## Attendance Planning & Decision Support System - Technology Architecture

Project Type: Progressive Web App (PWA)
Deployment: Static Hosting (No Backend Required for MVP)
Philosophy: Keep it simple, explainable, debuggable

## 1. TECHNOLOGY STACK OVERVIEW

### 1.1 Stack Summary

Layer	Technology	Rationale
Frontend Framework	React 18+ with Vite	Fast dev experience, component reusability, widely known
Styling	Tailwind CSS	Utility-first, rapid prototyping, small bundle size
State Management	Zustand	Simpler than Redux, perfect for this scale
Data Storage	LocalStorage + IndexedDB	No backend needed, offline-first
Date Handling	date-fns	Lightweight alternative to Moment.js
Charts/Visualization	Recharts	React-native charts, simple API
PDF Parsing	pdf.js (Mozilla)	Extract data from defaulter lists
Hosting	Vercel / Netlify	Free tier, automatic deployments
Version Control	Git + GitHub	Standard, required for collaboration

---

## 2. DETAILED TECHNOLOGY DECISIONS

### 2.1 Frontend Framework: React + Vite

Why React?
- ✅ Component-based (Subject Card, Dashboard are reusable)
- ✅ Large community (easy debugging)
- ✅ Hooks simplify state management
- ✅ Can be explained in viva (virtual DOM, component lifecycle)
- ✅ Familiar to most students

Why Vite over Create React App?
- ⚡ Faster dev server (instant HMR)
- 📦 Smaller build output
- 🔧 Better default config (no eject needed)
- 🚀 Modern tooling (ES modules)

Setup:
```bash
npm create vite@latest attendance-planner -- --template react
cd attendance-planner
```

npm install

Project Structure:
attendance-planner/
├── public/
│   ├── icons/
│   └── manifest.json   # PWA manifest
├── src/
│   ├── components/
│   │   ├── Dashboard.jsx
│   │   ├── SubjectCard.jsx
│   │   ├── SkipModal.jsx
│   │   └── RecoveryPlan.jsx
│   ├── utils/
│   │   ├── calculations.js   # Core logic
│   │   ├── dateHelpers.js
│   │   └── validation.js
│   ├── store/
│   │   └── attendanceStore.js # Zustand store
│   ├── data/
│   │   ├── semester.json
│   │   └── timetable.json
│   ├── App.jsx
│   ├── main.jsx
│   └── index.css
├── package.json
└── vite.config.js

### 2.2 Styling: Tailwind CSS

Why Tailwind?
- ✅ No context switching (HTML + CSS in one place)
- ✅ Utility classes prevent CSS bloat
- ✅ Responsive design built-in (`md:`, `lg:` prefixes)
- ✅ Customizable (theme colors match design doc)
- ✅ PurgeCSS removes unused styles (small bundle)

Installation:
```bash
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Tailwind Config (tailwind.config.js):
```javascript
```

```
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {
      colors: {
        critical: {
          DEFAULT: '#EF4444',
          light: '#FEE2E2',
        },
        caution: {
          DEFAULT: '#F59E0B',
          light: '#FEF3C7',
        },
        safe: {
          DEFAULT: '#10B981',
          light: '#D1FAE5',
        },
      },
      fontFamily: {
        sans: ['Inter', 'sans-serif'],
        mono: ['JetBrains Mono', 'monospace'],
      },
    },
  },
  plugins: [],
}
```

Example Usage:
```jsx
<div className="bg-critical-light border-l-4 border-critical p-4 rounded-lg">
  <h3 className="text-xl font-semibold text-gray-900">
    Machine Learning
  </h3>
  <p className="text-critical font-mono text-3xl">68.2%</p>
</div>
```

### 2.3 State Management: Zustand

Why Zustand over Redux?
- ✅ 10× less boilerplate
- ✅ No providers/context needed

- ✅ Works great with React hooks
- ✅ Easier to explain in viva
- ✅ Perfect for this app's complexity level

Installation:
```bash
npm install zustand
```

Store Implementation (src/store/attendanceStore.js):
```javascript
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

const useAttendanceStore = create(
  persist(
    (set, get) => ({
      // State
      subjects: [],
      semester: null,
      lastUpdated: null,

      // Actions
      addSubject: (subject) => set((state) => ({
        subjects: [...state.subjects, subject],
        lastUpdated: new Date().toISOString(),
      })),

      updateAttendance: (subjectCode, attended, conducted) => set((state) => ({
        subjects: state.subjects.map((sub) =>
          sub.code === subjectCode
            ? { ...sub, attended, conducted }
            : sub
        ),
        lastUpdated: new Date().toISOString(),
      })),

      setSemester: (semesterData) => set({ semester: semesterData }),

      // Computed values (selectors)
      getSubjectStatus: (subjectCode) => {
        const subject = get().subjects.find(s => s.code === subjectCode);
        // Calculate buffer, deficit, PNR, etc.
        return calculateStatus(subject);
      },
    }),
    {
      name: 'attendance-storage', // LocalStorage key
```

      version: 1,
    }
)
);

export default useAttendanceStore;

Usage in Components:
```jsx
import useAttendanceStore from './store/attendanceStore';

function Dashboard() {
  const subjects = useAttendanceStore((state) => state.subjects);
  const updateAttendance = useAttendanceStore((state) => state.updateAttendance);

  return (
    <div>
      {subjects.map(subject => (
        <SubjectCard key={subject.code} subject={subject} />
      ))}
    </div>
  );
}
```

### 2.4 Data Storage: LocalStorage + IndexedDB

Why No Backend?
- ✅ Faster development (no API to build)
- ✅ Works offline by default
- ✅ No hosting costs
- ✅ Privacy-friendly (data never leaves device)
- ✅ Perfect for single-user app

Storage Strategy:

LocalStorage (via Zustand persist):
- Small data (<5MB)
- User preferences, settings
- Current semester config
- Subject attendance data

IndexedDB (for larger data):
- Uploaded PDFs (defaulter lists)
- Historical semester data

- Exported reports

Implementation:
```javascript
// Wrapper for IndexedDB
import { openDB } from 'idb';

const DB_NAME = 'attendance-db';
const DB_VERSION = 1;

async function initDB() {
  return openDB(DB_NAME, DB_VERSION, {
    upgrade(db) {
      if (!db.objectStoreNames.contains('documents')) {
        db.createObjectStore('documents', { keyPath: 'id', autoIncrement: true });
      }
    },
  });
}

export async function saveDocument(file) {
  const db = await initDB();
  const arrayBuffer = await file.arrayBuffer();
  return db.add('documents', {
    name: file.name,
    type: file.type,
    data: arrayBuffer,
    uploadedAt: new Date().toISOString(),
  });
}

export async function getDocument(id) {
  const db = await initDB();
  return db.get('documents', id);
}
```

### 2.5 Date Handling: date-fns

Why date-fns?
- ✅ Lightweight (13KB vs Moment's 67KB)
- ✅ Tree-shakable (import only what you need)
- ✅ Immutable (no date mutation bugs)
- ✅ TypeScript support

Installation:

```bash
npm install date-fns
```

Usage Examples:
```javascript
import {
  parseISO,
  format,
  differenceInDays,
  addWeeks,
  isWithinInterval
} from 'date-fns';

// Calculate weeks elapsed
const semesterStart = parseISO('2026-01-19');
const today = new Date();
const weeksElapsed = Math.floor(differenceInDays(today, semesterStart) / 7);

// Format display dates
const displayDate = format(today, 'EEEE, d MMM yyyy'); // "Thursday, 23 Jan 2026"

// Check if date is a holiday
function isHoliday(date, holidays) {
  return holidays.some(holiday =>
    format(parseISO(holiday.date), 'yyyy-MM-dd') === format(date, 'yyyy-MM-dd')
  );
}

