IJCAT International Journal of Computing and Technology, Volume 1, Issue 3, April 2014
ISSN : 2348 - 6090
www.IJCAT.org

A Comparative Study of XML Parsers across Application

[1] Ms.V.M.Deshmukh [2] Dr. G.R. Bamnote [3] Ms.P.V.Kale

[1, 2, 3] Computer Science & Engineering Department, Prof. Ram Meghe Institute of Technology & Research, Badnera-Amravati.,MH.

Abstract - In today's world of high information sharing and transmission, XML plays a very important role as a universal format for data interchange, which allows users to transparently share XML documents. To achieve this transparency, XML documents need to be conformant with XML specifications. This specification conformance can be checked using an XML parser. The parser not only makes the data accessible, but it also ensures the validity of XML documents. Currently there are a lot of XML parsers, and most of them evolve, improve and become sophisticated. Though all the parsers serve the same purpose, they vary in terms of specification, performance, reliability and also conformance to standards. If a wrong choice has been made, it is highly possible to leads to the problem of excessive hardware requirement, which will resulted in productivity degradation**.** For XML document to be used, parsing is the most important process to be done. While parsing the document which parser is best suited and gives the more accurate result with respect to processing time , memory space that helps us to select the parser for our file to be parsed. Previously the comparison on various parsers implemented in java has been done, now in this dissertation we are going to analyze the comparison based on the parsers supporting .net as a platform.

Keywords - XML, XML Parser, Parsing

1. Introduction

XML is an Universal language (notation) for describing structured data. The data is stored together with the meta-information about it. Looks like HTML – text based, uses tags and attributes. It is used to describe other languages (formats) for data representation. Worldwide-affirmative standard, supported by the W3C (www.w3c.org). Platform, programming languages and OS independent. It is Simple, Extensible, Easy to process, Easy to generate and data interchange critical for networked applications ,"XML will be the ASCII of the Web: basic, essential, unexciting " by Tim Bray. So we can say that XML is a file extension for an Extensible Markup Language(XML) file format used to create common information formats and share both the format and data on World wide web ,intranet and anywhere using standard ASCII text. Parsing is the first step while doing the XML processing. Application developer must understand the operational and performance characteristics of XML processing. XML processing occurs in four stages: *parsing, access, modification,* and *serialization*.

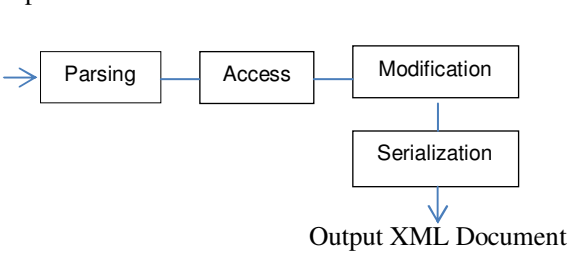


Fig 1 XML processing stages and parsing steps. processing

2. Literature Review

2.1 Comparison of Parsers for Different Languages

In 1999, Cooper [31] studied how parsing speeds vary with the programming language used for developing the parser. In this study, two java parsers, two C parsers, one perl and one python parser were used. Five XML documents with sizes ranging from 160 K to 5.0 MB were used in this study. This study concluded that C parsers are always faster compared to java, perl or python parsers.

2.2 Analysis of XML Parsers by Performing Tests

In 2002 , Srikanth Karre et. al explains [23] that the parsing of XML documents can be done using two approaches, *Event Based Parsing* and *Tree Based Parsing*. In Event Based Parsing, the XML data is parsed sequentially, one component at a time, and the parsing of events such as the start of a document, or the end of a document are reported directly to the application. SAX (Simple API for XML) is the standard API for event-driven parsing. In Tree Based Parsing, the XML document

is compiled into an internal tree structure and stored in main memory. Applications can then use this tree structure for navigation and data extraction. For example, the Document Object Model (DOM) uses tree based parsing, providing a standard set of objects for representing HTML and XML documents, a standard model of how these objects can be combined, and a standard interface for accessing manipulating them. Kai Ning et. al. proposed [6] a design scheme for DTD – based XML parser which has been implemented in the XML based network management R & D center.

