



Bid Buy-Sell system using client-server architecture, solution of concurrent users for the web application through optimistic locking and multithreading

Younten Tshering¹, Suyogya Ratna Tamrakar², Shubhangini Gontia³

¹Associate Lecturer, Dept. of IT, Jigme Namgyel Engineering College, Royal University of Bhutan

²Master's in Computer Engineering, Asian Institute of Technology

³Master's in Computer Science, Asian Institute of Technology

ABSTRACT

Nowadays, the online web-based auction system has become an extremely popular component in the electronic marketplace. A practical case study will be introduced in this research to highlight the best practices for analyzing and designing a bidding system. The proposed Bid Buy-Sell (BBS) - Bidding system was designed and implemented using the UML (to illustrate the architectural model), MySQL and Spring Boot with java programming language. In the proposed BBS, the UML offering several diagrams enabled the new functions to be updated and easily added. The proposed BBS will help the bidders bid in fast and increase their chances of making a successful bid by suggesting a bid price and help the seller to achieve maximum profit. The main architectural challenges addressed are handling multiple concurrent transactions, real-time information updates and handling online payment through secure payment gateways. Along with the testing tools and methods that have been used based on the analysis and implementation of architecture design, the proposed BBS offers excellent advantages for the control of concurrent users through optimistic locking and multithreading.

Keywords: Auction System, Bidding System, Concurrent users, and Unified Modeling Language (UML)

INTRODUCTION

In recent years, the electronic marketplace has been exponentially growing in usability, size and worth. It is expected that this trend will exaggerate in the upcoming years [1]. Because of the rapidly growing internet environment, the customer can conveniently obtain the products that he/she purchases from the traditional market by online systems. The online platform is a main component of the electronic marketplace that makes use of electronic commerce mechanisms. The idea of our research is like any e-commerce platform in which buying and selling of goods are done. But the core addition feature will be the system of bidding before buying anything. The products added will be subjected to a certain time frame where users most probably buyers will be allowed to bid on the products and can buy the product once the bid time closes or when there are no other bidders. In the case of the Asian Institute of Technology (AIT), this platform can be used to sell the used items. More than one user could be interested in the same product, but with this system, there will be a sense of competition as one user bids over another user's bid at the same time and the product value will also increase.

It is not uncommon to have hundreds or thousands of users independently and concurrently interacting with the [2] resources in modern software systems. We generally want to avoid situations when changes made by one client are overridden by another one without even knowing. The main architectural challenges that will be addressed is handling multiple concurrent transactions, real-time information updates and handling online payment through secure payment gateways.

We have used MVC with Client-Server architecture. System backup and recovery is not handled as there are no lengthy or long transactions in our purpose system. However, we store some instant information in the cache, so that with little interruptions like network connectivity issues users will be able to restore their session data. The application has used MySQL as a data store. For developing, we have used Java Spring Boot for backend and HTML & CSS for frontend. Bidding system was designed and implemented using the UML (to illustrate the architectural model). In the proposed



BBS, the UML with several diagrams like use case, sequence and class diagrams helped to enable new functions to be added and easily updated.

RELATED WORK

To prevent our data integrity from being violated we often use locking mechanisms provided by database engine, or even use abstractions provided by tools like JPA.

Optimistic locking lets every client read and write data with the restriction that just before committing the transaction we need to check whether a particular record has not been modified by someone else in the meantime [2]. This is usually done by adding the current version or last modification timestamp attribute. We need to compare all attributes of an aggregate, while in the locking, we only check if version and aggregate ID are the same. All attributes consistency and version-based consistency both define a pre-condition for the request to be fulfilled.

There is an explicit and standard way of dealing with conditional requests in HTTP protocol. RFC 7232 defines this concept including a set of metadata headers indicating the state of the resource. The ETag [3] or entity tag is part of HTTP, the protocol for the World Wide Web. It is one of several mechanisms that HTTP provides for Web cache validation, which allows a client to make conditional requests. This mechanism allows caches to be more efficient and saves bandwidth, as a Web server does not need to send a full response if the content has not changed. ETag can also be used for optimistic concurrency control [4] to help prevent simultaneous updates of a resource from overwriting each other.

For the request from the users will be considered as the threads and multiple requests will be treated as multithreading. All the threads will be inside a thread pool. A thread pool is a collection of threads on the same machine that operate by continually processing tasks. Logically, a thread pool is associated with a set of task types, and each thread in the pool executes a piece of code that consumes a task, processes it, and dispatches one or more outgoing tasks to a thread pool [5]. This way we can handle the multiple threads to process the requests sent by the client and give back the response according to the requests within its lifespan.

The UML is a language used to specify, visually model [7], and document the artifacts of an Object-Oriented system under development. It denotes several ideas unification from various methods. UML is used in the system design to improve its reusability and maintainability. Object-oriented analysis methods offer class, use case, state chart, sequence, and other diagrammatic notations for modeling [6]. UML has been employed effectively in many projects for modeling different requirements and architectures [7]. With this UML approach, it has eased the process of understanding the system and architectures design to be robust for the purpose of controlling concurrent user through sequence diagram.

The performance testing involves recording and monitoring the performance levels for any websites. Performance testing tools are used to determine the time required to perform a task by a system. [13] JMeter tool is used to test the performance of BBS.