// Calculate PNR date
function calculatePNR(subject, semesterEnd) {
  const sessionsUntilPNR = calculateSessionsUntilPNR(subject);
  const weeksNeeded = Math.ceil(sessionsUntilPNR / subject.sessionsPerWeek);
  return addWeeks(new Date(), weeksNeeded);
}
```

### 2.6 Visualization: Recharts

Why Recharts?
- ✅ React-native (built for React)
- ✅ Responsive out of the box
- ✅ Simple API
- ✅ SVG-based (crisp on retina displays)

Installation:

```bash
npm install recharts
```

Usage Example (Buffer Visualization):
```jsx
import { BarChart, Bar, XAxis, YAxis, Tooltip, ResponsiveContainer } from 'recharts';

function BufferChart({ subjects }) {
  const data = subjects.map(sub => ({
    name: sub.code,
    buffer: calculateBuffer(sub),
  }));

  return (
    <ResponsiveContainer width="100%" height={200}>
      <BarChart data={data}>
        <XAxis dataKey="name" />
        <YAxis />
        <Tooltip />
        <Bar
          dataKey="buffer"
          fill={(data) => data.buffer < 2 ? '#EF4444' : '#10B981'}
        />
      </BarChart>
    </ResponsiveContainer>
  );
}
```

### 2.7 PDF Parsing: PDF.js

Why PDF.js?
- ✅ Mozilla-backed (reliable)
- ✅ Pure JavaScript (no native dependencies)
- ✅ Supports text extraction
- ✅ Works in browser

Installation:
```bash
npm install pdfjs-dist
```

Usage Example (Extract Defaulter List):
```javascript
import * as pdfjsLib from 'pdfjs-dist/build/pdf';
```

```
pdfjsLib.GlobalWorkerOptions.workerSrc =
`//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjsLib.version}/pdf.worker.min.js`;

async function parseDefaulterPDF(file) {
  const arrayBuffer = await file.arrayBuffer();
  const pdf = await pdfjsLib.getDocument({ data: arrayBuffer }).promise;

  let fullText = '';
  for (let i = 1; i <= pdf.numPages; i++) {
    const page = await pdf.getPage(i);
    const textContent = await page.getTextContent();
    const pageText = textContent.items.map(item => item.str).join(' ');
    fullText += pageText + '\n';
  }

  // Parse text to extract subjects and attendance
  return parseAttendanceText(fullText);
}

function parseAttendanceText(text) {
  // Example: "Machine Learning  Conducted: 45  Attended: 38"
  const regex = /([A-Za-z\s]+)\s+Conducted:\s*(\d+)\s+Attended:\s*(\d+)/g;
  const subjects = [];

  let match;
  while ((match = regex.exec(text)) !== null) {
    subjects.push({
      name: match[1].trim(),
      conducted: parseInt(match[2]),
      attended: parseInt(match[3]),
    });
  }

  return subjects;
}
```

## 3. CORE CALCULATION ENGINE

### 3.1 Pure Functions (src/utils/calculations.js)

Philosophy: All calculations must be:
- Pure (same input → same output)
- Testable (unit tests for every function)
- Explainable (comments show math)

```javascript
/**
 * Calculate current attendance percentage
 * Formula: (attended / conducted) × 100
 */
export function calculatePercentage(attended, conducted) {
  if (conducted === 0) return 0;
  return parseFloat(((attended / conducted) * 100).toFixed(2));
}

/**
 * Calculate attendance buffer
 * Buffer = classes you can still miss while staying ≥75%
 *
 * Formula:
 * min_required = total × 0.75
 * buffer = (attended + remaining) - min_required
 */
export function calculateBuffer(subject, totalSessions) {
  const { attended, conducted } = subject;
  const remaining = totalSessions - conducted;
  const minRequired = Math.ceil(totalSessions * 0.75);

  const buffer = (attended + remaining) - minRequired;
  return Math.max(0, Math.floor(buffer)); // Can't have negative buffer
}

/**
 * Calculate attendance deficit
 * Deficit = additional classes needed to reach 75%
 *
 * Formula:
 * min_required = total × 0.75
 * deficit = min_required - attended
 */
export function calculateDeficit(subject, totalSessions) {
  const { attended } = subject;
  const minRequired = Math.ceil(totalSessions * 0.75);

  const deficit = minRequired - attended;
  return Math.max(0, Math.ceil(deficit)); // Can't have negative deficit
}

/**
 * Calculate Point of No Return date
 * PNR = date when (deficit > remaining sessions)
 *
 * Returns: Date object or null
 */
export function calculatePNR(subject, totalSessions, semesterEnd, sessionsPerWeek) {
  const { attended, conducted } = subject;
  const remaining = totalSessions - conducted;
  const deficit = calculateDeficit(subject, totalSessions);

  // Already impossible
  if (deficit > remaining) {
    return new Date(0); // "Already passed"
  }

  // No PNR (safe)
  if (deficit === 0) {
    return null;
  }

  // Calculate date
  const sessionsUntilPNR = remaining - deficit;
  const weeksUntilPNR = Math.floor(sessionsUntilPNR / sessionsPerWeek);

  return addWeeks(new Date(), weeksUntilPNR);
}

/**
 * Determine subject status
 * Returns: "CRITICAL" | "CAUTION" | "SAFE"
 */
export function getStatus(subject, totalSessions, pnrDate) {
  const percentage = calculatePercentage(subject.attended, subject.conducted);
  const buffer = calculateBuffer(subject, totalSessions);

  // Critical: Below 75% OR buffer ≤1
  if (percentage < 75 || buffer <= 1) {
    return 'CRITICAL';
  }

  // Caution: Buffer 2-4 OR PNR within 14 days
  const daysUntilPNR = pnrDate ? differenceInDays(pnrDate, new Date()) : Infinity;
  if (buffer <= 4 || daysUntilPNR <= 14) {
    return 'CAUTION';
  }

  // Safe: Buffer ≥5
  return 'SAFE';
}

/**
 * Generate recovery plan
 * Returns: { possible: boolean, plan: Array, finalPercentage: number }
 */
export function generateRecoveryPlan(subject, totalSessions, sessionsPerWeek,
weeksRemaining) {
  const deficit = calculateDeficit(subject, totalSessions);
  const remaining = totalSessions - subject.conducted;

  // Impossible case
  if (deficit > remaining) {
    const maxAchievable = calculatePercentage(
      subject.attended + remaining,
      totalSessions
    );

    return {
      possible: false,
      maxPercentage: maxAchievable,
      message: 'Recovery is mathematically impossible',
    };
  }

