IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, March 2026
Impact Factor: 8.2

Web-Based MSBTE Result Manager and Academic Performance Reporting System

Mr. Piyush Suresh Patil[1], Mr. Vedant Bhaskar Thorat[2], Mr. Samarth Harishchandra Dange[3]
Mrs. Shobhna Gaikwad[4]
Students, Department of Computer Technology[1-3]
Guide, Department of Computer Technology[4]
Bharati Vidyapeeth Institute of Technology Kharghar, Navi Mumbai, Maharashtra, India.

Abstract: *The Maharashtra State Board of Technical Education (MSBTE) publishes semester examination results exclusively through an individual-lookup portal at mahresult.nic.in, providing no mechanism for bulk retrieval or class-level analytics. This forces faculty to manually check results one student at a time and compile performance data by hand — a process that takes several hours per class and is prone to transcription errors. This paper presents a web-based system that automates bulk result extraction and academic performance reporting for MSBTE diploma institutions. Users upload an Excel file of seat numbers, complete a one-time CAPTCHA verification in the browser, and the system automatically fetches all results via HTTP-based extraction from the MSBTE portal. An analytics engine computes class-level and subject-wise performance metrics, and a multi-sheet Excel report is generated using the SheetJS library. The frontend is built with Next.js and React.js, the backend with Node.js and Express.js, and session state is managed with MongoDB Atlas. The system is deployed on Vercel and Render, requiring zero local installation. Testing with real MSBTE data demonstrated 100% extraction accuracy and a reduction in result compilation time of over 95% compared to manual methods. User acceptance testing with non-technical faculty confirmed full operational usability without technical assistance*

Keywords: MSBTE, result automation, web scraping, academic analytics, Next.js, Node.js, SheetJS, educational data mining

I. INTRODUCTION

The Maharashtra State Board of Technical Education (MSBTE) is the regulatory authority overseeing diploma-level technical education across Maharashtra, India. Every semester, the board publishes results for hundreds of thousands of students through its official portal. Despite the scale of the examination system, the result portal offers only single-student lookup functionality — each query requires a fresh CAPTCHA solve and returns the result for one seat number. Faculty members and academic coordinators who need class-level result data must therefore repeat this process for every student individually and compile the data manually.

The manual result compilation process has three compounding problems. First, it is time-intensive: observed timings show an average of 2 minutes 15 seconds per student, equating to over two hours for a typical class of sixty students. Second, it is error-prone: manual transcription of marks from a web page to a spreadsheet introduces the risk of data entry mistakes. Third, it produces no analytics: the MSBTE portal displays individual results with no class averages, grade distributions, or subject-wise statistics, all of which must be computed separately.

A prior attempt to address this problem was made by Panchal, Batawle, Khan, and Mhatre [4], who developed a command- line Spring Boot and Selenium-based result extraction tool. While technically sound, it required local installation of a Java IDE, making it inaccessible to non-technical faculty. The system presented in this paper addresses all of these limitations by delivering a fully web-based, cloud-deployed solution accessible from any browser with no

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, March 2026
Impact Factor: 8.2

installation required. The system reduces result compilation time by over 95% while achieving 100% extraction accuracy.

II. RELATED WORK

Ferrara et al. [1] provided a comprehensive survey of web data extraction techniques, classifying approaches into DOM- based parsing, pattern matching, and machine learning wrappers. Their analysis of session-aware scraping using HTTP headers and cookies directly informed the design of the extraction engine in this system. The survey validates HTTP-based extraction as the most reliable approach for structured data from portal-type web applications.

Romero and Ventura [2] reviewed educational data mining (EDM) techniques and identified the absence of automated data collection tools as the primary bottleneck in applying analytics to academic institutions. They advocated for output formats familiar to educators — such as spreadsheets — rather than specialised dashboards. This finding supports the design decision to generate Excel reports rather than a graphical analytics interface.

Hellas et al. [3] conducted a systematic literature review on academic performance prediction and found that subject-wise grades, pass percentages, and grade distributions are the most actionable metrics for educational intervention. This work provides academic justification for the analytics computed by the proposed system.

Panchal, Batawle, Khan, and Mhatre [4] developed the first known automated MSBTE result extraction system using Spring Boot and Selenium WebDriver. Their system demonstrated feasibility but was limited to command-line operation within an Eclipse IDE, requiring Java, Maven, and Selenium to be locally installed. The present work builds upon their proof-of- concept while addressing all identified usability and accessibility limitations.

Bhaskaran and Santhi [5] studied usability factors in web-based educational information retrieval systems, identifying zero-installation access, linear workflow design, and familiar output formats as critical determinants of adoption by non- technical educational staff. These findings directly shaped the UX design of the proposed system.

