International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 2, Mar-Apr 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

IdeaPoll: A React.js Based Community Idea Sharing and Voting Web Application

Arjun G*
*(Department of Computer Science and Engineering, Bharath Institute of Higher Education and Research, Chennai, India
Email: garjunarjun9@gmail.com

Abstract:
IdeaPoll is a modern, lightweight web application developed using React.js that enables users to share, discover, and vote on ideas within a collaborative community environment. The platform provides a seamless experience for posting ideas with categories, liking others' ideas, filtering content by topic, and sorting ideas by popularity or recency. The application employs browser localStorage as its data persistence layer, making it a fully client-side solution that requires no backend server or database infrastructure. The system incorporates key algorithms including Linear Search for real-time filtering, TimSort for sorting, and a Toggle Algorithm for the voting mechanism. With a polished dark-themed user interface and efficient state management using React Hooks, IdeaPoll demonstrates modern web development practices including component-based architecture and client-side persistence.

Keywords — React.js, Single Page Application, Idea Sharing, Community Voting, localStorage, Collaborative Platform, Component-Based Architecture, Client-Side Persistence, Web Application.

I. INTRODUCTION
In the digital age, collaborative platforms have fundamentally transformed how communities generate, share, and evaluate ideas. The proliferation of web technologies, particularly JavaScript frameworks such as React.js, has enabled developers to create highly interactive, responsive applications entirely on the client side.

IdeaPoll addresses the gap in lightweight, purpose-built idea sharing tools by providing a minimal, fast-loading web application that focuses exclusively on idea submission and community upvoting. All data is persisted in the browser localStorage, eliminating the need for backend infrastructure.

This paper presents the design, architecture, and implementation of IdeaPoll. Section II reviews relevant literature. Section III describes the problem statement. Section IV presents the proposed system. Section V outlines the system architecture. Section VI describes the modules. Section VII presents results and Section VIII concludes.

II. LITERATURE SURVEY
A. React.js and Component-Based Web Development
Maurya et al. [1] conducted a comprehensive study of web development using React.js, demonstrating that React's virtual DOM and component-based architecture significantly improve rendering performance and code reusability. Their findings directly informed IdeaPoll's use of functional components and React Hooks.

B. Single Page Application Architecture
Research on Single Page Application (SPA) architectures [4] establishes that SPAs deliver superior user experience by eliminating full page reloads and enabling dynamic content rendering. IdeaPoll follows SPA principles — a single HTML entry point and dynamic component rendering.

C. Web Storage API and Client-Side Persistence
The W3C Web Storage specification and subsequent research [5] define the localStorage and sessionStorage APIs for client-side data persistence. IdeaPoll leverages localStorage for

persisting idea data and user profiles across browser sessions without backend infrastructure.

D. Security of CORS and Client-Side Storage
Studies on CORS and localStorage security [3] highlight the importance of properly configuring cross-origin resource sharing when client-side applications communicate with backend services, essential for extending IdeaPoll with backend APIs.

E. Collaborative Filtering and Voting Systems
Resnick et al. [2] pioneered research in collaborative filtering and community voting, demonstrating that upvoting mechanisms effectively surface high-quality content. This work inspired IdeaPoll's like/unlike voting system and popularity-based sorting.

III. PROBLEM STATEMENT

Despite the wide availability of online platforms, several critical problems persist:

- Existing platforms like Reddit are feature-heavy and not purpose-built for focused idea sharing and voting.
- Most tools require mandatory registration, creating friction before participation.
- Nearly all platforms depend on backend servers and databases, increasing cost and complexity.
- No minimal, fast-loading tool exists that purely focuses on idea submission and community upvoting.
- Existing platforms like IdeaScale are enterprise-focused with paid plans unsuitable for individual or academic use.

IV. PROPOSED SYSTEM

IdeaPoll is a fully client-side web application built with React.js that provides the following features:

A. Quick User Setup
On first visit, users enter only a display name. A unique user ID is generated using a composite UID algorithm combining random base-36 encoding with Unix timestamps, persisted in localStorage. No email or password is required.

B. Idea Posting
Users post ideas with a title (up to 100 characters), a description (up to 500 characters), and a category from: Tech, Work, Society, Health, Education, Environment, and Other. New ideas render instantly without page reload.

C. Community Voting
Each idea features a like/unlike toggle button. A likedBy array tracks which users liked each idea, preventing duplicate likes and enabling accurate counts via the Toggle Algorithm.

D. Search and Filtering
A live search applies Linear Search across idea titles and descriptions. Category filters apply exact string matching. Sort options use JavaScript's built-in TimSort to order results by creation date or like count.

E. Data Persistence
All application data is serialized to JSON and stored in localStorage under keys ideapoll_data and ideapoll_user. On load, data is deserialized with seed ideas from db.json as fallback for first-time users.