  // Generate week-by-week plan
  const plan = [];
  let attendanceNeeded = deficit;
  let currentWeek = 1;

  while (attendanceNeeded > 0 && currentWeek <= weeksRemaining) {
    const sessionsThisWeek = Math.min(attendanceNeeded, sessionsPerWeek);
    plan.push({
      week: currentWeek,
      attend: sessionsThisWeek,
      skip: sessionsPerWeek - sessionsThisWeek,
    });

    attendanceNeeded -= sessionsThisWeek;
    currentWeek++;
  }

  const finalPercentage = calculatePercentage(
    subject.attended + deficit,
    totalSessions
  );

  return {
    possible: true,
    plan,
    finalPercentage,
    difficulty: deficit > sessionsPerWeek * 4 ? 'HARD' : deficit > sessionsPerWeek * 2 ?
'MEDIUM' : 'EASY',
  };
}
```

## 4. TESTING STRATEGY

### 4.1 Unit Tests: Vitest

Why Vitest?
- ✅ Vite-native (same config)
- ✅ Jest-compatible API (familiar)
- ✅ Fast (runs in Vite's dev server)

Installation:
```bash
npm install -D vitest @testing-library/react @testing-library/jest-dom
```

Example Test (src/utils/calculations.test.js):
```javascript
import { describe, it, expect } from 'vitest';
import {
  calculatePercentage,
  calculateBuffer,
  getStatus
} from './calculations';

describe('Attendance Calculations', () => {
  it('calculates percentage correctly', () => {
    expect(calculatePercentage(28, 41)).toBe(68.29);
    expect(calculatePercentage(0, 0)).toBe(0);
    expect(calculatePercentage(10, 10)).toBe(100);
  });

  it('calculates buffer correctly', () => {
    const subject = { attended: 60, conducted: 64 };
    const total = 80;
    // min_required = 80 × 0.75 = 60
    // buffer = (60 + 16) - 60 = 16
    expect(calculateBuffer(subject, total)).toBe(16);
  });

  it('determines status correctly', () => {
    expect(getStatus({ attended: 28, conducted: 41 }, 80, null)).toBe('CRITICAL');
```

    expect(getStatus({ attended: 62, conducted: 64 }, 80, null)).toBe('SAFE');
  });
});

Run Tests:
```bash
npm run test        # Run once
npm run test:watch  # Watch mode
npm run test:coverage # Coverage report
```

### 4.2 Component Tests: React Testing Library

Example (src/components/SubjectCard.test.jsx):
```jsx
import { render, screen } from '@testing-library/react';
import { describe, it, expect } from 'vitest';
import SubjectCard from './SubjectCard';

describe('SubjectCard', () => {
  it('renders critical status correctly', () => {
    const subject = {
      code: 'ML',
      name: 'Machine Learning',
      attended: 28,
      conducted: 41,
      status: 'CRITICAL',
      percentage: 68.2,
    };

    render(<SubjectCard subject={subject} />);

    expect(screen.getByText('Machine Learning')).toBeInTheDocument();
    expect(screen.getByText('68.2%')).toBeInTheDocument();
    expect(screen.getByText(/critical/i)).toBeInTheDocument();
  });
});
```

## 5. DEPLOYMENT & HOSTING

### 5.1 Hosting: Vercel (Recommended)

Why Vercel?
- ✅ Free for personal projects
- ✅ Automatic deployments from GitHub
- ✅ Global CDN
- ✅ HTTPS by default
- ✅ Perfect for React/Vite apps

Deployment Steps:
```bash
# 1. Push to GitHub
git add .
git commit -m "Initial commit"
git push origin main

# 2. Import to Vercel (via web UI)
# - Go to vercel.com
# - Click "Import Project"
# - Select GitHub repo
# - Framework: Vite
# - Build command: npm run build
# - Output directory: dist
# - Click "Deploy"

# 3. Custom domain (optional)
# - Add custom domain in Vercel dashboard
# - Update DNS records
```

Alternative: Netlify
```bash
# 1. Install Netlify CLI
npm install -g netlify-cli

# 2. Build
npm run build

# 3. Deploy
netlify deploy --prod --dir=dist
```

### 5.2 PWA Configuration

Make it installable on mobile:

public/manifest.json:
```json
```

```
{
  "name": "Attendance Planner",
  "short_name": "AttendancePlan",
  "description": "Smart attendance planning for students",
  "start_url": "/",
  "display": "standalone",
  "theme_color": "#3B82F6",
  "background_color": "#FFFFFF",
  "icons": [
    {
      "src": "/icons/icon-192.png
```
Attendance Planning & Decision Support System - Technology Architecture

Project Type: Progressive Web App (PWA)
Deployment: Static Hosting (No Backend Required for MVP)
Philosophy: Keep it simple, explainable, debuggable

## 1. TECHNOLOGY STACK OVERVIEW

### 1.1 Stack Summary

Layer	Technology	Rationale
Frontend Framework	React 18+ with Vite	Fast dev experience, component reusability, widely known
Styling	Tailwind CSS	Utility-first, rapid prototyping, small bundle size
State Management	Zustand	Simpler than Redux, perfect for this scale
Data Storage	LocalStorage + IndexedDB	No backend needed, offline-first
Date Handling	date-fns	Lightweight alternative to Moment.js
Charts/Visualization	Recharts	React-native charts, simple API
PDF Parsing	pdf.js (Mozilla)	Extract data from defaulter lists
Hosting	Vercel / Netlify	Free tier, automatic deployments
Version Control	Git + GitHub	Standard, required for collaboration

---

## 2. DETAILED TECHNOLOGY DECISIONS

### 2.1 Frontend Framework: React + Vite

Why React?
- ✅ Component-based (Subject Card, Dashboard are reusable)
- ✅ Large community (easy debugging)
- ✅ Hooks simplify state management
- ✅ Can be explained in viva (virtual DOM, component lifecycle)
- ✅ Familiar to most students

Why Vite over Create React App?
- ⚡ Faster dev server (instant HMR)
- 📦 Smaller build output
- 🔧 Better default config (no eject needed)
- 🚀 Modern tooling (ES modules)

Setup:
```bash
npm create vite@latest attendance-planner -- --template react
cd attendance-planner
npm install
```

Project Structure:

attendance-planner/ ├── public/ │ ├── icons/ │ └── manifest.json # PWA manifest ├── src/ │ ├── components/ │ │ ├── Dashboard.jsx │ │ ├── SubjectCard.jsx │ │ ├── SkipModal.jsx │ │ └── RecoveryPlan.jsx │ ├── utils/ │ │ ├── calculations.js # Core logic │ │ ├── dateHelpers.js │ │ └── validation.js │ ├── store/ │ │ └── attendanceStore.js # Zustand store │ ├── data/ │ │ ├── semester.json │ │ └── timetable.json │ ├── App.jsx │ ├── main.jsx │ └── index.css ├── package.json └── vite.config.js

---

### 2.2 Styling: Tailwind CSS

Why Tailwind?
- ✅ No context switching (HTML + CSS in one place)
- ✅ Utility classes prevent CSS bloat
- ✅ Responsive design built-in (`md:`, `lg:` prefixes)
- ✅ Customizable (theme colors match design doc)
- ✅ PurgeCSS removes unused styles (small bundle)

Installation:
```bash
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Tailwind Config (tailwind.config.js):
```javascript
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
```

  theme: {
    extend: {
      colors: {
        critical: {
          DEFAULT: '#EF4444',
          light: '#FEE2E2',
        },
        caution: {
          DEFAULT: '#F59E0B',
          light: '#FEF3C7',
        },
        safe: {
          DEFAULT: '#10B981',
          light: '#D1FAE5',
        },
      },
      fontFamily: {
        sans: ['Inter', 'sans-serif'],
        mono: ['JetBrains Mono', 'monospace'],
      },
    },
  },
  plugins: [],
}

Example Usage:
```jsx
<div className="bg-critical-light border-l-4 border-critical p-4 rounded-lg">
  <h3 className="text-xl font-semibold text-gray-900">
    Machine Learning
  </h3>
  <p className="text-critical font-mono text-3xl">68.2%</p>
</div>
```

### 2.3 State Management: Zustand

Why Zustand over Redux?
- ✅ 10× less boilerplate
- ✅ No providers/context needed
- ✅ Works great with React hooks
- ✅ Easier to explain in viva
- ✅ Perfect for this app's complexity level

