ITAHARI INTERNATIONAL COLLEGE

LONDON METROPOLITAN UNIVERSITY

Module Code & Module Title

CS6OO4NT Application Development

30% Individual Coursework

Submission: Final Submission

Academic Semester: Autumn Semester 2025

Credit: 30 credit yearlong module

Student Name: Karuna Giri

London Met ID: 23049261

College ID: np05cp4a04230127

Assignment Submission Date: January 28, 2025

Assignment Due Date: January 28, 2025

Submitted To: Mr. Khilendra Chaudhary

Git hub link: https://github.com/Karuna1630/DailyReflect_KarunaGiri

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The DailyReflect project is a coursework in the CS6004NT Application Development course that entails the designing and development of a secure desktop journaling prototype using networking 1.NET. It is a personal journaling app that uses features like individual entries per day with rich-text/Markdown support, extensive mood tracking in Positive/Neutral/Negative categories, intelligent tagging, analytics dashboard, calendar navigation and local SQLite storage alongside ensuring full privacy through password/PIN authentication.

This report is a record of the full process of development following the 3 stages, i.e., repository init and wireframes (Week 9), core feature implementation (Week 11), and full completion (Week 13). DailyReflect was developed using.NET MAUI as a cross-platform compatible application, so it adheres to software engineering standards such as modularity, managing versioning and maintenance with Git, error management, user experience design, and ensuring all 12 core features and quality attributes requirements are not only met but fulfilled in the marking scheme.

1.1 Project Overview

DailyReflect is a safe and well-rounded desktop journaling software, written in C#.NET, that aims to upgrade the old-fashioned method of personal journaling. The app will offer an individual and well-structured space where users can record their experiences, thoughts, and feelings, on a daily basis, in a steady and systematic way. To prevent inaccuracy in data and tracking habits, the system only lets one journal entry be created, updated, and deleted per day where all entries are automatically allocated system generated creation and update timestamps. Rich-text or Markdown support allows the creation of well-structured and expressive material.

One of the major characteristics of the DailyReflect is the extensive mood tracking and categorization mechanism. The users are asked to choose one main mood and are allowed to use an optional two additional moods, marked positive, neutral or negative. The additional sorting of journal entries may be achieved with custom or pre-defined tags and category levels, which improves the content management and search engine potential. The app has user-friendly calendar navigation, paginated view of journal and powerful search and filter functions to help users find their entries effectively according to date ranges, moods, tags or text.

1.2 Project Purpose and Scope

1.2.1 Project Purpose

The aim of the DailyReflect application is to redesign the traditional journaling experience with a secure, structured, and friendly desktop application that was designed with the C#.NET language. The application will serve to assist users to always track their day-to-day thoughts, experiences, and feelings and maintain accuracy, privacy, and meaningful self-reflection utilization of analytics and insights.

1.2.2 Project Scope

DailyReflect is a journal system that is a desktop-based system aimed at individual users. The app also permits the user to add, edit, and delete a journal entry in one day, which is enough to have a good habit-tracking history and streak-counting. Creation and update times are system generated in each journal entry. The system has support of rich-text or Markdown, which allows to create the content in a form of expression and organization.

The application will have mood tracking with one primary mood and two secondary moods, optional and with three categories positive, neutral, and negative. Custom or pre-defined tags and categories can be used by the users to categorize entries. It provides navigation using a calendar view, paginated list of journals and support advanced search and filtering.

The full data is safely stored in a locally running SQLite database, with the aid of password/PIN authentication. Other functions are customization of theme and exporting journal entries in PDF files in a given range of dates.

1.3 Aims and Objectives

1.3.1 Aim

To develop DailyReflect, a secure and full-featured C#.NET desktop journaling application rebranding personal reflection by adding a structured-daily reflections with support for rich-text / Markdown, mood tracking across Positive/Neutral/Negative, intelligent tagging and categorization, calendar based navigation, advanced search / filtering, streak detection, analytics dashboard showing mood distribution and word-count trends, theme customization, PDF export and strong PIN-based authentication - all locally stored in SQLite and therefore, providing full privacy and showing itself to be cross-platform.

1.3.2 Objectives

The major purposes of the DailyReflect application are:

- The functionality of this objective is to give a secure and confidential journaling platform based on local storage and authentication.
- To promote streak tracking and longest streak calculation as well as missed-day detection to promote daily journaling.
- To allow users to monitor and visualize emotional trends through mood-based analytics and visual dashboard indications.
- To enhance the content organization and retrieval using tags, categories, search and filter facilities.
- To provide usability and customization via theme customization and PDF export.
- To develop a good desktop application based on good software development practices.

1.4 Outcomes and Deliverables

1.4.1 Technology Stack Selection

For the completion of this project, the choice of the environment in which the development will be held is needed. To this, it resulted in the necessary research on the Framework and the external libraries that should be utilized. The reason behind this is to have adequate knowledge before the actual implementation of the project as the requirements should be learnt and a proper plan laid out on the same. The choice of the technology stack can be characterized in as follows:

Framework: The NET MAUI (.NET Multi-platform App Ui) is a tool that enables cross-platform application development on top of a single C# and XAML codebase by Microsoft to produce both native mobile and desktop applications. It allows the developers to target Android, iOS, macOS, and windows in a single project without having to maintain distinct code base. Since the development of Xamarin.Forms, .NET MAUI belongs to the larger single.NET platform, where it is possible to share the code, libraries and tools with all backend, web, and client applications. The system comes with native user interface controls, platform-specific API access, productivity features, including hot reload. .NET MAUI allows development streamlining and maintainability improvements and reduces long-term costs, and provides performant and native-feeling applications by sharing much of UI and business logic with many other platforms. (microsoft, 2025)

Mechanism of Persistence: SQLite is an in-memory relational database engine that is incorporated into the application instead of a standalone database server. It holds the complete database tables, indexes and data within a single cross-platform file on disk making it easy to duplicate, backup and implement. SQLite is a self-sufficient database engine that is serverless and needs no additional configuration and, therefore, allows applications to begin using it without needing to install a separate database service or manage it. It is fully ACID-compatible as well as a standard SQL despite being small, and is popular with desktops (software and browsers), mobile apps, and embedded systems due to its consistent reliability and efficiency. (sqlite, 2025)

External Libraries:

QuestPDF 2025.12.1 is used to generate professional PDFs through exporting journal data to complex layouts with tables, charts, and styled mood/tag reports with no reliance on any HTML. (Szymanowski, 2025)

The Quill.js 1.3.6 is a rich-text/Markdown editor with full formatting options (bold, italics, list, links), and based on the XML Delta data format to store the content in a consistent manner across platforms. (quilljs, 2026)

Bootstrap offers CSS framework that is responsive and grid system, components, and utilities needed to develop user-friendly dashboards, paginated journal lists, and the ability to switch between themes. (getbootstrap, 2026)

1.4.2 Git Repository

For version control and source-code submission, the repository with the name of Project and student have to be created in the GitHub. For this, the repo has been created with named “DailyReflect_KarunaGiri” which is mentioned below:

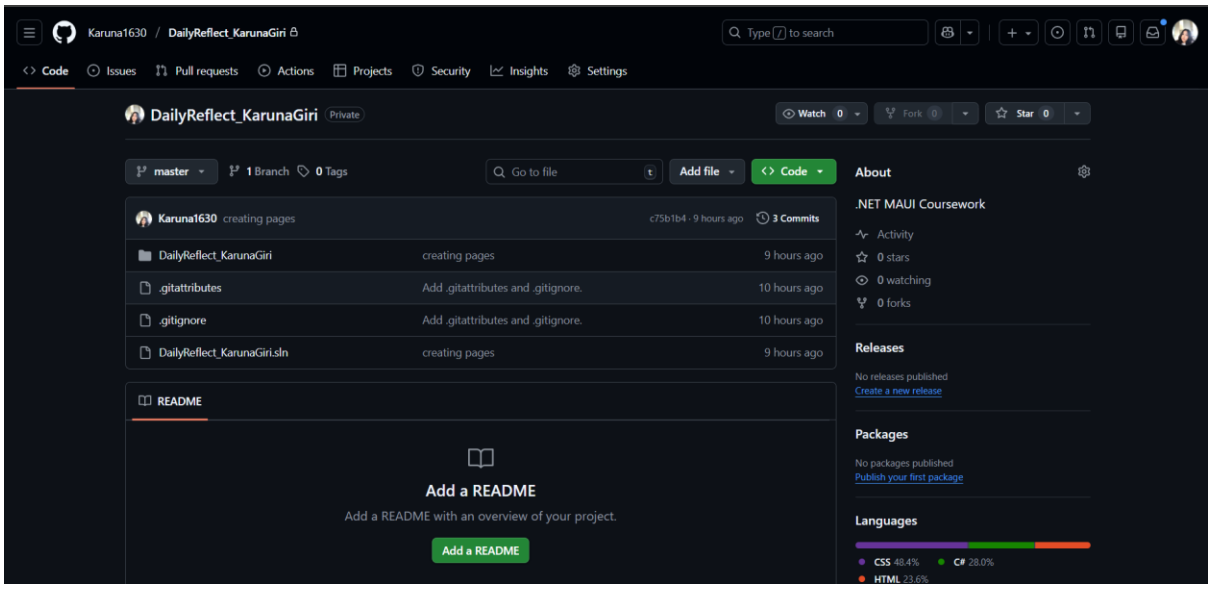

Figure 1: Screenshot of Repository

The repository link is mentioned below:
https://github.com/Karuna1630/DailyReflect_KarunaGiri

2 Features and Functionalities

2.1 Features Breakdown

Table 1: Table of Features Breakdown

S.N.	Features	Status	Description
1	Journal Entry Management	Completed	Create, update, delete a single entry everyday with auto-generated system created at and updated at timestamps to be able to keep track of habits correctly.
2	Rich Text/Markdown Writing	Completed	Expressive contents Have facilities to fully format (bold, italics, lists, headings, links) my articles through Quill.js 1.3.6 editor with real-time preview.
3	Mood Tracking	Completed	Emotional analytics What descriptions are needed Primary and at least 3 in the form of Positive/Neutral/Negative (Happy, Stressed, Calm, and...)?
4	Tagging System	Completed	20+ ready-made tags (Work, Health, Travel, Fitness, Relationships) and infinity number of custom tags to organize and filter the content.
5	Calendar Navigation	Completed	Visual calendar which shows the days of entry vs the days of missing direct date navigation to particular journals.
6	Paginated Journal View	Completed	The timeline/list view with pagination that can easily navigate through the entries of the history.
7	Search/Filter	Completed	Title/content search Full-text search by title/content along with date range search and tags (to accurately find an entry).
8	Streak Tracking	Completed	Live current daily streak and long streak and missed days display to help create consistency.
9	Theme Customization	Completed	Light/dark mode akin to all-purpose lights with default user setting to the best reading conditions.

10	Dashboard Analytics	Completed	Mood distribution charts, frequent moods/tags, Slice of category by date range.
11	Security/Privacy	Completed	SQLite encryption of PIN/Password protection with local data privacy.
12	Journal Export	Completed	Date range-based professional PDF generation with formatted content and moods and tags via Quest PDF.