METHODOLOGY

In modern software systems, it is common to have hundreds or thousands of users independently and concurrently interacting with the application. Therefore, we generally want to avoid situations when changes made by one client are overridden by another one and to control the concurrent user request or access to application.

In this Architectural design, application consists of 3 hierarchically ordered subsystems or three tiers such as Client Tier, Middle Tier and Data Tier. The middle Tier or subsystem services data requests between the user interface and the database subsystem. Three Layer architectural pattern is used for the development of our web app where:

- the Web Browser implements the user interface.
- the Web Server serves requests from the web browser.
- the Database manages and provides access to the persistent data.

The Three Tier Architecture Model (Bid Buy Sell)

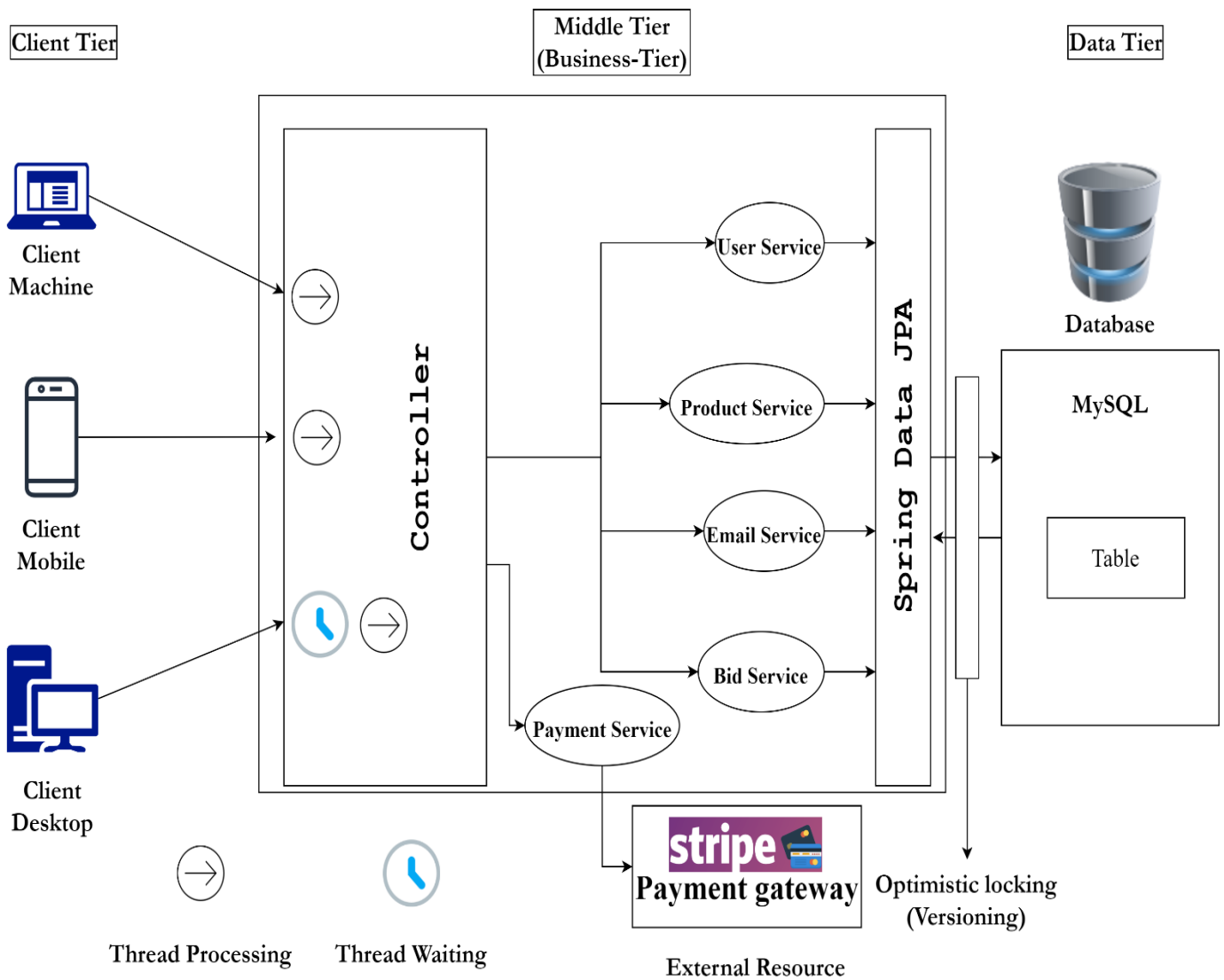


Figure 1: Architecture design

Use Case Diagram

The use case diagram is a visualization of a use-case [10, 6-9], i.e., the auction system interaction with the users. In the proposed Bid Buy-Sell, the use case mainly consists of register case, view products case, add product case, update product case, make a bid for products, specify time and price of bidding. Figure 2 shows the use case diagram for the actions that the actors (Seller, Bidder and Admin) can perform in our Bidding system.

Class Diagram

In Object-Oriented analysis and design, the class diagram is the most essential entity. It defines the kinds of objects that are present in the system and describes the static relationships between the system internal classes [10]. The operations and attributes of a class and the constraints that apply to the object's connection can be shown by the class diagram. Figure 3 displays the BBS class diagram in basic level to understand the classes. Figure 4 shows the BBS entities, such as product, user, and its detail with every interface and operation related to sequence diagram.