Nicola. et. al. [10] explains- study of XML parsing performance across a selection of commercial database applications has shown that XML parsing is the major bottleneck [4]. They found that parsing even small XML documents can increase the computation cost of a database transaction by 2 to 3 times. Robert A et. al. describes [12] a validating XML parsing method based on deterministic finite state automata (DFA). XML parsing and validation is performed by a schema-specific XML parser that encodes the admissible parsing states as a DFA.

2.3 Parallel XML Parsing Algorithm

Wei Lu et. al. [8] states that there are a number of ways to improve XML parsing performance. One of the approach would be to use pipelining. In this approach, XML parsing could be divided into a number of stages. Each stage would be executed by a different thread. This approach may provide speedup, but software pipelining is often hard to implement well, due to synchronization, load-balance and memory access costs.

2.6 VTD-XML-based Design and Implementation of GML Parsing Project

Lan et. al. [16] explains VTD-XML is a new open-source XML parsing model. It centers on a non-extractive XML parsing technique called Virtual Token Descriptor (VTD). It features random access capability, high performance, and low-memory consumption.

Su Cheng Haw and G. S. V. Radha Krishna Rao have presented a model called "Comparative Study and Benchmarking on XML Parser". In that, they compare the xerces and .NET parsers based on the performance, memory, usage and so on[2].

Bruno Oliveira1,Vasco Santos1 and Orlando Belo2 - explains in order to provide a benchmark of each one of APIs tested the used set of XML example files, which represents typical real-world applications. These files have several sizes categorized as *Small* (between 1,6 ~ 6,8 KB), *Medium* (10 ~ 1 MB) and *Big* (between 1 ~ 15MB). Tests were conducted with files in memory with the purpose of reducing I/O costs. XML parsing performance was conducted for testing latency, memory usage and navigation performance [4].The number of such techniques are available for doing the performance analysis on the individual basis in this we are comparing all those on a single XML document getting the overall review for time and space complexity.

V.M. Deshmukh, G.R. Bamnote- Campare the performance of different parsing like DOM, SAX for different data structures like linked list,stack, array and queue. Using the different data structure for accessing or reading the data from the database required less time as compared to others method.

3. Proposed Methodology

In this dissertation the XML document is parsed using different parsers available for ex. DOM(Document Object Model) , SAX (Simple API for XML), VTD (Virtual Token Descriptor), Xerces, Pull parser and so on. After parsing the performance of the XML parser is evaluated with respect to time and space in the form of graph showing which one is the best parser for parsing the given XML document. Java API's are being used for calculating the performance of the parsers. Thus a high-performance XML parsing and validation technique that is time and space optimal has been found out. This is to analyze which particular parser is suitable for certain applications.

A XML SAX parser has been implemented efficiently. The model also provides a novel mechanism to trace the stealing actions, and the equivalent sequential results are obtained by gluing the multiple parallel running results together. The basic idea of stealing based scheme is that every thread works on its own local task queue and whenever it runs out of the task it steals the task from other thread's task queue.

An XML parser can be built by extracting tokens (e.g. start and end tags) from a document by reading it from the beginning. With the help of data structure, we can parse it. For example, A DOM tree can be built by extracting tokens (e.g. start and end tags) from a document by reading it from the beginning. A stack (S) is maintained and is initially empty. This stack essentially stores the information of all the ancestors (in the DOM tree) of the current element being processed in the document. When a start element tag say <e> is read, a DOM node (de) is created for element (e) and any (attribute, value) pair that is associated with the element is parsed and stored, by creating the necessary DOM nodes. If S is not empty, then this implies that (de)'s parent node has already been created. If (e) encloses text,then a DOM node for the text

is also created and linked as a "text" child of de. When an end element tag say </e> is read, e is checked with the top of stack S. If the element names do not match, then the parsing is aborted as the document is not well formed. Otherwise, the top of S is popped and the parsing continues. After the last character of the document is processed, if S is empty, then the entire DOM tree has been constructed. Otherwise, the document is not well-formed.

4. Performance Analysis

Currently industries use XML files for permanently storing large amounts of data and XML files have not just remained a format for data interchange . And wherever these files are used, data from these files needs to be extracted and manipulated often. An XML parser is used to read the XML document. The document is divided into different parts such as element, attributes, etc., by these parsers. Then this information is passed to the application which needs it. In some cases, if the document is not well formed, the parser will send an error and stop parsing. In other cases, it will parse the whole document and send all the errors together. But if the error is encountered, it will only give information about the error and not the contents of the XML document.XML parsing can seriously affect overall performance for any project it is used in. Hence it is very important to study different APIs which have the capability of XML parsing and to compare the performance of the APIs to choose the best method for a given condition.