Installation:

```bash
npm install zustand
```

Store Implementation (src/store/attendanceStore.js):
```javascript
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

const useAttendanceStore = create(
  persist(
    (set, get) => ({
      // State
      subjects: [],
      semester: null,
      lastUpdated: null,

      // Actions
      addSubject: (subject) => set((state) => ({
        subjects: [...state.subjects, subject],
        lastUpdated: new Date().toISOString(),
      })),

      updateAttendance: (subjectCode, attended, conducted) => set((state) => ({
        subjects: state.subjects.map((sub) =>
          sub.code === subjectCode
            ? { ...sub, attended, conducted }
            : sub
        ),
        lastUpdated: new Date().toISOString(),
      })),

      setSemester: (semesterData) => set({ semester: semesterData }),

      // Computed values (selectors)
      getSubjectStatus: (subjectCode) => {
        const subject = get().subjects.find(s => s.code === subjectCode);
        // Calculate buffer, deficit, PNR, etc.
        return calculateStatus(subject);
      },
    }),
    {
      name: 'attendance-storage', // LocalStorage key
      version: 1,
    }
  )
);
```

export default useAttendanceStore;

Usage in Components:
```jsx
import useAttendanceStore from './store/attendanceStore';

function Dashboard() {
  const subjects = useAttendanceStore((state) => state.subjects);
  const updateAttendance = useAttendanceStore((state) => state.updateAttendance);

  return (
    <div>
      {subjects.map(subject => (
        <SubjectCard key={subject.code} subject={subject} />
      ))}
    </div>
  );
}
```

### 2.4 Data Storage: LocalStorage + IndexedDB

Why No Backend?
- ✅ Faster development (no API to build)
- ✅ Works offline by default
- ✅ No hosting costs
- ✅ Privacy-friendly (data never leaves device)
- ✅ Perfect for single-user app

Storage Strategy:

LocalStorage (via Zustand persist):
- Small data (<5MB)
- User preferences, settings
- Current semester config
- Subject attendance data

IndexedDB (for larger data):
- Uploaded PDFs (defaulter lists)
- Historical semester data
- Exported reports

Implementation:
```javascript
// Wrapper for IndexedDB
```

```
import { openDB } from 'idb';

const DB_NAME = 'attendance-db';
const DB_VERSION = 1;

async function initDB() {
  return openDB(DB_NAME, DB_VERSION, {
    upgrade(db) {
      if (!db.objectStoreNames.contains('documents')) {
        db.createObjectStore('documents', { keyPath: 'id', autoIncrement: true });
      }
    },
  });
}

export async function saveDocument(file) {
  const db = await initDB();
  const arrayBuffer = await file.arrayBuffer();
  return db.add('documents', {
    name: file.name,
    type: file.type,
    data: arrayBuffer,
    uploadedAt: new Date().toISOString(),
  });
}

export async function getDocument(id) {
  const db = await initDB();
  return db.get('documents', id);
}
```

### 2.5 Date Handling: date-fns

Why date-fns?
- ✅ Lightweight (13KB vs Moment's 67KB)
- ✅ Tree-shakable (import only what you need)
- ✅ Immutable (no date mutation bugs)
- ✅ TypeScript support

Installation:
```bash
npm install date-fns
```

Usage Examples:

```javascript
import {
  parseISO,
  format,
  differenceInDays,
  addWeeks,
  isWithinInterval
} from 'date-fns';

// Calculate weeks elapsed
const semesterStart = parseISO('2026-01-19');
const today = new Date();
const weeksElapsed = Math.floor(differenceInDays(today, semesterStart) / 7);

// Format display dates
const displayDate = format(today, 'EEEE, d MMM yyyy'); // "Thursday, 23 Jan 2026"

// Check if date is a holiday
function isHoliday(date, holidays) {
  return holidays.some(holiday =>
    format(parseISO(holiday.date), 'yyyy-MM-dd') === format(date, 'yyyy-MM-dd')
  );
}

// Calculate PNR date
function calculatePNR(subject, semesterEnd) {
  const sessionsUntilPNR = calculateSessionsUntilPNR(subject);
  const weeksNeeded = Math.ceil(sessionsUntilPNR / subject.sessionsPerWeek);
  return addWeeks(new Date(), weeksNeeded);
}
```

### 2.6 Visualization: Recharts

Why Recharts?
- ✅ React-native (built for React)
- ✅ Responsive out of the box
- ✅ Simple API
- ✅ SVG-based (crisp on retina displays)

Installation:
```bash
npm install recharts
```

Usage Example (Buffer Visualization):

```jsx
import { BarChart, Bar, XAxis, YAxis, Tooltip, ResponsiveContainer } from 'recharts';

function BufferChart({ subjects }) {
  const data = subjects.map(sub => ({
    name: sub.code,
    buffer: calculateBuffer(sub),
  }));

  return (
    <ResponsiveContainer width="100%" height={200}>
      <BarChart data={data}>
        <XAxis dataKey="name" />
        <YAxis />
        <Tooltip />
        <Bar
          dataKey="buffer"
          fill={(data) => data.buffer < 2 ? '#EF4444' : '#10B981'}
        />
      </BarChart>
    </ResponsiveContainer>
  );
}
```

### 2.7 PDF Parsing: PDF.js

Why PDF.js?
- ✅ Mozilla-backed (reliable)
- ✅ Pure JavaScript (no native dependencies)
- ✅ Supports text extraction
- ✅ Works in browser

Installation:
```bash
npm install pdfjs-dist
```

Usage Example (Extract Defaulter List):
```javascript
import * as pdfjsLib from 'pdfjs-dist/build/pdf';

pdfjsLib.GlobalWorkerOptions.workerSrc =
`//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjsLib.version}/pdf.worker.min.js`;

async function parseDefaulterPDF(file) {
  const arrayBuffer = await file.arrayBuffer();
  const pdf = await pdfjsLib.getDocument({ data: arrayBuffer }).promise;

  let fullText = '';
  for (let i = 1; i <= pdf.numPages; i++) {
    const page = await pdf.getPage(i);
    const textContent = await page.getTextContent();
    const pageText = textContent.items.map(item => item.str).join(' ');
    fullText += pageText + '\n';
  }

  // Parse text to extract subjects and attendance
  return parseAttendanceText(fullText);
}

function parseAttendanceText(text) {
  // Example: "Machine Learning  Conducted: 45  Attended: 38"
  const regex = /([A-Za-z\s]+)\s+Conducted:\s*(\d+)\s+Attended:\s*(\d+)/g;
  const subjects = [];

  let match;
  while ((match = regex.exec(text)) !== null) {
    subjects.push({
      name: match[1].trim(),
      conducted: parseInt(match[2]),
      attended: parseInt(match[3]),
    });
  }

  return subjects;
}
```

