CHAPTER 1: INTRODUCTION

1.1 Background of the System

LifeLog is a proposed wellness and habit tracking system designed to help individuals manage their emotional well-being, daily habits, and journaling activities in one integrated platform. In modern life, many people struggle with stress, lack of focus, and inconsistency in maintaining positive routines. These challenges often lead to reduced productivity and mental exhaustion.

LifeLog aims to address these issues by providing users with a simple, web-based platform where they can record their daily moods, track personal habits, and reflect through journaling. The system helps users visualize their progress, identify behavioral patterns, and stay motivated toward self-improvement.

It demonstrates how technology can be used to support emotional wellness, mindfulness, and self-discipline in an accessible and interactive way.

1.2 General Description of the Existing System

Currently, individuals who wish to manage their emotional health or track their daily habits rely on various disconnected tools such as:

- Traditional paper journals for daily reflections
- Mobile apps that focus only on one feature (either journaling or habit tracking)
- Generic note-taking applications without mood analysis features

These existing systems are often fragmented and require users to switch between multiple platforms. As a result, users find it difficult to maintain consistency, visualize emotional progress, or understand long-term behavioral trends. There is a clear need for a unified system that integrates mood tracking, journaling, and habit management in one environment.

1.3 Problems of the Existing System

The current tools used for journaling or tracking habits have several limitations, including:

- Lack of integration between journaling, mood, and habit tracking
- Difficulty in analyzing and visualizing emotional trends
- Inability to monitor long-term growth and consistency
- Risk of data loss or inaccuracy with manual entries
- Limited motivation and engagement features
- No unified, user-friendly platform for overall wellness tracking
- These problems make it challenging for individuals to develop self-awareness and remain consistent in improving their mental and emotional well-being.

1.4 Objective of the Project

1.4.1. General Objective

To design and develop a user-friendly, web-based system that allows users to track their moods, habits, and personal journals in one integrated platform.

1.4.2. Specific Objectives

- To create a simple and responsive interface for tracking moods, habits, and journals
- To enable users to visualize weekly progress and personal growth through charts
- To store user data using local storage for offline accessibility
- To encourage self-reflection, consistency, and mindfulness
- To demonstrate the potential of digital tools in promoting emotional wellness

1.5 Overview of the Proposed System

- ✓ The proposed LifeLog System will allow users to:
- ✓ Record their daily moods through an intuitive interface
- ✓ Add, edit, and delete journal entries for self-reflection
- ✓ Create and track personal habits with daily check-ins
- ✓ View weekly progress and emotional trends using visual summaries

All data will be stored locally in the browser, eliminating the need for an external database. The system will have a clean and responsive design that works across devices, making it convenient for users to maintain their daily logs anytime, anywhere.

Expected Benefits:

- Improved emotional awareness and mindfulness
- Simplified mood and habit tracking
- Motivation through progress visualization
- Increased consistency in personal development

1.6 Feasibility Analysis

1.6.1. **Economic Feasibility**

The development of LifeLog requires no cost since it relies on free and open-source web technologies such as HTML, CSS, and JavaScript. The system offers long-term benefits in efficiency, accessibility, and mental well-being support.

1.6.2. Technical Feasibility

LifeLog is technically feasible as it uses widely available and lightweight web technologies. No external servers or databases are needed, and the required development tools are easily accessible.

1.6.3. Operational Feasibility

The system is designed for simplicity and ease of use. Users do not require technical knowledge to operate it. It can be accessed through any modern web browser on desktop or mobile devices

1.6.4. Schedule Feasibility

The project can be completed within a reasonable timeframe using an iterative, Agile approach, allowing gradual testing, feedback, and improvement at each stage.

1.7 Relevance of the Proposed System

The proposed LifeLog system is highly relevant in today's fast-paced environment, where people often overlook their emotional and mental well-being. It encourages users to reflect, track, and improve their habits and moods through digital journaling. By integrating multiple wellness features into one platform, LifeLog supports personal growth, emotional awareness, and a balanced lifestyle.

1.8 Scope of the Proposed System

In Scope:

- Journal entry management (add, edit, delete)
- Mood tracking and analysis
- Habit creation and progress tracking
- Weekly summary visualization
- Local storage for data saving
- Responsive web interface

Out of Scope:

- Backend or cloud integration
- AI-based mood prediction
- Notifications and reminders
- Mobile application development

1.9 Project Schedule

Date Range	**Phase / Task**	**Expected Output**
Oct 21 – Oct 24	Proposal Development	Project proposal submitted
Oct 25 – Oct 31	Requirements Analysis & Planning	Requirements Document
Nov 1 – Nov 7	System Design	Wireframes, ERD, UI mockups
Nov 8 – Nov 21	Development Phase	Functional prototype
Nov 22 – Nov 28	Testing and Debugging	Bug-free working system
Nov 29 – Dec 30	Documentation & Presentation Prep	Final report + presentation slides

CHAPTER 2: CONTEXTUAL ANALYSISAN METHODOLOGICAL FRAMEWORK

2.1 Literature Review

Development of the digital mental health application requires a comprehensive awareness of what exists, where others have faltered, and where success has been achieved to develop an evidence-based approach to design.

2.1.1. **What Other Projects Have Failed?**

A literature review of digital mental health projects that have failed illustrates shortcomings of user engagement primarily and other derivative issues. First and foremost, user engagement suffers. Applications are developed based solely on clinically effective standards without the user experience (UX) design and appeal to retain users. This is why many cognitive behavioral therapy (CBT) apps no longer exist; they created interfaces that were complicated and conversations that felt too robotic for users to feel a sense of connectedness or motivation to continue effort in sessions [1]. Moreover, a lack of personal engagement results in project failure as well. Generic options that fail to personalize service application inevitably lose users over time. For example, applications that allowed users to go down static pathways or offer content without subsequent feedback or customized approaches diminished user efficacy over time [2]. However, it's not enough to fail to personalize; applications that lack socio-cultural considerations have found failure as well. For example, in Ethiopia, applications like Woebot and Ten Percent Happier fail because the language used, and metaphors/examples provided do not resonate with the users [3].

2.1.2. **What Other Projects Have Succeeded?**

Those that have succeeded have few things in common. Project success in the digital mental health realm stems from user-centered designs (UCD) from the start, whereby subsequent steps based on feedback involve the target population in every stage of production to ultimately create a highly intuitive and engaging product that meets the actual needs of those involved [4]. Furthermore, successful gamification and positive reinforcement efforts are critical. For example, apps like Habitica now gamify habit tracking and turn it into a role-playing game, so through points systems, badges, streaks, and a visualization bar, ongoing motivation can be sustained [5]. Successful applications also use visualization data; the more users can see mood trends or streaks over time translates to more tangible evidence of progress - an exciting motivating factor aligned with behavioral activation therapy [6].

2.1.3. **Benchmarking Relevance**

Life Log SAM will be developed relative to existing self-management platforms such as Daylio, Journey.Cloud, and Trello, which are widely used tools in the real world, for benchmarking purposes.
For benchmarking, the following aspects are considered:

• **Feature Coverage:** Daylio focuses mainly on mood tracking and visual summaries, Journey.Cloud emphasizes journaling with cloud backup, and Trello centers on task and to-do management. Life Log SAM will be benchmarked to integrate all these features—mood tracking, journaling, habit monitoring, task management, and progress visualization—within a single platform.

• **Integration Capability:** Unlike the existing platforms that operate as standalone solutions for specific needs, Life Log SAM aims to benchmark how well these features can be interconnected to provide a more holistic self-management experience.

• **User Experience:** Daylio and Trello are known for their simple and intuitive interfaces, while Journey.Cloud emphasizes clean journaling workflows. Life Log SAM will benchmark usability and ease of navigation to ensure a smooth and user-friendly experience.

• **Progress Visualization:** While Daylio provides visual mood summaries and Trello shows task progress, Life Log SAM will benchmark these approaches to deliver combined visual insights for mood trends, habits, and task completion over time.

• **Accessibility:** All benchmarked platforms are accessible online or via mobile applications. Life Log SAM will be benchmarked to remain lightweight, user-accessible, and suitable for environments with limited bandwidth, ensuring broader usability.

2.2 Research Methodology

2.2.1. **Research/Requirements Collection Methodology**

A blend of approaches will be used to garner an extensive array of requirements: · Surveys/Questionnaires: These will be distributed to a segmented population in

Addis Ababa to assess general smartphone usage, mental health (or awareness of wellness) and interest in a wellness app and what features would be desirable. · Semi-Structured Interviews: These will be collected from a subset of survey respondents and local community health workers for stronger qualitative analysis of cultural relevance, stigma, etc. · Existing Application Review/Analysis: We will consider features of applications already available (like Happify) and compare reviews noting both successful user experiences and common complaints.

2.2.2. **Analysis & Modeling Approach**

This project will be guided by the Agile/Scrum process. Development will occur over two-week sprints. The advantages of an iterative approach include: · Flexibility: Requirements are subject to change and refinement at any point in the development phase after user testing and demos. · Incremental Development: At the end of each sprint, a "potentially shippable" product increment will be added each step of the way from a simple journal to just tracking moods, then adding habits. · Early Risk Detection: Problems are noted and addressed sooner rather than later in the development phase, mitigating massive failure at a later date. The following models will be created for system modeling: · User Stories/Epic Stories: To clarify functional requirements from the user's perspective. · Use Case Diagrams: To depict interactions with users (actors) within the application. · Entity-Relationship Diagrams (ERD): To map the application database schema surrounding users, journals, moods and habits.

2.2.3. **Software Tool(s) To Be Used**

The following software tools will be used for modeling, documentation and development purposes: · Figma: Wireframing/prototyping/UI/UX software. · Draw.io / Lucidchart: For use case diagrams, ERD and other technical diagrams. · Trello / Jira: For Agile processes, backlog grooming and sprint planning. · Google Workspace: For shared documentation, survey creation (Google Forms) and data input (Sheets). · Git / GitHub: For version control.

CHAPTER 3: REQUIREMENT ANALYSIS AND SPECIFICATION

3.1 Requirement Analysis

3.1.1 **Analysis Approach (BPA/BPI/BPR)**

For this project, we chose to use the Business Process Improvement(BPI) approach.

BPI (Business process improvement) is a practice in which leaders analyze their processes using multiple techniques. This allows them to identify areas where they can improve their efficiency, precision, or usefulness. They can then revamp the procedures to consider the differences. BPI aims to pinpoint the skills or processes that need improvement in ensuring smooth procedures, efficient workflow, and business expansion.

On the other hand, BPA (Business Process Automation) would simply digitize traditional journaling and tracking manually done by users. And BPR (Business Process Reengineering) would require a complete redesign of how users approach daily reflection and productivity, which is beyond the project's current scope.

Therefore BPI analysis approach is chosen for this project to improves current habits and routines through digital integration, motivation tools, and data visualization.

3.1.2 Functional Requirements

The functional requirements describe **what the LifeLog system must do** to meet user needs.

1. User Registration, Login, and Authentication

- The system shall allow users to create an account using an email and password.
- The system shall allow registered users to log in securely.
- The system should ensure that usernames or emails are unique.
- The system should enforce strong password rules for security.