Considering the importance of parsers in this system the comparison is done on various XML files of varying file sizes as small ranges below 1 KB, medium range above 1 KB and Large range above 6 KB .

1. The input to the system is an .xml file which is valid.
2. This has been checked by one of the class of .net i.e. XMLTextReader class.
3. Once the file is supposed to be valid we can parser the file.
4. The tree view can be seen for each file.
5. Internal Processing can be checked for each and every file.
6. The graphical representation has been seen with the file size on x axis and different parameters as ParseTime ,ParseAttributes ,PageMemoryUsed on Y axix.
7. The size of XML documents to be considered

When parsing any XML document, the size of file can matter a lot.

For the event type are SAX, StAX, the size of XML document usually doesn't matter, since the parsing is done by streaming the XML data to the parser.On the other hand, for tree based, the tree representation of the XML document is created in the memory. Hence the bigger the XML file, the bigger the tree document that is formed. This can become a problem when the files are really very large. For these reasons, the size of an XML file plays an important role in choosing any kind of Parser.

8. A file of size 5 KB can be considered a small file, while a file of size 1 MB and 10 MB can contain a huge amount of data. Therefore for the experiments we will be using 3 files with different sizes 5KB, 1MB and 10 MB. This will give a good range for comparing the processing time results.

9. The output is in the form of a chart plotted across every file that has been parsed.

10. The graph is plotted for every file size small , medium and large for which the entry has been done in the database.
11. The entry has been done for the parser with its type i.e tree based , event based , simple , VTD etc
    - Tree based
      These APIs create a tree representation of the XML document in the memory. Data can be accessed by navigating through this tree. Document Object Model (DOM) is a tree based API. There are many other tree based APIs available.
    - Event based
      These APIs do not build a tree representation. They report different parsing events like start element, end element using the call back method. Handlers are designed to deal with the events. Simple API for XML (SAX) is the most common type of event-based.

5 Result Analysis

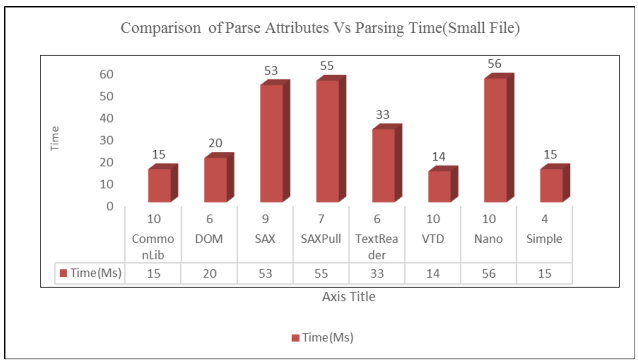
Result analysis is the important part of this dissertation as the project focused mainly on the comparison betweenthe available parsers. In this section we performthe graphical analysis by considering different factors as below. The below graphs are plotted for doing a comparison between different parameters such as Parsing Attributes vs Memory Space vs Parsing Time. The comparison has been done for the number of parsers available supporting to the .net platform. Such kind of comparison gives us the option to choose the best suited parser for our application.

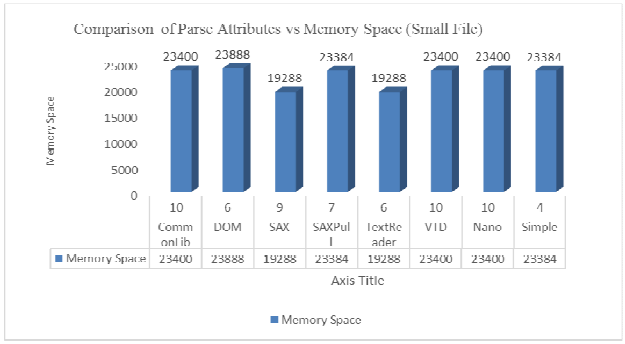
5.3.1 Graph for Small File Size

Assumption : Small file size below 1KB bytes

Parser	Parsing Attribute	Time	Memory Space
CommonLib	10	15	23400
DOM	6	20	23888
SAX	9	53	19288
SAXPull	7	55	23384
TextReader	6	53	19288
VTD	10	14	23400
Nano	10	56	23400
Simple	4	15	23384