## 3. CORE CALCULATION ENGINE

### 3.1 Pure Functions (src/utils/calculations.js)

Philosophy: All calculations must be:
- Pure (same input → same output)
- Testable (unit tests for every function)
- Explainable (comments show math)
```javascript
/**
 * Calculate current attendance percentage
 * Formula: (attended / conducted) × 100
 */
```

```
export function calculatePercentage(attended, conducted) {
  if (conducted === 0) return 0;
  return parseFloat(((attended / conducted) * 100).toFixed(2));
}

/**
 * Calculate attendance buffer
 * Buffer = classes you can still miss while staying ≥75%
 *
 * Formula:
 * min_required = total × 0.75
 * buffer = (attended + remaining) - min_required
 */
export function calculateBuffer(subject, totalSessions) {
  const { attended, conducted } = subject;
  const remaining = totalSessions - conducted;
  const minRequired = Math.ceil(totalSessions * 0.75);

  const buffer = (attended + remaining) - minRequired;
  return Math.max(0, Math.floor(buffer)); // Can't have negative buffer
}

/**
 * Calculate attendance deficit
 * Deficit = additional classes needed to reach 75%
 *
 * Formula:
 * min_required = total × 0.75
 * deficit = min_required - attended
 */
export function calculateDeficit(subject, totalSessions) {
  const { attended } = subject;
  const minRequired = Math.ceil(totalSessions * 0.75);

  const deficit = minRequired - attended;
  return Math.max(0, Math.ceil(deficit)); // Can't have negative deficit
}

/**
 * Calculate Point of No Return date
 * PNR = date when (deficit > remaining sessions)
 *
 * Returns: Date object or null
 */
export function calculatePNR(subject, totalSessions, semesterEnd, sessionsPerWeek) {
  const { attended, conducted } = subject;
  const remaining = totalSessions - conducted;
  const deficit = calculateDeficit(subject, totalSessions);
  // Already impossible
  if (deficit > remaining) {
    return new Date(0); // "Already passed"
  }

  // No PNR (safe)
  if (deficit === 0) {
    return null;
  }

  // Calculate date
  const sessionsUntilPNR = remaining - deficit;
  const weeksUntilPNR = Math.floor(sessionsUntilPNR / sessionsPerWeek);

  return addWeeks(new Date(), weeksUntilPNR);
}

/**
 * Determine subject status
 * Returns: "CRITICAL" | "CAUTION" | "SAFE"
 */
export function getStatus(subject, totalSessions, pnrDate) {
  const percentage = calculatePercentage(subject.attended, subject.conducted);
  const buffer = calculateBuffer(subject, totalSessions);

  // Critical: Below 75% OR buffer ≤1
  if (percentage < 75 || buffer <= 1) {
    return 'CRITICAL';
  }

  // Caution: Buffer 2-4 OR PNR within 14 days
  const daysUntilPNR = pnrDate ? differenceInDays(pnrDate, new Date()) : Infinity;
  if (buffer <= 4 || daysUntilPNR <= 14) {
    return 'CAUTION';
  }

  // Safe: Buffer ≥5
  return 'SAFE';
}

/**
 * Generate recovery plan
 * Returns: { possible: boolean, plan: Array, finalPercentage: number }
 */
export function generateRecoveryPlan(subject, totalSessions, sessionsPerWeek,
weeksRemaining) {
  const deficit = calculateDeficit(subject, totalSessions);
  const remaining = totalSessions - subject.conducted;

  // Impossible case
  if (deficit > remaining) {
    const maxAchievable = calculatePercentage(
      subject.attended + remaining,
      totalSessions
    );

    return {
      possible: false,
      maxPercentage: maxAchievable,
      message: 'Recovery is mathematically impossible',
    };
  }

  // Generate week-by-week plan
  const plan = [];
  let attendanceNeeded = deficit;
  let currentWeek = 1;

  while (attendanceNeeded > 0 && currentWeek <= weeksRemaining) {
    const sessionsThisWeek = Math.min(attendanceNeeded, sessionsPerWeek);
    plan.push({
      week: currentWeek,
      attend: sessionsThisWeek,
      skip: sessionsPerWeek - sessionsThisWeek,
    });

    attendanceNeeded -= sessionsThisWeek;
    currentWeek++;
  }

  const finalPercentage = calculatePercentage(
    subject.attended + deficit,
    totalSessions
  );

  return {
    possible: true,
    plan,
    finalPercentage,
    difficulty: deficit > sessionsPerWeek * 4 ? 'HARD' : deficit > sessionsPerWeek * 2 ?
'MEDIUM' : 'EASY',
  };
}
```

## 4. TESTING STRATEGY

### 4.1 Unit Tests: Vitest

Why Vitest?
- ✅ Vite-native (same config)
- ✅ Jest-compatible API (familiar)
- ✅ Fast (runs in Vite's dev server)

Installation:
```bash
npm install -D vitest @testing-library/react @testing-library/jest-dom
```

Example Test (src/utils/calculations.test.js):
```javascript
import { describe, it, expect } from 'vitest';
import {
  calculatePercentage,
  calculateBuffer,
  getStatus
} from './calculations';

describe('Attendance Calculations', () => {
  it('calculates percentage correctly', () => {
    expect(calculatePercentage(28, 41)).toBe(68.29);
    expect(calculatePercentage(0, 0)).toBe(0);
    expect(calculatePercentage(10, 10)).toBe(100);
  });

  it('calculates buffer correctly', () => {
    const subject = { attended: 60, conducted: 64 };
    const total = 80;
    // min_required = 80 × 0.75 = 60
    // buffer = (60 + 16) - 60 = 16
    expect(calculateBuffer(subject, total)).toBe(16);
  });

  it('determines status correctly', () => {
    expect(getStatus({ attended: 28, conducted: 41 }, 80, null)).toBe('CRITICAL');
    expect(getStatus({ attended: 62, conducted: 64 }, 80, null)).toBe('SAFE');
  });
});
```

Run Tests:

```bash
npm run test          # Run once
npm run test:watch  # Watch mode
npm run test:coverage # Coverage report
```

### 4.2 Component Tests: React Testing Library

Example (src/components/SubjectCard.test.jsx):
```jsx
import { render, screen } from '@testing-library/react';
import { describe, it, expect } from 'vitest';
import SubjectCard from './SubjectCard';

describe('SubjectCard', () => {
  it('renders critical status correctly', () => {
    const subject = {
      code: 'ML',
      name: 'Machine Learning',
      attended: 28,
      conducted: 41,
      status: 'CRITICAL',
      percentage: 68.2,
    };

    render(<SubjectCard subject={subject} />);

    expect(screen.getByText('Machine Learning')).toBeInTheDocument();
    expect(screen.getByText('68.2%')).toBeInTheDocument();
    expect(screen.getByText(/critical/i)).toBeInTheDocument();
  });
});
```

## 5. DEPLOYMENT & HOSTING

### 5.1 Hosting: Vercel (Recommended)

Why Vercel?
- ✅ Free for personal projects
- ✅ Automatic deployments from GitHub
- ✅ Global CDN
- ✅ HTTPS by default
- ✅ Perfect for React/Vite apps

Deployment Steps:
```bash
# 1. Push to GitHub
git add .
git commit -m "Initial commit"
git push origin main

# 2. Import to Vercel (via web UI)
# - Go to vercel.com
# - Click "Import Project"
# - Select GitHub repo
# - Framework: Vite
# - Build command: npm run build
# - Output directory: dist
# - Click "Deploy"

# 3. Custom domain (optional)
# - Add custom domain in Vercel dashboard
# - Update DNS records
```

Alternative: Netlify
```bash
# 1. Install Netlify CLI
npm install -g netlify-cli

# 2. Build
npm run build

# 3. Deploy
netlify deploy --prod --dir=dist
```

### 5.2 PWA Configuration

Make it installable on mobile:

public/manifest.json:
```json
{
  "name": "Attendance Planner",
  "short_name": "AttendancePlan",
  "description": "Smart attendance planning for students",
  "start_url": "/",
  "display": "standalone",
  "theme_color": "#3B82F6",
  "background_color": "#FFFFFF",
  "icons": [
    {
      "src": "/icons/icon-192.png
```

Attendance Planning & Decision Support System - Technology Architecture

Project Type: Progressive Web App (PWA)
Deployment: Static Hosting (No Backend Required for MVP)
Philosophy: Keep it simple, explainable, debuggable

---

1. TECHNOLOGY STACK OVERVIEW

1.1 Stack Summary

Layer	Technology	Rationale
Frontend Framework	React 18+ with Vite	Fast dev experience, component reusability, widely known
Styling	Tailwind CSS	Utility-first, rapid prototyping, small bundle size
State Management	Zustand	Simpler than Redux, perfect for this scale
Data Storage	LocalStorage + IndexedDB	No backend needed, offline-first
Date Handling	date-fns	Lightweight alternative to Moment.js
Charts/Visualization	Recharts	React-native charts, simple API
PDF Parsing	pdf.js (Mozilla)	Extract data from defaulter lists
Hosting	Vercel / Netlify	Free tier, automatic deployments
Version Control	Git + GitHub	Standard, required for collaboration

---

2. DETAILED TECHNOLOGY DECISIONS

2.1 Frontend Framework: React + Vite

Why React?