2.2 Description of Features

The DailyReflect app offers an integral collection of useful features that give way to safe, regular, and valuable daily journaling. These features are applied with respective system functionality and are accompanied by specific user interface components, as shown in the wireframes.

2.2.1 Authentication and Security

DailyReflect has the privacy of the user that is semi-secured by the usage of passes or PIN. The journal data can only be accessed by users after conducting an authentication to avoid unauthorized access. All journal data will be stored in a local SQLite database where devices will ensure the data is confidential.

Evidence:

Setting the pin for Journal Protection.

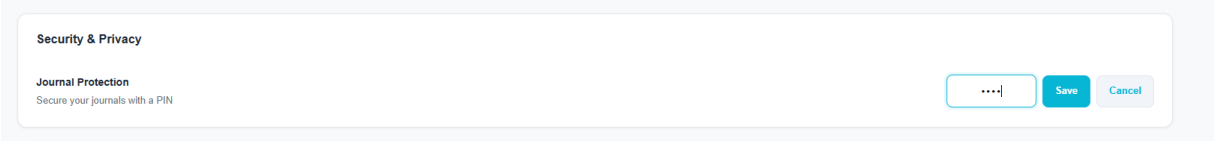

Figure 2: Figure of setting pin

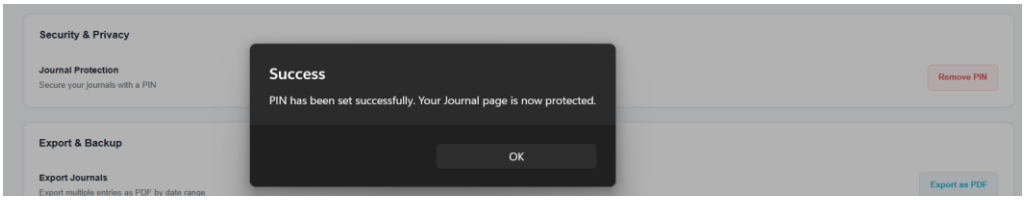

Figure 3: Successfully setting pin

Accessing the Journal by entering the pin.

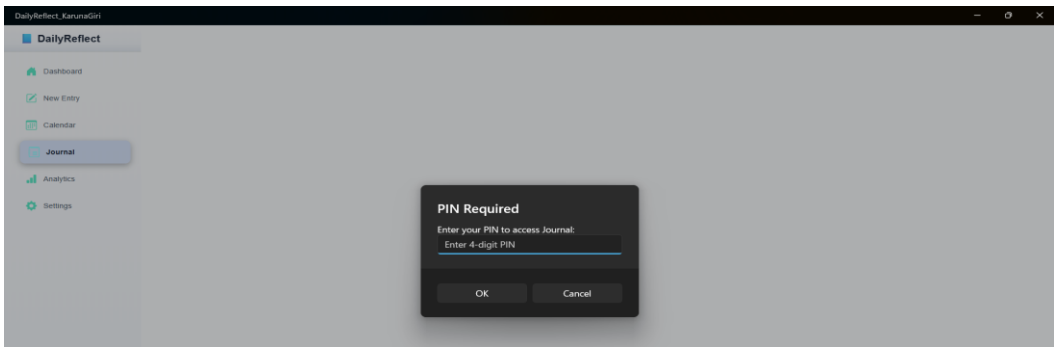

Figure 4: Figure of user entering pin

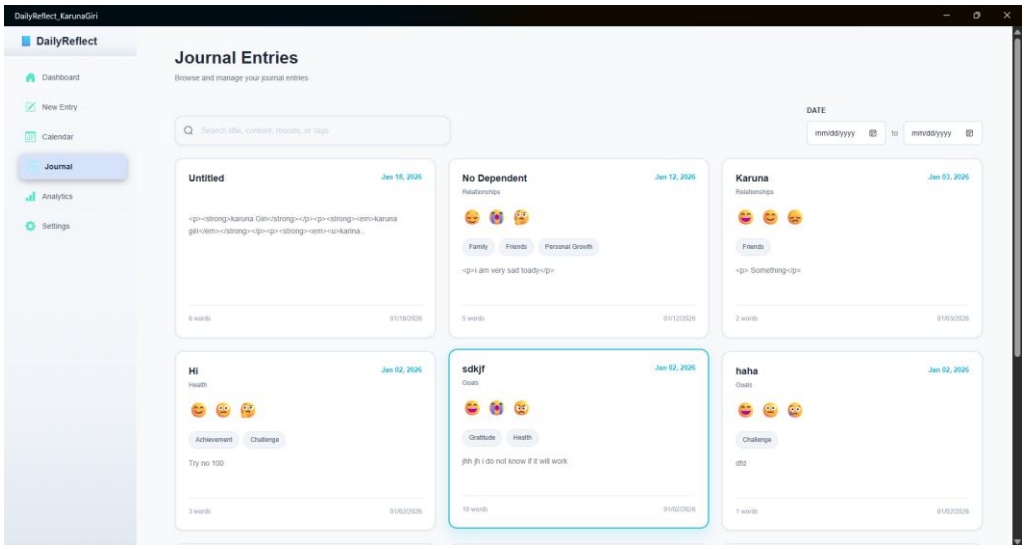

Figure 5: Successfully accessing journal

2.2.2 Journal Entry Management (CRUD)

The application will enable the user to add, edit, and remove one journal entry every day. This limitation leads to proper tracking of the habits and streaks. System-generated modified and modified dates are added to each record. Journal material advertises rich-text or Markdown coding which allows expressive writing and well-organised writing.

Evidence:

Creating a journal entry per day.

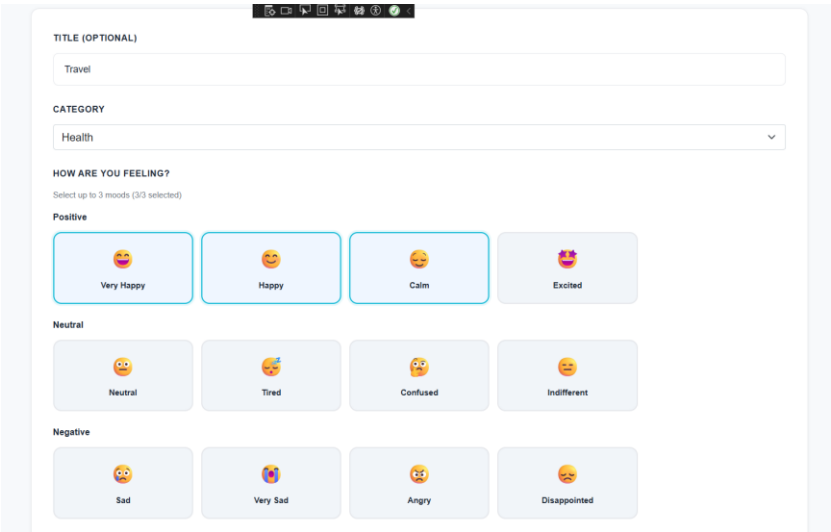

Figure 6: Figure of creating a journal

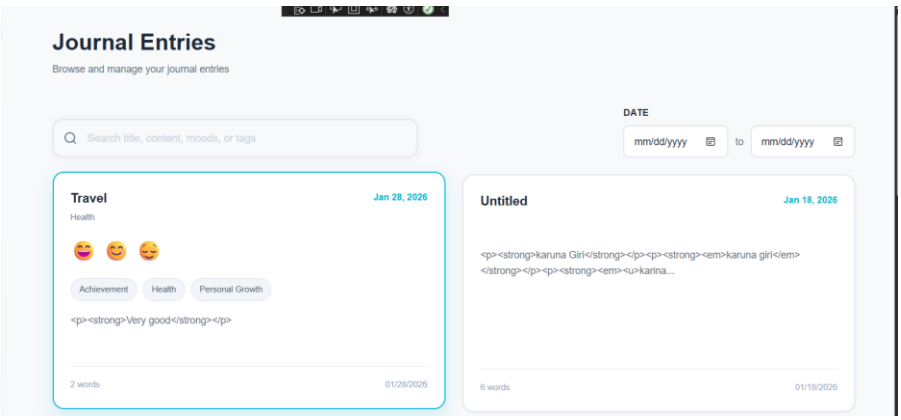

Figure 7: Journal appear in the Journal page

Updating only Recent added Journal Entry.

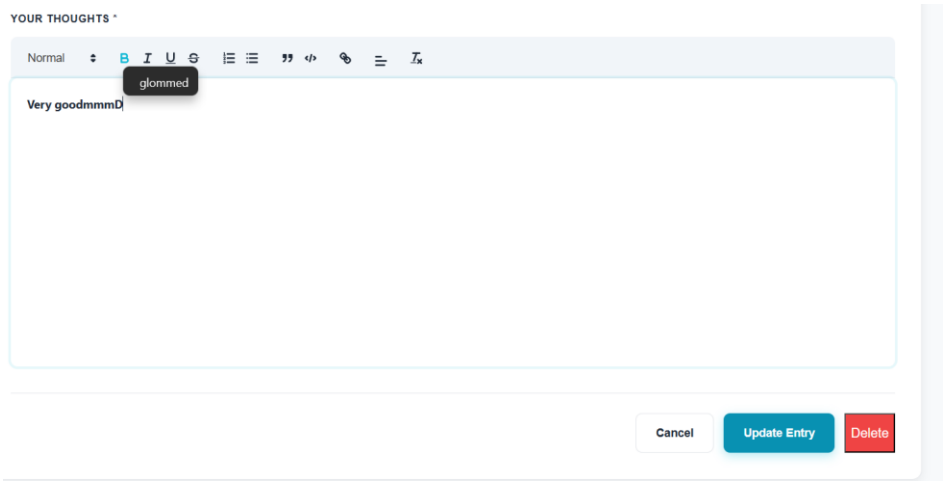

Figure 8: Updating Recent Journal Only

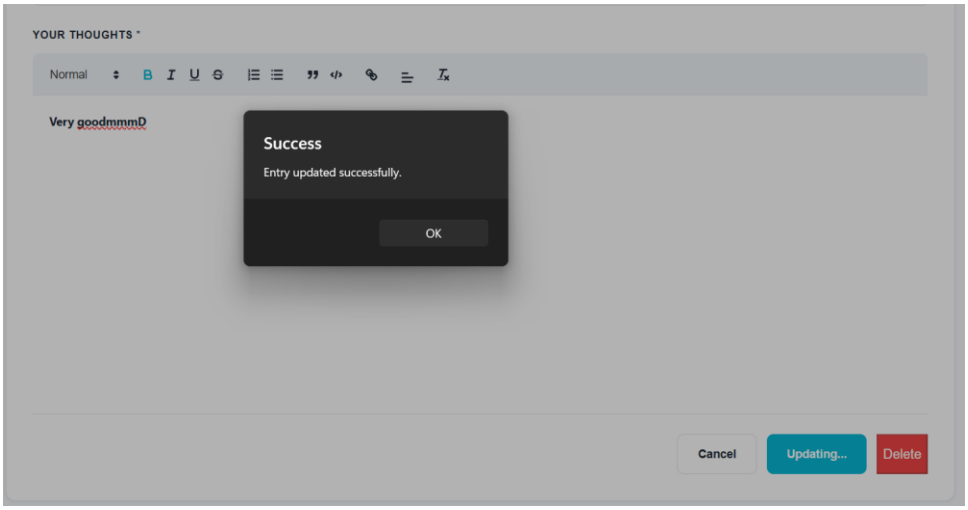

Figure 9: Successful message of Updated

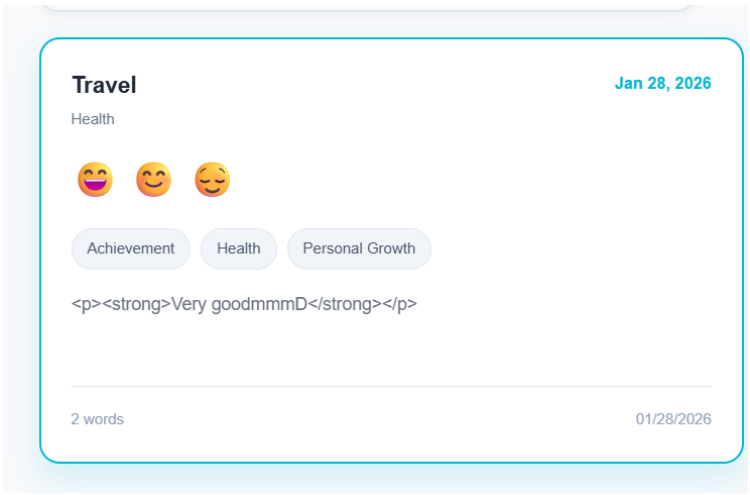

Figure 10: Recent Journal Updated Successfully

Removing Only Recent added Journal Entry.