Table for Small file Comparison





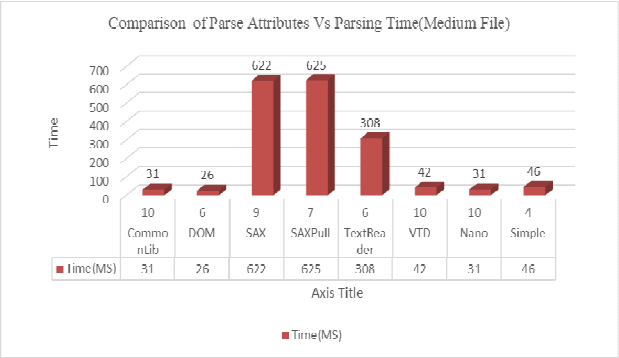
In the above graph we can see that the SAX parser and TextReader requires the same memory space for the small file size i.e. of 382 bytes. While considering for Time(ms) it was less for TextReader as compared to SAX. suited to parser the smaller file size. But memory space requirement is more.

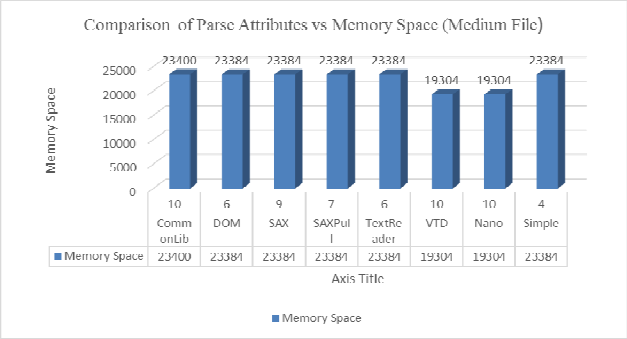
5.3.2 Graph for Medium File Size

Assumption : Medium file size above 1 KB bytes

Parser	Parsing Attributes	Time(ms)	Memory Space
CommonLib	10	31	23400
DOM	6	26	23384
SAX	9	622	23384
SAXPull	7	625	23384
TextReader	6	308	23384
VTD	10	42	19304
Nano	10	31	19304
Simple	4	26	23384

Table for medium file Comparison





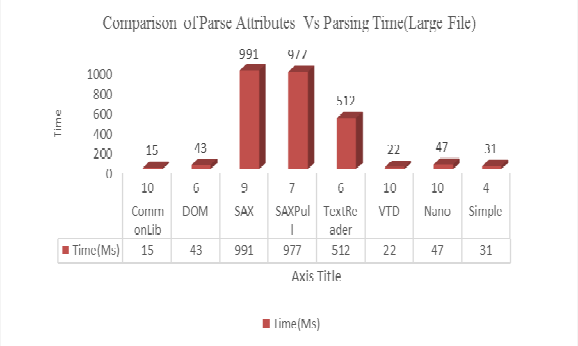
The above graph shows that Nano XML parser and VTD takes the same memory space and same parsing attributes but if we consider the time nano XML parser will take a less time than VTD. This comparison has been done on the medium file size .From the above comparison for medium file size we can conclude that nano XML will be the best parser to parse the file.

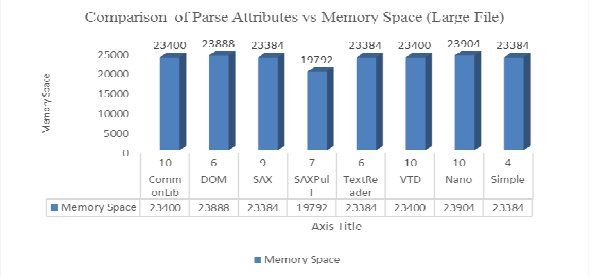
5.3.3 Graph for Large File Size

Assumption : Large file size above 6 KB bytes

Parser	Parsing Attributes	Time (ms)	Memory Space
CommonLib	10	15	23400
DOM	6	43	23888
SAX	9	991	23384
SAXPull	7	977	19792
TextReader	6	512	23384
VTD	10	22	23400
Nano	10	47	23904
Simple	4	31	23384

Table for Large file Comparison





The above graph shows that for the larger file size the SAX(Pull) parser takes the minimum memory allocation but the time required is more. The time taken for parsing for the simple parser is minimum but memory allocation was more.

