronic Commerce*, 1(2), 161-186.
- Houser, D. & Wooders, J. (2005). Hard and Soft Closes: A Field Experiment on Auction Closing Rules. In: R. Zwick and A. Rapoport. *Experimental Business Research*.2, 279-287.
- Kaye, B. K., & Medoff, N. J. (2000). *Just a click away: Advertising on the internet* (1st edition) . USA: Allyn and Bacon.
- Koppius, O. (2002). *Information Architecture and Electronic Market Performance*. Ph.D. thesis, Erasmus Research Institute of Management Ph.D. Series Research in Management.13.
- Kleinman, S. (2010). Basic nginx configuration. Retrieved October 27, 2011, from <http://library.linode.com/web-servers/nginx/configuration/basic>.
- Lastovicka, J. L., & Fernandez , K. V. (2005). Three paths to disposition: The movement of meaningful possessions to strangers. *Journal of Consumer Research*. 11, 813-823.
- Melnik, M. & Alm, J. (2002). Does a Seller's eCommerce Reputation Matter? Evidence from eBay Auctions. *The Journal of Industrial Economics*. 50(3), 337-349

- McGaw, J. (2009). *Beginning Django E-Commerce* (pp.1–339).USA: Apress.
- Nissanoff, D. (2006). *FutureShop: How the new auction culture will revolutionize the way we buy, sell and get the things we really want*. New York: The Penguin Press.
- Peters, C., & Bodkin, C. D. 2007. An exploratory investigation of problematic online auction behaviors: Experiences of eBay users. *Journal of Retailing & Consumer Services*.14(1), 1-16.
- Philip, A.; Hadzilacos, V. & Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*(pp.1-38). USA: Addison Wesley Publishing Company.
- Rafaeli, S., & Noy, A. (2002). Online auctions, messaging, communication and social facilitation: A simulation and experimental evidence. *European Journal of Information Systems*.11, 196-207.
- Rauniar, R.; Rawaski, G.; Crumbly, J. & Simms, J. (2009). C2C online auction website performance: Buyer's perspective. *Journal of Electronic Commerce Research*.10(2), 56-75.
- Resnick, P. & Zeckhauser, R. (2001). Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. Technical report, University of Michigan, February 5.
- Reynolds,S., & J. Wooders, (2002). *Ascending Bid Auction with a Buy-Now Price*, University of Arizona, Tucson.
- Segaran, T. (2007). *Programming collective intelligence : Building smart web 2.0 applications* (First Edition , pp. 1-28). United States of America: O'Reilly Media, Inc.
- Shklar , L., & Rosen , R. (2003). *Web application architecture: Principles, protocols and practices* (pp. 1-347). USA: John Wiley & Sons Inc.
- Sunderam, A. & Parkes, D. (2002). Preference elicitation in proxied multiattribute auction. tech. ref., division of engineering and applied sciences. harvard university, cambridge, MA. Retrieved November 17, 2003, from <http://www.eecs.harvard.edu/~parkes/pubs/proxysunderam.pdf>
- Tang, Y., & Forster, P. (2007). Exploring the value structure behind mobile auction adoption intention. *AMCIS 2007 Proceedings*.
- Teich, J. E., Wallenius, H., Wallenius, J., & Koppius, O. (2003). *Emerging Multiple Issue E-Auctions*, ERIM Report Series Research in Management ERS-2003-058-LIS, Erasmus Research Institute of Management (ERIM), Rotterdam School of Management, Erasmus University, Rotterdam, The Netherlands.
- Turban, E., & King, D. (2003). *Introduction to e-commerce*. Upper Saddle River, New Jersey: Prentice Hall.
- Tobias, J.; Lambertz, C. ;Spagnolo, G. & Konrad O. (2009). The actual structure of eBay's feedback mechanism and early evidence on the effects of recent changes. *Inderscience Enterprises Ltd*. Available also at <http://www.gianca.org/PapersHomepage/Klein%20et%20al.-%20Actual%20structure%20of%20ebay.pdf>.
- Weikum, G. , & Vossen, G. (2001): *Transactional Information Systems*, Elsevier, USA.
- Wood, C. M., & Sute, T. A. 2004. Making marketing principles tangible: Online auctions as living case studies. *Journal of Marketing Education*, 26(2), 137-144.
- Yen, C., & Lu, H. (2008). Factors influencing online auction repurchase intention. *Internet Research*, 18(1), 7-25.

Zhou, M., Dresner, M., & Windle, R. (2009). Revisiting feedback systems: Trust building in digital markets. *Information & Management*, 46(5), 279-284.

APPENDIX 1.0 - Bidding system

This is the core of the project. The source codes are available in the following files. They include:

- views.py
- models.py
- extra.py
- forms.py
- admin.py
- tasks.py

APPENDIX 1.1 (views.py)

”

These are the contents of views.py in the root of the project. This also contains views for creating of new users, registering new user, logging into the system, editing user's account, changing user's password and bid making processes e.t.c.