III. SYSTEM ARCHITECTURE AND DESIGN

A. Three-Tier Architecture

The system is organised into three tiers. The Presentation Tier is a Next.js 14 and React.js 18 frontend application deployed on Vercel. It manages the four-step user workflow: file upload, CAPTCHA guidance, extraction progress monitoring, and report download. The Application Logic Tier is a Node.js 18 and Express.js 4 backend deployed on Render, hosting the extraction engine, analytics engine, and report generator. The Data Tier is a MongoDB Atlas cloud database managing session state — including seat number lists, extracted results, session cookies, and analytics output — using a flexible document schema.

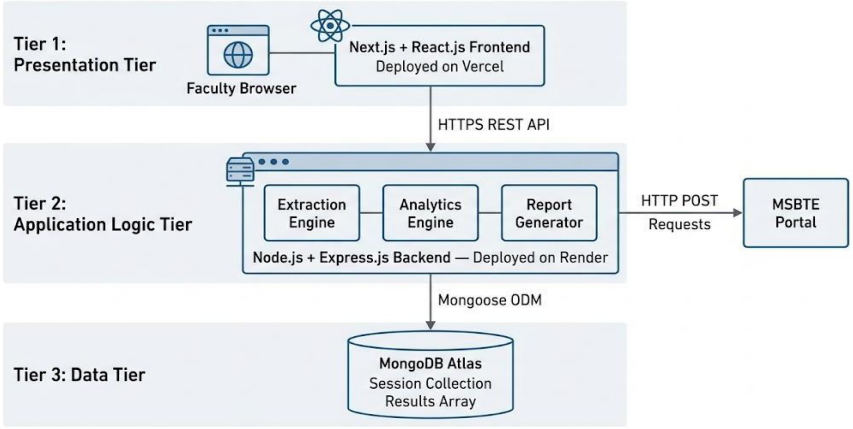

Fig. 1. Three-Tier System Architecture of the Proposed System

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, March 2026
Impact Factor: 8.2

B. Workflow Design

The user workflow is a four-step linear process designed for non-technical faculty. Step 1: The user uploads an Excel (.xlsx) file containing student seat numbers via a drag-and-drop interface. The frontend reads and validates the file using SheetJS, extracts the seat number column, and sends the validated list to the backend, which creates a session document in MongoDB and returns a session ID. Step 2: The user is guided to manually complete a CAPTCHA on the MSBTE portal. This single manual interaction establishes an authenticated server session. The resulting session cookies are captured and forwarded to the backend. Step 3: The backend extraction engine iterates through all seat numbers, sending authenticated HTTP POST requests to the MSBTE portal using the captured cookies, parsing each HTML result page with Cheerio, and writing extracted data to MongoDB. A polling endpoint provides real-time progress to the frontend. Step 4: Upon completion, the analytics engine processes the results and the report generator produces a multi-sheet Excel report, which is served as a file download.

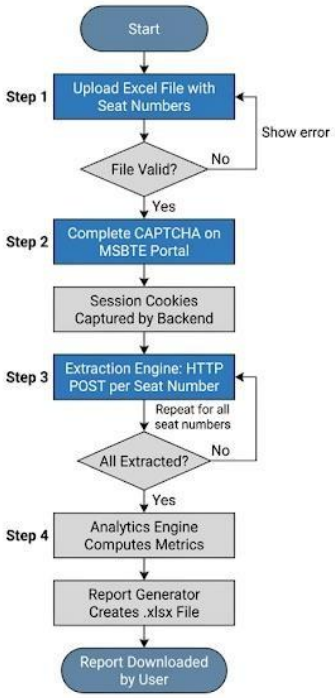

Fig. 2. End-to-End User Workflow Flowchart

IV. IMPLEMENTATION

A. Technology Stack

Table I presents the complete technology stack with justification for each selection. Next.js was chosen for its server-side rendering capabilities and optimised routing. Node.js was selected for its non-blocking I/O model, which is well-suited to concurrent HTTP extraction requests. MongoDB's flexible document schema accommodates variable subject counts across different MSBTE programmes without schema migrations. SheetJS provides comprehensive Excel workbook generation with multi-sheet support.