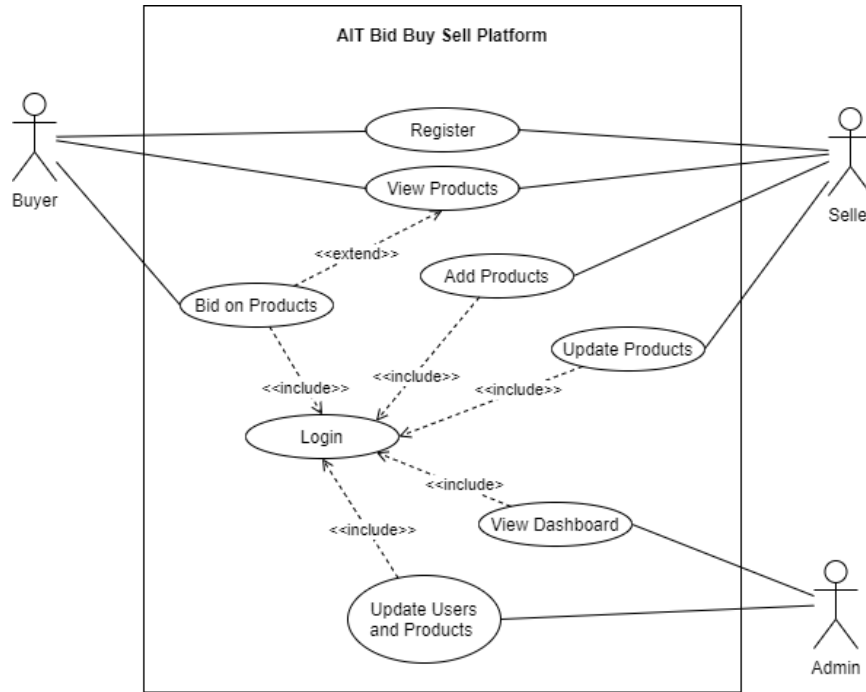


Figure 2: Use case diagram

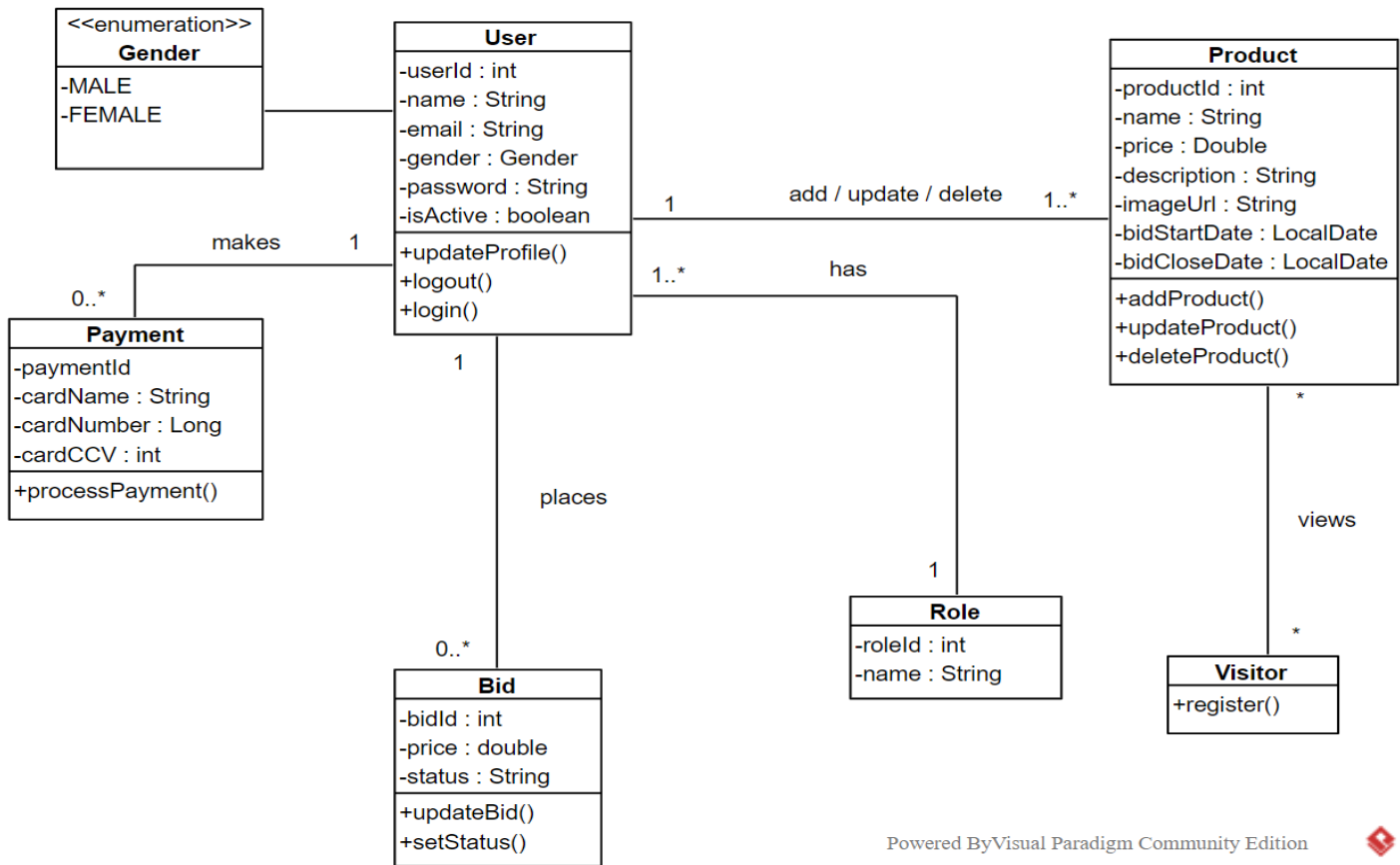


Figure 3: InitialClass Diagram



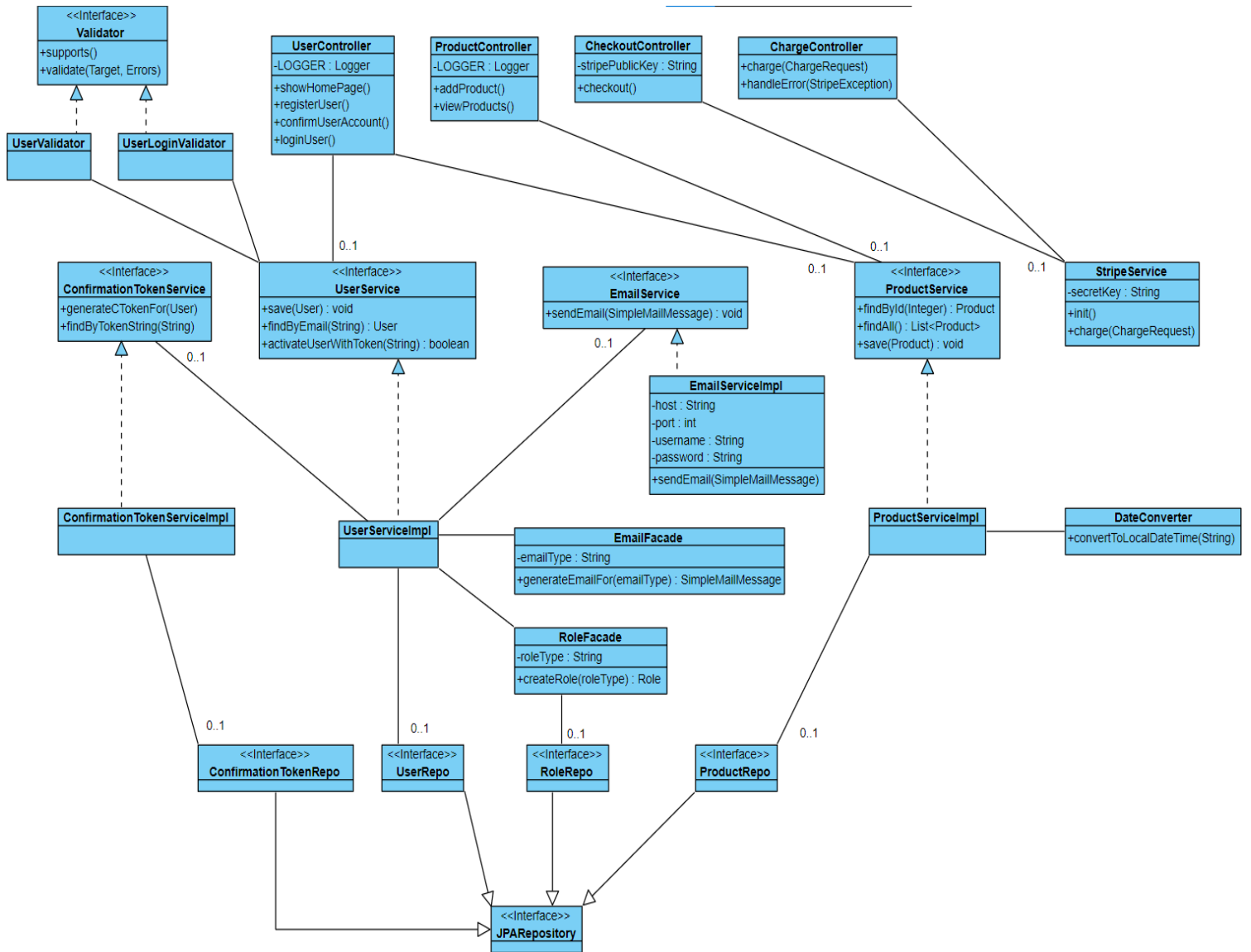


Figure 4: Detailed Class Diagram

Sequence Diagram

A sequence diagram is one of the UML dynamic models [6, 11], and it defines the interaction scene between the objects in time when the use case was executed and highlights the information sending time priority among objects. Usually, the sequence diagram illustrates the single use-case behavior. Figure 5 demonstrates the sequence diagram for the use case ‘Bid on Products’ in the proposed BBS.

Below sequence diagram is for the use case 'Bid on products'. Here we have shown two actors as client 1 and client 2 and bidController, server, bidRepo and bid (Entity) as the object for the sequence diagram. As shown in the sequence, when client 1 and client 2 both requests to update the bid amount simultaneously the timestamp taken here will be till nanosecond to diminish the possibility of clicking the bid at the same time. Once, the request is sent to the server, it will use locking to lock object and further proceed to update he bid entity. When the request from the second actor client 2 is sent to the server, say after 1 nanosecond, it will create another thread for the request and try to request the lock object, but since the thread 1 is already using it thread 2 will be sent to wait state until the lock is released by thread 1. After that, thread 2 will proceed to update the bid, and bid will be updated if the value is more than the latest bid amount.