6. Conclusion & Future Work

There have been a handful of studies and researches towards XML parsers. Nevertheless, most of them are not up to date. As XML parser is a technology, which is changing rapidly for the moment, there is no single study or research that would valid forever. The result of the study indicates that SAX has been the best parser in terms of performance. However, xParse outperformed in terms of supporting large-scale of dataset efficiently. Nevertheless, performance is not the only criteria;there are lots of factors to be considered when choosing XML parser, such as organization's need, API support, platforms and license fees.

Since the testing and benchmarking only involve in evaluating two most popular parsers, the .NET and Xerces parser, the study can be further extended in future. Some of the future approaches could includes 1): Compare the performance of new and established APIs DOM, SAX, StAX,Pull or electric XML together in a set of benchmarking tool,and 2): Compare and study the conformance of parsers to some of the new features such as support to new APIs and eXtensible Stylesheet Language Transformation (XSLT) Ability.

References

[1] Zisman, A., "An Overview of XML", Computing & Control Engineering Journal, 2000.

[2] Slominski, A., "Design of a Pull and Push Parser System for Streaming XML", Indiana University, Technical Report TR550, 2001.

[3] Srikanth Karre and Sebastian Elbaum," *An Empirical Assessment of XML Parsers*",2002.

[4] Kai Ning, Luoming Meng,"*Design and Implementation of DTD-based XML parser*", proceedings of ICCT2003.

[5] Chengkai Li,"XML Parsing, SAX/DOM",

[6] Nicola, M. and John, J., "*XML Parsing: a Threat to Database Performance*" International Conference on Information and Knowledge Management, 2003, pp. 175-178.

[7] Robert A. van Engelen, "*Constructing Finite State Automata for High-Performance XML Web Services*", in the proceedings of International Symposium on Web Services and Applications (ISWS) 2004.

[8] Wei Lu, Kenneth Chiu, "A Parallel Approach to XML Parsing", in the 7th International Conference on Grid Computing, IEEE/ACM 2006.

[9] Tong, T. et al, "*Rules about XML in XML*", Expert Systems with Applications, Vol. 30, No.2, 2006, pp. 397-411.

[10] Su Cheng Haw ,G. S. V. Radha Krishna Rao," *A Comparative Study and Benchmarking on XML Parsers*", Advanced Communication Technology, The 9th International Conference (Volume:1) ISSN :1738-9445 , 2-14 Feb. 2007 pp. 321 – 32.

[11] Su Cheng Haw ,G. S. V. Radha Krishna Rao," A Comparative Study and Benchmarking on XML Parsers", Advanced Communication Technology, The 9th International Conference _(Volume:1) ISSN :1738-9445 , 2-14 Feb. 2007 pp. 321 – 32

[12] Wei Lu, Dennis Gannon, "Parallel XML Processing by Work Stealing", SOCP'07, June 26, 2007, Monterey, California, USA.

[13] Wei Lu, Dennis Gannon, "*Parallel XML Processing by Work Stealing",* SOCP'07, June 26, 2007, Monterey, California, USA.

[14] Yinfei Pan, Wei Lu, Ying Zhang, Kenneth Chiu,"*A Static Load-Balancing Scheme for Parallel XML*

Parsing on Multicore CPU"s, Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07) 0-7695-2833-3/07 $20.00 © 2007

[15] Wei Zhang, Robert A. van Engelen, "An Adaptive XML Parser for Developing High-Performance Web Services", Fourth IEEE International Conference on eScience, 2008, pp 672-679

[16] Tak Cheung Lam and Jianxun Jason Ding Cisco Systems Jyh-Charn Liu Texas A&M University , "XML Document Parsing: Operational and Performance Characteristics ", published by IEEE computer society , October 7 , 2008

[17] Yusof Mohd Kamir, Mat Amin Mat Atar, "*High Performance of DOM Technique in XML for Data Retrieval"*, 2009 International Conference on Information and Multimedia Technology.

[18] Lan Xiaoji Su Jianqiang Cai Jinbao, "*VTD-XML-based Design and Implementation of GML Parsing Proj*ect", IEEE Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on 19 dec 2009 , pp.1 – 5.