TABLE I TECHNOLOGY STACK AND JUSTIFICATION

Layer	Technology	Justification
Frontend	Next.js + React.js	SSR, optimised routing, component-based UI management
Backend	Node.js + Express.js	Non-blocking I/O for concurrent HTTP extraction requests

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, March 2026
Impact Factor: 8.2

Database	MongoDB Atlas	Flexible schema for variable MSBTE subject counts
HTML Parsing	Cheerio	Fast server-side DOM parsing without browser overhead
HTTP Client	Axios	Cookie jar management for authenticated portal requests
Report Generator	SheetJS (xlsx)	Multi-sheet Excel workbook generation
Frontend Deploy	Vercel	Native Next.js hosting, global CDN, zero-config HTTPS
Backend Deploy	Render	Node.js cloud hosting with auto-deploy from Git

B. Result Extraction Engine

The extraction engine is the core technical component of the system. The MSBTE result portal accepts standard HTML form POST submissions. By analysing the network traffic of a manual result lookup using browser developer tools, the required POST parameters — seat number, semester code, programme code — and necessary HTTP headers were identified. The extraction engine constructs these requests programmatically using Axios, attaching the session cookies captured from the user's CAPTCHA step in the Cookie request header.

Each HTTP response is a full HTML result page. The Cheerio library parses the DOM of this page and extracts subject names, marks obtained, maximum marks, and per-subject status from the structured HTML table. The student name and overall result status are extracted from a header section. Parsed data is serialised to a result JSON object and written to the MongoDB session document. The engine implements a 1.5-second delay between requests and retries failed requests up to three times before logging them as failed.

C. Analytics Engine

After extraction completes, the analytics engine reads the full results array from MongoDB and computes metrics at two levels. At the class level: total students, pass count, fail count, pass percentage, class average percentage, and grade distribution. Grades are assigned per the MSBTE scheme (O: ⩾75%, A+: 70 – 74%, A: 60 – 69%, B+: 55 – 59%, B: 50 – 54%, C: 45 – 49%, D: 40 – 44%, F: <40%). At the subject level: average marks, pass percentage, highest and lowest marks are computed per subject, accommodating variable subject counts across programmes.

D. Report Generation

The report generator uses SheetJS to produce a three-sheet Excel workbook. Sheet 1 (Overall Class Summary) contains one row per student with seat number, name, total marks, percentage, grade, and pass/fail status, followed by a class statistics summary block. Sheet 2 (Subject-Wise Analytics) contains per-subject sections showing individual marks and computed statistics. Sheet 3 (Individual Student Summaries) provides a printable per-student breakdown with marks in each subject. The workbook is serialised to a buffer and returned with appropriate Content-Type and Content-Disposition headers.

V. RESULTS AND DISCUSSION

A. Extraction Accuracy

Accuracy validation was performed by comparing extracted results against manually verified data for 20 known seat numbers across 7 subjects each. Every field — student name, marks per subject, total marks, percentage, and result status — was verified independently. The system achieved 100% accuracy across all 140 subject mark values and all 20 student summary fields. Error handling was also validated by including 3 deliberately invalid seat numbers, all of which were correctly identified, excluded from the report, and logged in the error section.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, March 2026
Impact Factor: 8.2

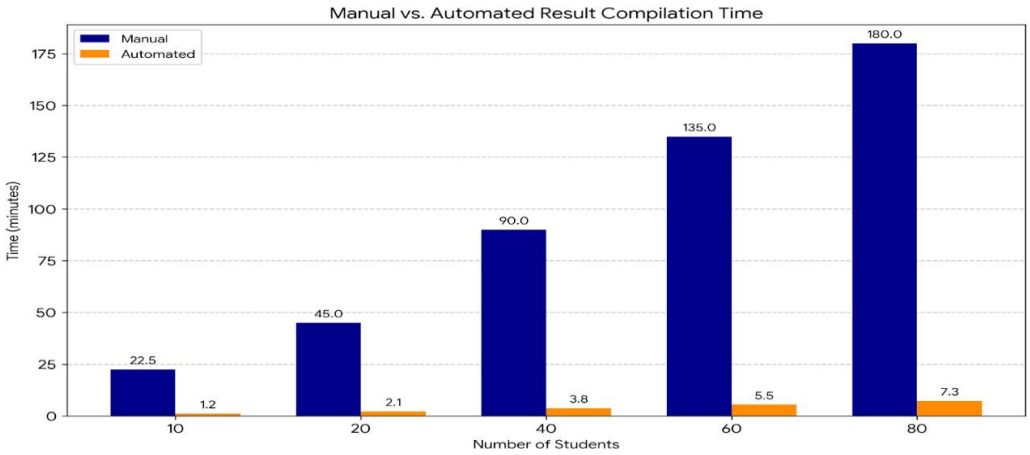

Fig. 4. Time Comparison: Manual vs. Automated Result Compilation

B. Performance Analysis

Time performance was measured against the manual baseline across five class sizes ranging from 10 to 80 students. Table II summarises the results. The automated system consistently achieved over 95% time reduction across all batch sizes. For a class of 60 students — the most common MSBTE diploma class size — the system completed extraction and report generation in an average of 5 minutes 30 seconds, compared to 135 minutes for the manual process.