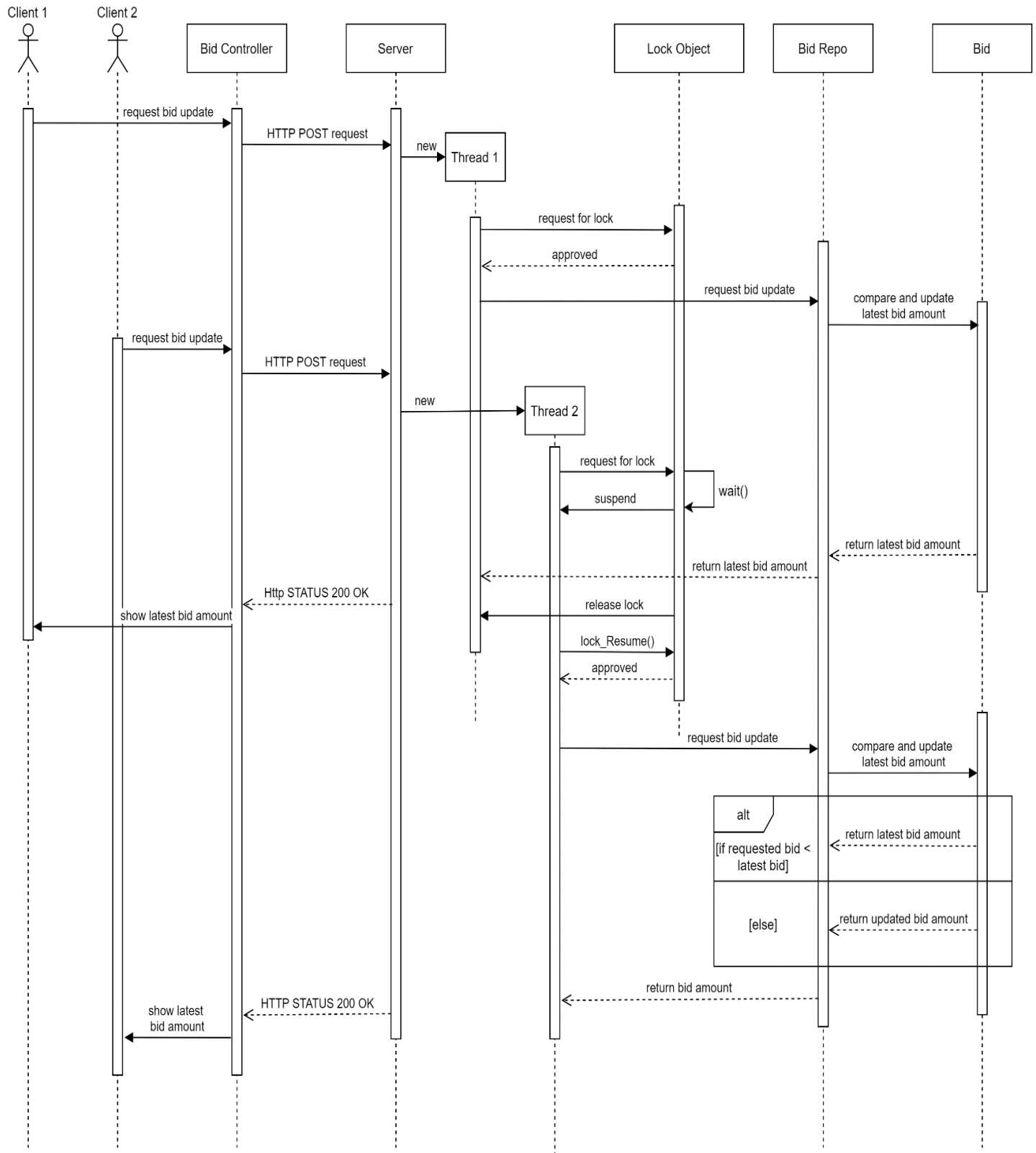


Figure 5: Sequence Diagram of Bid on Products (Use Case)

EXPERIMENT

The application testing was applied to our system (Bid Buy-Sell) for concurrent programs, the main architectural challenges are handling multiple concurrent transactions and real-time information updates. Handling multiple concurrent transactions is one of the factors to be checked where there will be many users active on the system and they will be able to bid on the products.

But multiple users may place bids on the same product at the same time. Another concern is updating the product price in real time for ongoing bidding processes. If the product is being bid by someone, there will be multiple users who will be on the same page waiting or trying to bid. If one bid is updated, other users must instantly see the updated bid amounts on their systems.

With JMeter, there are a variety of ways we can conduct a concurrent user test. For example, we can begin load testing with as few as 10 concurrent users and run these users for 5 minutes to establish our baseline performance metrics. After establishing a baseline, we can increase the number of concurrent users by 10 users a minute until we reach 100 concurrent users. We choose to follow that up with a test run for another 5 minutes for every 100 additional concurrent users to be sure that the results level out.

Some factors that may cause drops in webapp response time while adding concurrent users include additional allocation of memory on the web server or additional concurrent database connections on the backend. These could easily cause a drop in the average page load speed while waiting for the system resources to become free only to drop back to normal levels once the resources have been allocated.