”

```
from django.template import RequestContext, Context, loader, Template
from django.core.urlresolvers import reverse
from django.shortcuts import render_to_response, get_object_or_404
from django.http import HttpResponseRedirect, HttpResponse
from YAAS import extra
from YAAS.forms import LoginForm, RegistrationForm, EditForm, AuctionForm, BidForm,
PasswordForm
from YAAS.models import CustomUser, Item, Bid
from django.core.mail import send_mail, EmailMessage
from django.contrib.admin.views.decorators import staff_member_required
from django.core.files.base import ContentFile
from datetime import datetime, timedelta
from django.contrib.auth import logout
from django.contrib.auth.decorators import login_required
```

```

from django.utils.hashcompat import sha_constructor
import datetime, random
import decimal
from YAAS.advert.models import Advertisement

def login(request, template_name="account/login.html"):
    """
        This is form is used to make a user login.
    """
    if request.method == 'POST':
        postdata = request.POST.copy()
        page_title='Login form'
        form = LoginForm(request, postdata)
        if form.is_valid():
            un = postdata.get('username','')
            pw = postdata.get('password','')
            hashpw = extra.hashPassword(pw)
            from django.contrib.auth import login, authenticate
            new_user = authenticate(username=un, password=hashpw)
            if new_user and new_user.is_active:
                login(request, new_user)
                request.session['session_id'] = extra.generate_session_id()
                return HttpResponseRedirect(reverse('my_account'))
            else:
                return render_to_response('errors/login.html',
                    context_instance=RequestContext(request))
        else:
            form = LoginForm(request=request, label_suffix='')

            # set the test cookie on our first GET request
            request.session.set_test_cookie()
    return render_to_response(template_name, locals(),
        context_instance=RequestContext(request))

def register(request, template_name="account/register.html"):
    """
        This allows the anonymous user to become a registered user.
        This is the form used to register a new user and sends email with
        the action link with a time out.
    """
    if request.user.is_authenticated():
        # They already have an account; don't let them register again
        return HttpResponseRedirect(reverse('YAAS.views.my_account'))
    if request.method == 'POST':
        postdata = request.POST.copy()
        page_title='Registration form'
        form = RegistrationForm(postdata)
        if form.is_valid():

```

```

# Build the activation key for their account
human = True
un = postdata.get('user_name','')
pw = postdata.get('pass_word','')
em = postdata.get('email','')
fn = postdata.get('first_name','')
ln = postdata.get('last_name','')
pn = postdata.get('phone_number','')
sx = postdata.get('sex','')

salt = sha_constructor(str(random.random())).hexdigest()[:5]
activation_key = sha_constructor(salt+un).hexdigest()
key_expires = datetime.datetime.today() + datetime.timedelta(2)

# Create and save their profile
hashpw = extra.hashPassword(pw)
new_profile = CustomUser.objects.create_user(username=un, email=em,
password=hashpw)

new_profile.is_active = False
new_profile.first_name = fn
new_profile.last_name = ln
new_profile.activation_key = activation_key
new_profile.keyexpiry_date = key_expires
new_profile.phone_number = pn
new_profile.sex = sx

new_profile.save()

t = loader.get_template('registration/email.txt')
c = Context({
    'firstname':
        new_profile.first_name,
    'lastname':
        new_profile.last_name,
    'site_name':
        Auction Site',
    'username':
        new_profile.username,
    'activationkey':
        new_profile.activation_key,
    'admin':
        'Kenneth Odoh',
})

email_subject = 'Your new YAAS account'
send_mail(email_subject, t.render(c), 'account@example.com', [new_profile.email],
fail_silently=False)

```

```

        return HttpResponseRedirect(reverse('my_account'))
    else:
        #errors
        form = RegistrationForm()
        return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

'''
        This allows the newly registered user to activate their action
'''

def confirm(request, activation_key,template_name="account/confirm.html"):
'''
        This is used to generate the activation links. This has a expiry date.
'''

    page_title='User Confirmation form'
    if request.user.is_authenticated():
        has_account=True
        return HttpResponseRedirect(reverse('my_account'))
    user_profile = get_object_or_404(CustomUser, activation_key=activation_key)
    if user_profile.keyexpiry_date > datetime.datetime.today():
        expired=False
        user_profile.is_active = True
        user_profile.save()
        return HttpResponseRedirect(reverse('login'))
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def edit_account(request, template_name="account/update.html"):
'''
        This allows the registered user to alter user registered information. This is
used to changes to user's data.
'''

    if request.method == 'POST' and request.user.is_authenticated():
        page_title='Edit User Information form'
        postdata = request.POST.copy()
        form = EditForm(postdata)
        if form.is_valid():
            human = True
            user_profile = get_object_or_404(CustomUser, pk=request.user.id)

            email = postdata.get('email',"")
            first_name = postdata.get('first_name',"")
            last_name = postdata.get('last_name',"")
            phone_number = postdata.get('phone_number',"")
            sex = postdata.get('sex',"")

            my_list = []

```

```

if first_name <> "":
    user_profile.first_name = first_name
    my_list.append("first name")

if last_name <> "":
    user_profile.last_name = last_name
    my_list.append("last name")

if email <> "":
    user_profile.email = email
    my_list.append("email")

if phone_number <> "":
    user_profile.phone_number = phone_number
    my_list.append("phone name")

if sex <> "":
    user_profile.sex = sex
    my_list.append("sex")

user_profile.save()
t = loader.get_template('registration/update.txt')
c = Context({
    'firstname':
        new_profile.first_name,
    'lastname':
        new_profile.last_name,
    'site_name':
        Auction Site',
    'admin':
        'Kenneth Odoh',
    'my_list':
        my_list,
    })

email_subject = 'Your YAAS account has been updated'
send_mail(email_subject, t.render(c), 'account@example.com', [new_profile.email],
fail_silently=False)
return HttpResponseRedirect(reverse('my_account'))
else:
    #errors
    form = EditForm()
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def password_change(request, template_name="account/password_change.html"):
    """
        This allows the registered user to change their password