2. Logout and Account Deletion

- The system shall allow users to securely log out of their account.
- The system shall allow users to permanently delete their account.
- When an account is deleted, all related user data shall also be removed.

3. Profile Management

- The system shall allow users to view and edit their profile information.
- The system should allow users to upload or change a profile picture.

4. Habit Tracking

- The system shall allow users to create daily or weekly habits.
- The system shall allow users to edit or delete existing habits.
- The system should allow users to assign custom icons or colors to habits.
- The system shall allow users to set habit frequency (daily or weekly).

5. Daily Journal Logging

- The system shall allow users to write daily journal entries.
- The system shall automatically save the date and time of each entry.
- The system shall allow users to edit or delete journal entries.

6. Progress Visualization

- The system shall display progress using charts or summaries.
- The system should motivate users by showing habit completion and trends.

7. Data Storage

- The system shall store all user data securely.
- The system shall allow users to retrieve saved logs and progress data.
- Data must remain available even after the application is restarted.

8. History Visualization

- The system shall allow users to view past journal entries.
- The system should allow filtering logs by date or category.

9. Goal Setting

- The system shall allow users to set measurable personal goals.
- The system should allow users to track progress toward these goals.

3.1.3 Nonfunctional Requirements

The non-functional requirements describe **how well the system should perform**.

1. Usability

- The user interface shall be simple and easy to understand.
- Navigation shall be clear and consistent.
- Fonts and colors shall be readable and accessible.

2. Performance

- The system shall respond to user actions within **3–5 seconds**.
- The application should load pages efficiently.

3. Security

- User passwords shall be encrypted.
- User data shall be stored securely.
- Unauthorized access shall be prevented.

4. Compatibility

- The system shall work on desktops, tablets, and mobile devices.
- The system shall support different screen sizes and browsers.

5. Data Backup and Recovery

- The system should automatically back up user data daily.
- The system should provide recovery options for accidental data loss.

6. Availability

- The system shall be available **24/7**.
- Downtime should be minimal and handled gracefully.

7. Privacy

- User data shall not be shared without user consent.
- Personal information shall remain confidential.

8. Reliability

- The system shall operate consistently without frequent failures.
- The system shall recover smoothly from errors.

9. Scalability

- The system shall handle multiple users efficiently.
- Performance shall not degrade as user data grows.

10. Maintainability

- The system shall be easy to update and maintain.
- Updates shall not affect existing user data or core functionality.

3.2 Comparative Analysis and Solution Proposal

3.1.1 Benchmarking Analysis

Benchmarking is a strategic management approach that uses the method of comparing their practices, processes, and performance metrics with those of their industry counterparts or top performers. It's a powerful tool that allows companies to identify areas for improvement, set performance targets, and implement effective strategies to enhance overall performance[2]. Therefore to understand how Lifelog SAM will stand out, existing self-management websites like Daylio (daylio.net), Journey.Cloud (journey.cloud and Trello (trello.co) were analyzed.

Daylio provides simple mood tracking and daily journal ent ries with visual summaries, but it lacks habit management and task-tracking integration. Journey . Cloud allows online journaling with cloud backup and mood logging, yet it does not include habit tracking or goal-setting features. Trello provides a simple task and to-do management system but lacks mood tracking and journaling.

Lifelog SAM is designed to address these limitations by providing a comprehensive self-management solution that combines journaling, mood tracking, habit monitoring, task management, and progress visualization, all within one platform.

3.1.2 Gap Analysis

The gap analysis identifies limitations in existing self-management systems like Daylio, Journey.Cloud, and Trello, and shows how Lifelog addresses these gaps.

1. Feature Fragmentation:

- **Limitation:** None of these systems **i**ntegrate all core self-management features like journaling, mood tracking, habit tracking, task management, and goal setting in a single platform.
- **Lifelog Solution:** Lifelog SAM combines all these features, providing a unified system for daily self-monitoring.

2. Limited Progress Visualization and Insights:

- **limitation:** Daylio provides basic mood charts, Journey.Cloud has simple logs, Trello provides task status. But users cannot easily track patterns or see progress across moods, habits, and tasks.
- **Lifelog Solution**: Lifelog SAM provides visual dashboards, and trend graphs for moods, habits, and tasks, offering clear insights and motivation.

3. **Historical Review and Data Integration:**

- **Limitation:** These platforms allow viewing past entries separately. Daylio allows viewing past entries on mood history, Journey.Cloud allows past views on journal history, and Trello on task history. But users cannot analyze past entries of all activities in one place.
- **Lifelog SAM Solution:** Lifelog SAM provides integrated history visualization, letting users review moods, habits, journals, and tasks together over time.

3.1.3 Proposed Solution Based on Requirements

Based on the gaps identified above, Lifelog is designed to provide a comprehensive and user-friendly self-management solution.

1. Addressing Feature Fragmentation:

- **Functional Requirements Fulfilled:**
    - The system shall allow users to log daily journals, track moods, manage habits, and organize tasks all in one platform.
    - Goal setting: Users shall be able to set measurable goals and personalize their habits and tasks.
- **Benefit:** By combining all core self-management features, Lifelog eliminates the need for multiple separate tools.

2. Enhancing Progress Visualization and Insights:

- **Functional Requirements Fulfilled:**
    - **Progress Visualization**: The system shall display visual dashboards, graphs, streaks, and summaries for moods, habits, and task completion.
    - Users shall be able to analyze trends over time to gain insights into personal progress.
- **Nonfunctional Requirements Fulfilled:**
    - Usability: Simple and intuitive interface for easy interpretation of visual data.
    - Reliability: Accurate tracking and consistent display of user progress.
- **Benefit:** Users are motivated by clear visual feedback, enabling them to **track improvements and stay consistent** with their routines.

3. Integrated Historical Review and Data Analysis:

- **Functional Requirements Fulfilled:**
    - The system shall allow users to view historical data across all activities in one place.
    - Users shall be able to filter and search past entries by date, category, or type.
- **Nonfunctional Requirements Fulfilled:**
    - Security: Historical data is stored securely and protected from unauthorized access.
    - Availability: Users can access past data anytime, ensuring continuous monitoring.
- **Benefit:** Users gain a **c**omprehensive understanding of their patterns, which helps in making informed decisions about personal growth and habit improvement.

3.3. Requirement Modeling

3.31. **Structured Requirement Modeling**

Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a visual representation of how data moves through a system. It shows how information enters the system, how it is processed, and where it is stored. A DFD helps developers, analysts, and users understand the flow of information without needing to know the technical details of system implementation. It emphasizes processes, data stores, data flows, and external entities.

The main purpose of a DFD is to break down complex systems into smaller, more understandable parts. By doing so, it helps in identifying inefficiencies, analyzing requirements, and designing the system in a structured way.

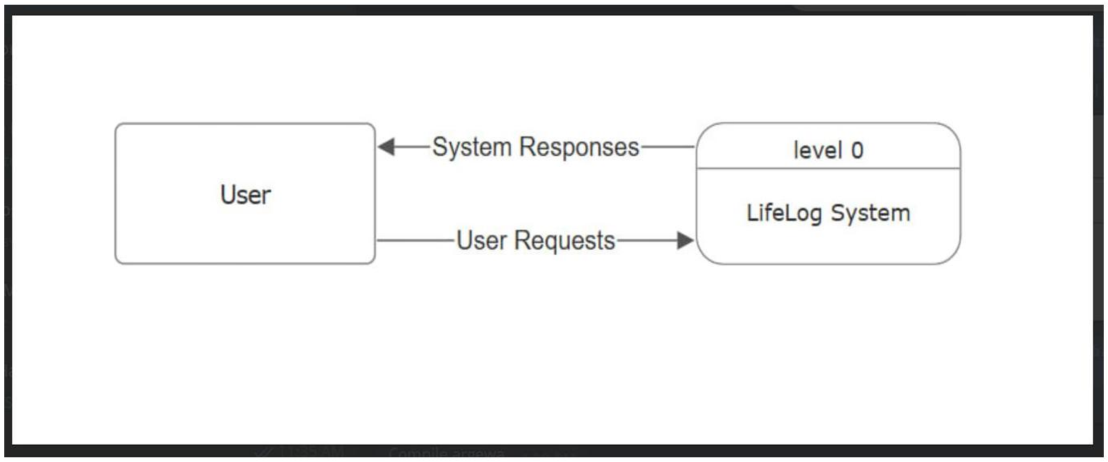
Classification of DFDs

DFDs are generally classified into three levels:

1. Level-0 (Context Diagram):
   This is the highest-level view of the system. The system is represented as a single process and shows only its interaction with external entities, such as users or other systems. It does not show internal details.
2. Level-1 DFD:
   At this level, the main system process is divided into sub-processes. It shows how the major parts of the system interact with each other and with the data stores. Level-1 diagrams give more detail than the context diagram but still provide an overview of the system's main functions.
3. Level-2 (and beyond) DFD:
   Level-2 diagrams further break down the Level-1 processes into more detailed subprocesses. This level shows exactly how data moves between specific parts of the system and gives a complete understanding of the internal operations. For complex systems, there may be additional levels, but Level-2 usually provides enough detail for most development purposes.

Level-0 DFD

The Level-0 Data Flow Diagram, often referred to as the context diagram, represents the LifeLog system at a very high level. It captures the system as a single process while showing its interactions with external entities, mainly the users. This level of DFD is essential for understanding the system's boundaries and overall scope without diving into detailed internal operations.