To test this, we have chosen to run a test of 500 to 1000 concurrent users, or until we feel we have adequately proven that our webapp is capable of handling peak user numbers. These tests can be used to identify both the volume of users that causes unacceptable page load speeds as well as the number of concurrent page requests that causes the web app to crash. This may be done by running additional load tests that start at a higher volume of users to push the system to its limits.

RESULT

JMeter is used for the purpose of load testing and considered as one of the best tools. Apache JMeter is a Java open-source GUI-based software that is used as a performance testing tool for analyzing and measuring the performance of our application. Performance testing can be categorized as load testing focused on testing whether the system is able to handle concurrent user accesses without breaking and stress testing focused on how our system copes with high load and limited resources under constrained conditions.

We configure 10000 threads in 4 loops, making it 400000 threads or users access the localhost. Our application was able to handle the multiple access over 10000. Then we configure 1000 group threads in 10 second (hold target rate), making it concurrent threads or users access the localhost. Our application was also able to handle more than 1000 concurrent access.

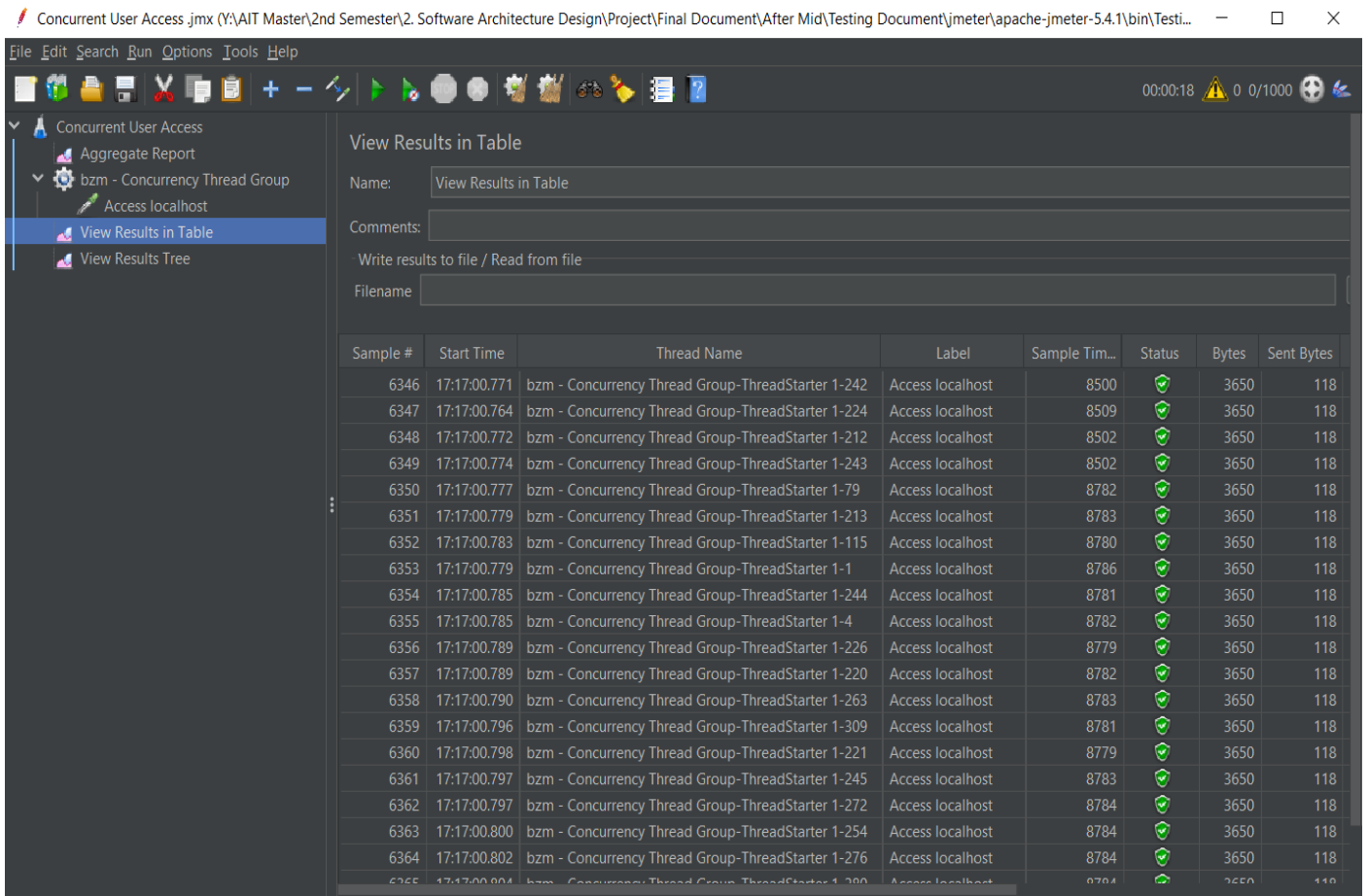
DISCUSSION

To check the architecture design; there is no better way to check than testing, rather than using theoretical claims. From aggregate report we can find that there is 15.56% error while the concurrent access with 1000 threads in 10 seconds (hold target rate) but when we have same 1000 threads in 10 minutes, it showed 0% of error.

The hold target rate — the duration which we want to run our tests for (how much time you want to keep adding new users each minute).