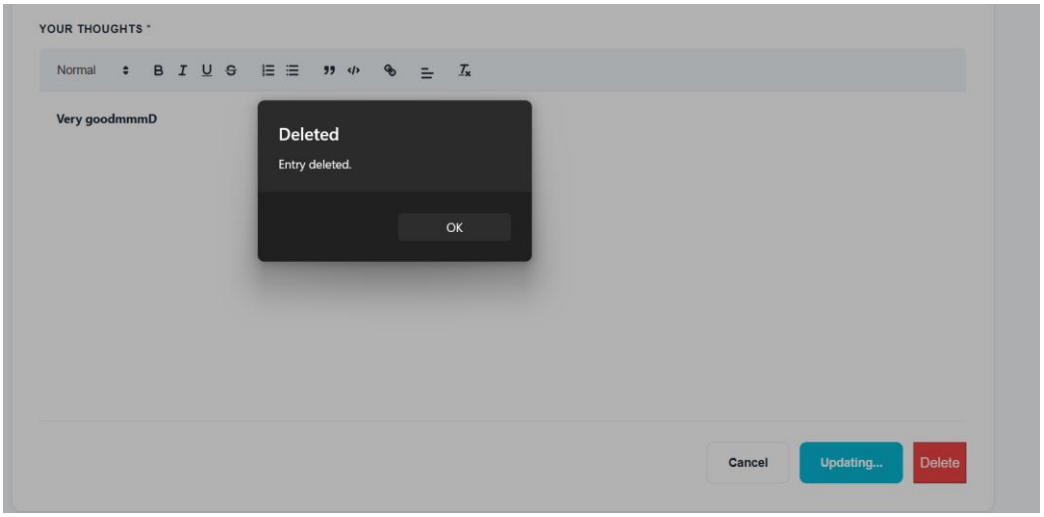

Figure 11: Removing Recent Journal

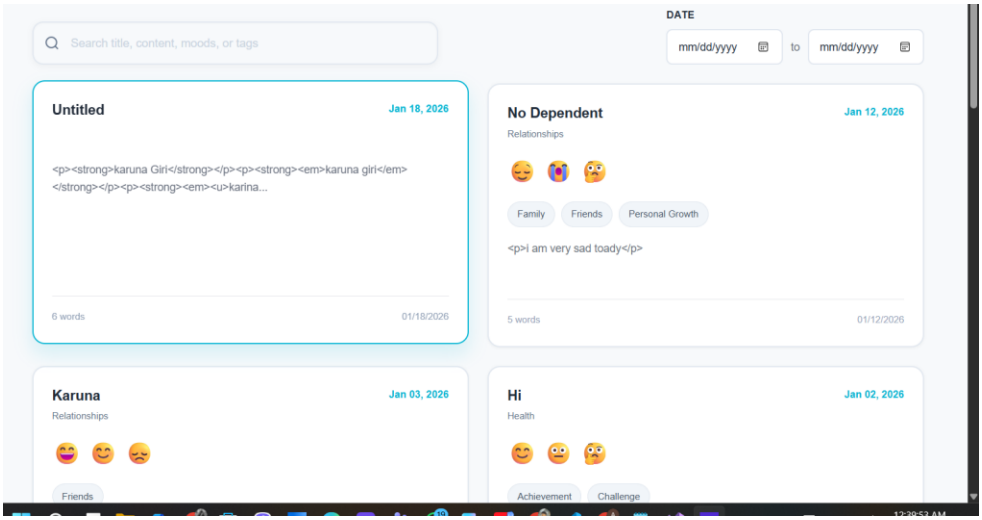

Figure 12: Journal Removed Successfully

	Id	EntryDate	Title	Category	Mood	
	Fil...	Filter	Filter	Filter	Filter	Filter
1	1	2026-01-02 00:00:00	Hi	Health	Happy,Neutral,Confused	Achievement,Cha
2	3	2026-01-02 00:00:00	sdkjf	Goals	Very Happy,Very Sad,Angry	Gratitude,Healt
3	4	2025-12-25 00:00:00	Christmas Day	Personal	Very Happy,Excited,Grateful	Family,Gratitud
4	5	2025-12-26 00:00:00	Boxing Day Reflections	Personal	Happy,Calm	Reflection,Fami
5	6	2025-12-27 00:00:00	Back to Routine	Work	Neutral,Tired	Work,Goals
6	7	2025-12-28 00:00:00	Gym Day	Health	Happy,Excited	Health,Achievem
7	8	2025-12-29 00:00:00	Year End Thoughts	Mindfulness	Calm,Confused	Reflection,Grat
8	9	2025-12-30 00:00:00	Final Day of 2025	Personal	Happy,Nostalgic	Reflection,Frie
9	10	2025-12-31 00:00:00	New Year Eve!	Goals	Very Happy,Excited,Confident	Goals,Personal
10	11	2026-01-01 00:00:00	Happy New Year 2026!	Personal	Very Happy,Excited	Goals,Gratitude
11	12	2026-01-02 00:00:00	haha	Goals	Very Happy,Neutral,Sad	Challenge
12	13	2026-01-03 00:00:00	Karuna	Relationships	Very Happy,Happy,Disappointed	Friends
13	14	2026-01-12 00:00:00	No Dependent	Relationships	Calm,Very Sad,Confused	Family,Friends,

Figure 13: Successfully Removed from Database

2.2.3 Mood Tracking System

The user is required to pick a single primary mood which is obligatory to analytics and optionally a secondary mood which can be picked twofold. The moods are sorted into positive, neutral and negative which helps analyse and visualise emotional patterns in analytics dashboard.

Evidence:

User can select up to 3 moods only at a time.

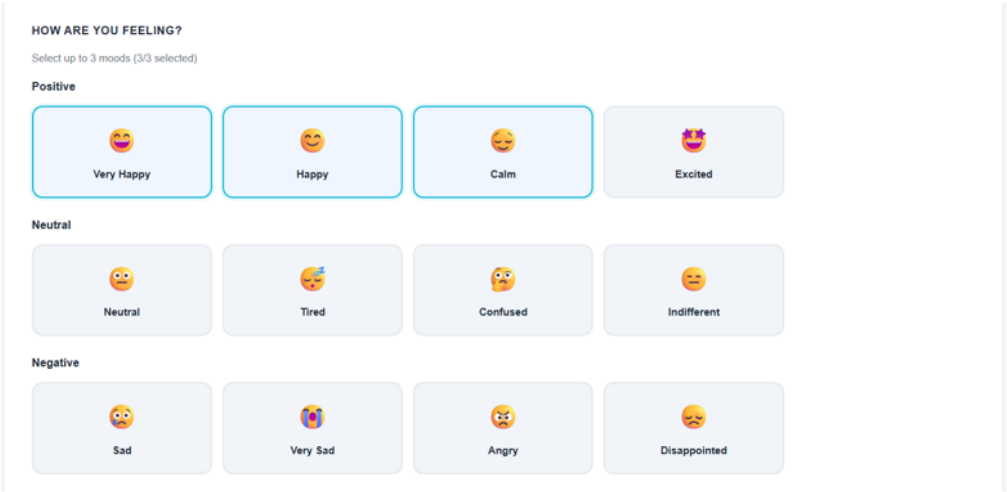

Figure 14: Figure of Mood Selection

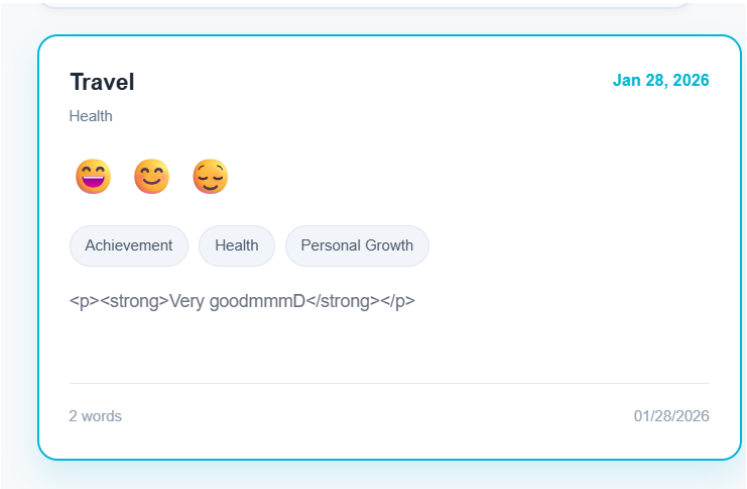

Figure 15: Successfully Mood Selection

2.2.4 Tag and Category Management

The entries in journals can be marked with custom or preset tags and categories that users set to enhance organisation and searchability. Health, Travel, Work and Personal Growth tags allow classification to be made efficiently and the features of filtering and analytics.

Evidence:

User can used different predefined tags as well.

