# Chapter 3: System Analysis

## 3.1 Study of Current System and Requirement of this System

### 3.1.1 Analysis of Existing Mood Tracking Solutions

Before developing the Pro Mood Tracker application, a comprehensive analysis of existing mood tracking solutions was conducted to identify strengths, weaknesses, and opportunities for improvement. The following table summarizes the findings from this analysis:

| Application | Key Features | Strengths | Limitations | Insights for Pro Mood Tracker |
| --- | --- | --- | --- | --- |
| Daylio | - Simple mood logging- Activity tracking- Basic statistics- Streak system | - Intuitive interface- Quick entry process- Good data visualization- Customizable activities | - Limited analytics- Basic visualization options- Minimal personalization- Limited export options | - Implement quick mood entry- Enhance with advanced analytics- Maintain simplicity while adding depth |
| Moodflow | - Mood tracking- Journal entries- Multiple parameters- Correlation analysis | - Comprehensive data collection- Correlation insights- Clean interface- Detailed graphs | - Overwhelming for new users- Long entry process- Limited customization- Requires premium for advanced features | - Balance data collection vs. usability- Make advanced features accessible- Implement correlation analysis |
| MoodKit | - CBT-based approach- Mood improvement activities- Thought journals- Mood tracking | - Evidence-based interventions- Educational content- Comprehensive toolkit- Privacy-focused | - Complex interface- Steep learning curve- Limited visualization- No social features | - Include evidence-based suggestions- Focus on both tracking and improvement- Maintain educational component |
| Mood Patterns | - Pattern recognition- Predictive analytics- Multiple data points- Detailed reports | - Advanced analytics- Predictive capabilities- Comprehensive reports- Scientific approach | - Complex for casual users- Data-heavy interface- Limited customization- Requires consistent use | - Implement pattern recognition- Balance data with user experience- Include predictive elements |
| Reflectly | - AI journaling- Guided reflection- Beautiful UI- Personalized content | - Aesthetically pleasing- Engaging experience- Personalized insights- Focus on well-being | - Limited data visualization- More journal than tracker- Subscription model- Limited export options | - Prioritize user interface design- Integrate journaling with tracking- Balance aesthetics with functionality |

### 3.1.2 User Needs Assessment

A user needs assessment was conducted through surveys, interviews, and analysis of reviews for existing mood tracking applications. The assessment revealed the following key user requirements:

1. **Simplicity and Efficiency**
   * Quick mood logging process (under 30 seconds)
   * Intuitive interface requiring minimal learning
   * Streamlined data entry process
2. **Visual Insights**
   * Clear visualization of mood patterns
   * Multiple chart types for different perspectives
   * Color-coded representation for quick understanding
3. **Meaningful Analytics**
   * Identification of patterns and correlations
   * Insights into factors affecting mood
   * Actionable recommendations based on data
4. **Personalization**
   * Customizable interface and themes
   * Adaptable to individual tracking preferences
   * Personalized insights and recommendations
5. **Privacy and Security**
   * Control over personal emotional data
   * Option for offline-only usage
   * Transparent data handling practices
6. **Engagement and Motivation**
   * Features to encourage consistent use
   * Rewarding experience for regular tracking
   * Visual progress indicators
7. **Comprehensive Tracking**
   * Ability to track contextual factors
   * Integration of relevant external data
   * Holistic view of emotional well-being
8. **Accessibility**
   * Cross-platform availability
   * Responsive design for various devices
   * Accommodation for different user abilities

### 3.1.3 Gap Analysis and Opportunities

The analysis of existing solutions and user needs identified several gaps and opportunities for the Pro Mood Tracker application:

| Gap in Existing Solutions | Opportunity for Pro Mood Tracker |
| --- | --- |
| Limited visualization options for mood data | Implement multiple chart types tailored to different analytical needs |
| Complex data entry processes | Create streamlined, emoji-based mood logging with optional detailed input |
| Minimal personalization options | Develop comprehensive theme system with multiple customization options |
| Basic or limited analytics | Implement advanced pattern recognition and correlation analysis |
| Lack of engagement features | Design rewards system with badges, streaks, and points |
| Poor integration of contextual factors | Incorporate weather data, activities, and other context elements |
| Limited data export capabilities | Create comprehensive export options in multiple formats |
| Overwhelming interfaces for casual users | Design progressive complexity with intuitive default options |
| Limited offline functionality | Implement local storage for full offline capability |

## 3.2 Data Dictionary

The Pro Mood Tracker application uses a structured data model to store and manage various types of information. The following data dictionary provides a comprehensive overview of the data entities, attributes, and relationships within the system.