- ✅ Component-based (Subject Card, Dashboard are reusable)
- ✅ Large community (easy debugging)
- ✅ Hooks simplify state management
- ✅ Can be explained in viva (virtual DOM, component lifecycle)
- ✅ Familiar to most students

Why Vite over Create React App?

- ⚡ Faster dev server (instant HMR)
- 📦 Smaller build output
- 🔧 Better default config (no eject needed)
- 🚀 Modern tooling (ES modules)

Setup:

npm create vite@latest attendance-planner -- --template react
cd attendance-planner
npm install

Project Structure: attendance-planner/ ├── public/ │ ├── icons/ │ └── manifest.json # PWA manifest ├── src/ │ ├── components/ │ │ ├── Dashboard.jsx │ │ ├── SubjectCard.jsx │ │ ├── SkipModal.jsx │ │ └── RecoveryPlan.jsx │ ├── utils/ │ │ ├── calculations.js # Core logic │ │ ├── dateHelpers.js │ │ └── validation.js │ ├── store/ │ │ └── attendanceStore.js # Zustand store │ ├── data/ │ │ ├── semester.json │ │ └── timetable.json │ ├── App.jsx │ ├── main.jsx │ └── index.css ├── package.json └── vite.config.js

---

2.2 Styling: Tailwind CSS

Why Tailwind?

- ✅ No context switching (HTML + CSS in one place)
- ✅ Utility classes prevent CSS bloat
- ✅ Responsive design built-in (`md:`, `lg:` prefixes)
- ✅ Customizable (theme colors match design doc)
- ✅ PurgeCSS removes unused styles (small bundle)

Installation:

npm install -D tailwindcss postcss autoprefixer

```
npx tailwindcss init -p
```

Tailwind Config (tailwind.config.js):

```
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {
      colors: {
        critical: {
          DEFAULT: '#EF4444',
          light: '#FEE2E2',
        },
        caution: {
          DEFAULT: '#F59E0B',
          light: '#FEF3C7',
        },
        safe: {
          DEFAULT: '#10B981',
          light: '#D1FAE5',
        },
      },
      fontFamily: {
        sans: ['Inter', 'sans-serif'],
        mono: ['JetBrains Mono', 'monospace'],
      },
    },
  },
  plugins: [],
}
```

Example Usage:

```
<div className="bg-critical-light border-l-4 border-critical p-4 rounded-lg">
  <h3 className="text-xl font-semibold text-gray-900">
    Machine Learning
  </h3>
  <p className="text-critical font-mono text-3xl">68.2%</p>
</div>
```

2.3 State Management: Zustand

Why Zustand over Redux?

- ✅ 10× less boilerplate
- ✅ No providers/context needed
- ✅ Works great with React hooks
- ✅ Easier to explain in viva
- ✅ Perfect for this app's complexity level

Installation:

```
npm install zustand
```

Store Implementation (src/store/attendanceStore.js):

```
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

const useAttendanceStore = create(
  persist(
    (set, get) => ({
      // State
      subjects: [],
      semester: null,
      lastUpdated: null,

      // Actions
      addSubject: (subject) => set((state) => ({
        subjects: [...state.subjects, subject],
        lastUpdated: new Date().toISOString(),
      })),

      updateAttendance: (subjectCode, attended, conducted) => set((state) => ({
        subjects: state.subjects.map((sub) =>
          sub.code === subjectCode
            ? { ...sub, attended, conducted }
            : sub
        ),
        lastUpdated: new Date().toISOString(),
      })),

      setSemester: (semesterData) => set({ semester: semesterData }),

      // Computed values (selectors)
      getSubjectStatus: (subjectCode) => {
        const subject = get().subjects.find(s => s.code === subjectCode);
        // Calculate buffer, deficit, PNR, etc.
        return calculateStatus(subject);
      },
    }),
    {
      name: 'attendance-storage', // LocalStorage key
      version: 1,
    }
  )
);

export default useAttendanceStore;
```

Usage in Components:

```
import useAttendanceStore from './store/attendanceStore';

function Dashboard() {
  const subjects = useAttendanceStore((state) => state.subjects);
  const updateAttendance = useAttendanceStore((state) => state.updateAttendance);

  return (
    <div>
      {subjects.map(subject => (
        <SubjectCard key={subject.code} subject={subject} />
      ))}
    </div>
  );
}
```

2.4 Data Storage: LocalStorage + IndexedDB

Why No Backend?

- ✅ Faster development (no API to build)
- ✅ Works offline by default
- ✅ No hosting costs
- ✅ Privacy-friendly (data never leaves device)
- ✅ Perfect for single-user app

Storage Strategy:

LocalStorage (via Zustand persist):

- Small data (<5MB)
- User preferences, settings
- Current semester config
- Subject attendance data

IndexedDB (for larger data):

- Uploaded PDFs (defaulter lists)
- Historical semester data
- Exported reports

Implementation:

```
// Wrapper for IndexedDB
import { openDB } from 'idb';

const DB_NAME = 'attendance-db';
const DB_VERSION = 1;

async function initDB() {
  return openDB(DB_NAME, DB_VERSION, {
    upgrade(db) {
      if (!db.objectStoreNames.contains('documents')) {
        db.createObjectStore('documents', { keyPath: 'id', autoIncrement: true });
      }
    },
  });
}

export async function saveDocument(file) {
  const db = await initDB();
  const arrayBuffer = await file.arrayBuffer();
  return db.add('documents', {
    name: file.name,
    type: file.type,
    data: arrayBuffer,
    uploadedAt: new Date().toISOString(),
  });
}

export async function getDocument(id) {
  const db = await initDB();
  return db.get('documents', id);
}
```

2.5 Date Handling: date-fns

Why date-fns?

- ✅ Lightweight (13KB vs Moment's 67KB)
- ✅ Tree-shakable (import only what you need)

- ✅ Immutable (no date mutation bugs)
- ✅ TypeScript support

Installation:

```
npm install date-fns
```

Usage Examples:

```
import {
  parseISO,
  format,
  differenceInDays,
  addWeeks,
  isWithinInterval
} from 'date-fns';

// Calculate weeks elapsed
const semesterStart = parseISO('2026-01-19');
const today = new Date();
const weeksElapsed = Math.floor(differenceInDays(today, semesterStart) / 7);

// Format display dates
const displayDate = format(today, 'EEEE, d MMM yyyy'); // "Thursday, 23 Jan 2026"

// Check if date is a holiday
function isHoliday(date, holidays) {
  return holidays.some(holiday =>
    format(parseISO(holiday.date), 'yyyy-MM-dd') === format(date, 'yyyy-MM-dd')
  );
}

// Calculate PNR date
function calculatePNR(subject, semesterEnd) {
  const sessionsUntilPNR = calculateSessionsUntilPNR(subject);
  const weeksNeeded = Math.ceil(sessionsUntilPNR / subject.sessionsPerWeek);
  return addWeeks(new Date(), weeksNeeded);
}
```

2.6 Visualization: Recharts

Why Recharts?

- ✅ React-native (built for React)
- ✅ Responsive out of the box
- ✅ Simple API

- ✅ SVG-based (crisp on retina displays)

Installation:

```
npm install recharts
```

Usage Example (Buffer Visualization):

```
import { BarChart, Bar, XAxis, YAxis, Tooltip, ResponsiveContainer } from 'recharts';

function BufferChart({ subjects }) {
  const data = subjects.map(sub => ({
    name: sub.code,
    buffer: calculateBuffer(sub),
  }));

  return (
    <ResponsiveContainer width="100%" height={200}>
      <BarChart data={data}>
        <XAxis dataKey="name" />
        <YAxis />
        <Tooltip />
        <Bar
          dataKey="buffer"
          fill={(data) => data.buffer < 2 ? '#EF4444' : '#10B981'}
        />
      </BarChart>
    </ResponsiveContainer>
  );
}
```

2.7 PDF Parsing: PDF.js

Why PDF.js?