```

```

'''
if request.method == 'POST' and request.user.is_authenticated():
    page_title='Password Change form'
    postdata = request.POST.copy()
    form = PasswordForm(CustomUser,postdata)
    if form.is_valid():
        human = True
        pw = postdata.get('new_password',"")
        user_profile = get_object_or_404(CustomUser, pk=request.user.id)
        hashpw = extra.hashPassword(pw)
        user_profile.set_password(hashpw)
        user_profile.save()

                                #force user log out
        return HttpResponseRedirect(reverse('login'))
    else:
        form = PasswordForm(CustomUser)
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

#           This allows the registered user to log out of the site

def logoutview(request, template_name="account/logout.html"):
    page_title='Logout form'
    try:
        del request.session['session_id']
    except KeyError:
        pass
    logout(request)
    return render_to_response(template_name,locals(),
context_instance=RequestContext(request))

@login_required(login_url='/account/login/')
def makebid(request, item_id, template_name="auction/makebid.html"):
    '''
        This allows the registered user to make bids on the site. The bids are arranged
        in a list that deletes duplicate. Every new bids must be higher than all previous bids. However,
        since duplicates are eliminated every bid is unique. We cannot have a bid with the same
        amount twice. The bids are sorted according to bid amount and bid date. Therefore the first
        element in the list is always the highest bid. We can obtain the value use the index 0. Email is
        sent whenever a bid is made. You cannot bid on the item that you created.
    '''
    itemid = item_id
    item = get_object_or_404(Item, pk=itemid)
    page_title='Make Bid form'

    if request.method == 'POST' and request.user.is_authenticated():
        postdata = request.POST.copy()

```



```

form = BidForm(postdata)
user_profile = get_object_or_404(CustomUser, pk=request.user.id)
bidQuerySet = Bid.active.filter(item=item)
highest_bid_list = list(bidQuerySet.order_by('-bid_price').values_list('bid_price', flat=True))

if form.is_valid() and user_profile != item.owner:
    human = True
    amount = postdata.get('amount')
    decAmount= decimal.Decimal(str(amount))
    dechighBid= decimal.Decimal(str(item.highestbid))
    status=True
    if highest_bid_list is None:
        highest_bid_list.append(item.highestbid)
    if decAmount > dechighBid:
        highest_bid_list = extra.uniq(highest_bid_list)
        if highest_bid_list:
            for bidprice in highest_bid_list:
                if decimal.Decimal(str(bidprice)) == decAmount:
                    status=False
                    break

    if status:
        bid = Bid(bid_price=amount , user=user_profile, item=item)
        bid.save()
        highest_bid_list = extra.uniq(highest_bid_list)
        if highest_bid_list:
            item.highestbid = highest_bid_list[0]
            item.save()
            #send email to bidderin
            item_name = bid.item.name
            owner_email = bid.item.owner.email
            recipientlist = list(bidQuerySet.values_list('user__email', flat=True))
            recipientlist.append(owner_email)

        t = loader.get_template('registration/makebid.txt')
        c = Context({
            amount,
            item_name,
            'Kenneth Odoh',
            datetime.datetime.now(),
        })

        email_subject = 'A new bid has been made on the auction'
        core_msg = EmailMessage(subject=email_subject,
            from_email='account@example.com', to=recipientlist,
            body=t.render(c),
            'amount':
            'item_name':
            'admin':
            'timestamp':

```

```

        core_msg.send(fail_silently=False)
        return HttpResponseRedirect(reverse('showCurrentBid', args=(bid.id,)))
                                   #render bid less than highest bid to template
    else:
        return render_to_response('errors/makebid.html',
                                   context_instance=RequestContext(request))
    else:
        form = BidForm()
        return render_to_response(template_name, locals(),
                                   context_instance=RequestContext(request))

def auction_create(request, template_name="auction/createauction.html"):
    """
    This is used to create new items(Auctions). The created items are automatically displayed
    on the latest item and can be searchable. Email is sent to the person who created the item once
    the item is created.
    """
    if request.method == 'POST' and request.user.is_authenticated():
        page_title='Auction Creation form'
        postdata = request.POST.copy()
        form = AuctionForm(postdata, request.FILES)
        if form.is_valid():
            human = True
            name = postdata.get('name','')
            description = postdata.get('description','')
            minimum_price = postdata.get('minimum_price',0.01)
            ownerid = request.user.id
            user_profile = get_object_or_404(CustomUser, pk=ownerid)

            item = Item(name=name, description=description, minimum_price=minimum_price,
            owner=user_profile, highestbid=minimum_price)
            item.save()

            file_content = ContentFile(request.FILES['image'].read())
            item.image.save(request.FILES['image'].name, file_content, save=False)
            item.save()

            t = loader.get_template('registration/createAuction.txt')
            c = Context({
                'firstname':
                    user_profile.first_name,
                'lastname':
                    user_profile.last_name,
                'site_name':
                    'YAAS
Auction Site',
                'admin':
                    'Kenneth Odoh',
            })

```

```

        email_subject = 'Your have created a new auction'
        send_mail(email_subject, t.render(c), 'account@example.com', [user_profile.email],
fail_silently=False)
        return HttpResponseRedirect(reverse('showCurrentItem', args=(item.id,)))
    else:
        form = AuctionForm()
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

```

```

def auction_update(request, item_id, template_name="auction/updateauction.html"):
    """

```

This is used to update new items(Auctions). The currently updated items are automatically displayed on the latest item and can be searchable. Email is sent to the person who updated the item once the item is created. The item can only be updated by the creator of the item. The other users won't even see the link to update an item that they have not created.

```

    """
    itemid = item_id
    item = get_object_or_404(Item, pk=itemid )
    page_title='Auction Update form'
    if request.method == 'POST' and request.user.is_authenticated():
        postdata = request.POST.copy()
        form = AuctionForm(postdata, request.FILES)
        user_profile = get_object_or_404(CustomUser, pk=request.user.id)
        if form.is_valid() and item.owner == user_profile:
            human = True
            name = postdata.get('name',")
            description = postdata.get('description',")
            minimum_price = postdata.get('minimum_price',0.01)
            image = request.FILES['image']

    my_list = []
    if item:
        #update
        if name <> "":
            item.name = name
            my_list.append("name")

        if description <> "":
            item.description = description
            my_list.append("description")

        if minimum_price <> 0.01:
            item.minimum_price = minimum_price
            my_list.append("minimum price")

        if image:
            file_content = ContentFile(request.FILES['image'].read())