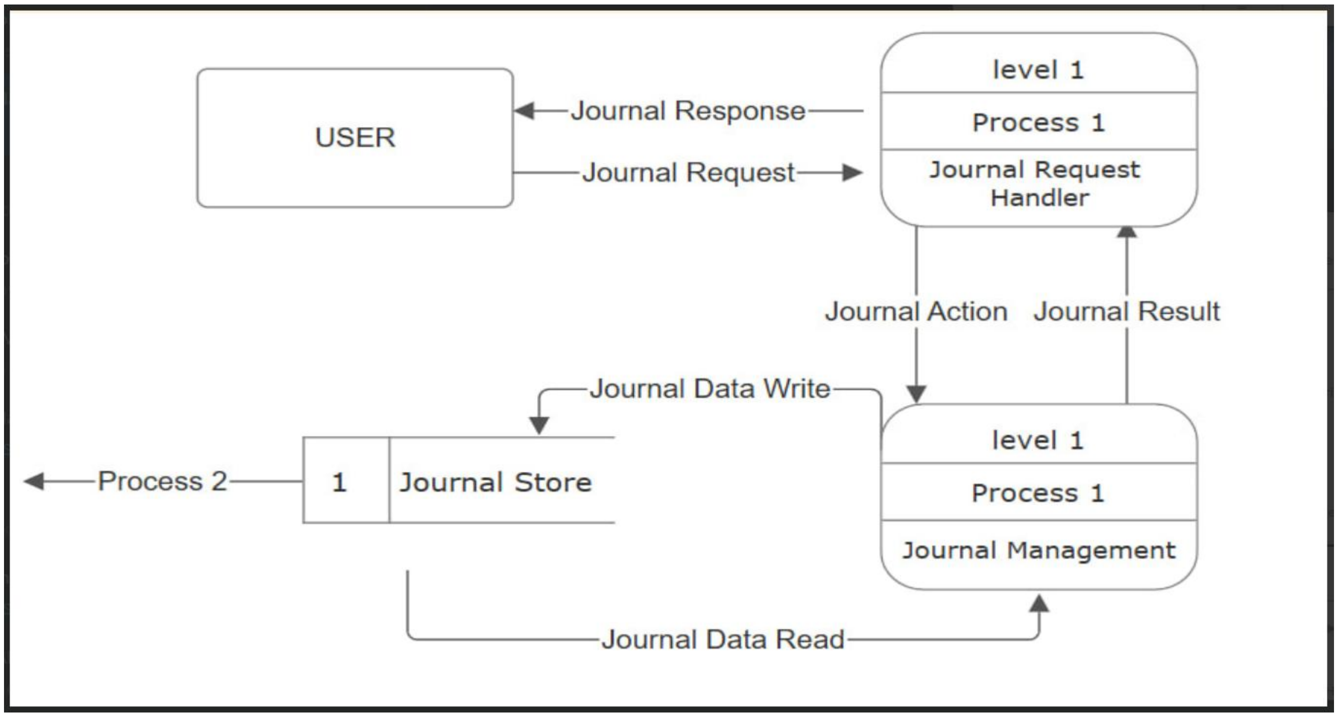


Level - 0 DFD

Level-1 DFD

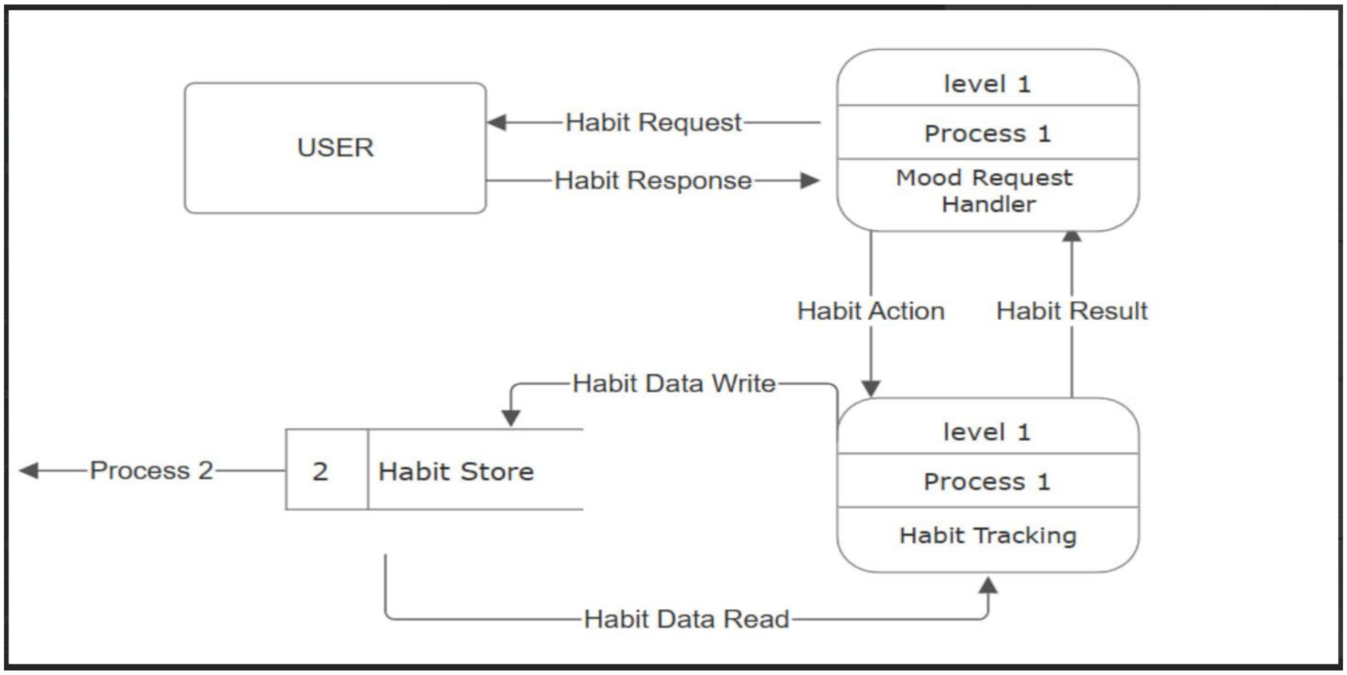
The Level-1 DFD breaks down the LifeLog system into its main functional modules, showing how data flows between them and with external entities. Unlike Level-0, which represents the system as a single process, Level-1 provides a more detailed view of the system's internal operations, yet still at a high level.