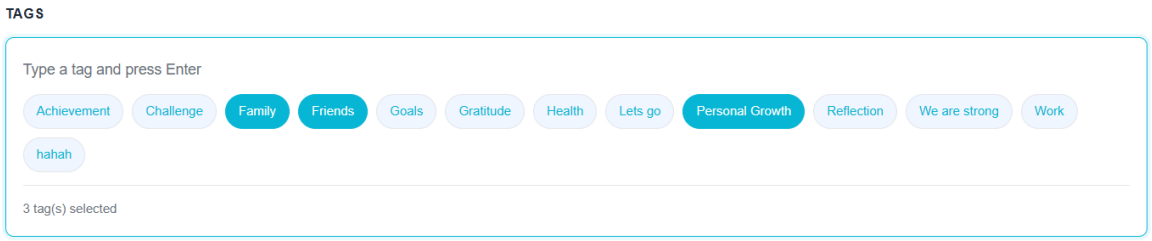

Figure 16: Figure of predefined tags

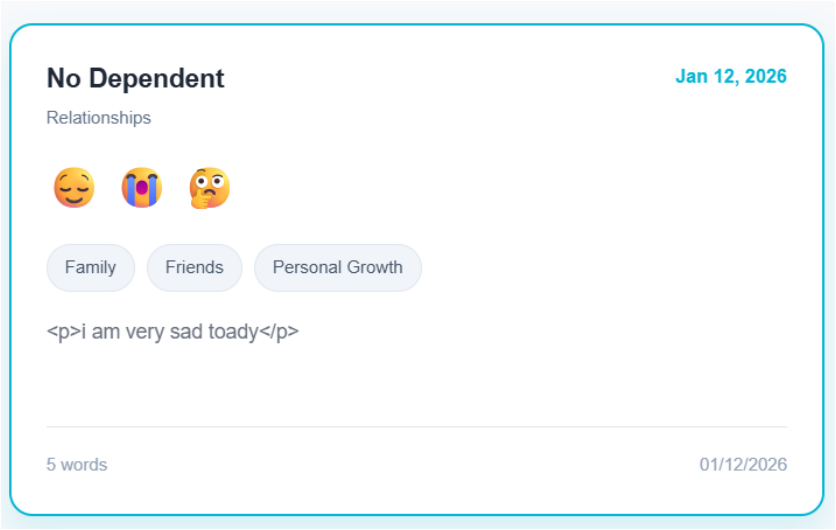

Figure 17: Successfully attach in Journal

2.2.5 Calendar-Based Navigation

The application also has a calendar view which is visually used to show the days with journal entries and missed days. Users are able to jump into certain dates very fast and accessibility is enhanced but journaling is also encouraged.

Evidence:

It shows the days in which journal entries had done with blue tick.

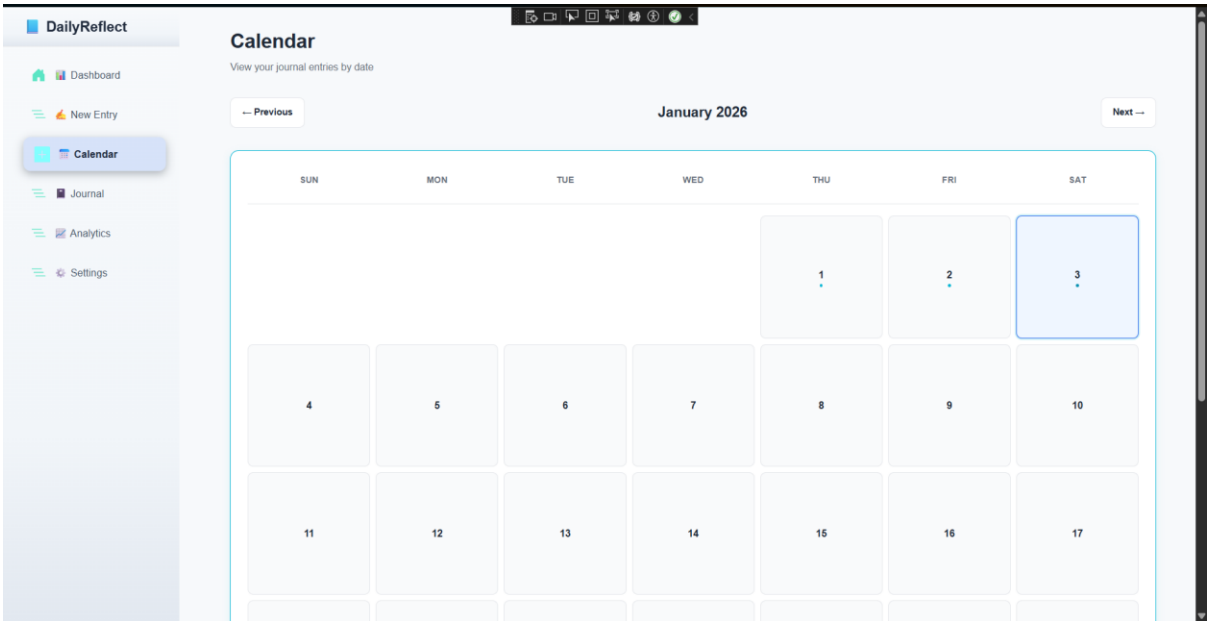

Figure 18: Figure of calendar navigation for journal entry

2.2.6 Paginated Journal View

DailyReflect offers a journal list view which can be seeing in a structured and readable format. Pagination has better performance and usability in cases where there are many entries.

Evidence:

User can adjust entries per page.

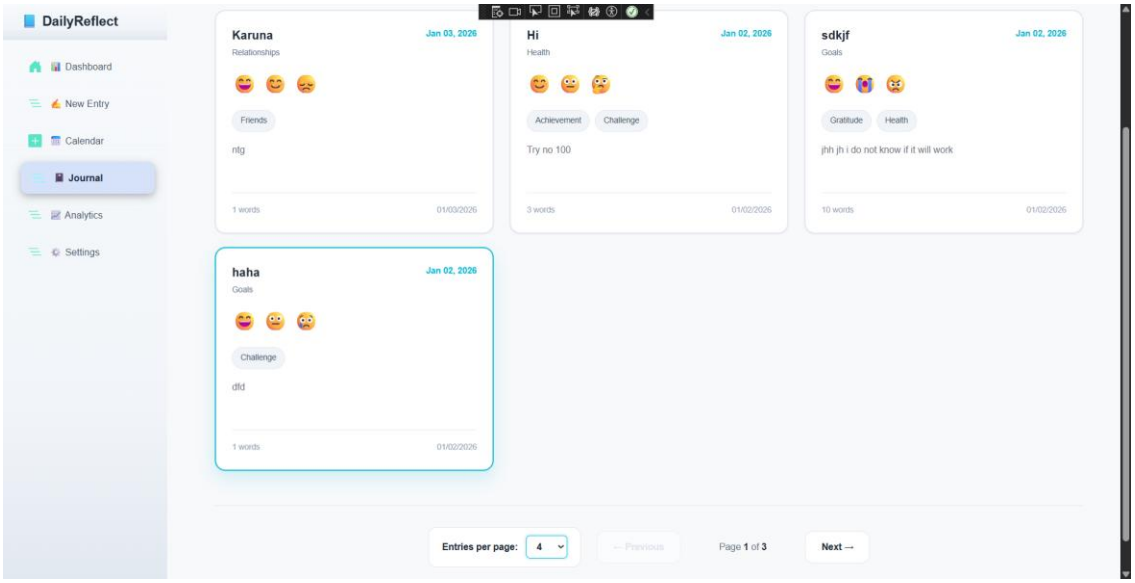

Figure 19: Figure of Paginated Journal View

2.2.7 Filtering and Search Capability.

The user is able to filter journal entries by title or content along with filters by date range, moods or tags. Such option enables the users to find desired entries quickly and compare certain time or mood.

Evidence:

User can search journal by using journal title and data as well.

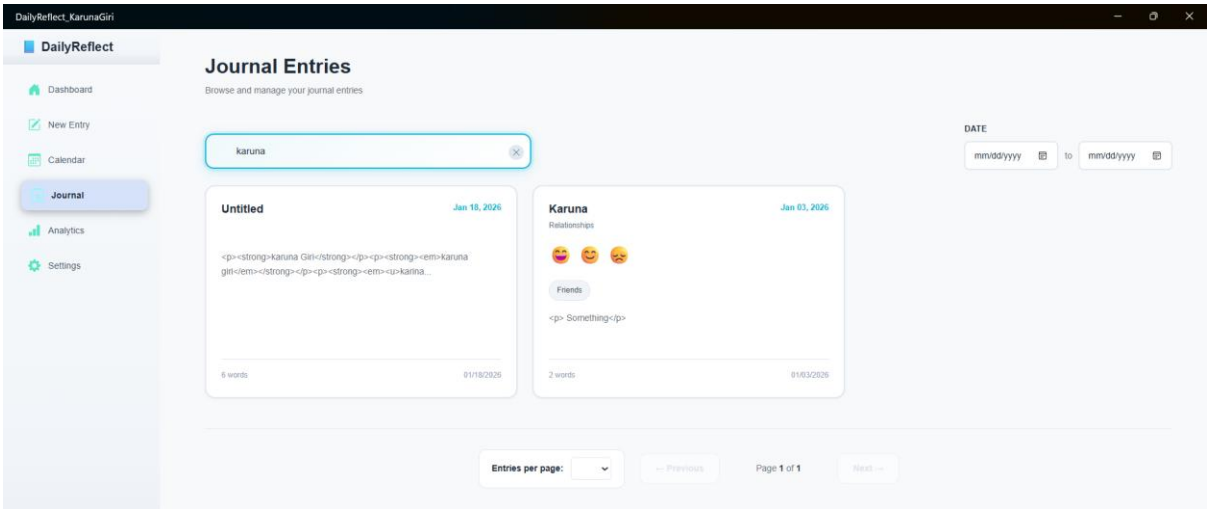

Figure 20: Searching Journal by Title and Date

2.2.8 Streak Tracking System

The app also monitors the streak that the user has, the longest streak and missed days in a day. The given aspect encourages regular journaling through visualizing the progress and habits.

Evidence:

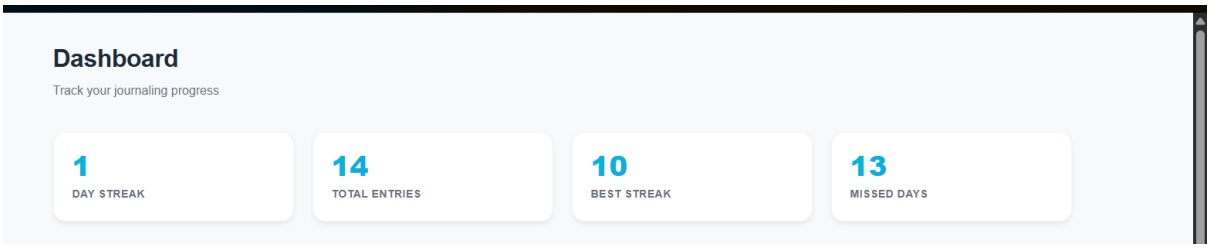

Figure 21: Figure of Streak Track

2.2.9 Analytics Dashboard

The DailyReflect has an interactive analytics dashboard to give information about the journaling habits. The most important analytics is mood distribution, most common moods, most popular tags, tag breakdown and trends of word counts over time. All analytics may be filtered to an analysis with specific date range.