### 3.2.1 User Data

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| id | String | Unique identifier for the user | Primary key, UUID format |
| username | String | User’s chosen display name | 3-50 characters, alphanumeric |
| email | String | User’s email address | Valid email format |
| createdAt | String | ISO timestamp of account creation | ISO date string |
| points | Number | Accumulated reward points | Integer, ≥ 0 |
| settings | Object | User preference settings | See UserSettings |

### 3.2.2 Mood Entry

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| id | String | Unique identifier for the mood entry | Primary key, UUID format |
| date | String | Date of the mood entry | ISO date string |
| timeOfDay | String | Time period of the mood entry | One of: ‘morning’, ‘afternoon’, ‘evening’, ‘night’, ‘full-day’ |
| mood | String | Mood level recorded | One of: ‘Very Bad’, ‘Bad’, ‘Okay’, ‘Good’, ‘Very Good’ |
| notes | String | Additional notes about the mood | Optional, 0-500 characters |
| customCategory | String | User-defined mood category | Optional |
| time | String | Specific time of entry | Optional, HH:MM format |
| journal | String | Longer journal entry | Optional, 0-2000 characters |
| activities | Array of Strings | Activities associated with the mood | Optional |
| weather | Object | Weather data at time of entry | Optional, see Weather Data |

### 3.2.3 Weather Data

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| temperature | Number | Temperature in degrees Celsius | Optional, decimal number |
| condition | String | Weather condition description | Optional, e.g., ‘Sunny’, ‘Rainy’ |
| humidity | Number | Humidity percentage | Optional, 0-100 |

### 3.2.4 Badge

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| id | String | Unique identifier for the badge | Primary key, UUID format |
| name | String | Display name of the badge | 2-50 characters |
| description | String | Description of how to earn the badge | 10-200 characters |
| imageUrl | String | URL to the badge image | Valid URL or image path |
| criteria | String | Technical criteria description | Internal reference |
| points | Number | Points awarded for earning the badge | Integer, > 0 |
| tier | String | Badge tier or difficulty level | One of: ‘bronze’, ‘silver’, ‘gold’, ‘platinum’ |
| earnedAt | String | ISO timestamp when badge was earned | Optional, ISO date string |

### 3.2.5 User Settings

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| theme | String | User’s preferred theme | One of: ‘dark’, ‘light’, ‘ocean’, ‘sunset’, ‘forest’, ‘pastel’ |
| enableWeatherTracking | Boolean | Whether to track weather data | true/false |
| enablePredictions | Boolean | Whether to show mood predictions | true/false |
| location | Object | User’s location information | Optional, see Location |

### 3.2.6 Location

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| city | String | User’s city | Optional |
| country | String | User’s country | Optional |
| lat | Number | Latitude coordinate | Optional, decimal number |
| lon | Number | Longitude coordinate | Optional, decimal number |

### 3.2.7 Custom Theme

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| id | String | Unique identifier for the theme | Primary key, UUID format |
| name | String | Display name of the custom theme | 2-50 characters |
| primary | String | Primary color | Hex color code |
| secondary | String | Secondary color | Hex color code |
| background | String | Background color | Hex color code |
| paper | String | Paper/card background color | Hex color code |
| text | String | Text color | Hex color code |
| createdAt | String | ISO timestamp of theme creation | ISO date string |

### 3.2.8 Mood Streak

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| type | String | Type of streak | e.g., ‘daily’, ‘weekly’ |
| count | Number | Current streak count | Integer, ≥ 0 |
| startDate | String | ISO timestamp of streak start | ISO date string |
| endDate | String | ISO timestamp of streak end | ISO date string |

### 3.2.9 Mood Stats

| Attribute | Data Type | Description | Constraints |
| --- | --- | --- | --- |
| averageMoodByDay | Object | Average mood score by day of week | Key-value pairs: day -> number |
| averageMoodByTimeOfDay | Object | Average mood score by time of day | Key-value pairs: timeOfDay -> number |
| moodCountByType | Object | Count of each mood type | Key-value pairs: mood -> number |
| mostFrequentMood | String | Most frequently recorded mood | One of mood types |
| moodTrend | String | Overall trend in mood data | One of: ‘improving’, ‘declining’, ‘stable’ |
| moodCorrelations | Object | Correlation data for various factors | Complex structure |

## 3.3 Modules and Their Description of System

The Pro Mood Tracker application is organized into several functional modules, each responsible for specific aspects of the system’s functionality. This modular architecture enhances maintainability, scalability, and separation of concerns.

