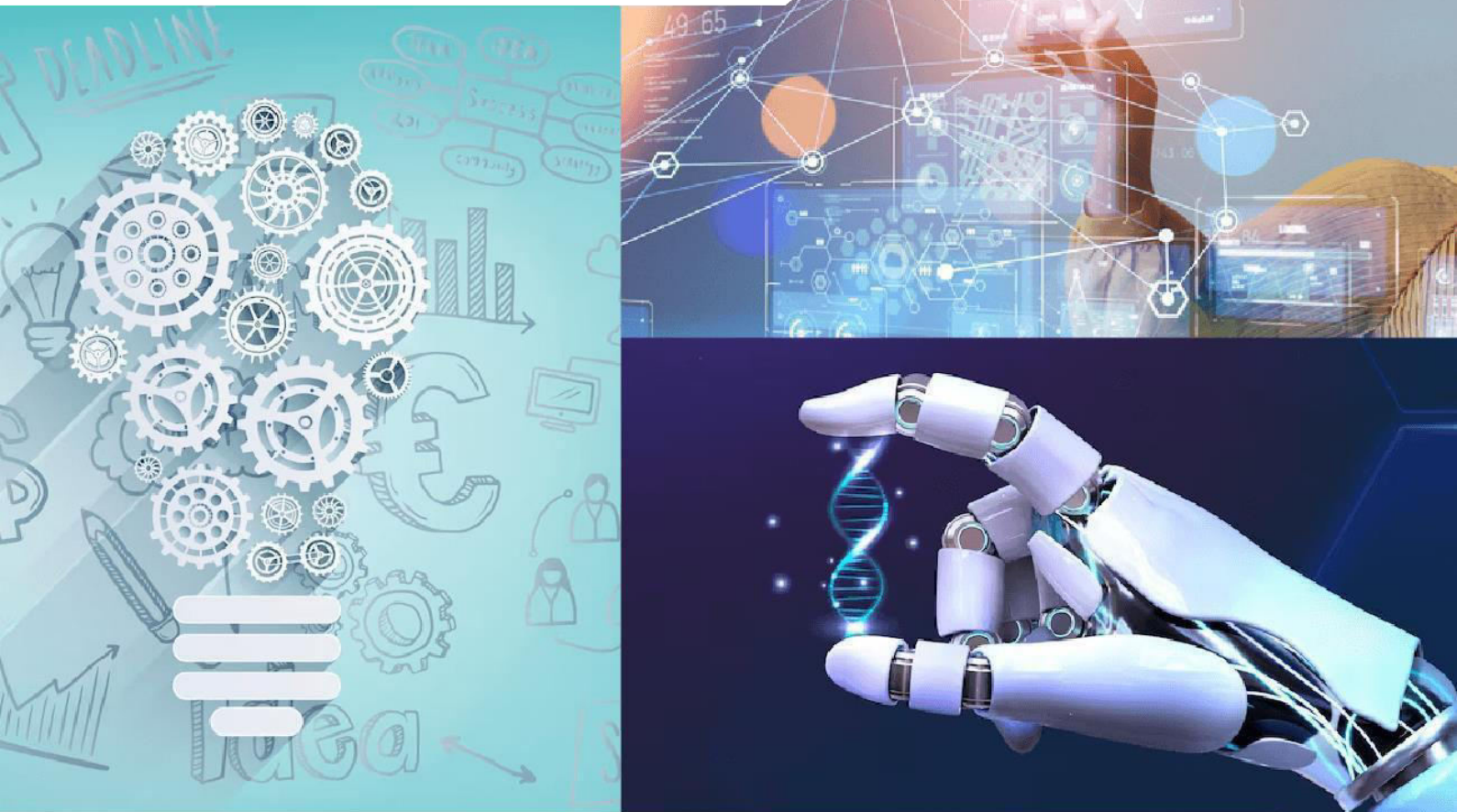




International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 3, May - June 2026

Impact Factor: 9.971



Data Analytics for E-Commerce: A Browser-Based Interactive Intelligence Platform

Dheemathi T, Gnaneswari Gnanaguru

Department of Master of Computer Applications, CMR Institute of Technology, Bengaluru, Karnataka, India
Associate Professor, Department of Master of Computer Applications, CMR Institute of Technology, Bengaluru,
Karnataka, India

ABSTRACT: This paper presents a lightweight, browser-based analytics platform that transforms raw CSV records into interactive visual reports for e-commerce businesses. Built with React.js, Vite, PapaParse, and Chart.js, the system displays key performance indicators—total revenue, order trends, product performance, regional sales, and customer behaviour—immediately upon data upload, without requiring back-end infrastructure or domain expertise. The platform serves small e-commerce businesses, data analysts, and academic learners alike.