- ✅ Mozilla-backed (reliable)
- ✅ Pure JavaScript (no native dependencies)
- ✅ Supports text extraction
- ✅ Works in browser

Installation:

```
npm install pdfjs-dist
```

Usage Example (Extract Defaulter List):

```
import * as pdfjsLib from 'pdfjs-dist/build/pdf';

pdfjsLib.GlobalWorkerOptions.workerSrc =
`//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjsLib.version}/pdf.worker.min.js`;

async function parseDefaulterPDF(file) {
  const arrayBuffer = await file.arrayBuffer();
  const pdf = await pdfjsLib.getDocument({ data: arrayBuffer }).promise;

  let fullText = '';
  for (let i = 1; i <= pdf.numPages; i++) {
    const page = await pdf.getPage(i);
    const textContent = await page.getTextContent();
    const pageText = textContent.items.map(item => item.str).join(' ');
    fullText += pageText + '\n';
  }

  // Parse text to extract subjects and attendance
  return parseAttendanceText(fullText);
}

function parseAttendanceText(text) {
  // Example: "Machine Learning  Conducted: 45  Attended: 38"
  const regex = /([A-Za-z\s]+)\s+Conducted:\s*(\d+)\s+Attended:\s*(\d+)/g;
  const subjects = [];

  let match;
  while ((match = regex.exec(text)) !== null) {
    subjects.push({
      name: match[1].trim(),
      conducted: parseInt(match[2]),
      attended: parseInt(match[3]),
    });
  }

  return subjects;
}
```

---

3. CORE CALCULATION ENGINE

3.1 Pure Functions (src/utils/calculations.js)

Philosophy: All calculations must be:

- Pure (same input → same output)

- Testable (unit tests for every function)
- Explainable (comments show math)

```
/**
 * Calculate current attendance percentage
 * Formula: (attended / conducted) × 100
 */
export function calculatePercentage(attended, conducted) {
  if (conducted === 0) return 0;
  return parseFloat(((attended / conducted) * 100).toFixed(2));
}

/**
 * Calculate attendance buffer
 * Buffer = classes you can still miss while staying ≥75%
 *
 * Formula:
 * min_required = total × 0.75
 * buffer = (attended + remaining) - min_required
 */
export function calculateBuffer(subject, totalSessions) {
  const { attended, conducted } = subject;
  const remaining = totalSessions - conducted;
  const minRequired = Math.ceil(totalSessions * 0.75);

  const buffer = (attended + remaining) - minRequired;
  return Math.max(0, Math.floor(buffer)); // Can't have negative buffer
}

/**
 * Calculate attendance deficit
 * Deficit = additional classes needed to reach 75%
 *
 * Formula:
 * min_required = total × 0.75
 * deficit = min_required - attended
 */
export function calculateDeficit(subject, totalSessions) {
  const { attended } = subject;
  const minRequired = Math.ceil(totalSessions * 0.75);

  const deficit = minRequired - attended;
  return Math.max(0, Math.ceil(deficit)); // Can't have negative deficit
}

/**
 * Calculate Point of No Return date
 * PNR = date when (deficit > remaining sessions)
 *
 * Returns: Date object or null
 */
export function calculatePNR(subject, totalSessions, semesterEnd, sessionsPerWeek) {
  const { attended, conducted } = subject;
  const remaining = totalSessions - conducted;
  const deficit = calculateDeficit(subject, totalSessions);

  // Already impossible
  if (deficit > remaining) {
    return new Date(0); // "Already passed"
  }

  // No PNR (safe)
  if (deficit === 0) {
    return null;
  }

  // Calculate date
  const sessionsUntilPNR = remaining - deficit;
  const weeksUntilPNR = Math.floor(sessionsUntilPNR / sessionsPerWeek);

  return addWeeks(new Date(), weeksUntilPNR);
}

/**
 * Determine subject status
 * Returns: "CRITICAL" | "CAUTION" | "SAFE"
 */
export function getStatus(subject, totalSessions, pnrDate) {
  const percentage = calculatePercentage(subject.attended, subject.conducted);
  const buffer = calculateBuffer(subject, totalSessions);

  // Critical: Below 75% OR buffer ≤1
  if (percentage < 75 || buffer <= 1) {
    return 'CRITICAL';
  }

  // Caution: Buffer 2-4 OR PNR within 14 days
  const daysUntilPNR = pnrDate ? differenceInDays(pnrDate, new Date()) : Infinity;
  if (buffer <= 4 || daysUntilPNR <= 14) {
    return 'CAUTION';
  }

  // Safe: Buffer ≥5
  return 'SAFE';
}
```

```
/**
 * Generate recovery plan
 * Returns: { possible: boolean, plan: Array, finalPercentage: number }
 */
export function generateRecoveryPlan(subject, totalSessions, sessionsPerWeek,
weeksRemaining) {
  const deficit = calculateDeficit(subject, totalSessions);
  const remaining = totalSessions - subject.conducted;

  // Impossible case
  if (deficit > remaining) {
    const maxAchievable = calculatePercentage(
      subject.attended + remaining,
      totalSessions
    );

    return {
      possible: false,
      maxPercentage: maxAchievable,
      message: 'Recovery is mathematically impossible',
    };
  }

  // Generate week-by-week plan
  const plan = [];
  let attendanceNeeded = deficit;
  let currentWeek = 1;

  while (attendanceNeeded > 0 && currentWeek <= weeksRemaining) {
    const sessionsThisWeek = Math.min(attendanceNeeded, sessionsPerWeek);
    plan.push({
      week: currentWeek,
      attend: sessionsThisWeek,
      skip: sessionsPerWeek - sessionsThisWeek,
    });

    attendanceNeeded -= sessionsThisWeek;
    currentWeek++;
  }

  const finalPercentage = calculatePercentage(
    subject.attended + deficit,
    totalSessions
  );

  return {
    possible: true,
    plan,
    finalPercentage,
    difficulty: deficit > sessionsPerWeek * 4 ? 'HARD' : deficit > sessionsPerWeek * 2 ?
'MEDIUM' : 'EASY',
  };
}
```

---

4. TESTING STRATEGY

4.1 Unit Tests: Vitest

Why Vitest?

- ✅ Vite-native (same config)
- ✅ Jest-compatible API (familiar)
- ✅ Fast (runs in Vite's dev server)

Installation:

```
npm install -D vitest @testing-library/react @testing-library/jest-dom
```

Example Test (src/utils/calculations.test.js):

```
import { describe, it, expect } from 'vitest';
import {
  calculatePercentage,
  calculateBuffer,
  getStatus
} from './calculations';

describe('Attendance Calculations', () => {
  it('calculates percentage correctly', () => {
    expect(calculatePercentage(28, 41)).toBe(68.29);
    expect(calculatePercentage(0, 0)).toBe(0);
    expect(calculatePercentage(10, 10)).toBe(100);
  });

  it('calculates buffer correctly', () => {
    const subject = { attended: 60, conducted: 64 };
    const total = 80;
    // min_required = 80 × 0.75 = 60
    // buffer = (60 + 16) - 60 = 16
    expect(calculateBuffer(subject, total)).toBe(16);
  });

  it('determines status correctly', () => {
    expect(getStatus({ attended: 28, conducted: 41 }, 80, null)).toBe('CRITICAL');
    expect(getStatus({ attended: 62, conducted: 64 }, 80, null)).toBe('SAFE');
  });
});
```

Run Tests:

```
npm run test        # Run once
npm run test:watch  # Watch mode
npm run test:coverage # Coverage report
```

4.2 Component Tests: React Testing Library

Example (src/components/SubjectCard.test.jsx):

```
import { render, screen } from '@testing-library/react';
import { describe, it, expect } from 'vitest';
import SubjectCard from './SubjectCard';

describe('SubjectCard', () => {
  it('renders critical status correctly', () => {
    const subject = {
      code: 'ML',
      name: 'Machine Learning',
      attended: 28,
      conducted: 41,
      status: 'CRITICAL',
      percentage: 68.2,
    };

    render(<SubjectCard subject={subject} />);

    expect(screen.getByText('Machine Learning')).toBeInTheDocument();
    expect(screen.getByText('68.2%')).toBeInTheDocument();
    expect(screen.getByText(/critical/i)).toBeInTheDocument();
  });
});
```

---

5. DEPLOYMENT & HOSTING

5.1 Hosting: Vercel (Recommended)

Why Vercel?