```

```

        item.image.save(request.FILES['image'].name, file_content, save=False)
        item.save()
        my_list.append("image")

    item.save()

    t = loader.get_template('registration/updateAuction.txt')
    c = Context({
        'firstname':
            user_profile.first_name,
        'lastname':
            user_profile.last_name,
        'site_name': 'YAAS
Auction Site',
        'admin':
            'Kenneth Odoh',
        'my_list':
my_list,
    })

    email_subject = 'Your have updated your auction'
    send_mail(email_subject, t.render(c), 'account@example.com', [item.owner.email],
fail_silently=False)
    return HttpResponseRedirect(reverse('showCurrentUpdatedItem', args=(item.id,)))
else:
    form = AuctionForm()
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def showAllCreatedAuction(request, template_name="auction/showauction.html"):
    """
    This is used to display all the created auctions.
    """
    page_title='Display all auctions'
    myItem = Item.active.filter(owner=request.user)
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def showAllMadeBid(request, template_name="auction/showbid.html"):
    """
    This is used to display all the bids.
    """
    page_title='Show my created item'
    myBid = Bid.active.filter(user=request.user)
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def showCurrentBid(request, bid_id, template_name="auction/showNewbid.html"):
    """
    This is used to display currently made bid.

```

```

'''
page_title='Show currently created item'
newBid = get_object_or_404(Bid, pk=bid_id, user=request.user)
return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def showCurrentItem(request, item_id, template_name="auction/showNewCreatedItem.html"):
'''
    This is used to display currently created item.
'''
newItem = get_object_or_404(Item, pk=item_id, owner=request.user)
return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def showCurrentUpdatedBid(request, bid_id, template_name="auction/showupdatedbid.html"):
'''
    This is used to display currently updated bid.
'''
page_title='Show currently updated bid'
updatedBid = get_object_or_404(Bid, pk=bid_id, user=request.user)
return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def showCurrentUpdatedItem(request, item_id,
template_name="auction/showupdatedItem.html"):
'''
    This is used to display currently updated item.
'''
page_title='Show currently updated item'
updatedItem = get_object_or_404(Item, pk=item_id, owner=request.user)
return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

@login_required(login_url='/account/login/')
def my_account(request, template_name="account/my_account.html"):
'''
    This is used to create user profile.
'''
page_title='Account Page'
name = request.user.username
advertItems = Advertisement.active.all()
items = Item.active.all()
num_of_item=items.count()
item_id = None
recommendedItem = {}
mostbiddeditem = None
result = {}
itemobjDict = {}
try:

```

```

    featureditems = Item.active.all()[:5]
    if num_of_item > 10:
        for item in items:
            bids = Bid.active.filter(item = item)
            bidcount = bids.count()
            result[item.id] = bidcount

            item_id , bidcount = extra.maxValueBid(result)
            mostbiddeditem = get_object_or_404(Item, pk=item_id)
            recommendedItem = extra.sortedItemDictionary(result)
            for key, value in recommendedItem:
                itemObj = get_object_or_404(Item, pk=key)
                itemobjDict[itemObj] = value
    except ValueError:
        pass
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def getBidsOnItem(request, itemid, template_name="auction/bidsfromitem.html"):
    """
    This is used to get bids from the item.
    """
    page_title='Bids on item information'
    bids = Bid.active.filter(item__id=itemid)
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def getItemDetails(request, item_id, template_name="auction/itemdetails.html"):
    """
    This is used to item details.
    """
    page_title="Item's Details"
    userid = request.user.id
    item = get_object_or_404(Item, pk=item_id)
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def getPicture(request, item_id, template_name="auction/itempictures.html"):
    """
    This is used to get pictures of item.
    """
    page_title='Pictures Details'
    item = get_object_or_404(Item, pk=item_id)
    return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

def indexView(request, template_name="index.html"):
    """
    This is used to create the index page.
    """

```

```

page_title='Index Page'
latestItem = Item.active.all()[:5]
advertItems = Advertisement.active.all()
return render_to_response(template_name, locals(),
context_instance=RequestContext(request))

```

APPENDIX 1.2 (models.py)

models.py

'''

This customizes the user object available in django by inheritance to add extra fields. This also saved the image in the file system instead of the database. We can achieve some performance by this method. This contains all the database components.

'''

```

from django.contrib.auth.models import User , UserManager # UserManager
from django.db import models
from YAAS import extra
from YAAS.extra import ContentTypeRestrictedFileField
from datetime import datetime, timedelta
from YAAS.customerReview.models import ProductReview

```

```

def get_image_path(instance, filename):
    return 'pictures/%s_%s' % (extra.createRandom(), filename)

```

```

class CustomUser(User):
    """User with app settings."""
    STATUS_CHOICES = (
        ('M', 'Male'),
        ('F', 'Female'),
    )
    created_at = models.DateTimeField(default=datetime.now(),blank=True,
editable=False)
    updated_at = models.DateTimeField(auto_now_add=True, blank=True)
    activation_key = models.CharField(null=True , blank=True, max_length=50)
    keyexpiry_date = models.DateTimeField(null=True , blank=True )
    phone_number = models.CharField(max_length=50)
    sex = models.CharField(max_length=1, choices=STATUS_CHOICES)

    #Use UserManger to get the create_user method, etc.
    objects = UserManager()