[19] Xiaosong Li, Hao Wang, Taoying Liu, Wei Li*," Key Elements Tracing Method for Parallel XML Parsing in Multi-core System",* 2009 International Conference on Parallel and Distributed Computing, Applications and Technologies, 978-0-7695-3914-0/09 $26.00 © 2009 IEEE DOI 10.1109/PDCAT.2009.64

[20] M. Van Cappellen, Z. H. Lui, J. Melton, and Maxim Orgiyan, "*XQJ - XQuery Java API is Completed*", SIMOD Record, vol. 38, no. 4, 2009.

[21] Shu Yuan-zhong,"Research of optimizing device description technology based on XML in EPA" 2009 Second International Symposium on Electronic Commerce and Security.

[22] Gong Li and Liu Gao-Feng, Liu Zhong and An Ru-Kui, "*XML Processing by Tree-Branch symbiosis algorithm*", 2010 2nd International Conference on Future Computer and Communication, Volume 1.

[23] Rami Alnaqeib, Fahad H.Alshammari, M.A.Zaidan," An Overview: Extensible Markup Language Technology", Journal of computing, Volume 2, Issue 6, June 2010, ISSN 2151-9617,pp -177-181

[24] V.M. Deshmukh, G.R. Bamnote, "*DESIGN AND DEVELOPMENT OF AN EFFICIENT XML PARSING ALGORITHM: A REVIEW"*, International Journal of Applied Science and Advance Technologyz, January-June 2012, Vol. 1, No. 1, pp. 5-8.

[25] Cheng-Han You and Sheng-De Wang," A Data Parallel Approach to XML Parsing and Query" 2011 IEEE International Conference on High Performance Computing and Communications pp-520-527

[26] Girish Tere Bharat Jadhav," Efficient Processing of XML Documents" International Conference on Technology Systems and Management (ICTSM) 2011 Proceedings published by International Journal of Computer Applications® (IJCA*).*

[27] Ms. V.M.Deshmukh, Dr. G.R.Bamnote, "*An Empirical Study: XML Parsing using Various Data Structures*", International Journal of Computer Science and Applications, Vol. 6, No.2, Apr 2013.

[28] Jie Tang, Shaoshan Liu, Chen Liu, Zhimin Gu, and Jean-Luc Gaudiot," *Acceleration of XML Parsing through Prefetching*", IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 8, AUGUST 2013

[29] Mohammad Khabbaz**,** Dirar Assi**,**Reda Alhaj, Moustafa Hammad," *Parse Tree Based Approach for Processing XML Streams*", IEEE IRI 2013, August 14-16, 2013, San Francisco, California, USA 978-1-4799-1050-2/13/$31.00 ©2013 IEEE

[30] Bruno Oliveira1,Vasco Santos1 and Orlando Belo2," *Processing XML with Java – A Performance Benchmark*", International Journal of New Computer Architectures and their Applications (IJNCAA) 3(1): 72-85 The Society of Digital Information and Wireless Communications (SDIWC) 2013 (ISSN: 2220-9085) ,pp. 72-85.

[31] Michael R. Head† Madhusudhan Govindaraju, "*Parallel Processing of Large-Scale XML-Based Application Documents on Multi-core Architectures with PiXiMaL*", Fourth IEEE International Conference on eScience.

[32] Wei Lu , Kenneth Chiu, Yinfei Pan, "*A Parallel Approach to XML Parsing*".

[33] Li Zhao , Laxmi Bhuyan , "Performance Evaluation and Acceleration for XML Data Parsing,"http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.5330 &rep=rep1&type=pdf

[34] W3C, "Extensible Markup Language (XML)". [Online].Available:http://www.w3.org/XML.

[35] Xml Pull Parser, http://www.extreme.indiana.edu/xgws/xsoap/xpp/

[36] libxml2, http://www.xmlsoft.org/

[37] Xml Pull Parser, http://www.extreme.indiana.edu/xgws/xsoap/xpp/

Author Profile

First Author: Prof. Ms.V.M.Deshmukh ,Head of Information Technology ,P.R.M.I.T & R,Badnera
Second Author Dr.G.R.Bamnote Head of Computer Science & Engg ,P.R.M.I.T & R,Badnera
Third Author Ms. Prachi V.Kale ,Student M.E Computer Science & Engg ,P.R.M.I.T & R,Badnera