**Project Report**

**Mindora: Progressive Web Application**

**1. Abstract**

Mindora is a modern, AI-powered app tailored for students to efficiently manage tasks, take notes, review study material, and organize their academic schedule. It combines a Progressive Web App (PWA) architecture with AI capabilities to offer offline support, intuitive navigation, and intelligent features such as smart task suggestions. Developed by a team working on frontend, backend, UI/UX, and AI modules, Mindora is a holistic platform built with React, IndexedDB, and Google Generative AI. This report documents the technical journey, challenges overcome, and the key outcomes of the project.

**2. Introduction**

**2.1 Motivation**

Students often juggle multiple academic responsibilities, requiring a centralized tool for task management, note-taking, revision, and scheduling. Existing apps are often fragmented or lack AI integration and offline capabilities.

**2.2 Problem Statement**

There is a gap in student-focused productivity tools that are intelligent, accessible offline, and integrative.

**2.3 Objectives**

* Develop an all-in-one student productivity suite
* Ensure offline functionality and fast performance
* Integrate AI for smarter study planning
* Design a clean, accessible UI for better usability

**3. Team Structure & Contributions**

Our team was divided into domain-specific roles to ensure a focused and efficient development process:

**Frontend Development – Krishna Kumar Singh (2401410003)**

* Built the core React structure
* Developed responsive UI with Tailwind CSS
* Set up routing and theme toggle functionality

**UI/UX Design – Deepti Shaw (2401410005)**

* Designed app interface and layout
* Focused on accessibility standards
* Managed design consistency and documentation

**AI Integration – Shashank Kumar (2401410007)**

* Integrated Google’s Generative AI SDK
* Built AI assistant interface
* Optimized task suggestion and response logic

**Backend & PWA Integration – Tanya Shivhare (2401410010)**

* Integrated IndexedDB for offline storage
* Implemented service workers
* Designed data persistence layer and APIs

**4. Technical Architecture**

**4.1 Frontend Framework**

* **React 18 + TypeScript**
* **Vite** for fast bundling
* **React Router v6** for navigation
* **Context API** for global state

**4.2 UI Styling**

* **Tailwind CSS** for utility-first design
* **Dark/Light Theme** switcher
* **Lucide React** for iconography

**4.3 Progressive Web App (PWA)**

* Offline installability
* Service workers
* Local storage via IndexedDB

**4.4 AI Integration**

* Google Generative AI (@google/generative-ai)
* Smart task and schedule suggestions
* Natural language understanding

**5. Key Features**

**5.1 Task Management**

* Create, update, and delete tasks
* Categorize and set priorities
* Due date and status tracking

**5.2 Note Taking**

* Rich-text editing
* Image and media support
* Organized by subject or category

**5.3 Flashcards**

* Create and study flashcards
* Progress tracking with spaced repetition
* Export/import support

**5.4 Calendar Integration**

* Event scheduling and reminders
* Visual deadline tracking
* Daily, weekly, monthly views

**5.5 PDF Storage**

* Upload and organize academic PDFs
* Searchable archive
* Offline access to documents

**5.6 AI Assistant**

* Context-aware natural language assistant
* Intelligent task suggestions
* Schedule optimizations and study tips

**6. Technical Innovations**

**6.1 Performance Optimizations**

* Code splitting and lazy loading
* Optimized asset delivery
* Efficient render strategies

**6.2 Offline Capabilities**

* IndexedDB for persistent local data
* Background sync when reconnected
* Full PWA installability

**6.3 State Management**

* Context API with modular state slices
* Custom hooks for reusability
* Persistent state across reloads

**7. Development Process**

**7.1 Initial Setup**

* Vite + TypeScript scaffold
* ESLint and Prettier setup
* GitHub repo with branch workflow

**7.2 Workflow**

* Component-driven architecture
* Agile sprint-based development
* Code reviews and versioning

**7.3 Testing**

* Unit testing using Jest
* Accessibility testing (contrast, keyboard nav)
* Performance benchmarks and CI checks

**8. Challenges & Solutions**

**8.1 Offline Functionality**

* Implemented IndexedDB wrapper
* Custom sync logic
* Efficient caching strategies

**8.2 AI Integration**

* Dealt with token rate limits and response delays
* Created fallback systems
* Optimized query format for better results

**8.3 Performance Tuning**

* Minimized bundle size with tree-shaking
* Prioritized rendering of critical UI
* Deferred non-essential scripts

**9. Future Roadmap**

**9.1 Planned Features**

* User authentication and login
* Cloud sync across devices
* Advanced full-text search
* Native mobile app (React Native or Flutter)

**9.2 Improvements**

* More personalized AI features
* Push notifications
* Better PDF annotation tools
* Study session analytics

**9. Project Impact**

* Unified productivity for students
* Cross-platform availability
* Enhanced study methods via AI
* Robust offline support
* Improved academic organization and time management

**10. References**

* React Documentation – <https://react.dev>
* Tailwind CSS – <https://tailwindcss.com>
* Google Generative AI SDK – <https://ai.google.dev>
* Vite – <https://vitejs.dev>
* IndexedDB API – <https://developer.mozilla.org>

**11. Appendices**

* GitHub Repository: *[Insert Link]*
* Design Wireframes: *[Insert Link or Screenshot]*
* Architecture Diagram: *[Insert if available]*