### 3.3.1 Authentication Module

**Purpose**: Manages user authentication, registration, and session handling.

**Key Components**: - Login Component: Handles user login with email/username and password - Registration Component: Facilitates new user registration - Auth Context: Manages authentication state across the application - Session Management: Handles user session persistence

**Core Functionality**: - User authentication with secure credential verification - New user registration with validation - Session persistence using local storage - Protected route management - Password reset functionality (future implementation)

**Integration Points**: - Interacts with User Module for profile management - Provides authentication state to the entire application - Controls access to protected features and routes

### 3.3.2 Mood Tracking Module

**Purpose**: Enables users to log, edit, and manage mood entries.

**Key Components**: - Mood Logger: Primary interface for recording new mood entries - Mood Selector: Component for selecting mood level with visual indicators - Time Period Selector: Component for specifying time of day - Journal Entry: Rich text editor for detailed mood journaling - Activity Selector: Interface for tagging activities with moods

**Core Functionality**: - Quick mood recording with emoji-based interface - Detailed mood entry with optional contextual information - Weather data integration (optional) - Activity tagging and categorization - Journaling capabilities for detailed expression

**Integration Points**: - Provides data to Visualization Module for charts and graphs - Feeds information to Analytics Module for processing - Connects with Rewards Module for streak tracking - Utilizes Theme Module for visual presentation

### 3.3.3 Visualization Module

**Purpose**: Presents mood data in various visual formats for pattern recognition and analysis.

**Key Components**: - Mood Calendar: Calendar view with color-coded mood indicators - Trend Charts: Line and bar charts for trend visualization - Distribution Graphs: Pie and bar charts showing mood distribution - Time Analysis Charts: Specialized charts for time-based analysis - Correlation Visualizations: Charts showing relationships between factors

**Core Functionality**: - Multiple chart types for different analytical perspectives - Interactive visualizations with filtering capabilities - Time-range selection for focused analysis - Responsive design for various screen sizes - Export functionality for charts and visualizations

**Integration Points**: - Consumes data from Mood Tracking Module - Utilizes Analytics Module for processed data - Adapts to Theme Module for visual styling - Provides visual elements for Dashboard Module

### 3.3.4 Analytics Module

**Purpose**: Processes mood data to derive insights, identify patterns, and generate recommendations.

**Key Components**: - Statistical Processor: Calculates statistical measures from mood data - Pattern Recognition: Identifies patterns and trends in mood entries - Correlation Analyzer: Identifies relationships between mood and other factors - Prediction Engine: Generates mood forecasts based on historical data - Insight Generator: Creates actionable insights from analysis

**Core Functionality**: - Statistical analysis of mood patterns - Identification of mood triggers and influencing factors - Correlation of mood with activities, weather, and time - Prediction of future mood states based on patterns - Generation of personalized recommendations

**Integration Points**: - Processes data from Mood Tracking Module - Provides analytical results to Visualization Module - Feeds insights to Recommendations Module - Supplies data for Reports Module

### 3.3.5 Rewards Module

**Purpose**: Implements gamification elements to encourage consistent application use.

**Key Components**: - Streak Tracker: Monitors and manages user streaks - Badge System: Assigns and displays achievement badges - Points Manager: Calculates and tracks reward points - Achievement Detector: Identifies when achievement criteria are met - Rewards Display: Visualizes user’s rewards and achievements

**Core Functionality**: - Daily streak tracking for consistent mood logging - Badge awarding for reaching milestones - Points system for engagement and achievements - Visual representation of progress and accomplishments - Notification of new achievements and rewards

**Integration Points**: - Monitors Mood Tracking Module for logging activity - Adapts to Theme Module for visual presentation - Provides motivation elements for Dashboard Module - Future integration with potential social sharing features

### 3.3.6 Theme Module

**Purpose**: Manages application appearance and visual customization.

**Key Components**: - Theme Provider: Context provider for theme settings - Theme Selector: Interface for choosing predefined themes - Color Customizer: Tools for creating custom color schemes - Theme Previewer: Real-time preview of theme changes - CSS Variable Manager: Handles dynamic CSS variable updates

**Core Functionality**: - Multiple predefined theme options - Custom theme creation and management - Dynamic application of theme settings - Persistent theme preferences - Accessibility considerations in theme design

**Integration Points**: - Provides theming context to all UI components - Interacts with User Module for preference storage - Applies consistent styling across all modules - Ensures proper contrast and readability

### 3.3.7 Data Management Module

**Purpose**: Handles data storage, retrieval, export, and backup.

**Key Components**: - Local Storage Manager: Interfaces with browser storage APIs - Data Export Engine: Generates data export files - Import Processor: Handles importing of previously exported data - Data Validator: Ensures data integrity and format compliance - Email Integration: Facilitates sharing data via email

