Software Design Document - Use Case 3: Mood Insight

1. System Overview

IdeaPulse is a web-based Progressive Web Application (PWA) that allows users to capture, reflect, and analyze their creative ideas. This document outlines the third core use case: **Mood Insight**.

The Mood Insight feature enables users to visualize and understand the emotional patterns behind their saved ideas. By tagging each idea with a mood, users can track which emotions dominate their creative flow and how those emotions evolve over time.

2. System Context

This feature operates entirely within the application, without relying on external systems or APIs. The architecture remains consistent with the core application:

- **User**: The only actor, responsible for tagging ideas with moods.
- **Browser**: The platform executing the app and rendering visual feedback.
- **localStorage**: Used to persist both the idea content and associated mood tags.

3. Key Functions and Functionalities

The Mood Insight use case includes the following core functionalities:

- **Mood Tagging**: Users assign a mood label (e.g., "Happy", "Sad", "Excited", "Stressed") when saving a new idea.
- **Data Aggregation**: The system parses all saved ideas from `localStorage`, extracting and counting the occurrence of each mood.

- **Visual Feedback**: Mood frequencies are displayed using pie charts or bar charts to give users a clear visual summary of their emotional distribution.
- **Dynamic Updates**: Whenever a new idea is added, deleted, or edited, the mood chart updates accordingly to reflect the latest data.
- **Navigation Integration**: Mood Insight is accessible from the main menu or sidebar.

4. Assumptions and Dependencies

Assumptions:

- Users consistently tag their ideas with valid mood labels.
- The browser supports HTML Canvas or SVG for rendering charts.
- Users are on a modern browser that supports ES6 and localStorage.

Dependencies:

- **Next.js / React**: UI framework and rendering logic.
- **Typescript**: Type safety and development stability.
- **Tailwind CSS**: Used for consistent layout and styling.
- **uuid**: Generates unique identifiers for mood-tracked ideas.
- **Next-pwa**: Provides offline support and persistent data access.
- **Chart.js / Recharts** (or similar): Library for rendering mood charts.

5. Architectural Design

The Mood Insight module is a client-side feature. It reads from the localStorage on component mount, processes mood data using a reducer or aggregation function, and passes the result to a chart-rendering component. All operations are performed client-side for speed and offline compatibility.

6. Component Design

1. Stats Component (`Stats.tsx`)

- Main component for mood insight and analysis.
- Displays mood distribution using a grid layout.
- Each mood is shown as a row with a label, a colored progress bar, and a percentage.
- Used for visual feedback and emotional pattern recognition.

2. Sidebar Component (`Sidebar.tsx`)

- Enables filtering ideas by mood.
- Lists available moods along with the number of ideas tagged with each.
- Users can click on a mood to filter the idea list accordingly.

3. MoodSelector Component (`MoodSelector.tsx`)

- Used when creating or editing an idea.
- Provides selectable mood options, each with an emoji, name, and description.
- Highlights the currently selected mood.

4. IdeaList Component (`IdeaList.tsx`)

- **Displays all saved ideas in a list.**
- **Shows the selected mood as an emoji next to each idea entry.**

Summary

- **Stats: Core for mood insights, visualizes mood data with bars and percentages.**
- **Sidebar: Mood-based filter with counts for navigation.**
- **MoodSelector: Mood selection interface for user input.**
- **IdeaList: Displays idea entries with mood indicators.**

7. Data Design

Ideas are stored in localStorage with the following shape:

- export interface Idea {
-     id: string;
-     content: string;
-     mood: 'inspired' | 'neutral' | 'tired' | 'excited';
-     tags: string[];
-     timestamp: string; // ISO format
- }

Mood Insight operates by grouping the `mood` field and counting occurrences.

8. Design Patterns

- **Strategy Pattern**:
  The Strategy Pattern is used to support multiple visualization styles (e.g., pie chart, bar chart) for displaying mood data.
  Each chart type is implemented as a separate strategy class that adheres to a common interface.
  This allows the MoodInsightPage component to remain decoupled from the specific rendering logic of each chart type. The appropriate strategy is selected dynamically based on user preferences or context.

- **Presentational/Container Pattern (React idiom)**

  What it is: Separates UI (presentational) from logic/state (container).

  How it's used in IdeaList.tsx:

  `IdeaList` is a presentational component: it receives all data (`ideas`) and handlers (`onDelete`, `onEdit`, `onTagClick`) as props.

  It does not fetch or own the main data; it just renders and delegates actions upward.

9. Implementation Notes

- Ensure the mood field is always present for each idea; fallback to "Unspecified" if absent.
- Consider performance optimizations for large datasets by memoizing aggregation results.
- For accessibility, charts should include alternative textual descriptions.

10. User Interface Design

- **Navigation**: “Mood Insights” is accessible via a sidebar or top menu.
- **Layout**: Includes a title, chart selector (e.g., pie/bar), and the chart itself.
- **Responsiveness**: Fully responsive on both desktop and mobile views.
- **Colors**: Each mood is assigned a distinct color for better visual separation.