KEYWORDS: e-commerce analytics, data visualisation, React.js, CSV parsing, interactive dashboard, browser-based analytics, Chart.js, PapaParse

I. INTRODUCTION

A. Project Description

E-commerce generates large volumes of transactional data daily, yet many small online stores rely on spreadsheets and manual methods to extract insights—a workflow that is time-consuming and error-prone. The Data Analytics for E-Commerce platform addresses this gap by converting raw CSV records into actionable insights: revenue totals, purchase patterns, product-level results, geographic sales, and buyer behaviour. The application is a single-page web app (SPA) running entirely in the browser, eliminating backend infrastructure and adapting responsively across desktop, tablet, and mobile viewports.

II. LITERATURE SURVEY

A. Existing and Proposed System

Traditional e-commerce analysis relies on spreadsheet tools and static reporting software, which degrade at scale: trend spotting is laborious, formula errors distort results, and reports offer limited interactivity. The proposed system resolves these shortcomings through a web-driven, upload-and-explore paradigm. Users supply a CSV file; the platform validates, parses, and renders interactive charts automatically—surfacing revenue, top products, regional performance, and order trends—without installation overhead.

B. Feasibility Study

1) Technical Feasibility

The project is technically feasible using mature, open-source technologies: React.js/Vite for the front-end, Chart.js for visualisation, and PapaParse for client-side CSV parsing. No high-end hardware or proprietary software is required.

2) Operational Feasibility

Users upload a CSV file and immediately view dashboard results. The clean interface requires no specialist training; graceful error handling prevents crashes on malformed data.

3) Economic Feasibility

All tools are open-source, eliminating licensing fees. Browser-based execution removes dedicated server costs; standard hardware suffices.

4) Schedule Feasibility

Modular component architecture enables parallel development tracks; familiarity with selected technologies makes semester-end delivery realistic.

C. Tools and Technologies

Frontend: React.js (component-based UI), Vite (fast HMR build tool), JavaScript ES6+, Vanilla CSS. Data processing: PapaParse (browser-native CSV parser), Chart.js / React-Chartjs-2 (bar, line, and pie charts). State management: React Context API (global state, no prop-drilling). Development tools: VS Code, Node.js, Chrome/Firefox/Edge for cross-browser testing.

D. Hardware and Software Requirements

Hardware: Intel Core i3+, 4 GB RAM, 500 MB storage, 1366×768 display, stable internet. Software: Windows 10/11, Linux, or macOS; Node.js v16+; React.js, Vite, PapaParse, Chart.js, React-Chartjs-2, Vanilla CSS, React Context API; VS Code; latest Chrome, Firefox, or Edge.

III. SOFTWARE REQUIREMENTS SPECIFICATION

A. Users

Business Admin — Uploads CSV datasets, monitors overall revenue and order volume. Data Analyst — Examines detailed charts and prepares evidence-based reports. Inventory Manager — Reviews product-level sales to support stock decisions.

B. Functional Requirements

- Data Upload: Users select a CSV via file-picker; system validates format before ingestion.
- Data Parsing & Processing: CSVs parsed immediately; sales, product, timestamp, and geographic data extracted to memory.
- Dashboard Display: Responsive dashboard renders total revenue, order count, and derived metrics post-parsing.
- Data Visualisation: Bar, line, and pie charts update dynamically on data load.
- Navigation: Users switch between overview, trend, product, and regional views via navigation panel.
- User Interaction: Charts respond to hover/click events revealing granular values.
- Error Handling: Invalid files trigger descriptive messages; application does not crash.
- Data Refresh: New CSV upload clears prior session data and re-renders all visualisations.