For more details on testing, check the link given below:

https://github.com/Younten-Tshering/Projects_and_related_works/blob/main/Bid-Buy-Sell-Project/Related%20Documents/Final%20Project%20Submission/Testing%20Documents/BSS%20Testing%20Document.pdf



View Results in Table

Name: View Results in Table

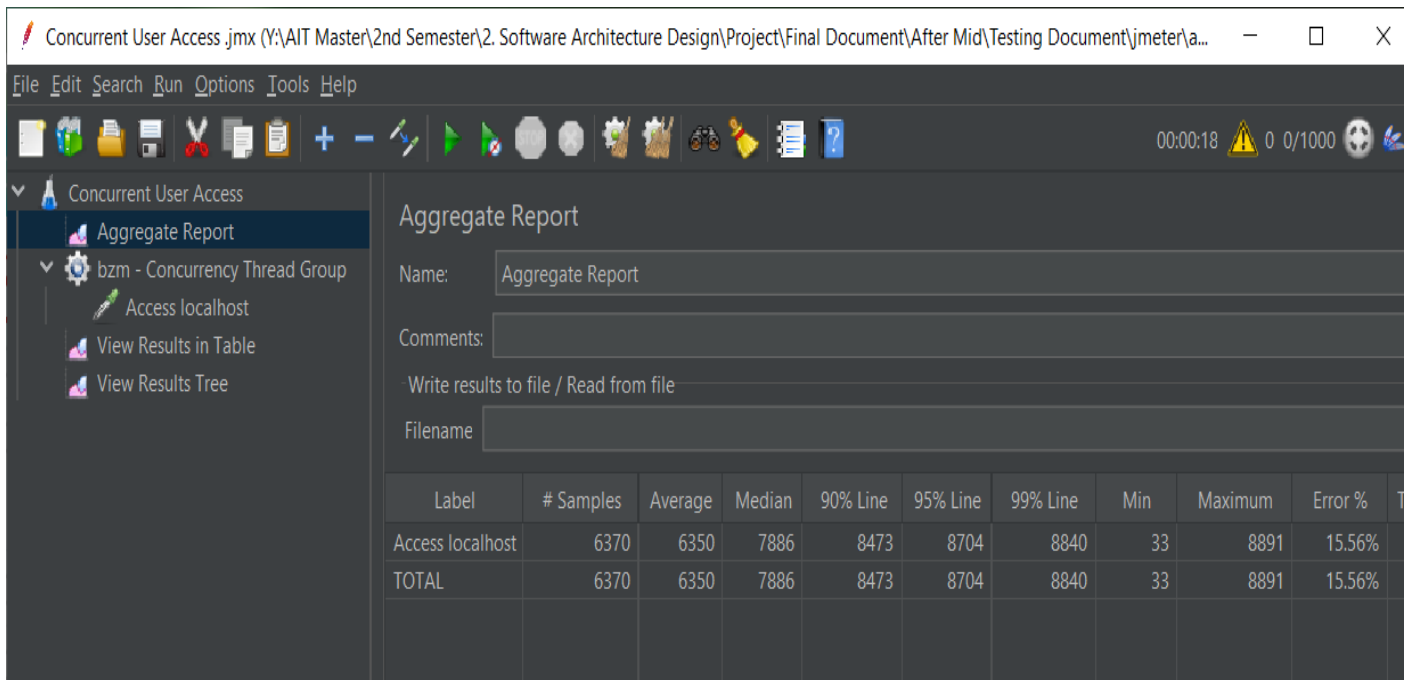
Comments:

Write results to file / Read from file

Filename:

Sample #	Start Time	Thread Name	Label	Sample Tim...	Status	Bytes	Sent Bytes
6346	17:17:00.771	bzm - Concurrency Thread Group-ThreadStarter 1-242	Access localhost	8500	✓	3650	118
6347	17:17:00.764	bzm - Concurrency Thread Group-ThreadStarter 1-224	Access localhost	8509	✓	3650	118
6348	17:17:00.772	bzm - Concurrency Thread Group-ThreadStarter 1-212	Access localhost	8502	✓	3650	118
6349	17:17:00.774	bzm - Concurrency Thread Group-ThreadStarter 1-243	Access localhost	8502	✓	3650	118
6350	17:17:00.777	bzm - Concurrency Thread Group-ThreadStarter 1-79	Access localhost	8782	✓	3650	118
6351	17:17:00.779	bzm - Concurrency Thread Group-ThreadStarter 1-213	Access localhost	8783	✓	3650	118
6352	17:17:00.783	bzm - Concurrency Thread Group-ThreadStarter 1-115	Access localhost	8780	✓	3650	118
6353	17:17:00.779	bzm - Concurrency Thread Group-ThreadStarter 1-1	Access localhost	8786	✓	3650	118
6354	17:17:00.785	bzm - Concurrency Thread Group-ThreadStarter 1-244	Access localhost	8781	✓	3650	118
6355	17:17:00.785	bzm - Concurrency Thread Group-ThreadStarter 1-4	Access localhost	8782	✓	3650	118
6356	17:17:00.789	bzm - Concurrency Thread Group-ThreadStarter 1-226	Access localhost	8779	✓	3650	118
6357	17:17:00.789	bzm - Concurrency Thread Group-ThreadStarter 1-220	Access localhost	8782	✓	3650	118
6358	17:17:00.790	bzm - Concurrency Thread Group-ThreadStarter 1-263	Access localhost	8783	✓	3650	118
6359	17:17:00.796	bzm - Concurrency Thread Group-ThreadStarter 1-309	Access localhost	8781	✓	3650	118
6360	17:17:00.798	bzm - Concurrency Thread Group-ThreadStarter 1-221	Access localhost	8779	✓	3650	118
6361	17:17:00.797	bzm - Concurrency Thread Group-ThreadStarter 1-245	Access localhost	8783	✓	3650	118
6362	17:17:00.797	bzm - Concurrency Thread Group-ThreadStarter 1-272	Access localhost	8784	✓	3650	118
6363	17:17:00.800	bzm - Concurrency Thread Group-ThreadStarter 1-254	Access localhost	8784	✓	3650	118
6364	17:17:00.802	bzm - Concurrency Thread Group-ThreadStarter 1-276	Access localhost	8784	✓	3650	118
6365	17:17:00.804	bzm - Concurrency Thread Group-ThreadStarter 1-280	Access localhost	8784	✓	3650	118