Evidence

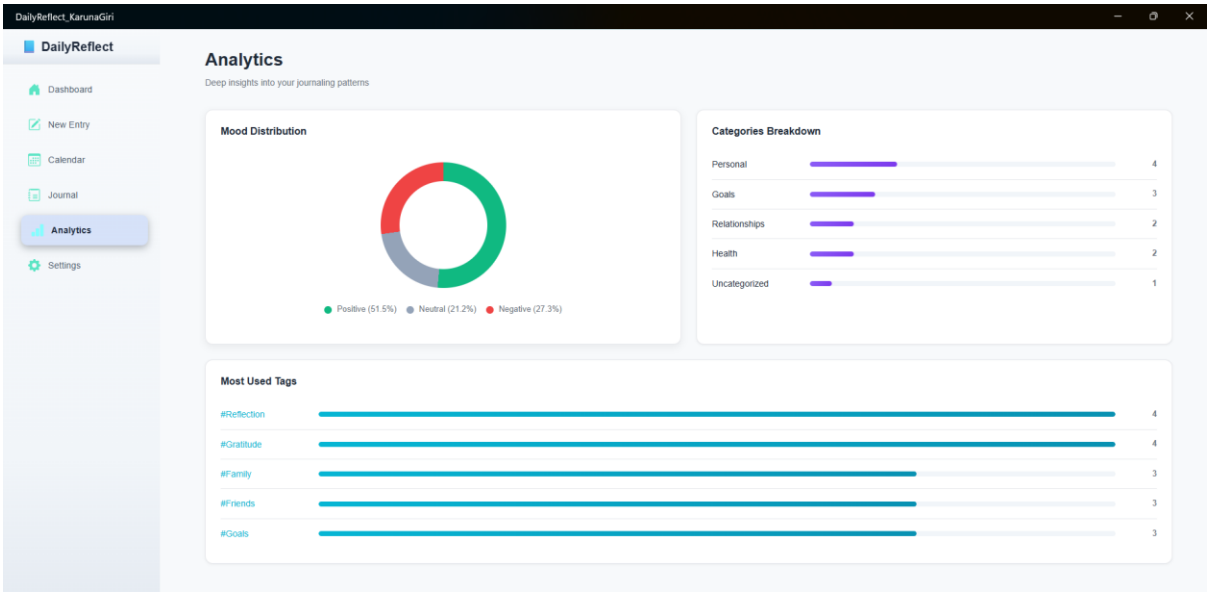

Figure 22: Figure of Analytics and Insights

2.2.10 Theme Customization

The application interface is customizable allowing users to change their themes to light and dark, and this increases the accessibility and user comfort.

Evidence:

User can customize theme as white and black for better visual UI appearance.

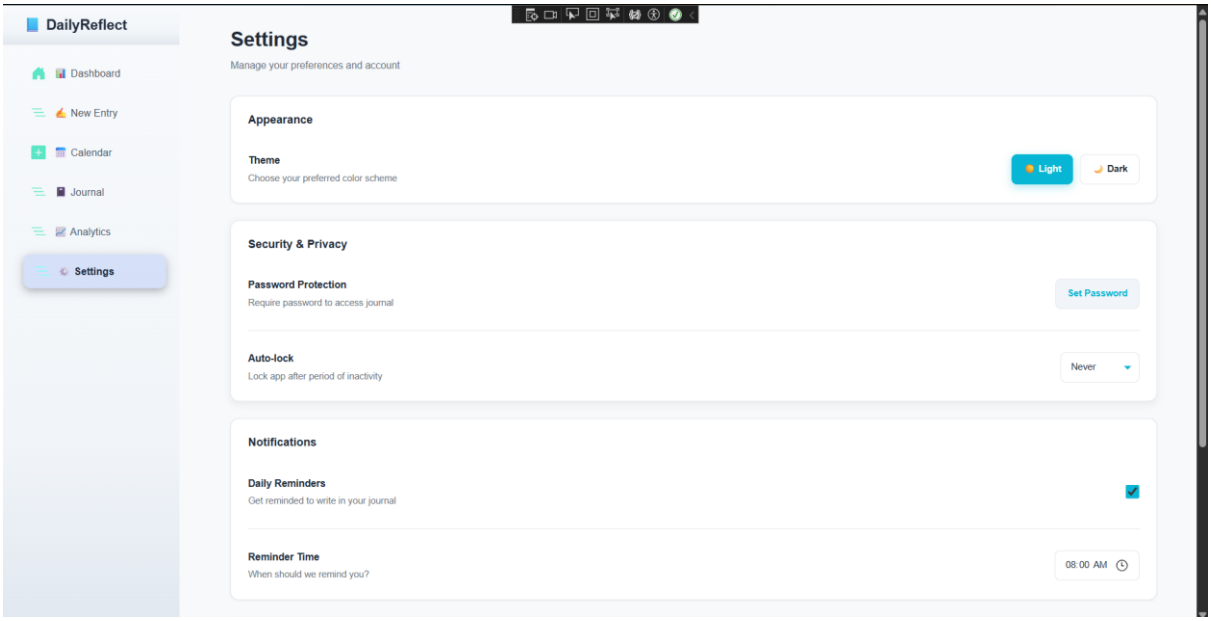

Figure 23: UI with Light Theme

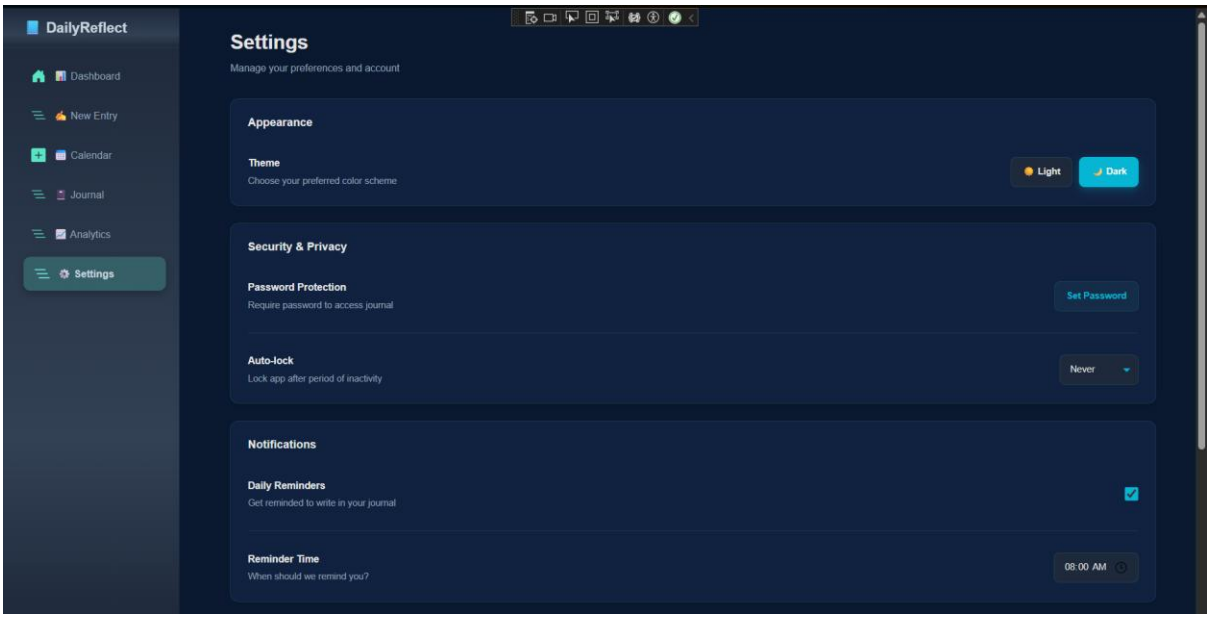

Figure 24: UI with Dark Theme

2.2.11 Journal Export as PDF

The app permits the user to print out journal entries in the form of PDF files depending on the date range selected. It has the advantage of enhancing data portability and allowing users to store their journal records safely or share them with others.

Evidence:

User can select journal as per also for generating pdf.

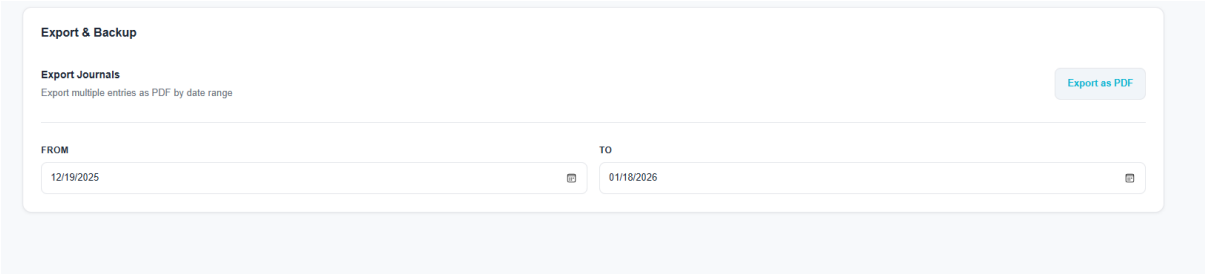

Figure 25: Figure of Selecting Journal by Date for PDF

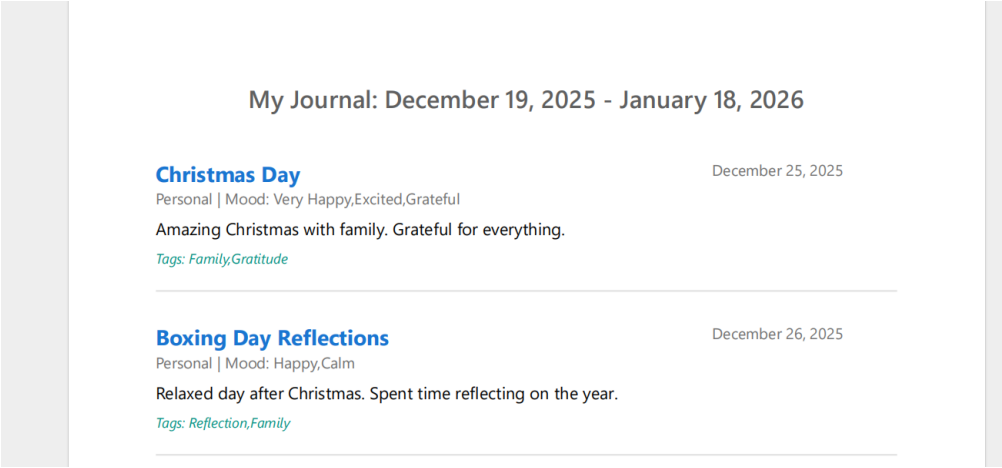

Figure 26: Successfully exporting in PDF

2.2.12 Rich Text/Markdown Writing

The user is able to use rich formatting with bold, italics, lists, headings, and links using user-friendly Quill.js 1.3.6 toolbar with live preview. JSON Delta format content stores that guarantee a perfect level of fidelity to the editor and display to PDF exports and to SQLite storage. It has a complex nested formatting ability that offers professional, expressive daily journal entries that are necessary in meaningful personal reflections.

Evidence:

User can select different font for entering journal.