- ✅ Free for personal projects
- ✅ Automatic deployments from GitHub
- ✅ Global CDN
- ✅ HTTPS by default
- ✅ Perfect for React/Vite apps

Deployment Steps:

```
# 1. Push to GitHub
git add .
git commit -m "Initial commit"
git push origin main

# 2. Import to Vercel (via web UI)
# - Go to vercel.com
# - Click "Import Project"
# - Select GitHub repo
# - Framework: Vite
# - Build command: npm run build
# - Output directory: dist
# - Click "Deploy"

# 3. Custom domain (optional)
# - Add custom domain in Vercel dashboard
# - Update DNS records
```

Alternative: Netlify

```
# 1. Install Netlify CLI
npm install -g netlify-cli

# 2. Build
npm run build

# 3. Deploy
netlify deploy --prod --dir=dist
```

5.2 PWA Configuration

Make it installable on mobile:

public/manifest.json:

```
{
```

```
"name": "Attendance Planner",
"short_name": "AttendancePlan",
"description": "Smart attendance planning for students",
"start_url": "/",
"display": "standalone",
"theme_color": "#3B82F6",
"background_color": "#FFFFFF",
"icons": [
  {
    "src": "/icons/icon-192.png
```

Updated Sections: **Project Setup (Updated Commands)**

```
# Create project
npm create vite@latest 75guard -- --template react
cd 75guard
npm install

# Install dependencies
npm install zustand date-fns recharts pdfjs-dist idb
npm install -D tailwindcss postcss autoprefixer vitest
```

### Project Structure (Updated)

```
75guard/
├── public/
│   ├── icons/
│   │   ├── 75guard-icon-192.png
│   │   ├── 75guard-icon-512.png
│   │   └── 75guard-favicon.svg
│   ├── manifest.json
│   └── sw.js
├── src/
│   ├── components/
│   │   ├── Dashboard.jsx
│   │   ├── SubjectCard.jsx
│   │   ├── SkipModal.jsx
│   │   ├── RecoveryPlan.jsx
│   │   └── Header.jsx
│   ├── utils/
│   │   ├── calculations.js
│   │   ├── dateHelpers.js
│   │   └── validation.js
│   ├── store/
│   │   └── attendanceStore.js
│   ├── data/
│   │   ├── semester.json
│   │   └── timetable.json
│   ├── assets/
│   │   ├── logo.svg
│   │   └── shield-icon.svg
│   ├── App.jsx
│   ├── main.jsx
│   └── index.css
├── package.json
├── vite.config.js
└── README.md
```

Updated package.json

```
{
  "name": "75guard",
  "version": "1.0.0",
  "description": "Smart attendance planning for students - Guard your 75%",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "test": "vitest",
    "test:coverage": "vitest --coverage",
    "lint": "eslint src --ext js,jsx",
    "format": "prettier --write \"src/**/*.{js,jsx,json,css}\""
  },
  "keywords": [
    "attendance",
    "student",
    "planning",
    "decision-support",
    "75-percent"
  ],
  "author": "TE DS Batch 2023-27",
  "license": "MIT",
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "zustand": "^4.4.7",
    "date-fns": "^3.0.6",
    "recharts": "^2.10.3",
    "pdfjs-dist": "^4.0.269",
    "idb": "^8.0.0"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.2.1",
    "vite": "^5.0.8",
    "tailwindcss": "^3.4.0",
    "postcss": "^8.4.32",
    "autoprefixer": "^10.4.16",
    "vitest": "^1.1.0",
    "@testing-library/react": "^14.1.2",
    "@testing-library/jest-dom": "^6.1.5",
    "eslint": "^8.56.0",
    "prettier": "^3.1.1"
  }
}
```

Updated manifest.json:

```
{
  "name": "75Guard - Attendance Planner",
  "short_name": "75Guard",
  "description": "Guard Your Attendance. Plan Your Freedom.",
  "start_url": "/",
  "display": "standalone",
  "theme_color": "#3B82F6",
  "background_color": "#FFFFFF",
  "orientation": "portrait",
  "categories": ["education", "productivity"],
  "icons": [
    {
      "src": "/icons/75guard-icon-192.png",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/icons/75guard-icon-512.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ]
}
```

Updated LocalStorage Key:

```
// In attendanceStore.js
const useAttendanceStore = create(
  persist(
    (set, get) => ({
      // ... state and actions
    }),
    {
      name: '75guard-storage', // Updated key
      version: 1,
    }
  )
);
```

Updated IndexedDB Name

javascript
```
const DB_NAME = '75guard-db'; // Updated database name

const DB_VERSION = 1;
```

Updated Service Worker Cache javascript // public/sw.js self.addEventListener('install', (event) => { event.waitUntil(caches.open('75guard-v1').then((cache) => { // Updated cache name return cache.addAll(['/', '/index.html', '/src/main.jsx', '/src/App.jsx',]); })); }); Updated README.md markdown

🛡️ 75Guard

Guard Your Attendance. Plan Your Freedom.

A smart attendance planning and decision support system for college students.

What is 75Guard?

75Guard helps students make informed decisions about class attendance by answering critical questions:

- "Can I skip today's class safely?"
- "When does recovery become mathematically impossible?"
- "Which subject needs urgent attention?"

Unlike traditional attendance trackers that only show percentages, 75Guard provides forward-looking insights and recovery plans.

Features

- 🚦 **Traffic Light Dashboard** - Instant visual status of all subjects
- 🎯 **Skip Impact Simulator** - See consequences before skipping
- ⏰ **Point of No Return Alerts** - Know your deadlines
- 📊 **Recovery Path Generator** - Concrete plans to reach 75%
- 📱 **Mobile-First PWA** - Works offline, installable

Tech Stack

- React 18 + Vite
- Tailwind CSS
- Zustand (State Management)
- date-fns (Date handling)
- Recharts (Visualizations)
- LocalStorage + IndexedDB (Offline-first)

Getting Started

```
# Clone repository
git clone https://github.com/yourusername/75guard.git
cd 75guard

# Install dependencies
npm install

# Run development server
npm run dev

# Build for production
npm run build
```

Project Structure

```
75guard/
├── src/
│   ├── components/      # React components
│   ├── utils/           # Calculation engine
│   ├── store/           # State management
│   └── App.jsx          # Main app
├── public/
│   ├── icons/           # PWA icons
│   └── manifest.json    # PWA config
└── package.json
```

Testing

```
npm run test           # Run tests
npm run test:coverage    # Coverage report
```

Deployment

Deployed on Vercel: 75guard.vercel.app

Mini-Project Details

- **Course:** TE Data Science
- **Semester:** 6th (Even Sem 2025-26)
- **Team:** [Your Names]
- **Guide:** [Guide Name]

License

MIT License - Built for educational purposes

Made with ❤️ by TE DS Students