V. SYSTEM ARCHITECTURE

IdeaPoll follows a layered client-side architecture as shown in Table I:

TABLE I
SYSTEM ARCHITECTURE LAYERS

LAYER	DESCRIPTION
Presentation	React.js components: App.js, IdeaCard, PostModal, DetailModal, NameModal.
Logic	React Hooks (useState, useEffect, useCallback) for state and business logic.
Helper	uid(), timeAgo(), loadData(), saveData() utility functions.
Persistence	localStorage: ideapoll_data (ideas), ideapoll_user (user profile).

The application entry point is index.js, which mounts the root App component. App.js serves as the main controller managing global state. Data flows unidirectionally from App state to child components via props.

VI. MODULES

A. User Module

Handles user identity management. On first visit, NameModal collects the display name. A uid() algorithm generates a unique ID, and an avatar character is derived from the first letter of the name. User profile is saved to local Storage.

B. Idea Module

Manages idea creation and display. Post Modal provides a form for idea submission. New ideas are prepended to the ideas array in state and immediately rendered. Each idea is displayed as an IdeaCard component.

C. Voting Module

Implements community voting. The handle Like() function applies a toggle algorithm checking if the user's ID exists in the idea's liked By array. If present, the user is removed (unlike); if absent, the user is added (like).

D. Search and Filter Module

Provides real-time content discovery using Linear Search (O(n)) for text matching, exact string matching for category filtering, and TimSort O(n log n) for sorting by date or popularity.

E. Persistence Module

saveData() serializes state to JSON and writes to localStorage. loadData() reads and deserializes on app load. Falls back to seed ideas from db.json if localStorage is empty.

F. UI/UX Module

Delivers dark-themed interface with purple accents, animated modals, category pills, toast notifications, a statistics bar, and a responsive layout for desktop and mobile browsers.

VII. RESULTS AND DISCUSSION

IdeaPoll was successfully implemented as a fully functional React.js web application. Table II shows the feature testing results:

TABLE II
FEATURE TESTING RESULTS

FEATURE	EXPECTED	RESULT
Idea Posting	Instant render	Pass
Like/Unlike Toggle	Real-time update	Pass
Duplicate Like Prevention	One like/user	Pass
Search Filter	Live results	Pass
Category Filter	Exact match	Pass
Sort by Newest	Desc. date order	Pass
Data Persistence	Survives refresh	Pass

All core features functioned as expected. The application loads within 1-2 seconds on standard broadband. React's virtual DOM ensures efficient re-rendering, with only affected components updating on state change.

VIII. CONCLUSION

This paper presented IdeaPoll, a lightweight, fully client-side community idea sharing and voting web application built with React.js. The system successfully addresses complex existing platforms, mandatory registration, and backend infrastructure dependencies by delivering a simple, fast, and free solution running entirely in the browser.

The application demonstrates the power of modern React.js development using functional components, React Hooks, and the browser localStorage API to deliver a complete collaborative platform without server infrastructure.

Future work includes backend REST API integration, OAuth authentication, comment threads, WebSocket notifications, and an admin moderation dashboard.

ACKNOWLEDGMENT

The author would like to thank the faculty of the Department of Computer Science and Engineering for their guidance throughout the development of this project. Special thanks to the open-source React.js community and authors of the referenced works.

REFERENCES

[1] P. Maurya, A. Katiyar, P. Keshari and P. Kumar, "Web Development Using ReactJS," 2023 Int. Conf. on Artificial Intelligence, Innovation and High-Technology (ICAIIHI), 2023, pp. 1-5, doi: 10.1109/ICAIIHI57871.2023.10541743.

[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in Proc. ACM Conf. Computer Supported Cooperative Work (CSCW), Chapel Hill, NC, 1994, pp. 175-186.

[3] Author(s), "Security of CORS on LocalStorage," in 2021 Int. Seminar on Application for Technology of Information and Communication (iSemantic), 2021, doi: 10.1109/ISEMANTIC52711.2021.9525853.

[4] Author(s), "User Interface Design Practices in Simple Single Page Web Applications," in Proc. IEEE Int. Conf. e-Business Engineering (ICEBE), 2008, doi: 10.1109/ICEBE.2008.4664349.

[5] Author(s), "Toward User's Devices Collaboration to Distribute Securely the Client Side Storage," in Proc. IEEE Int. Conf. on Data Science (ICDS), 2015, doi: 10.1109/ICDS.2015.7293479.

[6] React Documentation, "Hooks API Reference," Facebook Inc., 2024. [Online]. Available: https://react.dev/reference

[7] Mozilla Developer Network, "Web Storage API," MDN Web Docs, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