**Core Functionality**: - Persistent storage of application data - Data export in multiple formats (JSON, CSV) - Optional cloud backup functionality (future implementation) - Data sharing via email or download - Data integrity and validation

**Integration Points**: - Provides data persistence for all modules - Interfaces with external services for data sharing - Ensures data availability for all application features - Manages user data privacy and security

### 3.3.8 User Profile Module

**Purpose**: Manages user profile information and preferences.

**Key Components**: - Profile Editor: Interface for updating user information - Preferences Manager: Controls for setting user preferences - Settings Panel: Comprehensive settings interface - User Context: Provides user information across the application - Profile Display: Visual representation of user profile

**Core Functionality**: - User profile management and updates - Preference setting and storage - Application settings control - User statistics and overview - Account management options

**Integration Points**: - Connects with Authentication Module for user identity - Provides user context to other modules - Stores preferences for Theme Module - Manages settings for all application features

### 3.3.9 Notifications Module

**Purpose**: Manages application notifications and reminders.

**Key Components**: - Notification Manager: Handles creation and display of notifications - Reminder Scheduler: Sets and manages reminder schedules - Notification Center: Central interface for notification history - Push Notification Handler: Manages browser push notifications - Settings Controller: Controls for notification preferences

**Core Functionality**: - In-app notifications for achievements and milestones - Reminder system for consistent mood logging - Notification history and management - Customizable notification preferences - Silent vs. alert notifications

**Integration Points**: - Receives triggers from Rewards Module - Integrates with User Profile for preferences - Provides notification services to all modules - Future integration with external notification services

## 3.4 Database Design

Although the Pro Mood Tracker application primarily uses client-side storage rather than a traditional server-based database, the data structure design follows database design principles. This section outlines the logical data model, relationships, and storage implementation.

### 3.4.1 Storage Strategy

The application utilizes the following storage mechanisms:

1. **Local Storage**: Primary storage method using the browser’s localStorage API
   * Persistent across sessions and browser restarts
   * Limited to string data (JSON serialization used)
   * Approximately 5-10MB capacity depending on browser
2. **Session Storage**: Used for temporary session-specific data
   * Cleared when browser session ends
   * Same string-only limitation as localStorage
   * Used for temporary form data and session state
3. **Indexed DB** (future implementation):
   * Object-oriented database for more complex data structures
   * Larger storage capacity
   * Support for complex queries and indexing

### 3.4.2 Data Relationships

The following entity-relationship diagram illustrates the relationships between key data entities in the application:

+---------------+ +---------------+ +---------------+  
| | | | | |  
| User |<------+ MoodEntry +------>| Weather |  
| | | | | |  
+-------+-------+ +-------+-------+ +---------------+  
 | |  
 | |  
 v v  
+-------+-------+ +-------+-------+ +---------------+  
| | | | | |  
| Badge | | Activity |<----->| MoodEntry |  
| | | | | |  
+---------------+ +---------------+ +---------------+  
 ^  
 |  
+-------+-------+  
| |  
| Achievement |  
| |  
+---------------+

### 3.4.3 Data Storage Implementation

The application uses a structured approach to local storage organization:

1. **Namespace Prefixing**: All storage keys are prefixed with ‘proMoodTracker:’ to avoid conflicts
2. **Entity-based Storage**: Data is organized by entity type:
   * proMoodTracker:user - User profile and settings
   * proMoodTracker:moodEntries - Array of mood entry objects
   * proMoodTracker:badges - Earned badges and achievements
   * proMoodTracker:themes - Custom theme definitions
   * proMoodTracker:stats - Cached statistical calculations
3. **Data Serialization**: All data is serialized to JSON format for storage and deserialized when retrieved
4. **Storage Optimization**:
   * Non-essential data is stored separately to minimize main data payload
   * Historical entries beyond a certain age may be archived separately
   * Optional compression for large datasets (future implementation)

### 3.4.4 Data Integrity and Security

To ensure data integrity and security, the application implements several measures:

1. **Validation**: All data is validated before storage to ensure format compliance
2. **Error Handling**: Comprehensive error handling for storage failures
3. **Backup Reminders**: Users are periodically reminded to export their data
4. **Data Encryption** (future implementation): Sensitive data will be encrypted
5. **Version Tagging**: Data structures include version information for migration support
6. **Atomic Updates**: Critical updates use temporary storage to prevent data corruption

### 3.4.5 Data Migration Strategy

As the application evolves, data structure changes are managed through:

1. **Schema Versioning**: Data structures include version information
2. **Migration Functions**: Code to upgrade data from previous formats
3. **Backward Compatibility**: New versions can read old data formats
4. **Migration Notifications**: Users are informed when data migration occurs
5. **Rollback Support**: Critical migrations maintain backup of original data