```

```

#This customizes the item to show only visible items
class ActiveItemManager(models.Manager):
    def get_query_set(self):
        return super(ActiveItemManager,
self).get_query_set().filter(status=True)

#This can be alternatively called auctions
class Item(models.Model):
    """
    True         visible
    False        invisible
    """
    owner = models.ForeignKey(CustomUser, db_index=True)
    name = models.CharField(max_length=30, help_text='Enter the name of the
item', db_index=True)
    description = models.CharField(max_length=150, help_text='Enter the description
of the item', db_index=True)
    minimum_price = models.DecimalField(max_digits=9, decimal_places=2,
default=0.00)
    highestbid = models.DecimalField(max_digits=9, decimal_places=2, default=0.00)
    status = models.BooleanField(default=True)
    created_at = models.DateTimeField(default=datetime.now(), blank=True, editable=False)
    updated_at = models.DateTimeField(auto_now_add=True, blank=True)
    end_date = models.DateTimeField( default=datetime.now()+timedelta(days=3) )
    image = ContentTypeRestrictedFileField(
upload_to=get_image_path,
content_types=['image/jpeg' , 'image/jpeg', 'image/tiff', 'image/png', 'image/gif'],
max_upload_size=2.5*1024*1024,
blank=True,
null=True
)

    objects = models.Manager()
    active = ActiveItemManager()
    class Meta:
        ordering = ['-created_at']

    def __unicode__(self):
        return self.name

    #make bid
    @models.permalink
    def get_absolute_url(self):
        return ('YAAS.views.makebid', [str(self.id)])

#This customizes the item to show only visible items
class ActiveBidManager(models.Manager):
    def get_query_set(self):
        return super(ActiveBidManager,
self).get_query_set().filter(bid_status=False).filter(availability=True)

class Bid(models.Model):
    """

```



```

        #bid_status
        #False      waiting
        #True       resolved

        #payment_status
        #False      not paid
        #True       paid

        #availability :needed during maintenance
        #False      invisible
        #True       visible
    """
    user = models.ForeignKey(CustomUser, db_index=True)
    item = models.ForeignKey(Item, db_index=True)
    review = models.ForeignKey(ProductReview, null=True, db_index=True)
    bid_status = models.BooleanField(default = False)
    bid_price = models.DecimalField(max_digits=9,decimal_places=2, default=0.01)
    availability = models.BooleanField(default = True)
    created_at = models.DateTimeField(default=datetime.now(),blank=True)
    is_winner = models.BooleanField(default = False)
    is_emailed = models.BooleanField(default = False)
    objects = models.Manager()
    active = ActiveBidManager()
    class Meta:
        ordering = ['-bid_price' , '-created_at']

```

APPENDIX 1.3 (url.py)

```

"""

```

This make the views available in url.

```

"""

```

```

#from django.conf.urls.defaults import patterns, include, url
import os
#from YAAS.views. import *
from YAAS import settings
from django.conf.urls.defaults import *

# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()

skip_last_activity_date = [
    #Your expressions go here
]

```

```

urlpatterns = patterns("",
    # Examples:
    # url(r'^$', 'YAAS.views.home', name='home'),
    # url(r'^YAAS/', include('YAAS.foo.urls')),

    # Uncomment the admin/doc line below to enable admin documentation:
    # url(r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    # url(r'^admin/', include(admin.site.urls)),
    # (r'^catalog/$', 'YAAS.views.catalog'),

#)

urlpatterns = patterns("",
    # other commented code here
    (r'^admin/', include(admin.site.urls)),

    #(r'^accounts/', include('django.contrib.auth.urls')),

    (r'^static/(?P<path>.*)*$', 'django.views.static.serve',
        { 'document_root' : os.path.join(settings.CURRENT_PATH, 'static')
    }
    ),
    (r'^account/login/$', 'YAAS.views.login',{ 'template_name': 'account/login.html',
'SSL': settings.ENABLE_SSL}, 'login'),

    (r'^account/register/$', 'YAAS.views.register',{ 'template_name':
'account/register.html', 'SSL': settings.ENABLE_SSL}, 'register'),

    (r'^account/confirm/(?P<activation_key>\w+)/$',
'YAAS.views.confirm',{ 'template_name': 'account/confirm.html', 'SSL': settings.ENABLE_SSL },
'confirm'),

    (r'^account/edit/$', 'YAAS.views.edit_account',{ 'template_name':
'account/update.html', 'SSL': settings.ENABLE_SSL}, 'edit_account'),

    (r'^account/passwordchange/$',
'YAAS.views.password_change',{ 'template_name': 'account/password_change.html', 'SSL':
settings.ENABLE_SSL}, 'password_change'),

    (r'^account/logout/$', 'YAAS.views.logoutview', { 'template_name':
'account/logout.html', 'SSL': settings.ENABLE_SSL}, 'logout'),

    (r'^auction/makebid/(?P<item_id>\d+)/$',
'YAAS.views.makebid',{ 'template_name': 'auction/makebid.html'}, 'makebid'),

    (r'^auction/createauction/$', 'YAAS.views.auction_create',{ 'template_name':
'auction/createauction.html'}, 'auction_create'),