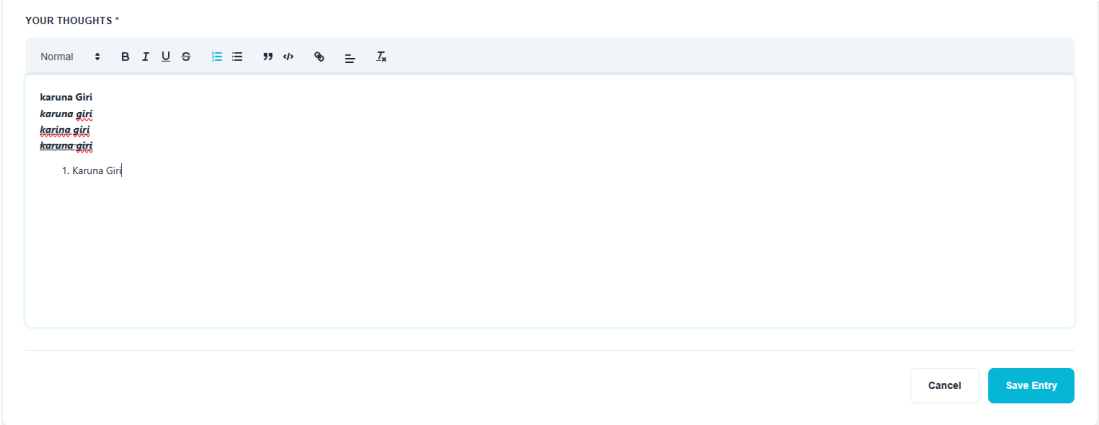

Figure 27: Figure of selecting different format

3 Proof of Work

3.1 Designs

3.1.1 UI Design (Wireframe)

User Interface (UI) design is the process of designing the visual and interactive components of software, websites or applications that involve people to interact with digital products. It is basically where the users interact with the technology and it includes layouts, buttons, menus, input fields and other features that enable the navigation to be easy and effective. The design of the UI determines the appearance, behaviour and feel of a product and has a direct impact on whether a user finds an application to be easy to use and visually balanced. Although it is interaction oriented and presentation oriented, it is used in collaboration with larger usability objectives to ensure a smooth interaction. An excellent design of UI emphasizes the simplicity, clearness, and responsiveness to help the user to attain the desired objective without frustration or irritation. Through the careful arrangement of visual content and the preservation of a steady structure, UI designers contribute to both the functionality and friendliness of applications and this is critical towards the end user satisfaction and success of the products. (Staff, 2025)

3.1.1.1 Login Page

In the DailyReflect Journal desktop application it contains login form which consist of only one input field named “Password/PIN” and its wireframe is mentioned below:

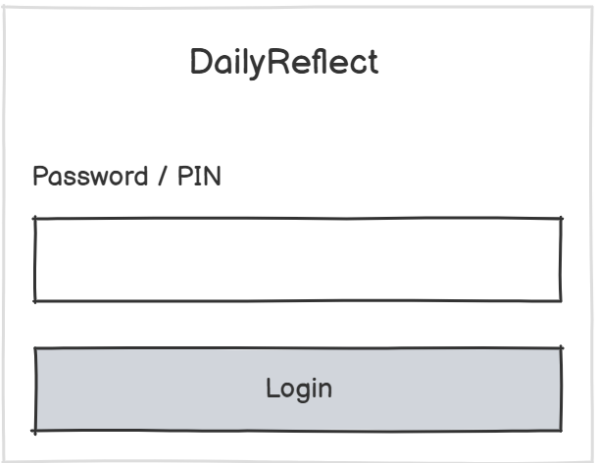

Figure 28: Wireframe of Login page

3.1.1.2 Dashboard Page

The dashboard page of DailyReflect Journal contains side menu bar in the left side of the page and its contents such as Daily Streak, Longest Streak, Total, Entries, Mood Distribution Chart etc. in the right section of the page and its wireframe is mentioned below:

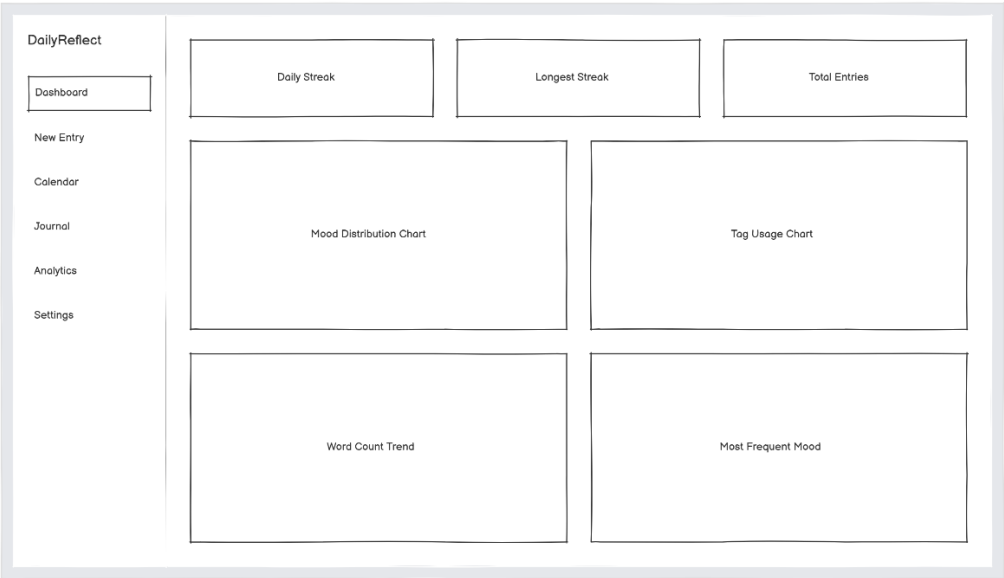

Figure 29: Wireframe of Dashboard page

3.1.1.3 New Entry Page

The New Entry Page of the DailyReflect Journal contains the new entry form for journal of the user which contains different section of the input fields such as Date, Title, Category, Mood Selection, Tags and so on which wireframe is mentioned below:

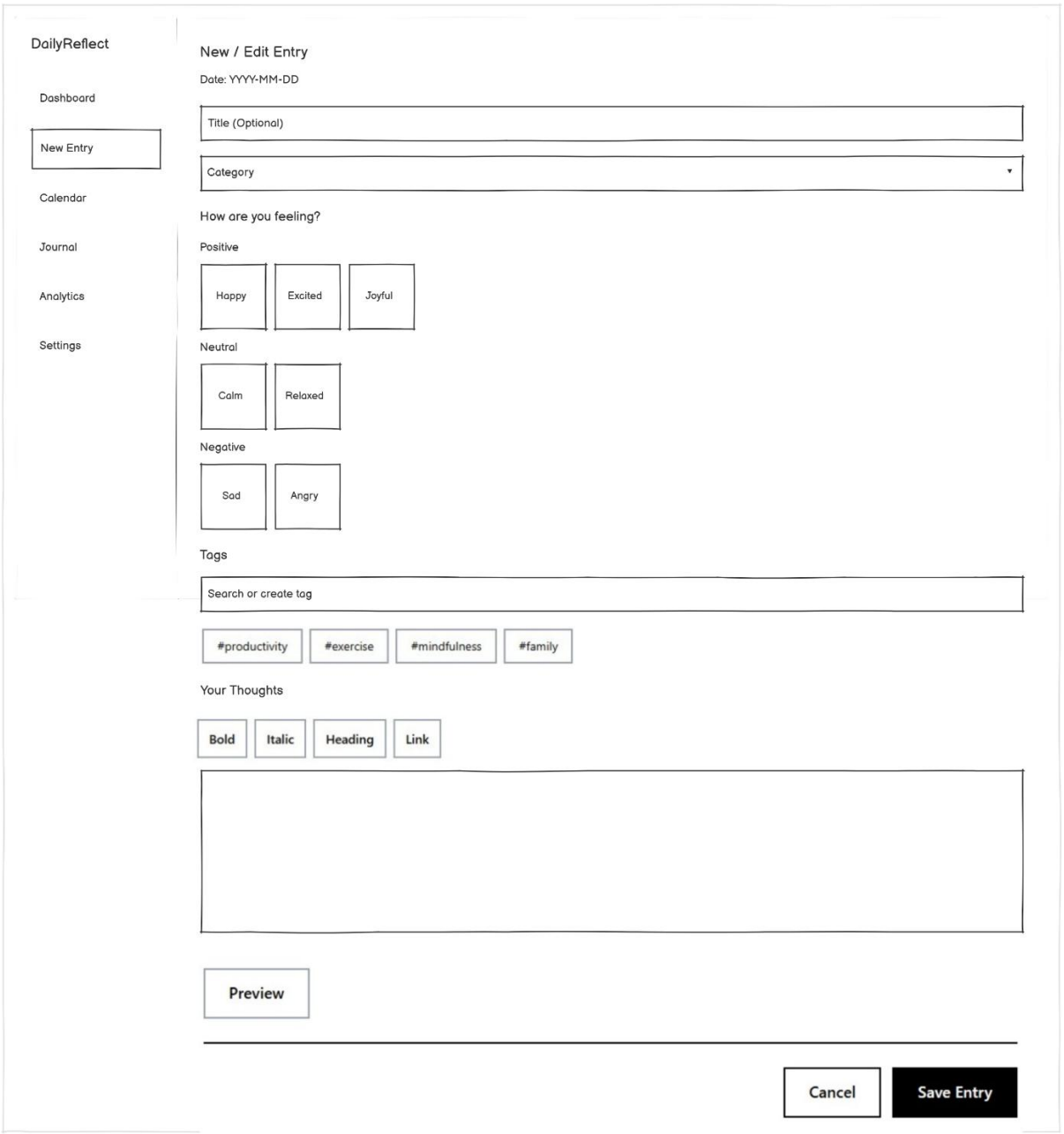

Figure 30: Wireframe of New Entry Page

3.1.1.4 Calendar Page

The DailyReflect Journal also has the calendar page which show when user entry journal and miss the journal date and its wireframe is mentioned below:

Figure 31: Wireframe of Calendar Page

3.1.1.5 Journal Page

The journal page of the DailyReflect contains all journal in short description with its necessary data and it can be filter and search by the date and tag as well. Its wireframe is mentioned below:

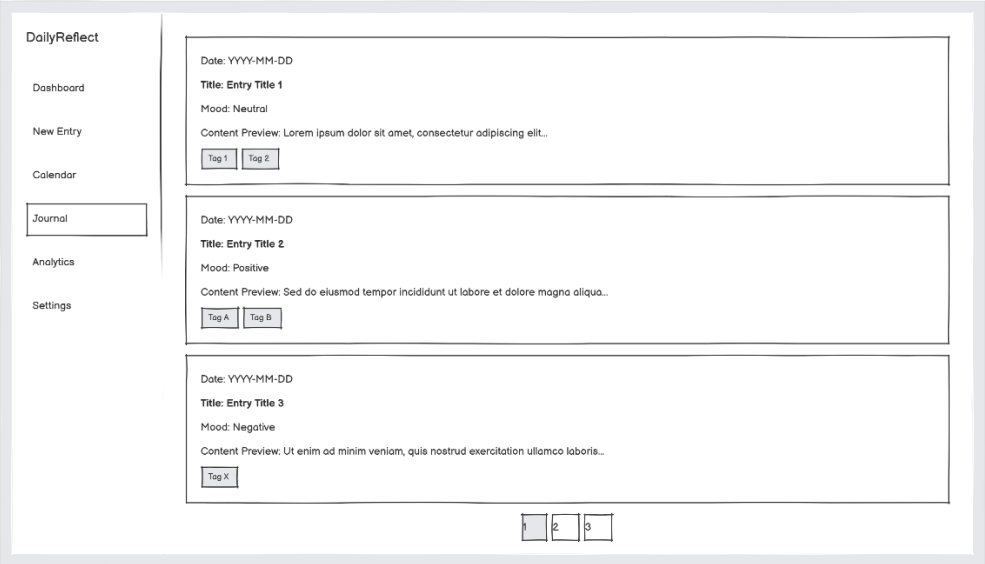

Figure 32: Wireframe of Journal Page

3.1.1.6 Analytics Page

The analytics page of the DailyReflect journal shows the detailed habit and pattern analysis of the user which wireframe is mentioned below:

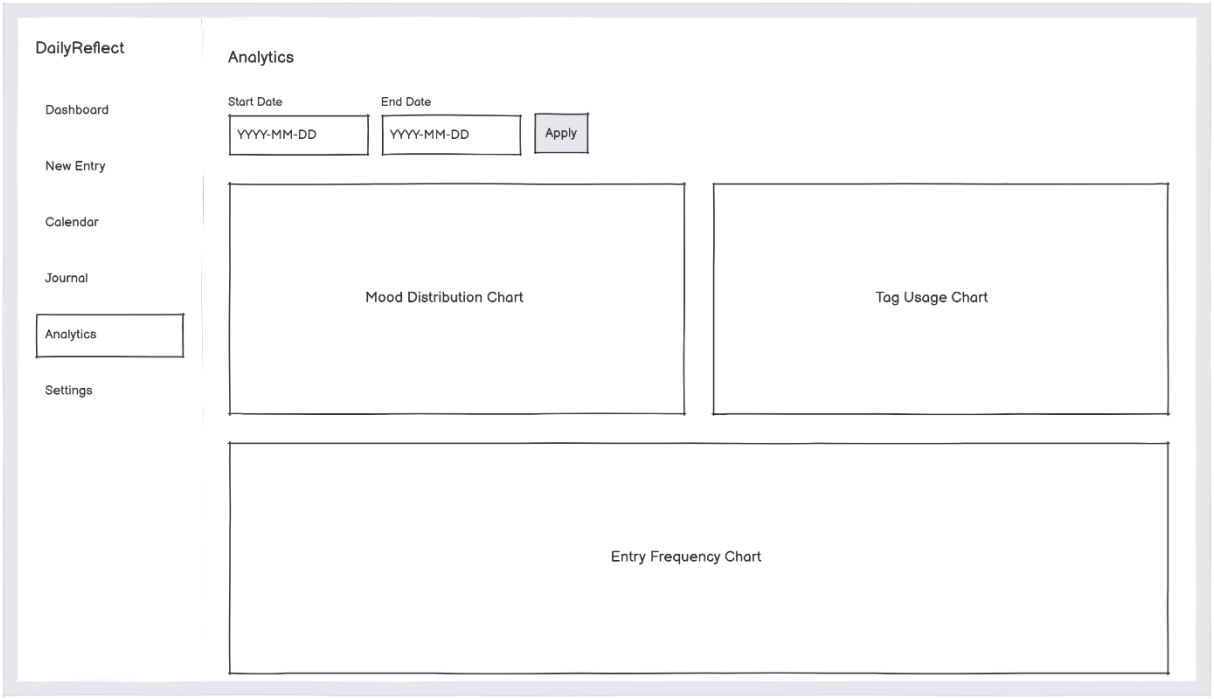

Figure 33: Wireframe of Analytics Page

3.1.1.7 Settings Page

The setting page of the DailyReflect contains all the setting features such as changing password, theme changing and converting journal into pdf and able to download it. Its wireframe is mentioned below:

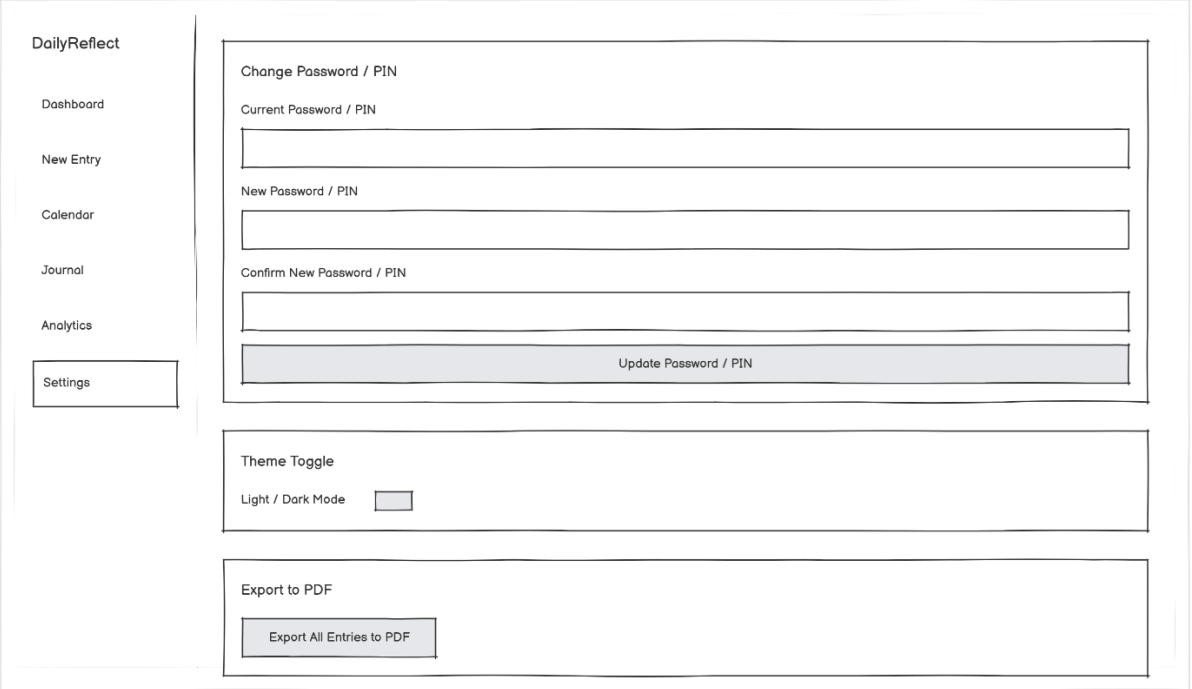

Figure 34: Wireframe of Setting page

3.1.2 Entity Relationship Diagram

A data or entity model is a conceptual definition of the information that a system needs and the organization and association of the information. It establishes the important entities, their properties and how they are connected with each other and they help bring a clear picture of real-life concepts in a software application. Such models define technology-independent method to elaborate data structures prior to implementing it in both a physical database and an application layer, based on the Microsoft concept of Entity Data Model (EDM) models.

When referring to the DailyReflect application, the data entity model serves as a template in the SQLite database and domain objects design. It provides uniformity, information integrity and clarity in journaling entries, moods, tags, categories and user details. The model ensures better communication throughout the development process in that entities and relationships are well defined and can be used to facilitate maintainable and scalable application design. (microsoft, 2021)

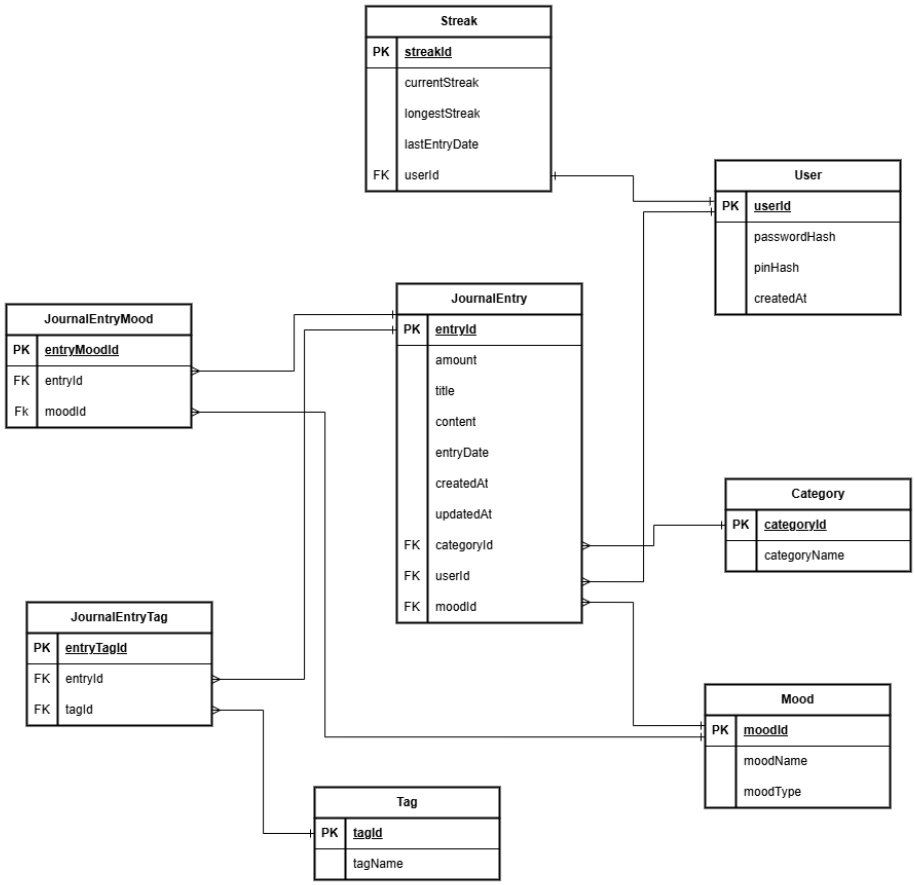

Figure 35: Figure of ERD

3.2 Quality

3.2.1 Code Readability

To enhance code readability in my project, the more attention has given to the fact that both the C# code and Razor markup in my project were always kept clean, consistently organized and self-explaining. I gave meaningful names to the variables, methods, and CSS classes (such as filterStartDate, hasPinProtection, GetPaginatedEntries, entry-moods, tag-badge etc.) such that their use is evident without additional comments. The elements had divided like (moods, tags, content preview,Headers) into recognizable units divided by comments at the UI layer, and any indentation and spacing to the if conditions, foreach loops, and nested HTML, which made the layout simple to read, comprehend, and maintain.

```
        }
    }
}
private int totalPages = 1;

// Storing date range filter values
private DateTime? filterStartDate = null;
private DateTime? filterEndDate = null;

// Tracking PIN-based access protection for the page
private bool hasPinProtection = false;
private bool isPageAccessGranted = false;

// Checking for PIN protection and loading entries
protected override async Task OnInitializedAsync()
{
    // Checking if user has enabled PIN protection for journal access
    hasPinProtection = Preferences.ContainsKey("JournalPin");

    if (hasPinProtection)
    {
        // Prompting for PIN before granting page access
        await VerifyPinForPageAccess();
    }
    else
    {
        // Granting access immediately when no PIN protection
        isPageAccessGranted = true;
        await LoadEntries();
    }
}
```

Figure 36: Figure of Code Readability

3.2.2 Code Efficiency

To enhance the efficiency of the code, centralized had done all the journal filtering, pagination processing into one method to be reused rather than write all similar queries to make them each search or filter choice. The approach uses date range, key word, mood and tag filters sequentially with the aid of LINQ with the result being that only the conditions needed are tested and the minimal result set processed. I also avoid doing unnecessary work because system resources can be short-circuited (by skipping filters when the value is null or empty) and because loading entries after authentication of PIN has occurred, which will avoid expensive database calls and user interface rendering when a page is still open.

```
// Loading all journal entries for the current user
private async Task LoadEntries()
{
    var userId = AuthState.CurrentUserId ?? 0;
    allEntries = await JournalService.GetEntriesAsync(userId);
    filteredEntries = allEntries;
    currentPage = 1;
    UpdateTotalPages();
}

// Filtering entries based on search query and date range
private void FilterEntries()
{
    filteredEntries = allEntries;

    // Applying text search across title, content, moods, and tags
    if (!string.IsNullOrWhiteSpace(searchQuery))
    {
        var query = searchQuery.ToLower();
        filteredEntries = filteredEntries
            .Where(e =>
                (e.Title?.ToLower().Contains(query) ?? false) ||
                (e.Content?.ToLower().Contains(query) ?? false) ||
                (e.Tags?.ToLower().Contains(query) ?? false) ||
                (e.Mood?.ToLower().Contains(query) ?? false))
            .ToList();
    }

    // Applying start date filter
    if (filterStartDate.HasValue)
    {
        filteredEntries = filteredEntries
            .Where(e => e.EntryDate >= filterStartDate.Value.Date)
            .ToList();
    }
```

Figure 37: Figure of Code Efficiency

3.2.3 Code Modularity

To achieve code modularity, the project had arranged in such a way that every responsibility is separated into a layer and a folder to make the application easier to read and maintain. UI pages have dedicated Razor components (i.e., dashboard, new entry, calendar, all entries, settings), whereas data selection and business functions are adjusted in others Data, Models, and Services. This isolation so that any modification to any of the system components, like modification of database logic or the addition of a new feature, may be done with minimal effects on the other components, enhances reusability, testability, and scalability of the entire application.