1. **Journal**



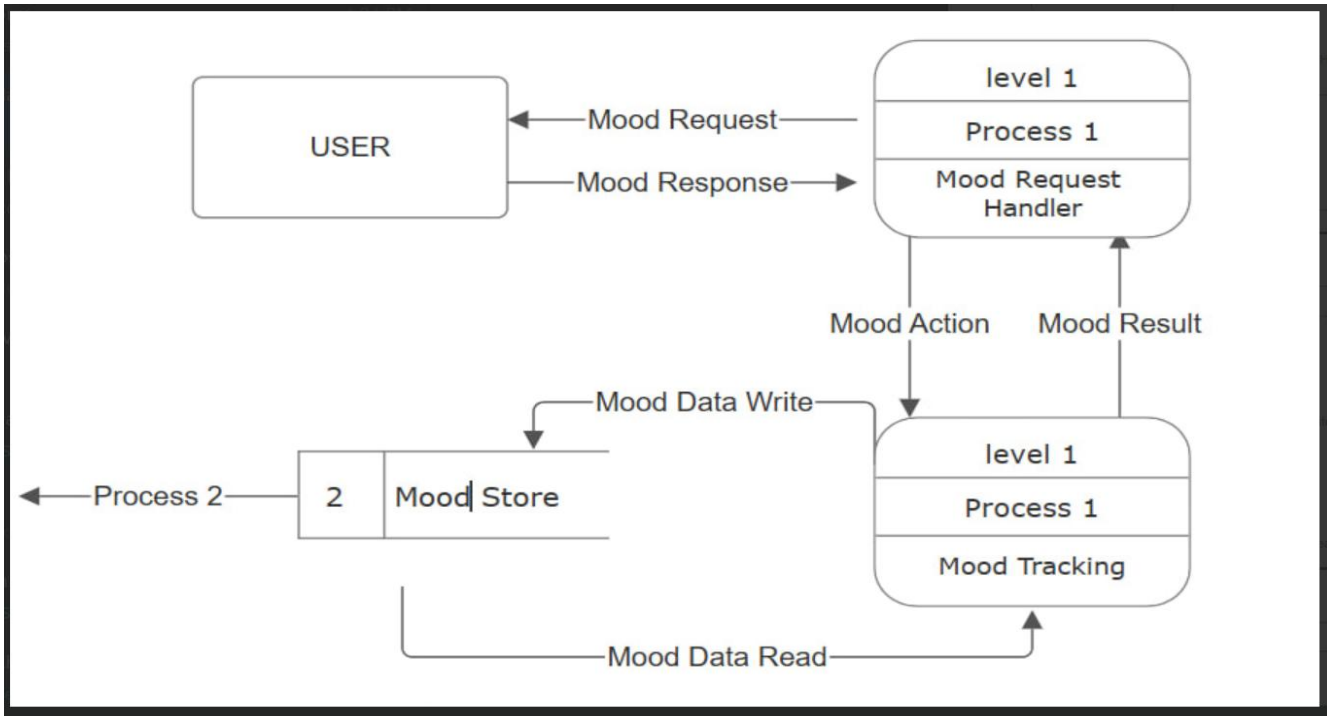
Level - 0 DFD

2. **Habit**



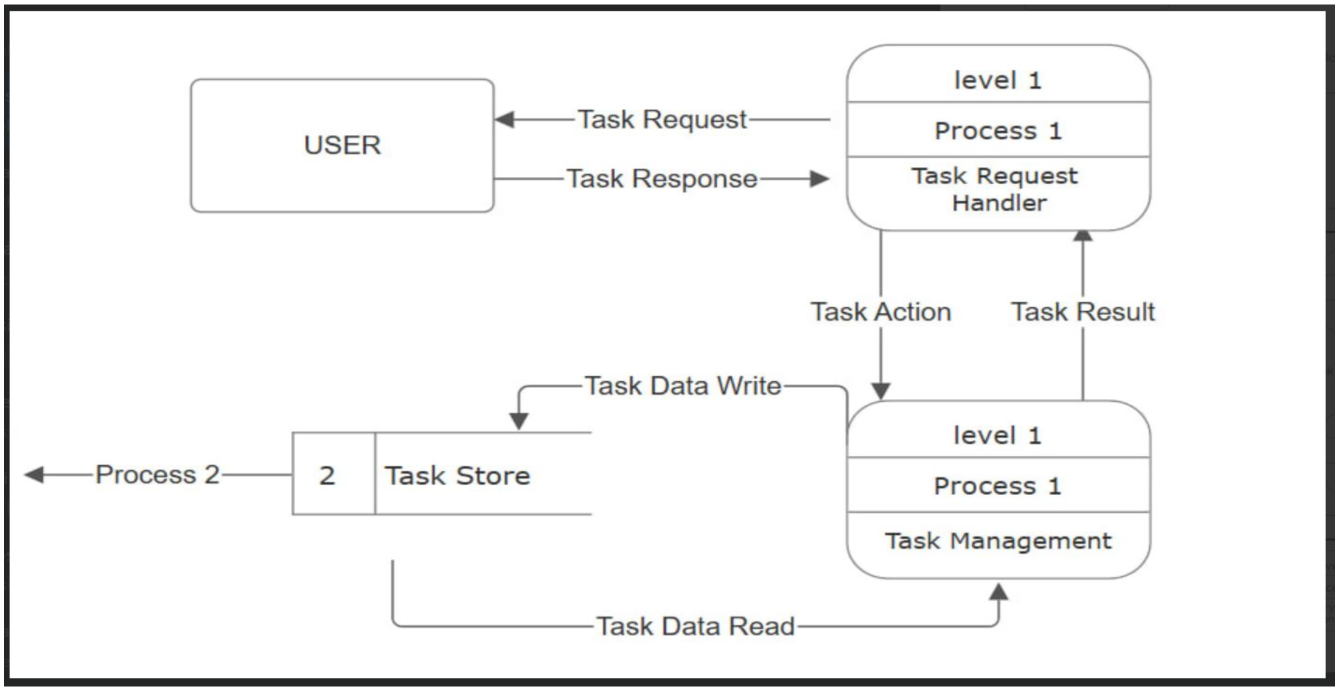
Level – 1 Habit DF

3. **mood**



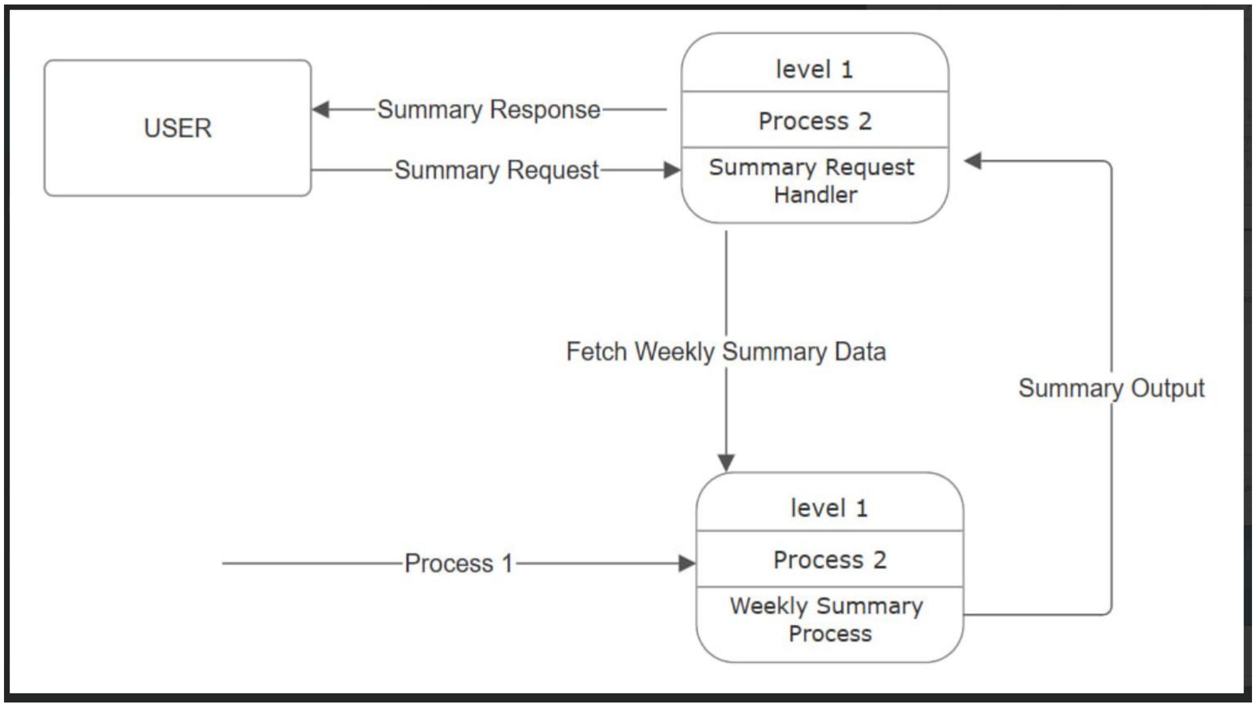
Level – 1 Mood DFD

4. todo list



Level – 1 TO-DO list DFD

5.weekly summary

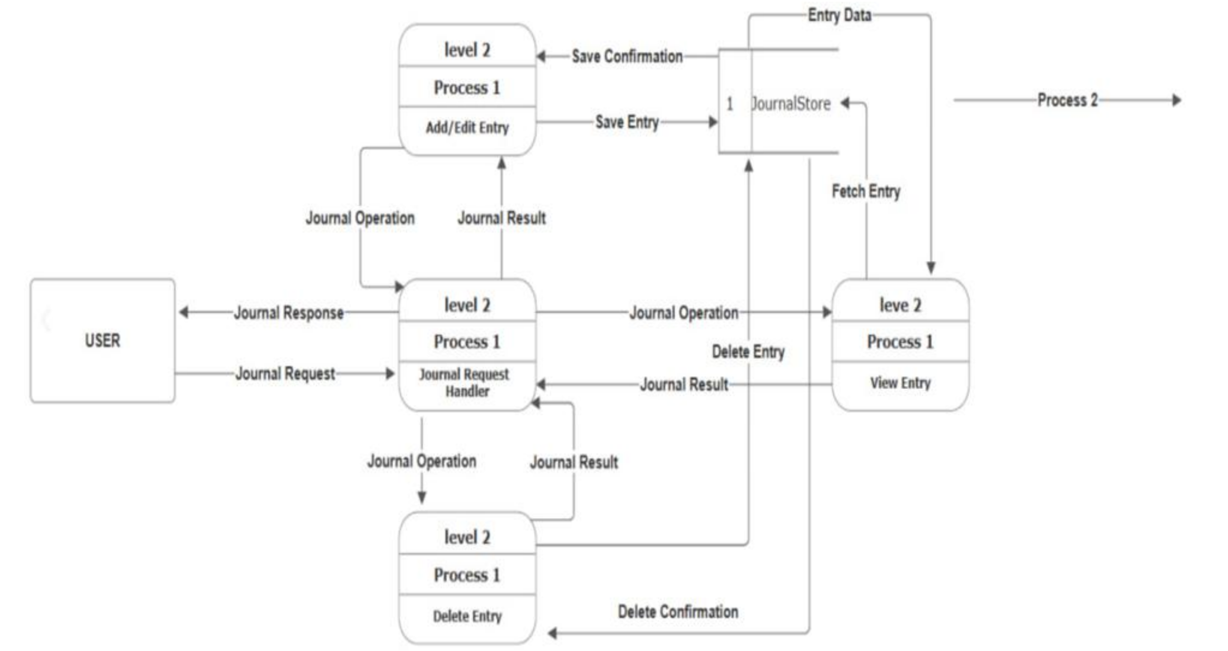


Level – 1 Weekly Summary DFD

Level-2 DFD

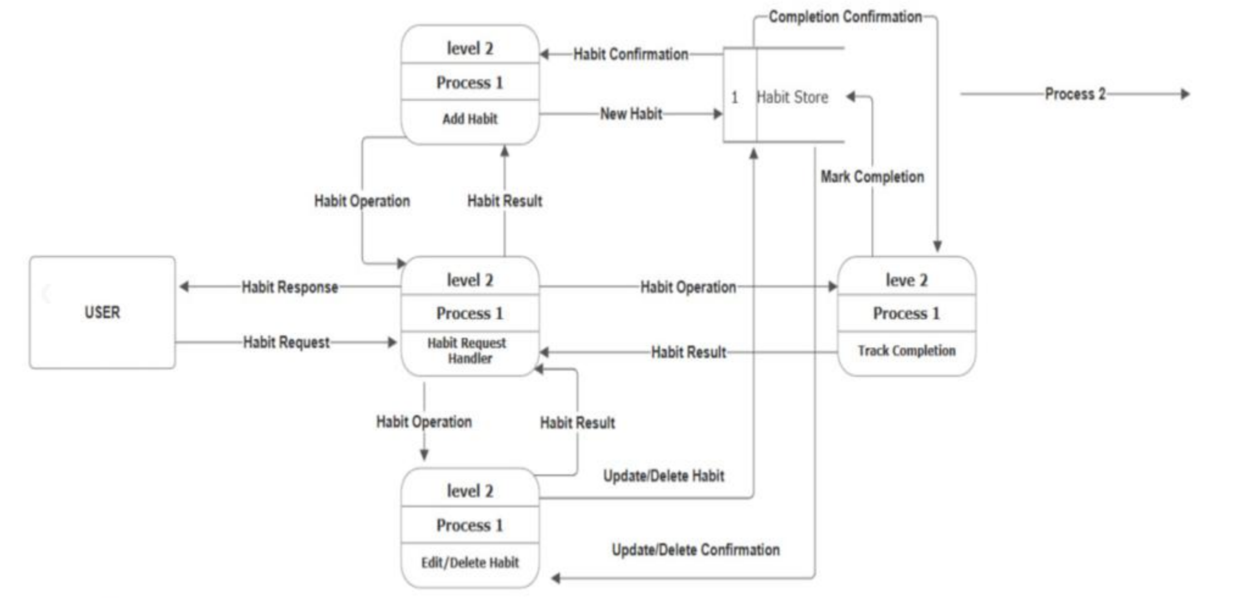
The Level-2 DFD expands the Level-1 modules into their internal sub-processes, showing how data moves within each part of the LifeLog system. This level provides a detailed view of the operations of each module, the data stores involved, and the flow of information between sub-processes.