C. Non-Functional Requirements

Performance: loads within seconds; CSV processing completes without perceptible delay. Security: data processed locally; invalid formats rejected. Usability: clean layout; no training required. Reliability: errors produce clear messages; data is consistent with upload. Maintainability: modular code enables independent component updates. Portability: runs on Chrome, Firefox, Edge across Windows, Linux, macOS. Scalability: architecture accommodates larger datasets and additional modules.

IV. SYSTEM DESIGN

The system comprises three layers: (1) Presentation Layer—React.js/Vanilla CSS, handling file upload, metric tiles, chart panels, and navigation. (2) Application Layer—JavaScript modules for CSV ingestion, statistical computation, and React Context API state management. (3) Data Layer—parsed records held in browser memory for the session; no persistent storage; a new upload replaces prior data.

A. System Perspective

Three cooperating subsystems—React.js/CSS frontend, JavaScript processing layer, and transient in-memory data store—produce a lean, modular system suitable for academic and small-business deployment.

B. Context Diagram

The system receives inputs from three user types (Business Admin, Data Analyst, Inventory Manager) and uploaded CSV sources, providing dashboards, charts, and KPI summaries in return. All parsing, computation, and visualisation logic is contained within the system boundary; no data crosses to external servers.

V. DETAILED DESIGN

A. Use Case Diagram

Three actors interact with eight primary use cases. Table 1 summarises these interactions.

Table 1: Use Case Summary

Use Case	Actor	Description
Upload Sales Data	Admin	Uploads CSV; system validates and loads data
View Dashboard	Admin	Views revenue, orders, and trends
Update Dataset	Admin	New CSV replaces prior data
Study Sales Trends	Analyst	Reviews trends by time, product, and region
Compare Metrics	Analyst	Compares KPIs using charts
Check Product Perf.	Inv. Mgr	Reviews product-wise sales
Identify Demand	Inv. Mgr	Finds high- and low-demand items
View Reports	All	Views visual analytical reports

B. Sequence Diagrams

Five workflows are modelled: (1) Login—credentials validated; dashboard rendered or error shown. (2) Sign Up—email uniqueness checked; account created or duplicate error shown. (3) Upload CSV—file validated; parsed and stored in memory. (4) Data Processing—totals, averages, and trends computed; visualisations rendered. (5) Logout—session cleared; user redirected to login.

C. Collaboration & Activity Diagrams

Collaboration diagrams highlight object relationships for Login, Sign Up, CSV Upload, Data Processing, and Dashboard View. Activity diagrams with decision nodes cover seven workflows: Login, Sign Up, Upload CSV, Data Processing, View Dashboard, Refresh Dataset, and Logout.

D. Database Design

A conceptual schema is defined for future extensibility. Entities: User (userId PK, name, email, password, dateOfRegistration), Dataset (datasetId PK, userId FK, fileName, uploadDate, fileType), SalesData (recordId PK, datasetId FK, productName, category, quantity, revenue, orderDate), DashboardMetrics (metricId PK, datasetId FK, totalRevenue, totalOrders, averageSales, generatedDate), ProductPerformance (productId PK, datasetId FK, productName, totalSales, demandLevel). Relationships: User–Dataset (1:N), Dataset–SalesData (1:N), Dataset–DashboardMetrics (1:1), Dataset–ProductPerformance (1:N).

VI. IMPLEMENTATION

The entry point App.jsx composes five analytical agents (DataEngineerAgent, BusinessAnalystAgent, DemandForecastAgent, OperationsAgent, CustomerIntelligenceAgent) within a DataProvider context, rendering an agents sidebar alongside a workspace containing DataUploader, DatasetSelector, analysis controls, and ResultsPanel.

```
import { DataProvider } from './context/DataContext';
import DataUploader from './components/DataUploader';
import ResultsPanel from './components/ResultsPanel';
```

```
const agents = [
  new DataEngineerAgent(), new BusinessAnalystAgent(),
  new DemandForecastAgent(), new OperationsAgent(),
  new CustomerIntelligenceAgent()
];
```

```
export default function App() {
  return <DataProvider><AppContent /></DataProvider>;
}
```

Listing 1: App.jsx – simplified structure

The application bootstraps via main.jsx using React StrictMode. All CSS uses Vanilla CSS custom properties (--space-md, --font-size-lg, etc.) for consistent theming.