```

```

        (r'^auction/updateauction/(?P<item_id>\d+)/$',
'YAAS.views.auction_update',{template_name': 'auction/updateauction.html'}, 'auction_update'),

        (r'^auction/showmycreatedauction/$',
'YAAS.views.showAllCreatedAuction',{template_name': 'auction/showauction.html'},
'showAllCreatedAuction'),

        (r'^auction/showmybid/$', 'YAAS.views.showAllMadeBid',{template_name':
'auction/showbid.html'}, 'showAllMadeBid'),

        (r'^auction/showcurrentbid/(?P<bid_id>\d+)/$',
'YAAS.views.showCurrentBid',{template_name': 'auction/showNewbid.html'}, 'showCurrentBid'),

        (r'^auction/showcurrentitem/(?P<item_id>\d+)/$',
'YAAS.views.showCurrentItem',{template_name': 'auction/showNewCreatedItem.html'},
'showCurrentItem'),

        (r'^auction/showcurrentupdatedbid/(?P<bid_id>\d+)/$',
'YAAS.views.showCurrentUpdatedBid',{template_name': 'auction/showupdatedbid.html'},
'showCurrentUpdatedBid'),

        (r'^auction/showcurrentupdateditem/(?P<item_id>\d+)/$',
'YAAS.views.showCurrentUpdatedItem',{template_name': 'auction/showupdatedItem.html'},
'showCurrentUpdatedItem'),

        (r'^account/my_account/$', 'YAAS.views.my_account',{template_name':
'account/my_account.html'}, 'my_account'),

        (r'^auction/getbidsonitem/(?P<itemid>\d+)/$',
'YAAS.views.getBidsOnItem',{template_name': 'auction/bidsfromitem.html'}, 'getBidsOnItem'),

        (r'^auction/getitemdetails/(?P<item_id>\d+)/$',
'YAAS.views.getItemDetails',{template_name': 'auction/itemdetails.html'}, 'getItemDetails'),

        (r'^auction/getpicture/(?P<item_id>\d+)/$',
'YAAS.views.getPicture',{template_name': 'auction/itempictures.html'}, 'getItemPicture'),

        (r'^index/$', 'YAAS.views.indexView',{template_name': 'index.html'}, 'index'),

#search functionality
(r'^search/', include('YAAS.search.urls')),

#advertisement functionality
(r'^advert/', include('YAAS.advert.urls')),

#chat functionality
(r'^comment/', include('YAAS.comment.urls')),

#capcha to prevent spam

```

```

(r'^captcha/', include('YAAS.captcha.urls')),

        #generate statistical reports
(r'^stats/', include('YAAS.stats.urls')),

        #generate statistical graphs
(r'^graph/', include('YAAS.graph.urls')),

        #generate customer review
(r'^review/', include('YAAS.customerReview.urls')),
)

```

APPENDIX 1.4 (extra.py)

'''

This contains all the helper methods used throughout the project. Even a validation checking data type for images named `ContentTypeRestrictedFileField`, create random number needed to create unique names for the images saved in the file system of the server and some session management methods.

'''

```

import hashlib
from django.db.models import ImageField
from django.forms import forms
from django.template.defaultfilters import filesizeformat
from django.utils.translation import ugettext_lazy as _
import random
import operator

class ContentTypeRestrictedFileField(ImageField):
    """
    Same as FileField, but you can specify:
    * content_types - list containing allowed content_types. Example: ['application/pdf',
'image/jpeg']
    * max_upload_size - a number indicating the maximum file size allowed for upload.
    2.5MB - 2621440
    5MB - 5242880
    10MB - 10485760
    20MB - 20971520
    50MB - 5242880
    100MB 104857600
    250MB - 214958080
    500MB - 429916160
    """

```

```

"""
def __init__(self, *args, **kwargs):
    self.content_types = kwargs.pop("content_types")
    self.max_upload_size = kwargs.pop("max_upload_size")

    super(ContentTypeRestrictedFileField, self).__init__(*args, **kwargs)

def clean(self, *args, **kwargs):
    data = super(ContentTypeRestrictedFileField, self).clean(*args, **kwargs)

    file = data.file
    try:
        content_type = file.content_type
        if content_type in self.content_types:
            if file._size > self.max_upload_size:
                raise forms.ValidationError(_('Please keep filesize under %s. Current filesize %s')
% (filesizeformat(self.max_upload_size), filesizeformat(file._size)))
            else:
                raise forms.ValidationError(_('Filetype not supported.))
        except AttributeError:
            pass

    return data

#This is needed to hash the user password.
def hashPassword(password):
    """
    This is used to hash the password. The password is not saved in clear text.
    """
    hashed_password = hashlib.sha1(password).hexdigest()
    return hashed_password

#This is needed for random number generation
def createRandom():
    """
    This is used to generate random number
    """
    strlenLimit, randomValue = 18, ""
    val = random.randint(1, 1000000000000000000)
    strLenght = len(str(val))
    if (strLenght < strlenLimit):
        randomValue = ('0' * abs(strLenght - strlenLimit)) + str(val)
    return randomValue

#This is used for session id generation
def generate_session_id():
    """

```

This is used to generate unique session id that can be used to conveniently identify unique user sessions.

```

"""
    session_id = ""
    characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890!@#$%^&*()'
    session_id_length = 50
    for y in range(session_id_length):
        session_id += characters[random.randint(0, len(characters)-1)]
    return session_id

```

```

#get maximum key, value on dictionary
def maxValueBid(stats):
    #http://stackoverflow.com/questions/268272/getting-key-with-maximum-value-in-dictionary
    key = max(stats.iteritems(), key=operator.itemgetter(1))[0] #item id
    value = max(stats.iteritems(), key=operator.itemgetter(1))[1] #bid count
    return key , value

```

```

#sort dictionary by value
def sortedItemDictionary(x):
    sorted_x = sorted(x.iteritems(), key=operator.itemgetter(1), reverse=True)[:10]
    return sorted_x

```

```

def uniq(alist):
    """
        This is used to make a list element unique. This is implemented by enclosing a set in a list.
        This deletes duplicate
    """
    set = {}
    return [set.setdefault(e,e) for e in alist if e not in set]

```

APPENDIX 1.5 (forms.py)

```

"""
This contains all the forms needed for the bidding system.
"""

```

```

from django import forms
from django.forms.widgets import PasswordInput
from YAAS.models import Bid, CustomUser
from YAAS import extra
from YAAS.captcha.fields import CaptchaField