1. **Journal**



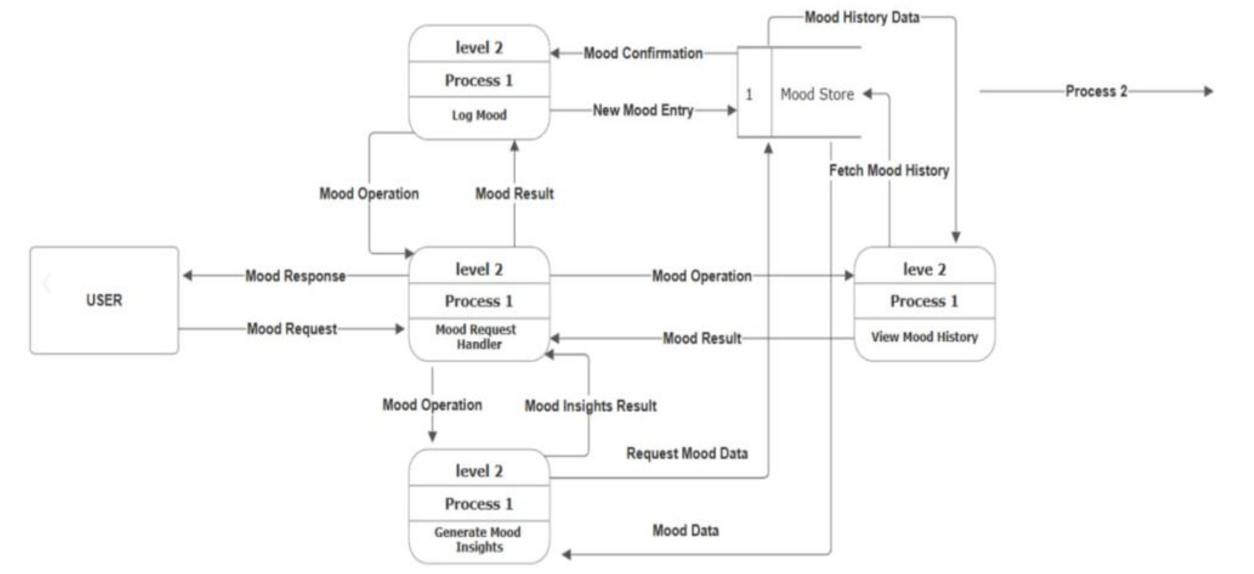
Level – 2 Journal DFD

1. **habit**



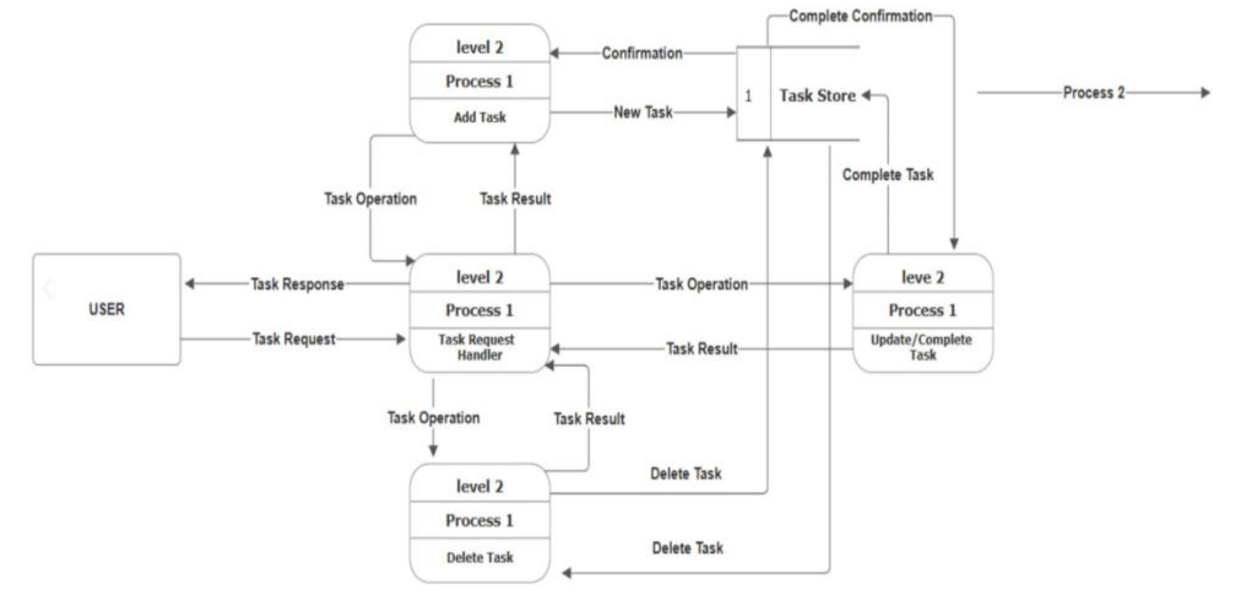
Level – 2 Habit DFD

1.1.1. mood



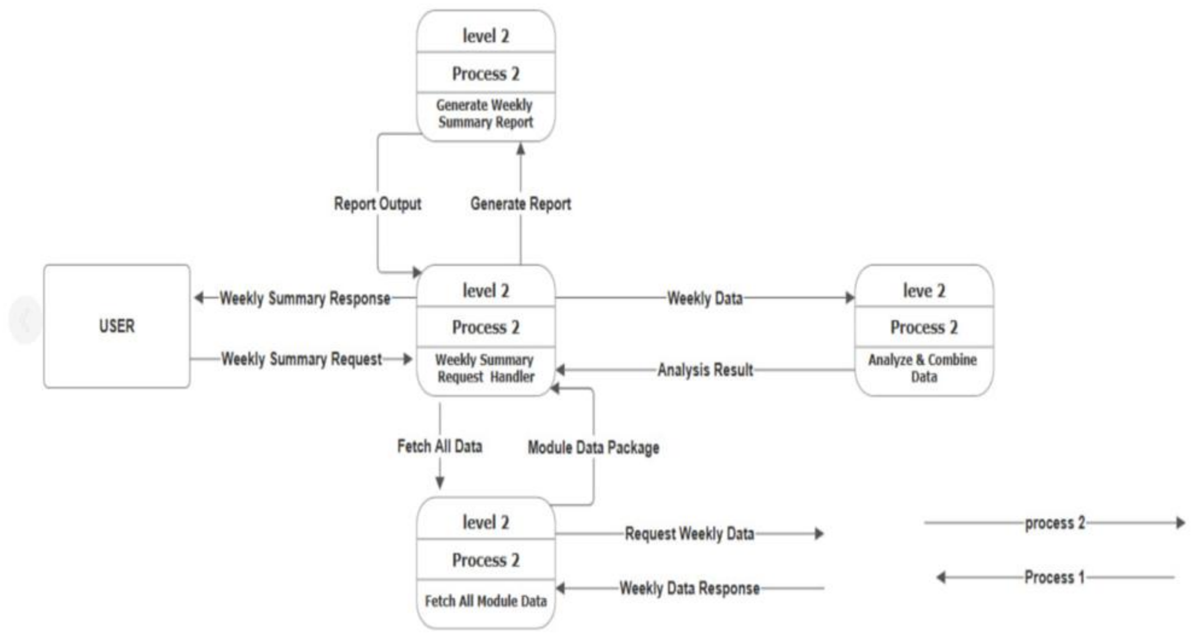
Level – 2 Mood DFD

1.1.2. task



Level – 2 TO-DO list DFD

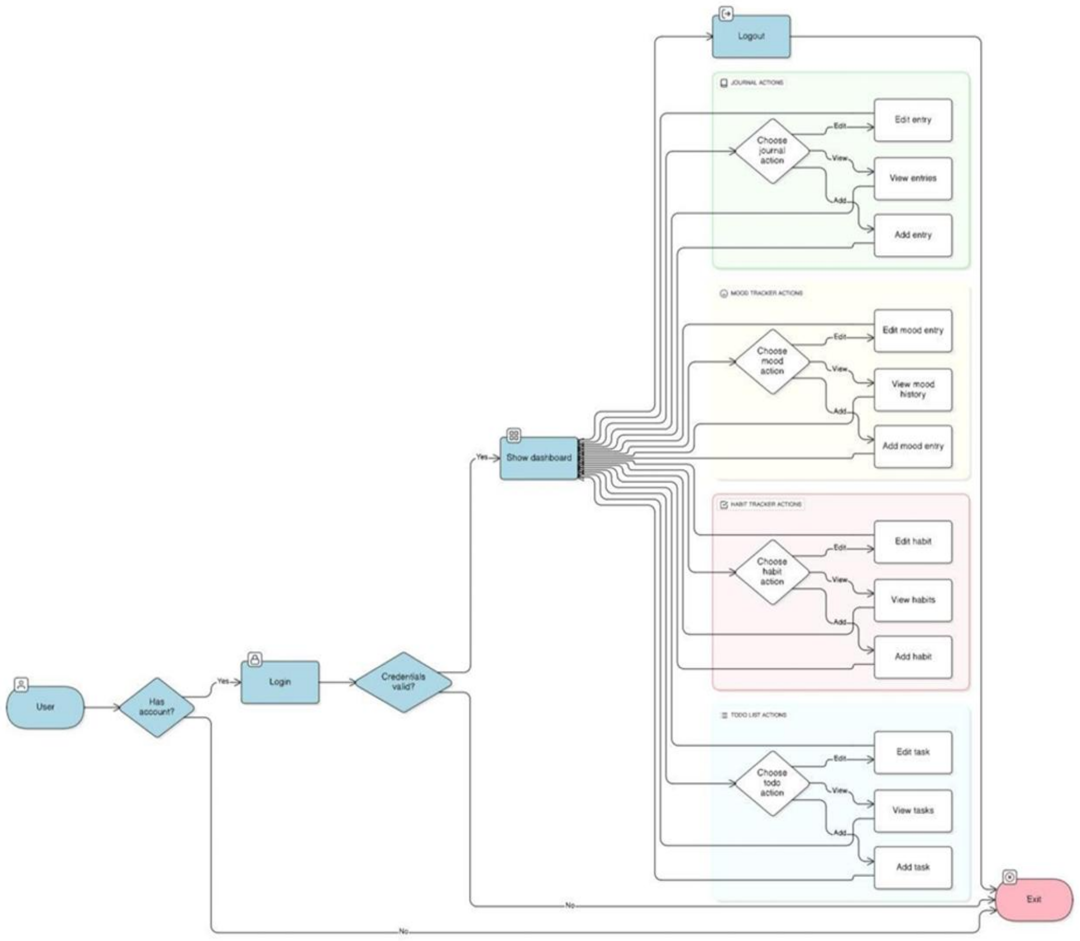
1.1.3. weekly summary



Level – 2 Weekly Summary DFD

The LifeLog DFDs provide a complete view of the system, from high-level interactions to detailed processes. Level-0 shows the overall system handling Journal, Habit, Mood, To-Do List, and Weekly Summary. Level-1 breaks the system into main modules, showing data flow between users, modules, and data stores. Level-2 details each module's sub-processes, illustrating precise operations and data interactions. Together, these diagrams ensure a clear understanding of LifeLog's functionality, supporting efficient design and implementation.