VII. SOFTWARE TESTING**A. Introduction**

Testing validated the system against specified requirements, covering authentication, dataset upload, data processing, dashboard rendering, and session management.

B. Testing Strategy

A multi-layered strategy was employed: Black-Box Testing (input–output validation), White-Box Testing (CSV parsing and metric logic paths), Unit Testing (individual modules), Integration Testing (end-to-end flows), System Testing (full application under realistic conditions), and User Acceptance Testing (UAT).

C. Test Plan

Test items: Login, Sign Up, Dataset Upload, Data Processing, Dashboard Visualisation, Logout. Environment: React/Vanilla CSS frontend; JavaScript processing; Windows 10; Chrome and Edge.

D. Test Cases**Table 2: Selected Test Cases and Results**

ID	Description	Expected Output	Status
TC01	Valid login	Redirect to dashboard	Pass
TC02	Invalid password	Error message displayed	Pass
TC03	Empty login fields	Warning displayed	Pass
TC04	Valid sign up	Account created	Pass
TC05	Duplicate email	Error message displayed	Pass
TC07	Valid CSV upload	Upload successful	Pass
TC08	Non-CSV file	Error displayed	Pass
TC10	Correct metric calc	Accurate KPIs generated	Pass
TC11	Dataset with nulls	Handled without error	Pass
TC12	Dashboard with data	Charts rendered correctly	Pass
TC13	Dashboard, no data	Upload prompt shown	Pass
TC14	Logout action	Session terminated	Pass
TC15	Back after logout	Redirect to login	Pass

E. Test Results and Conclusions

All 15 test cases passed. Two minor issues were identified and resolved: (1) large CSV files showed elevated upload times on slow connections—resolved by optimising validation logic; (2) occasional chart render delays—resolved by improving state management. No critical defects remained open.

VIII. CONCLUSION

The Data Analytics for E-Commerce platform demonstrates that modern browser technologies can bridge the gap between raw transactional data and actionable business intelligence without back-end infrastructure or domain expertise. The system fulfils all stated requirements: secure authentication, rapid CSV ingestion, accurate metric computation, and responsive visual reporting. Testing confirmed stability, usability, and correctness. The platform is a viable lightweight analytics tool for small e-commerce operations.

IX. FUTURE ENHANCEMENTS

- Mobile Application: Native Android/iOS app for on-the-go analysis.
- Advanced Analytics: Predictive sales trends and demand forecasting using machine learning.
- Smart Recommendations: Automated insight suggestions—top products, seasonal trends—from uploaded data.
- Multi-Format Support: Accept Excel (XLS/XLSX) files and direct database connections.
- Personalised Dashboards: User-saved chart and metric preferences for recurring analysis.

REFERENCES

- [1] I. Sommerville, Software Engineering, 10th ed. Harlow, UK: Pearson Education, 2015.
- [2] R. S. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach, 8th ed. New York, NY: McGraw-Hill, 2014.
- [3] Object Management Group (OMG), "Unified Modeling Language Specification," [Online]. Available: <https://www.omg.org/spec/UML/>. [Accessed: 2025].
- [4] MDN Web Docs, "JavaScript Reference," Mozilla, [Online]. Available: <https://developer.mozilla.org/>. [Accessed: 2025].
- [5] React Team, "React – A JavaScript Library for Building User Interfaces," [Online]. Available: <https://react.dev/>. [Accessed: 2025].
- [6] PapaParse, "Fast and Powerful CSV Parser for JavaScript," [Online]. Available: <https://www.papaparse.com/>. [Accessed: 2025].
- [7] Chart.js Contributors, "Chart.js Documentation," [Online]. Available: <https://www.chartjs.org/docs/>. [Accessed: 2025].
- [8] E. You, "Vite – Next Generation Frontend Tooling," [Online]. Available: <https://vitejs.dev/>. [Accessed: 2025].
- [9] MDN Web Docs, "Web Standards and Browser Compatibility," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web>. [Accessed: 2025].

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 9.971