Image 1. Result of concurrent user access



Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename:

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %
Access localhost	6370	6350	7886	8473	8704	8840	33	8891	15.56%
TOTAL	6370	6350	7886	8473	8704	8840	33	8891	15.56%

Image 2. Aggregate report



For code and implementation, follow the link given below:

https://github.com/Younten-Tshering/Projects_and_related_works/tree/main/Bid-Buy-Sell-Project/Main_Project

CONCLUSION

The system 'Bid Buy-Sell' is especially designed to be used by students and employees (staff and faculties) of AIT to sell the products which can be new or used. The products added will be subjected to a certain time frame for buyers to bid on the products and are able to buy the product once the bid time closes or when there are no other bidders.

The Spring Boot based application was developed mainly to give reliable and handle multiple concurrent transactions, real-time information updates and handling online payment through secure payment gateways. With this application (BBS), concurrent users can access and manipulate the webapp with optimistic locking and multithreading concept.

In future, the application has a provision to integrate payment gateways and perform database performance testing. Therefore, bidding system was designed and implemented using the UML to illustrate the architectural model and this app will assist concurrent users to access and bid on products.

ACKNOWLEDGEMENT

The success and outcome of this work required immense guidance and assistance from many people, and it is always a pleasure to remind the fine people for their sincere guidance we received to uphold our work.

With much respect and reverence, we would like to pay our heartfelt gratitude to our Professor Dr. Chaklam Silpasuwanchai for his constant encouragement, support and guidance who helped us in successfully completing our work and to our friends who have made valuable comments and suggestion and inspired us to improve the quality.

Finally, we would also like to thank ourselves for manifesting excellent team spirit and cooperation.

REFERENCES

- [1] Majadi N, Trevathan J and Bergmann N. uAuction: Analysis, Design, and Implementation of a Secure Online Auction System. In Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C, pp. 278-285.
- [2] Slota,B. (2020, June 3). Concurrency control in REST API with spring framework. Dzone: Java Zone. Retrieved from <https://dzone.com/articles/concurrency-control-in-rest-api-with-spring-framew>.
- [3] Schworer,C. (2017, March 22). Managing concurrency in a distributed RESTful environment with Spring Boot and Angular 2. NOVATEC. Retrieved from <https://www.novatec-gmbh.de/en/blog/managing-concurrency-in-a-distributed-restful-environment-with-spring-boot-and-angular2/>.
- [4] "Editing the Web – Detecting the Lost Update Problem Using Unreserved Checkout". W3C Note. 10 May 1999.
- [5] Matt Welsh, Steven D. Gribble, Eric A. Brewer, and David Culler, "A Design Framework for Highly Concurrent Systems"
- [6] Dick J, Hull E and Jackson K. Requirements engineering. 2017.
- [7] [62] Bello S I, Bello R O, Babatunde A O, Olugbebi M and Bello B O. A University Examination Web Application Based on Linear-Sequential Life Cycle Model. 2017.
- [8] ALMRASHDEH I A, SAHARI N, ZIN N A M and ALSMADI M. DISTANCE LEARNING MANAGEMENT SYSTEM REQUIREMENTS FROM STUDENT'S PERSPECTIVE. Journal of Theoretical & Applied Information Technology, 2011, 24(1).
- [9] Almarashde I, Althunibat A and Fazidah El N. Developing a Mobile Portal Prototype for E-government Services. Journal of Applied Sciences, 2014, 14: 791-797.
- [10] Rajagopal D and Thilakavalli K. A Study: UML for OOA and OOD. International Journal of Knowledge Content Development & Technology, 2017, 7(2): 5-20



- [11] Karim S, Liawatimena S, Trisetyarso A, Abbas B S and Suparta W. Automating functional and structural software size measurement based on XML structure of UML sequence diagram. In Cybernetics and Computational Intelligence (CyberneticsCom), 2017 IEEE International Conference on, pp. 24-28.
- [12] R. Beulah and Dr. M. Soranamageswari, "Performance and Comparative Study of Functionality Testing Tools: Win Runner and QTP in IT World", International Journal of Advanced Research in Computer and Communication Engineering, ISSN (Online) 2278-1021 ISSN(Print) 2319 5940, Vol. 4, Issue 7, July 2015
- [13] Shagun Bhardwaj and Dr. Aman Kumar Sharma, "Performance Testing Tools: A Comparative Analysis", International Journal of Engineering Technology, Management and Applied Sciences, ISSN 2349-4476, Volume 3 Issue 4, April 2015.