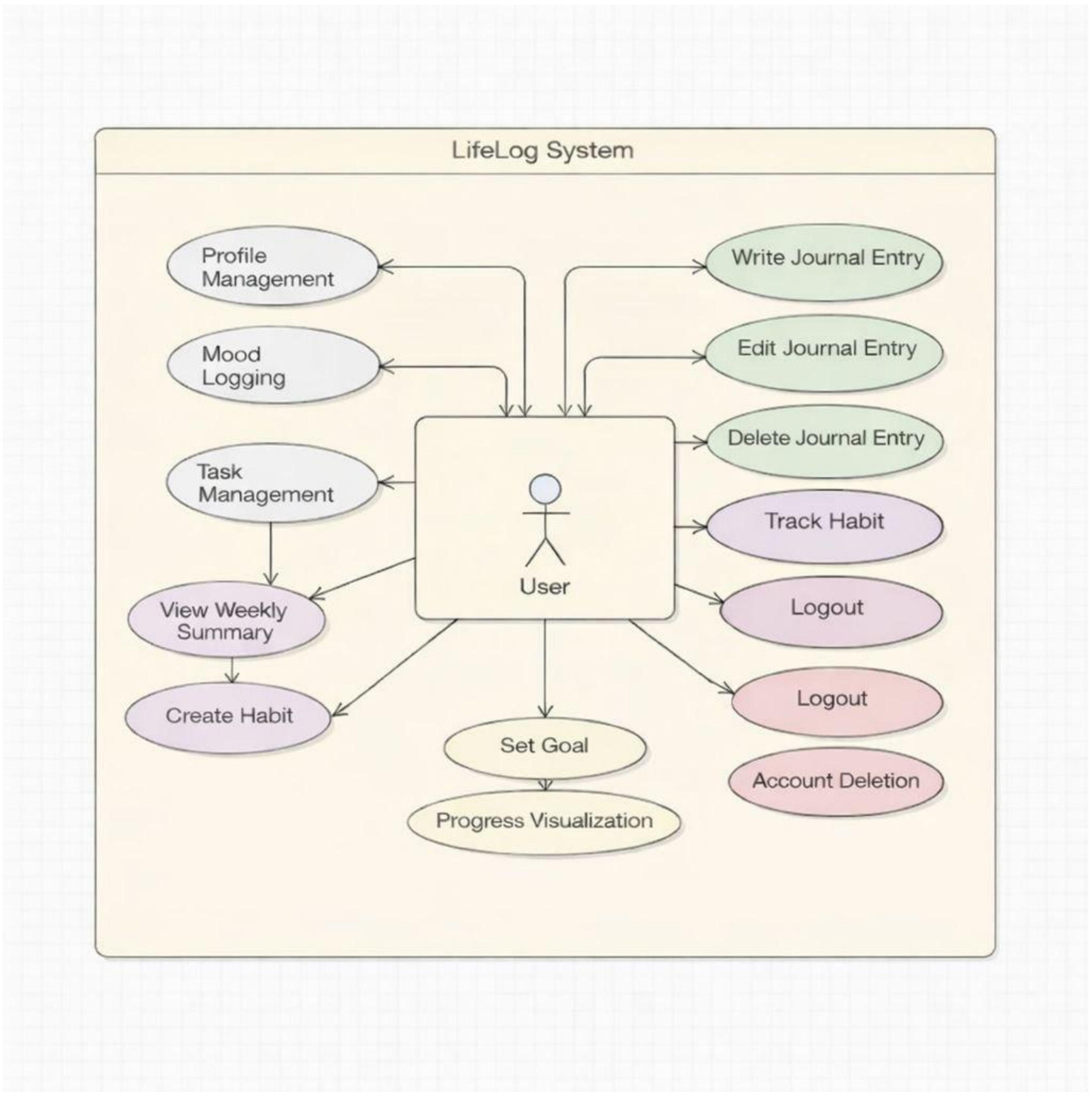
DECISION TREE DIAGRAM



3.3.2. <u>**OO Requirement Modeling**</u>

<u>USE CASE</u>

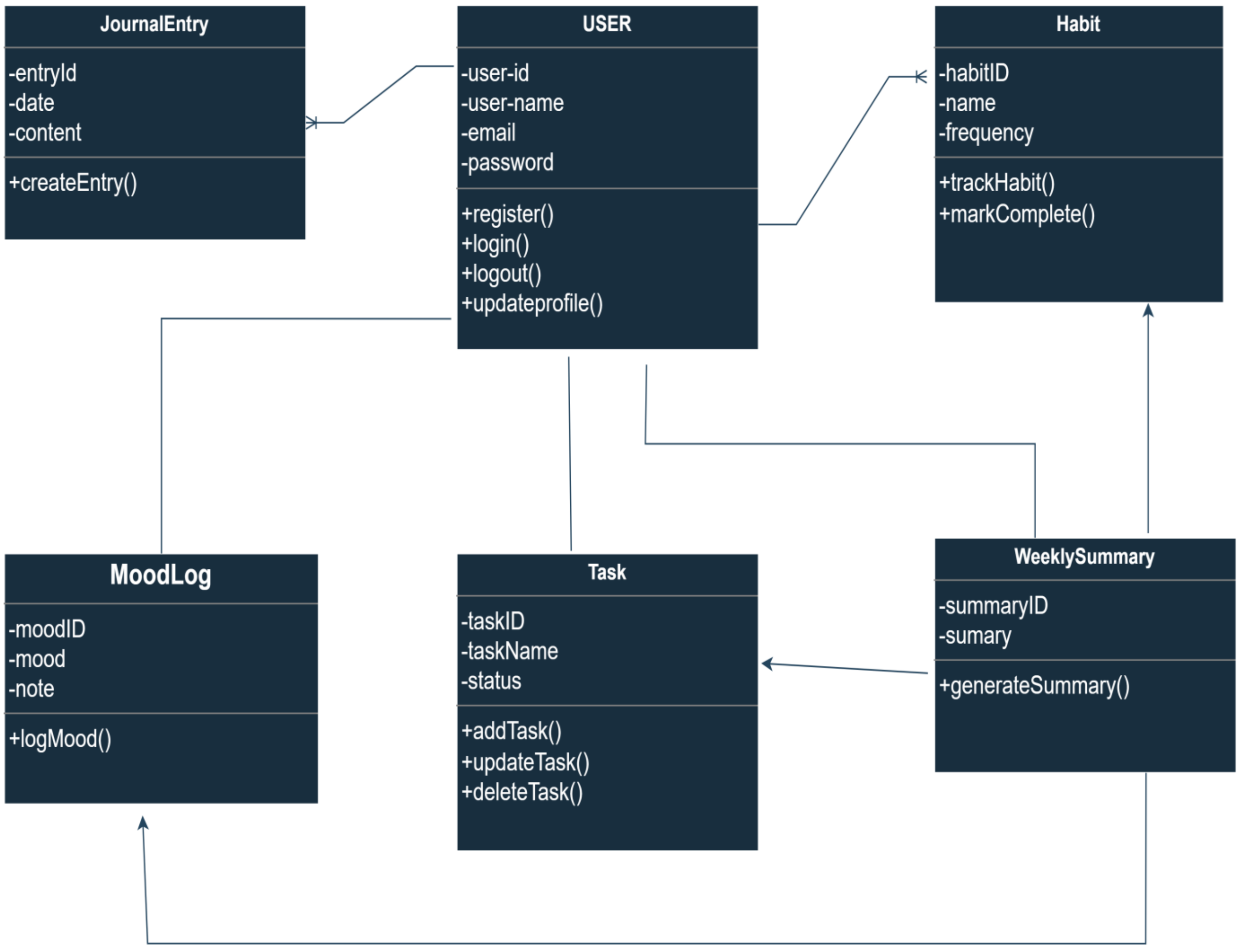
The Use Case Diagram shows the interaction between the User and the LifeLog System. The user is the primary actor who performs all system operations. The main use cases include user registration and profile management, logging moods, writing journal entries, tracking habits, managing tasks, and viewing weekly summaries. These use cases are enclosed within the system boundary, showing that all functionalities are provided by the LifeLog system. The diagram clearly represents the functional scope of the system and how users interact with different features.



CLASS DIAGRAM

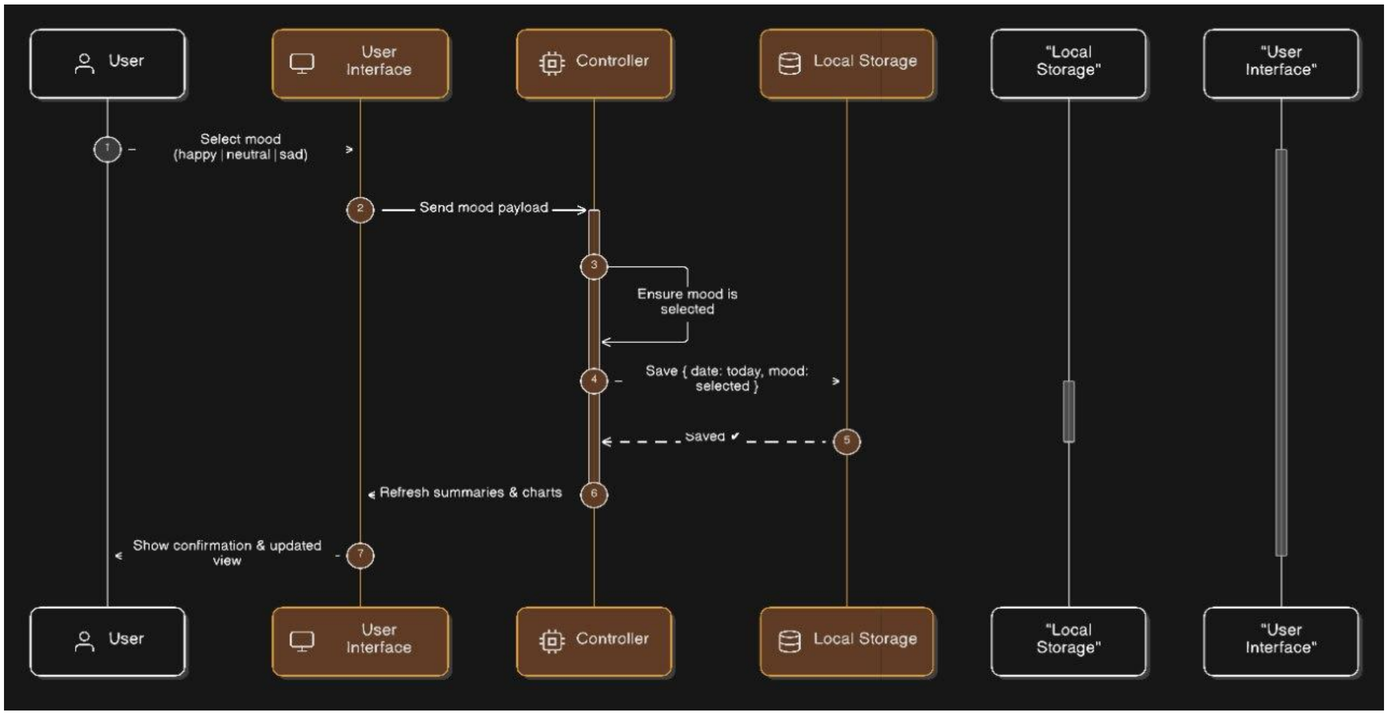
The Class Diagram represents the static structure of the LifeLog system. The **User** class stores user-related information and handles authentication and profile management. The **JournalEntry**,

Habit, **MoodLog**, and **Task** classes represent the core system entities derived from the system's data structures. Each class contains attributes and relevant operations that define its behavior. The **WeeklySummary** class is responsible for generating progress overviews. Relationships between classes show how user data is associated with journals, habits, moods, and tasks.



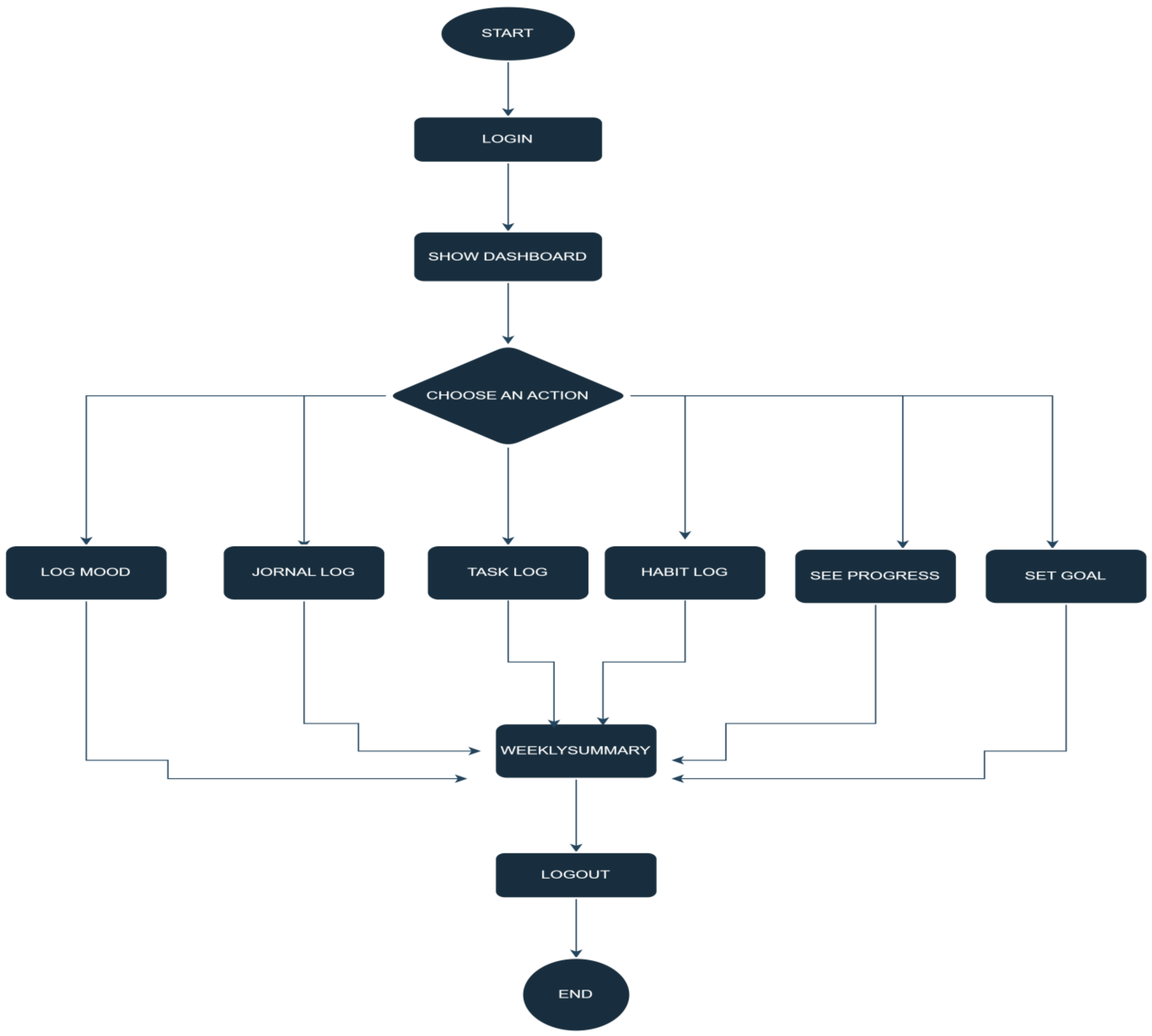
SEQUENCE DIAGRAM(for moodTracker)

