# Table 2.1: Popular Methods/Techniques for Data Storage and State Management in React Applications

| Method/Technique | Description | Advantages | Disadvantages | Usage in Pro Mood Tracker |
| --- | --- | --- | --- | --- |
| Local Storage | Browser-based key-value storage that persists across sessions | - Persists after browser restarts- Simple API- Widely supported | - Limited storage capacity (5-10MB)- String-only storage- No indexing capabilities | Primary storage for user mood data and application settings |
| Session Storage | Similar to localStorage but cleared when session ends | - Same API as localStorage- Improved security for sensitive data | - Lost on tab/browser close- Same limitations as localStorage | Used for temporary session data and form state persistence |
| IndexedDB | Browser-based object store with advanced querying | - Large storage capacity- Complex data structure support- Indexing and searching | - Complex API- Requires more boilerplate code | Considered for future implementation for larger datasets |
| React Context API | Built-in state management for React components | - Native to React- No external dependencies- Hierarchical state | - Can cause re-renders- Not optimized for frequent updates | Used for theme, auth, user, and mood state management |
| Redux | Centralized state management library | - Predictable state updates- Time-travel debugging- Middleware support | - Verbose boilerplate- Learning curve- Additional dependency | Not used in current implementation |
| MobX | Reactive state management library | - Less boilerplate- Automatic tracking- Simplified API | - Less explicit data flow- Can be harder to debug | Not used in current implementation |
| React Query | Data fetching and caching library | - Automatic caching- Background updates- Loading/error states | - External dependency- Primarily for server data | Not used in current implementation |
| Zustand | Simple state management | - Minimal API- No providers needed- Good performance | - Relatively new- Less community support | Not used in current implementation |
| Recoil | Facebook’s experimental state management | - Atom-based approach- Fine-grained updates- Concurrent mode compatible | - Experimental status- Evolving API | Not used in current implementation |
| Custom Hooks | Encapsulated state logic in reusable hooks | - Highly customizable- No dependencies- Focused functionality | - Can lead to duplication- Requires careful design | Used for specific functionality like window size detection and form handling |