```

```

class LoginForm(forms.Form):
    """
        This is used to enable user login
    """
    username = forms.CharField(label="Username", max_length=20)
    password = forms.RegexField(label="Password",
    regex=r'^(?=.*\W+).*$', widget=forms.PasswordInput, min_length=6)

    def __init__(self, request=None, *args, **kwargs):
        self.request = request
        super(LoginForm, self).__init__(*args, **kwargs)

    def clean(self):
        if self.request:
            if not self.request.session.test_cookie_worked():
                raise forms.ValidationError("Cookies
must be enabled.")
        return self.cleaned_data

```

```

class RegistrationForm(forms.Form):
    """
        This is used to register a new user who does not have an account.
    """
    STATUS_CHOICES = (
        ('M', 'Male'),
        ('F', 'Female'),
    )
    user_name = forms.CharField(label="Username", max_length=20)
    pass_word = forms.RegexField(label="Password",
    regex=r'^(?=.*\W+).*$', widget=forms.PasswordInput, min_length=6, help_text='Password must
be six characters long and contain at least one non-alphanumeric character.')
    retype_password = forms.RegexField(label="Password confirmation",
    regex=r'^(?=.*\W+).*$', widget=forms.PasswordInput, min_length=6, help_text='Password must
be six characters long and contain at least one non-alphanumeric character.')
    email = forms.EmailField(label="Email", max_length="50")
    first_name = forms.CharField(label="First name", max_length=20)
    last_name = forms.CharField(label="Last name", max_length=20)
    phone_number = forms.CharField(label="Phone number", max_length=20)
    sex = forms.ChoiceField(label="Sex", choices=STATUS_CHOICES,
    widget=forms.RadioSelect)
    captcha = CaptchaField()

```

```

def clean_pass_word(self):
    if self.data['pass_word'] != self.data['retype_password']:
        raise forms.ValidationError('Passwords do not
match!')
    return self.data['pass_word']

def isValidUsername(self):
    try:
        CustomUser.objects.get(username=self.data['user_name'])
        raise forms.ValidationError('The username is
already taken. Please choose another')
    except:
        pass
    return

def clean(self):
    self.clean_pass_word()
    self.isValidUsername()
    return self.cleaned_data

class EditForm(forms.Form):
    """
        This is used to edit the form and this is used an interface.
    """
    STATUS_CHOICES = (
        ('M', 'Male'),
        ('F', 'Female'),
    )
    email = forms.EmailField(label="Email", max_length="50", required=False)
    first_name = forms.CharField(label="First name", max_length=20,
required=False)
    last_name = forms.CharField(label="Last name", max_length=20,
required=False)
    phone_number = forms.CharField(label="Phone number", max_length=20,
required=False)
    sex = forms.ChoiceField(label="Sex", choices=STATUS_CHOICES,
widget=forms.RadioSelect, required=False)
    captcha = CaptchaField()

class AuctionForm(forms.Form):
    """
        This is the form used to make an auction.
    """
    name = forms.CharField(label="Auction name", max_length=30 )
    description = forms.CharField(label="Description", max_length=150, widget=forms.Textarea)
    minimum_price = forms.DecimalField(label="Minimum Price", max_digits=9,
decimal_places=2)
    image = forms.ImageField(label="Photo")
    captcha = CaptchaField()

```



```

def clean_image(self):
    image = self.cleaned_data.get('image',False)
    if image:
        if image._size > 2.5*1024*1024:
            raise ValidationError("Image file too large ( > 2.5mb )")
        return image
    else:
        raise ValidationError("Couldn't read uploaded image")

#Bid form not necessary
class BidForm(forms.Form):
    """
        This is the form used to make a bid.
    """
    amount = forms.DecimalField(max_digits=9, decimal_places=2)
    captcha = CaptchaField()

class PasswordForm(forms.Form):
    """
        This is the form used to change password.
    """
    old_password = forms.RegexField(label="Old Password",
    regex=r'^(?=.*\W+).*$', widget=forms.PasswordInput, min_length=6)

    new_password = forms.RegexField(label="New Password",
    regex=r'^(?=.*\W+).*$', widget=forms.PasswordInput, min_length=6)

    retype_newpassword = forms.RegexField(label="New Password Confirmation",
    regex=r'^(?=.*\W+).*$', widget=forms.PasswordInput, min_length=6)
    captcha = CaptchaField()

    def __init__(self, user=None, *args, **kwargs):
        self.user = user
        super(PasswordForm, self).__init__(*args, **kwargs)

    def clean_password(self):
        oldpass = self.cleaned_data['old_password']
        hashpw = extra.hashPassword(oldpass)
        valid = self.user.check_password(hashpw)
        if not valid:
            raise forms.ValidationError("Password Incorrect")

    def clean_pass_word(self):
        if self.data['new_password'] != self.data['retype_newpassword']:
            raise forms.ValidationError('Passwords do not
match!')

        return self.data['new_password']

```

```
def clean(self):
    self.clean_pass_word()
    return self.cleaned_data
```

APPENDIX 1.6 (admin.py)

```
'''
```

This is used to set up the administrator interface.

```
'''
```

```
from django.contrib import admin
from YAAS.models import CustomUser, Item, Bid

#This allows CustomUser to be hooked in the admin
admin.site.register(CustomUser)

#This allows Item to be hooked in the admin
admin.site.register(Item)

#This allows Bid to be hooked in the admin
admin.site.register(Bid)
```