The Sequence Diagram demonstrates the interaction flow when a user logs a mood in the LifeLog system. The process begins when the user enters a mood through the user interface. The UI forwards the request to the controller, which processes the data and stores it in Local Storage. Once the mood is saved, the system updates the summary and displays confirmation to the user. This diagram shows the time-ordered communication between system components and highlights how user actions trigger internal processing.



ACTIVITY DIAGRAM

The Activity Diagram illustrates the overall workflow of a user interacting with the LifeLog system. The process starts with user login, followed by dashboard display. From the dashboard, the user can choose to log moods, write journal entries, track habits, or manage tasks. After completing an activity, the system saves the data and allows the user to view summaries. The diagram includes decision points and clearly shows the sequential and parallel flow of activities until the process ends.



CHAPTER 4: SYSTEM DESIGN

4.1 System Architecture

Client–Server Model The system uses a purely client-side architecture. There is no backend server or external API. All application logic, data processing, and storage occur in the user's browser using HTML, CSS, and JavaScript. The browser acts as both the client (UI rendering) and the server (data handling), making it a self-contained application. Database Design Since no real database is used, the system stores all data in the browser's Local Storage. Each module (Journal, Habit Tracker, To-Do List, and Mood Tracker) maintains structured JSON data. Local Storage acts as a key–value store where each feature has its own entry, such as journalEntries, habits, tasks, and moodLogs. System Components The system is divided into functional modules:

● Authentication Module: Handles registration and login using Local Storage to store user credentials.

● Dashboard Module: The main interface that provides navigation to all features.

● Journal Module: Allows users to create, edit, and view journal entries.

● Habit Tracker Module: Tracks daily habits and completion status.

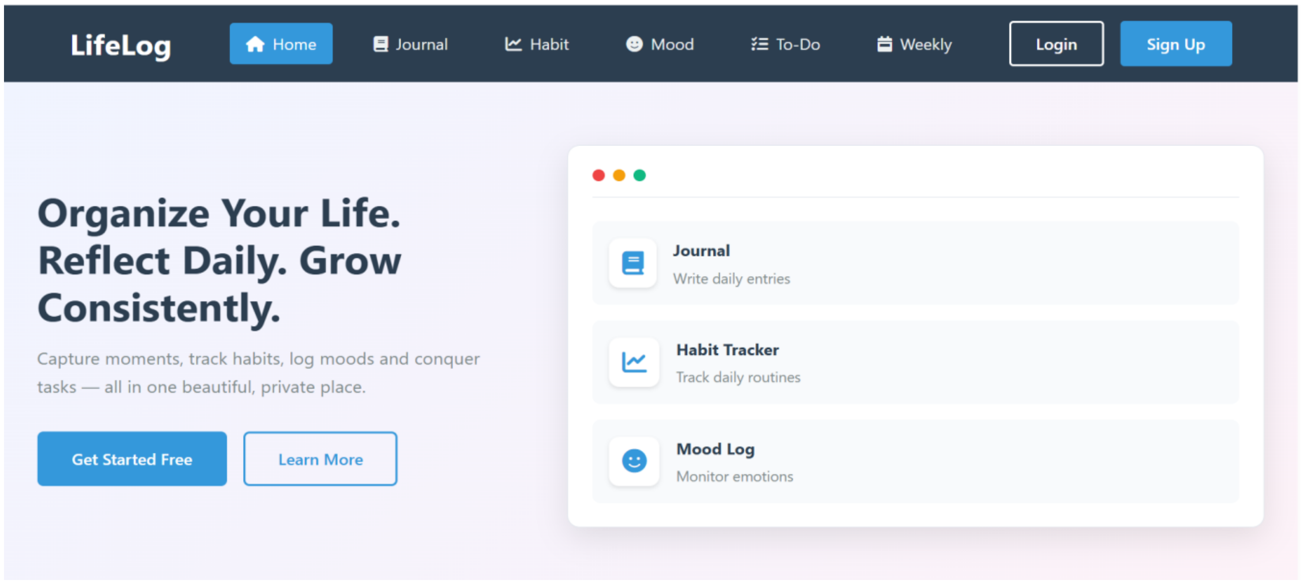
● To-Do List Module: Manages tasks with add, update, and delete operations.

● Mood Tracker Module: Records and displays daily mood logs.

● Summary Module: Provides simple data visualization using charts generated with JavaScript. Each component works independently but shares the same Local Storage layer. Communication Flow User actions trigger JavaScript functions that process inputs, update Local Storage, and refresh the UI. When the application loads, each module reads stored data and displays it. When a user adds or updates something, it is immediately written back to Local Storage, and the display is updated. There is no network communication, as everything happens within the browser.

4.2 Interface Design

Login and Registration Interface The login/registration page provides a simple form where users enter a username and password. The design focuses on clarity and ease of use, with minimal visual distractions. Error messages appear clearly when a user enters invalid credentials. After successful login, the system loads the dashboard. Dashboard Interface The dashboard serves as the central navigation point of the application. It displays shortcuts or buttons to the main modules: Journal, Habit Tracker, To-Do List, Mood Tracker, and Summary. The layout is simple and visually organized so users can quickly access any feature. A visible header or menu includes options like "Home" and "Logout." Journal Interface The journal interface includes a text area for writing entries, input fields for titles or dates, and a list of past entries. Users can create, edit, and delete entries. The design emphasizes readability and a clean writing environment to encourage reflection and emotional expression. Habit Tracker Interface The habit page allows users to add habits and mark daily completion. The interface includes a list of habits, a weekly progress view, and clear buttons for marking habits as done. The layout highlights progress to motivate consistency. To-Do List Interface The to-do list interface features input fields for adding tasks and a list where users can mark tasks as completed or remove them. The design focuses on productivity and simplicity, making task management intuitive and fast. Mood Tracker Interface The mood tracker page presents selectable mood options (e.g., happy, neutral, sad). Users choose a mood, and the system logs it with the current date. The interface may include simple icons or colors for quick mood selection and easy daily usage. Summary / Data Visualization Interface The summary page displays simple charts representing mood trends, habit consistency, or task completion. Since the system uses pure JavaScript, lightweight visualizations (like bar or line charts) are shown using HTML canvas or basic chart libraries that work without frameworks. The design focuses on clarity so users can understand their patterns at a glance.



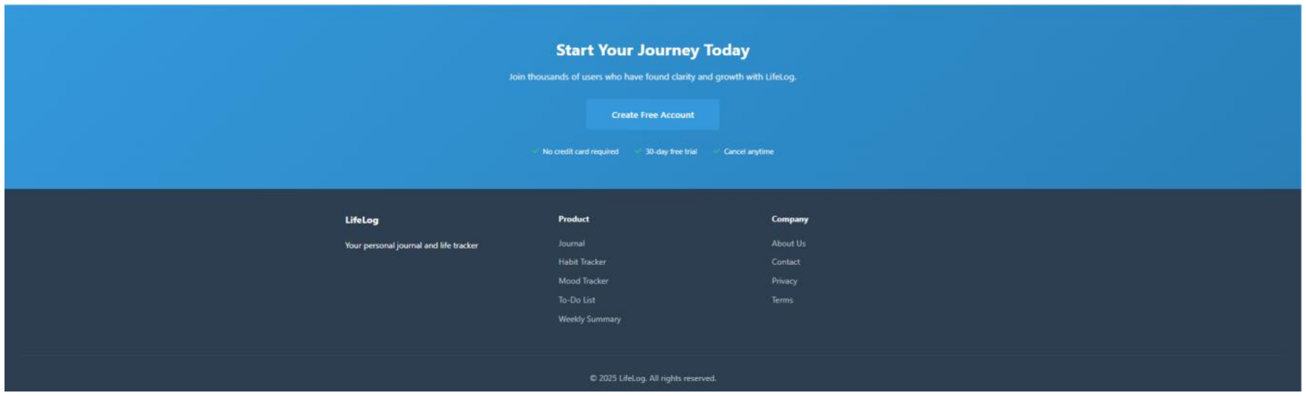
Organize Your Life. Reflect Daily. Grow Consistently.

Capture moments, track habits, log moods and conquer tasks — all in one beautiful, private place.

Journal
Write daily entries

Habit Tracker
Track daily routines

Mood Log
Monitor emotions



Start Your Journey Today

Join thousands of users who have found clarity and growth with LifeLog.

Create Free Account

✓ No credit card required ✓ 30-day free trial ✓ Cancel anytime

LifeLog

Your personal journal and life tracker

Product

Journal

Habit Tracker

Mood Tracker

To-Do List

Weekly Summary

Company

About Us

Contact

Privacy

Terms

© 2025 LifeLog. All rights reserved.

4.3 Database / Data Structure Design

Since the project uses Local Storage instead of a traditional database, all data is stored in the browser as key–value pairs in JSON format. Each module has its own structured data model defined using JavaScript objects and arrays. The following are the main data structures used in the system:

1. **User Data Local Storage Key: users Fields:**
    - username — string — e.g., "john123"
    - password — string — stored as plain text or hashed Purpose: Used for login and basic authentication.
2. **Journal Entry Data Local Storage Key: journalEntries Fields:**
    - id — number — unique ID like 1, 2, 3
    - title — string — e.g., "My Day"
    - content — string — user's journal text
    - date — string — e.g., "2025-01-10" Purpose: Stores all journal entries for writing and editing.
3. **Habit Tracker Data Local Storage Key: habits Fields:**
    - id — number — unique habit ID
    - name — string — e.g., "Exercise"
    - progress — object containing 7 keys (Mon–Sun) with boolean values Example values: Mon: true, Tue: false, etc. Purpose: Tracks daily progress and weekly completion.