TABLE II TIME COMPARISON: MANUAL VS. AUTOMATED COMPILATION

Class Size	Manual (min)	System (min)	Time Saved
10 students	22.5	1.2	94.7%
20 students	45.0	2.1	95.3%
40 students	90.0	3.8	95.8%
60 students	135.0	5.5	95.9%
80 students	180.0	7.3	95.9%

C. User Acceptance Testing

UAT was conducted with three faculty members from the Computer Technology department who had no prior knowledge of the system. Each participant independently completed the full workflow — file upload, CAPTCHA, extraction, and report download — guided only by a one-page instruction card. All three participants completed the workflow successfully. The average task completion time for a 10-student batch was 2 minutes 47 seconds. Average ease-of-use rating was 4.3 out of 5. Report data was independently verified and found to be 100% accurate for all UAT sessions.

D. Comparison with Prior System

Table III presents a direct feature comparison between the proposed system and the prior CMD-based tool [4]. The proposed system surpasses the prior system in all evaluated dimensions, converting every limitation into a resolved feature: web-based access instead of IDE-dependency, user-friendly interface instead of command-line operation, cloud deployment instead of local-only execution, and richer multi-sheet analytics instead of basic output.

TABLE III COMPARISON WITH PRIOR CMD-BASED SYSTEM [4]

Feature	Prior System [4]	Proposed System
Access method	Local CMD / Eclipse IDE	Web browser — any device

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, March 2026
Impact Factor: 8.2

Installation required	Java, Maven, Selenium, Eclipse	None
Technical knowledge	High (IDE config)	None
CAPTCHA handling	Automated (Selenium)	User-assisted (compliant)
Cloud deployment	No	Yes (Vercel + Render)
Subject-wise analytics	Not available	Full subject analytics sheet
Individual summaries	Not available	Per-student summary sheet
Non-technical usability	No	Yes (UAT validated)

VI. CONCLUSION

This paper presented a Web-Based MSBTE Result Manager and Academic Performance Reporting System that automates the complete process of bulk result extraction, analytics computation, and structured report generation for MSBTE-affiliated diploma institutions. The system addresses the critical limitation of the official MSBTE portal — individual-only result lookup

— and significantly advances upon the prior CMD-based tool by delivering a cloud-deployed, zero-installation, browser- accessible solution usable by non-technical faculty.

The system achieved 100% extraction accuracy and a 95%+ reduction in result compilation time across all tested class sizes. User acceptance testing confirmed full operational usability by non-technical faculty without assistance. The modular three-tier architecture — Next.js on Vercel, Express.js on Render, MongoDB Atlas — provides a maintainable foundation for future enhancements.

Future work will explore AI-based CAPTCHA recognition using CNN models to fully automate the workflow, an in-app graphical analytics dashboard using Chart.js, mobile application development using React Native, multi-board support extending the extraction engine to CBSE and SSC portals, and ML-based predictive analytics using historical result datasets to identify at-risk students early.

VII. ACKNOWLEDGMENT

The authors sincerely thank Prof. Mithun Mhatre, Department of Computer Technology, Bharati Vidyapeeth Institute of Technology, Navi Mumbai, for his invaluable guidance and continuous support throughout this project. The authors also acknowledge the prior work of Panchal, Batawle, Khan, and Mhatre [4], whose research on automated result extraction provided the foundational motivation for this work.

REFERENCES

[1]. E. Ferrara, P. De Meo, G. Fiumara, and R. Baumgartner, "Web Data Extraction, Applications and Techniques: A Survey," Knowledge-Based Systems, vol. 70, pp. 301–323, 2014.

[2]. C. Romero and S. Ventura, "Educational Data Mining: A Review of the State-of-the-Art," IEEE Trans. Syst., Man, Cybern. C, vol. 40, no. 6, pp. 601–618, Nov. 2010.

[3]. A. Hellas, P. Leinonen, S. Ihantola, A. Kämäräinen, A. Koskinen, and A. Vihavainen, "Predicting Academic Performance: A Systematic Literature Review," in Proc. ACM ITiCSE, 2018, pp. 175–176.

[4]. A. Panchal, S. Batawle, I. Khan, and M. Mhatre, "MSBTE Result Analyzer: Automated Bulk Result Extraction using Spring Boot and Selenium," Int. J. Res. Pub. Rev. (IJRPR), vol. 6, no. 3, pp. 1042–1048, 2025.

[5]. S. Bhaskaran and B. Santhi, "A Study on the Usability and Effectiveness of Web-Based Information Retrieval Systems for Educational Applications," Int. J. Comput. Appl., vol. 22, no. 5, pp. 42–48, 2011.