APPENDIX 1.7 (tasks.py)

```
'''
```

This contains all the scheduled events for resolving the bids and for user / business analytics information gathering. The resolving of bids action takes place by making the items invisible to the user accessing the site. The user with the highest bid has his bid on the top of the bid queue. We can select this bid and identify the user. Then set his status to winner. We need to iterate through the bid again searching for the winner of the bid. Only the winner is found. An congratulatory email message is sent this user. The object users are sent message they they did not win the bid. We have to change this user emailed status to true. Otherwise, we keep spamming the user. If the winner is not able to pay we can modify the algorithm and choose the next potential winner on the bid queue.

```
'''
```

```
#http://bitkickers.blogspot.com/2010/07/djangocelery-quickstart-or-how-i.html
```

```
from YAAS.models import Item, Bid
from YAAS.stats.models import RegisteredUser, OnlineUser, StatBid
from YAAS.stats import stat
from django.template import RequestContext, Context, loader
from django.core.mail import send_mail, EmailMessage
from celery.task.schedules import crontab
from celery.decorators import periodic_task
from datetime import datetime, timedelta
from django.shortcuts import get_object_or_404

#make the items invisible
def makeAllItemsInvisible():
    itemQuerySet = Item.objects.all()
    itemQuerySet.update(status=False)

#make the bids invisible
def makeAllBidsInvisible():
    bidQuerySet = Bid.objects.all()
    bidQuerySet.update(availability=False)

#make the items visible
def makeAllItemsVisible():
    itemQuerySet = Item.objects.all()
    itemQuerySet.update(status=True)

#make the bids visible
def makeAllBidsVisible():
    bidQuerySet = Bid.objects.all()
    bidQuerySet.update(availability=True)

#resolve auction
@periodic_task(run_every=crontab(hour=1, minute=15, day_of_week="*"))
def resolveAuction():
    #make item and bid invisible
    makeAllItemsInvisible()
    makeAllBidsInvisible()

    #change bid status to resolved after time lapse
    threedays = datetime.today() - timedelta(days=3)

    myItem = Item.objects.filter(end_date__lte=threedays)
    for item in myItem:
        myBid_id = Bid.objects.filter(item=item).order_by('-bid_price')
        for b_id in myBid_id:
            bid_obj = get_object_or_404( Bid, pk=int(b_id.id) )
            bid_obj.is_winner=True
```

```

        bid_obj.save()
    break

    realBid =
    Bid.objects.exclude(is_emailed=True).filter(item__end_date__lte=threedays)
    for bid in realBid:
        bid_obj = get_object_or_404( Bid, pk=int( bid.id ) )
        if bid_obj.is_winner:
            #send email to winner
            t = loader.get_template('registration/winnerBid.txt')
            c = Context({
                'firstname':
                bid_obj.user.first_name,
                'lastname':
                bid_obj.user.last_name,
                'site_name': 'YAAS
Auction Site',
                'admin':
                'Kenneth Odoh',
                'bid_id':
                bid_obj.id,
            })
            email_subject = 'Your have won the auction YAAS
account'
            send_mail(email_subject, t.render(c),
'account@example.com', [bid_obj.user.email], fail_silently=False)
            bid_obj.is_emailed=True

        else:
            t = loader.get_template('registration/otherBidder.txt')
            c = Context({
                'firstname':
                bid_obj.user.first_name,
                'lastname':
                bid_obj.user.last_name,
                'site_name': 'YAAS
Auction Site',
                'admin':
                'Kenneth Odoh',
            })

            recipientlist = list(Bid.objects.exclude(is_emailed=True).values_list('user__email',
flat=True))
            bid_obj.is_emailed=True
            email_subject = 'Your auction has been resolved'
            core_msg = EmailMessage(subject=email_subject, body=t.render(c),
from_email='account@example.com', to=recipientlist)
            core_msg.send(fail_silently=False)

```

```

#auction has been resolved so maintenance is over
makeAllItemsVisible()
makeAllBidsVisible()

@periodic_task(run_every=crontab(hour=1, minute=25, day_of_week="*"))
def makeExpiredItemsBidInvisible():
    threedays = datetime.today() - datetime.timedelta(days=3)
    myItem = Item.objects.filter(end_date__lte=threedays )
    myItem.update(status=False)
    myBid = Bid.objects.filter(item__in=myItem)
    myBid.update(availability=False)

@periodic_task(run_every=crontab(hour=1, minute=30, day_of_week=0))
def deleteOldItemsandBids():
    hunderedandtwentydays = datetime.today() - datetime.timedelta(days=120)
    myItem = Item.objects.filter(end_date__lte=hunderedandtwentydays ).delete()
    myBid = Bid.objects.filter(end_date__lte=hunderedandtwentydays ).delete()

#populate the registereduser and onlineuser model at regular intervals

@periodic_task(run_every=crontab(hour=1, minute=45, day_of_week="*"))
def fillRegisterUserModel():
    regnum = stat.getNumofRegisteredUser()
    n_day = stat.getDay()
    n_month = stat.getMonth()
    n_year = stat.getYear()
    n_week = stat.getWeek(n_day, n_month, n_year)
    regUsrObj = RegisteredUser(no_of_reg_user=regnum, day=n_day,
    month=n_month, year=n_year, week=n_week)

    regUsrObj.save()

def fillOnlineUserModel():
    onlnum = stat.getNumofOnlineUser()
    n_day = stat.getDay()
    n_month = stat.getMonth()
    n_year = stat.getYear()
    n_week = stat.getWeek(n_day, n_month, n_year)
    onlinUsrObj = OnlineUser(no_of_online_user=onlnum, day=n_day,
    month=n_month, year=n_year, week=n_week)

    onlinUsrObj.save()

def fillStatBidModel():
    numbid = stat.getNumofBid()

```

```
n_day = stat.getDay()
n_month = stat.getMonth()
n_year = stat.getYear()
n_week = stat.getWeek(n_day, n_month, n_year)
bidObj = StatBid(no_of_bids=numbid, day=n_day, month=n_month,
year=n_year, week=n_week)

bidObj.save()
```

```
@periodic_task(run_every=crontab(hour="*/7", minute=50, day_of_week="**"))
def realwork():
    fillOnlineUserModel()
    fillStatBidModel()
```