4 **To-Do List Task Data Local Storage Key: tasks Fields:**

- id — number — unique task ID

- task — string — e.g., “Finish homework”
- completed — boolean — true or false Purpose: Stores user tasks and their completion status.

5. **Mood Log Data Local Storage Key: moodLogs Fields:**

- date — string — e.g., “2025-01-10”
- mood — string — e.g., “happy”, “sad”, “neutral” Purpose: Records the user’s daily mood for visualization.

4.4 Algorithm / Program Design

This section presents the key algorithms that drive the system’s main features. Each algorithm is written in simple pseudocode to describe how the system processes user actions, updates Local Storage, and displays results.

1. **User Login Algorithm Purpose**: Validate user credentials and allow access. Steps:

- Receive username and password from the login form.
- Load stored user accounts from Local Storage.
- Compare the input credentials with the stored credentials.
- If a match is found:
    - Save the logged-in username as the current active user.
    - Redirect the user to the dashboard.
- If no match is found:
    - Display an “Invalid Credentials” message.

2. J**ournal Entry Creation Algorithm Purpose**: Save a new journal entry. Steps:

- Receive the journal title and content from the input form.
- Generate a new entry containing ID, title, content, and date.
- Load existing journal entries from Local Storage.
- Add the new entry to the list.

- Save the updated list back into Local Storage.
- Refresh the journal display to show the new entry.

3. **Habit Completion Update Algorithm Purpose**: Record a habit as completed on a selected day. Steps:

- Receive the habit ID and the current day.
- Load all habit data from Local Storage.
- Locate the habit that matches the provided ID.
- Update its progress for the selected day to “true.”
- Save the updated habits back to Local Storage.
- Update the progress UI to reflect the change.

4. **Task Creation Algorithm (To-Do List) Purpose**: Add a new task. Steps:

- Receive the task description from the user.
- Generate a task object with ID, description, and a “not completed” status.
- Load existing tasks from Local Storage.
- Append the new task to the task list.
- Save the updated list back into Local Storage.
- Refresh the task list display.

5. **Task Completion Toggle Algorithm Purpose**: Mark a task as completed or uncompleted. Steps:

- Receive the task ID to modify.
- Load tasks from Local Storage.
- Find the task matching the ID.
- Change its “completed” status to the opposite value.
- Save the updated list back to Local Storage.
- Refresh the task display to show the updated status.

6. **Mood Logging Algorithm Purpose**: Store the user's mood for the current day. Steps:

- Receive the selected mood from the user.
- Create a mood entry containing the current date and mood.
- Load existing mood logs from Local Storage.
- Check if a mood entry for today already exists:
    - If yes: update the existing entry.
    - If no: add a new mood entry.
- ● Save the updated list back into Local Storage.
- ● Refresh mood charts or history views.

7. **Summary Visualization Algorithm Purpose**: Convert stored data into meaningful visual charts. Steps:

- Load journals, habits, tasks, and mood logs from Local Storage.
- Compute required summary values:
        - Frequency of each mood type.
        - Percentage of habit completion for each habit.
        - Number of completed vs incomplete tasks.
- Pass these summary values to chart-rendering functions.
- Display the charts on the summary/analytics page.

CHAPTER 5: RISK MITIGATION STRATEGY

5.1 Risk Identification

In the LifeLog project, the main risks that could affect system success include:

1. **Technical/Platform Risk** – The application may not work properly on older browsers or low-spec devices.
2. **Data Loss/Storage Risk** – Using browser Local Storage may result in data loss if the user clears cache or changes devices.

3. **User Engagement Risk** – Users may not consistently use the app, reducing its effectiveness.
4. **Requirement Change/Scope Creep Risk** – New feature requests during development could extend the timeline.
5. **Schedule/Time-Delay Risk** – Delays in development could affect delivery.
6. **Resource/Team Risk** – Lack of technical skills or unavailability of team members may hinder progress.
7. **Security/Privacy Risk** – Weak passwords or shared devices could lead to unauthorized access.
8. **Compatibility Risk** – The interface may not display well across all devices.
9. **Maintenance/Support Risk** – Bugs or user errors may require continuous support.
10. **Cultural/Localization Risk** – The design may not fully match local user preferences (e.g., language, routines).

5.2 Risk Assessment

Each risk was evaluated based on its likelihood and potential impact on the project:

- **Technical/Platform Risk:** Medium likelihood, medium impact. The project depends entirely on browser functionality.
- **Data Loss/Storage Risk:** Medium likelihood, high impact, because lost user data can affect trust.
- **User Engagement Risk:** High likelihood, high impact, since the system's success depends on consistent user use.
- **Requirement Change/Scope Creep Risk:** Medium likelihood, medium impact. Minor changes can be accommodated with Agile sprints.
- **Schedule/Time-Delay Risk:** Medium likelihood, high impact, as delays could postpone deployment.
- **Resource/Team Risk:** Medium likelihood, medium impact, manageable with clear task assignment and planning.
- **Security/Privacy Risk:** Low likelihood, high impact, as user data must remain secure.
- **Compatibility Risk:** Medium likelihood, medium impact; extensive testing can reduce this risk.
- **Maintenance/Support Risk:** Medium likelihood, medium impact, addressed through modular system design.

Cultural/Localization Risk: Low likelihood, medium impact, since LifeLog targets a general audience but may need minor adjustments.

5.3 Mitigation Strategies

To reduce or prevent risks, the following strategies will be implemented:

- **Technical/Platform Risk:** Conduct regular testing during development, use well-supported web technologies (HTML, CSS, JavaScript), and follow best coding practices.
- **Data Loss/Storage Risk:** Implement frequent local backups and encourage users to export their data periodically.
- **User Engagement Risk:** Design a user-friendly interface, provide motivational features such as visual progress tracking, and collect user feedback for improvements.
- **Requirement Change/Scope Creep Risk:** Use Agile methodology with short sprints, allowing flexible and incremental feature development.
- **Schedule/Time-Delay Risk:** Develop a realistic project schedule, monitor progress regularly, and adjust priorities if needed.
- **Resource/Team Risk:** Assign clear responsibilities to each team member, maintain communication, and provide necessary training.
- **Security/Privacy Risk:** Encrypt sensitive user data, ensure secure handling of credentials, and follow privacy best practices.
- **Compatibility Risk:** Test the application on multiple browsers and devices during development.
- **Maintenance/Support Risk:** Document the system design and code clearly, making future updates easier.
- **Cultural/Localization Risk:** Conduct user testing and gather feedback from the target audience to ensure the interface is understandable and usable.

By following these mitigation strategies, LifeLog aims to minimize risks and ensure successful deployment and sustained use.

CHAPTER 6: CONCLUSION & RECOMMENDATIONS

6.1 conclusion

The LifeLog system was created as an all-in-one digital solution designed to help users improve productivity, emotional awareness, and personal organization. It brings together journaling, mood tracking, habit monitoring, and task management into one simple, easy-to-use platform. Using the Business Process Improvement (BPI) approach, the project examined weaknesses in current self-management tools and introduced a system that enhances user interaction, data visualization, and overall self-tracking efficiency. Findings from the comparative and gap analyses revealed that existing applications such as Daylio, Journey.Cloud, and Trello each cover only specific aspects of self-management. LifeLog overcomes this limitation by integrating all essential features—habit tracking, journaling, mood monitoring, and goal management—within a single, browser-based system. Built entirely with HTML, CSS, and JavaScript, it operates smoothly without requiring any external server, making it lightweight, platform-independent, and easily accessible. User data, including habits, mood entries, and progress records, are stored locally using the browser's Local Storage, ensuring efficient and secure information handling. The system was developed in accordance with both functional and non-functional requirements, prioritizing usability, speed, data safety, and dependability. Its modular architecture and well structured data design allow for simple maintenance and future upgrades. In addition, potential risks such as data loss, storage limitations, and user retention challenges were carefully identified and addressed through mitigation strategies. In summary, LifeLog successfully delivers a user-friendly and reliable self-management system that supports users in tracking their habits, emotions, and goals in one place. It encourages personal growth and consistency through visual progress tracking and an accessible interface. Looking ahead, the system can be further improved by adding cloud synchronization, AI-based personalized insights, and a mobile version to enhance flexibility and reach a wider audience.

6.2. Recommendations

Based on the development and evaluation of the LifeLog system, the following recommendations are suggested for future improvements and additional features:

1. Cloud Synchronization: Implement cloud storage to allow users to access their data across multiple devices, ensuring seamless continuity and backup in case of device failure.

2. Mobile Application: Develop a dedicated mobile version (iOS and Android) to increase accessibility and provide users with on-the-go tracking capabilities.

3. AI-Powered Insights: Integrate AI algorithms to analyze user data and provide personalized recommendations, trend predictions, and habit improvement suggestions.

4. Enhanced Data Visualization: Introduce interactive charts, heatmaps, and dashboards to give users deeper insights into their moods, habits, and goal progress.

5. Social or Community Features: Allow optional sharing of achievements or participation in challenges to enhance user motivation and engagement.

6. Reminder and Notification System: Implement intelligent reminders for habits, tasks, and journaling to encourage consistency without being intrusive.

These improvements aim to increase flexibility, personalization, and overall user engagement while maintaining the system's simplicity and usability.

6.3. Plan for System Construction, Testing, Conversion & Support

1. **System Construction:**

    - Use HTML, CSS, and JavaScript for front-end development, with Local Storage for initial data handling.
    - For future versions, integrate cloud storage APIs for synchronization across devices.

2. **Testing:**

- Conduct unit testing for individual modules (habit tracker, mood journal, task manager).
- Perform integration testing to ensure smooth interaction between modules.
- Carry out user acceptance testing (UAT) with real users to gather feedback on usability and functionality.
- Test performance, security, and data storage limits to prevent potential data loss.

3. **Conversion & Deployment:**

- Deploy the browser-based system on a web server or host it as a Progressive Web App (PWA) for easy access.
- For mobile versions, convert the system into native or hybrid apps for iOS and Android platforms.
- Provide data migration tools for users to transfer existing data to future cloud-enabled versions.

4. **Support & Maintenance:**

- Offer user guides, FAQs, and in-app tutorials to assist new users.
- Monitor system usage and collect feedback for ongoing improvements.
- Regularly update the system to fix bugs, enhance security, and introduce new features.
- Maintain modularity in code to facilitate updates without affecting existing functionality.

This structured approach ensures that LifeLog remains reliable, user-friendly, and adaptable to evolving user needs

Reference

[1] https://dynamics.folio3.com/blog/bpm-vs-bpi-vs-bpr-vs-bpe/

[2] https://www.geeksforgeeks.org/business-studies/benchmarking-steps-and-types/

[3] https://daylio.net/faq/docs/daylio-faq/tutorials/

[4] https://trello.com/

[5] https://journey.cloud/

[6] https://www.sciencedirect.com/topics/computer-science/risk-management

[7]https://uir.unisa.ac.za/items/9f65dbd0-30ea-4990-95d1-4864901033ff

[9] https://www.ijert.org/role-of-risk-management-in-development-of-software-projects

[4] https://ouci.dntb.gov.ua/en/works/4g5PW309/