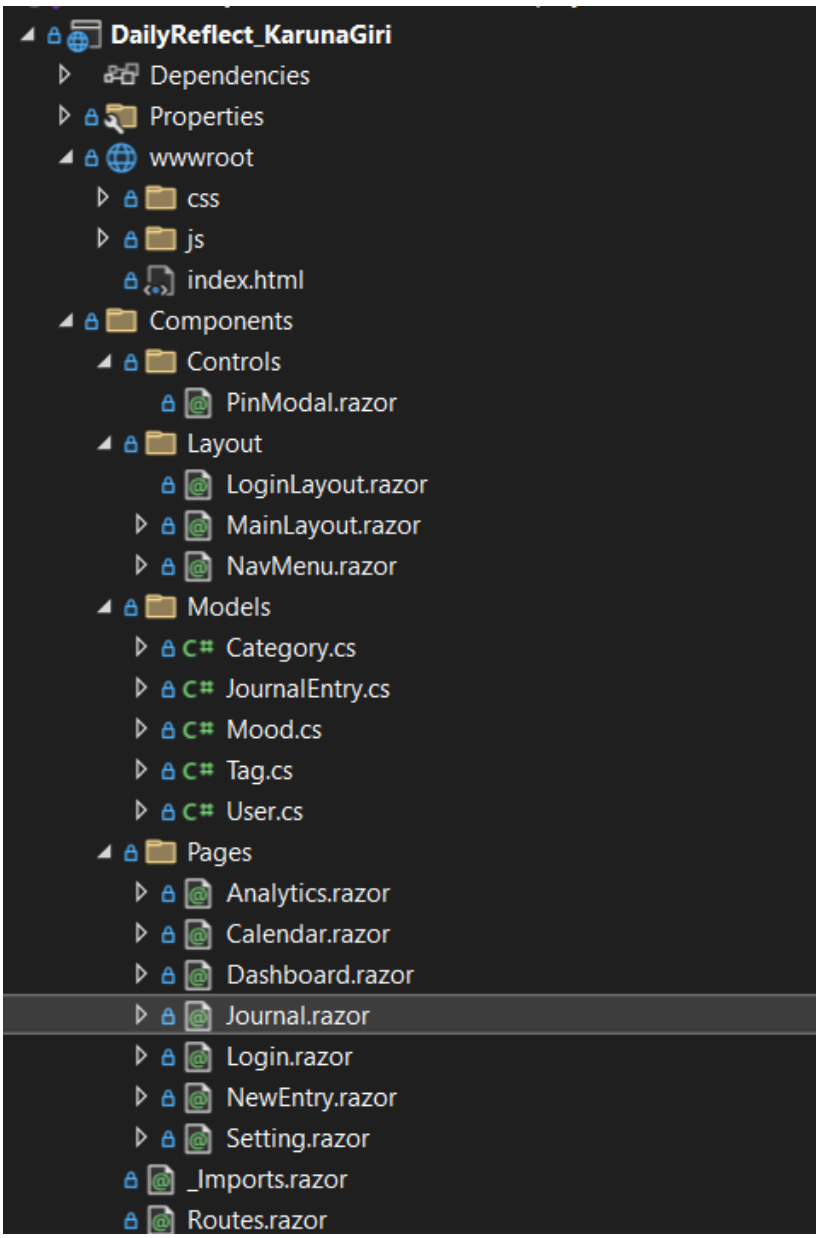

Figure 38: Figure of Code Modularity

3.2.4 Error Handling

To avoid errors, the prioritized has given on stability and user-friendliness of the application by integrating compile and runtime protection. When a project was being developed, I look and correct warnings and errors on the IDE Error List frequently to ensure that the project will build successfully and makes less unexpected failures. During runtime, there is utilization of input validation and try catch blocks in reference to important operations such as login and data loading, provided a descriptive message on invalid credentials or some unexpected exceptions rather than a crash, and safely stopping loading states to get clear feedback to the user and to ensure that the app is responsive.

```
    // Restricting deletes to today's entry only
    if (existingEntry.EntryDate.Date != DateTime.Today)
    {
        statusMessage = "⚠ You can only delete today's entry.";
        return;
    }

    try
    {
        isSaving = true;
        var userId = AuthState.CurrentUserId ?? 0;
        var ok = await JournalService.DeleteEntryAsync(existingEntry.Id, userId);
        if (ok)
        {
            statusMessage = "Entry deleted.";
            await Application.Current!.Windows[0].Page!.DisplayAlert("Deleted", statusMessage, "O
            Navigation.NavigateTo("/journal");
        }
        else
        {
            statusMessage = "Could not delete entry.";
        }
    }
    catch (Exception ex)
    {
        var innerMessage = ex.InnerException?.Message ?? ex.Message;
        statusMessage = $"Delete failed: {innerMessage}";
    }
    finally
    {
        isSaving = false;
    }
}
```

Figure 39: Figure of Error Handling

3.3 Testing

3.4 Features

3.4.1 Journal Entry Management

Table 2:Test-1: Journal Entry Management

Feature	Journal Entry Management
Test Case	Create a new journal, update and delete it.
Expected Result	New journal should be created, can update it and also removed it.
Actual Result	Entry was created, updated and deleted successfully.
Conclusion	Test was successful.

Evidence:

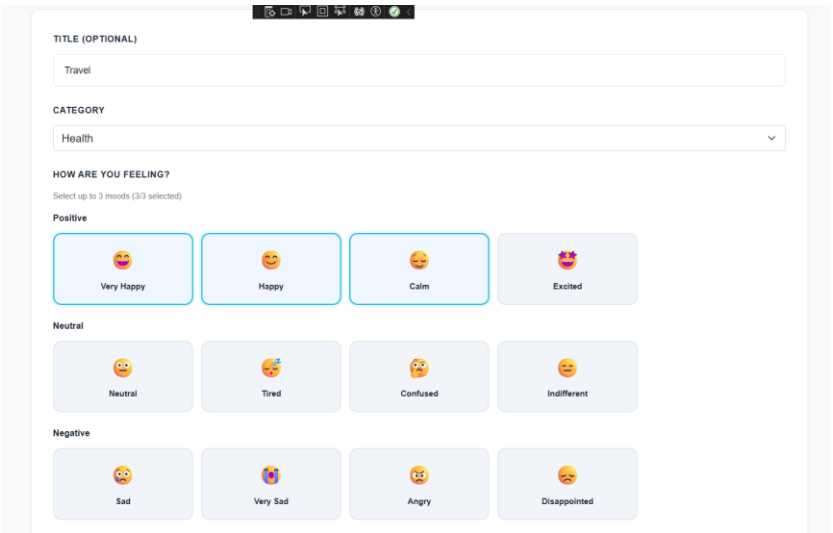

Figure 40: Testing-Figure of creating a journal

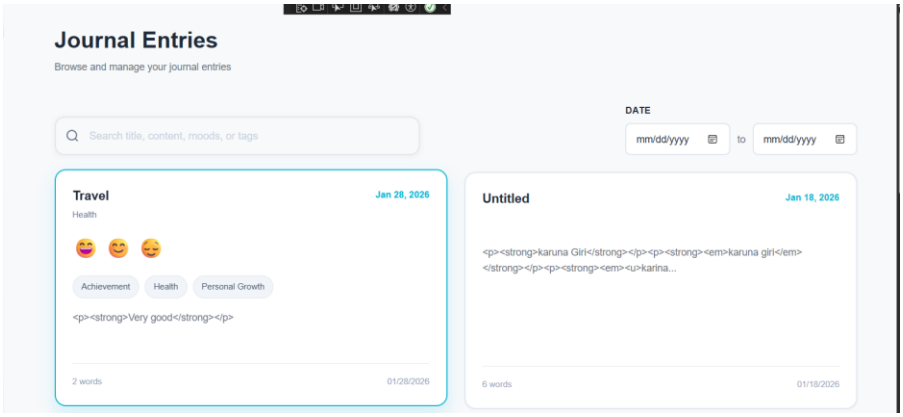

Figure 41: Testing-Journal appear in the Journal page

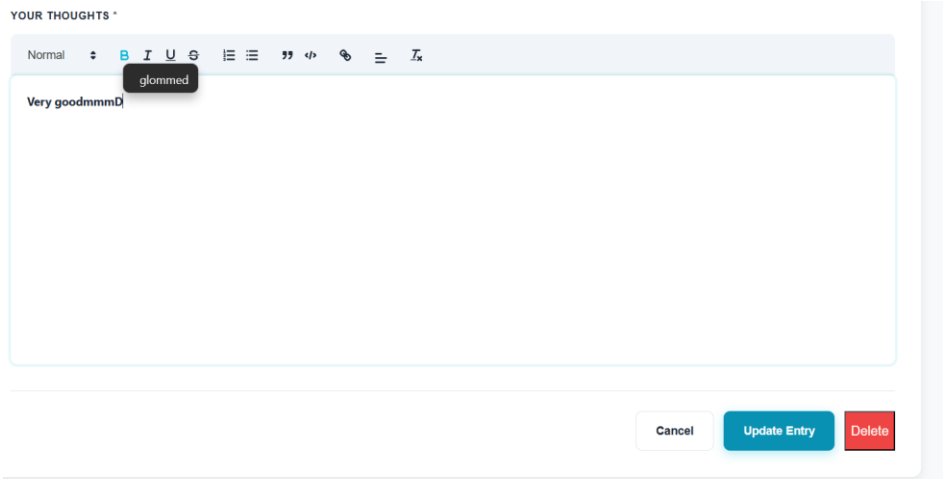

Figure 42: Testing-Updating Recent Journal Only

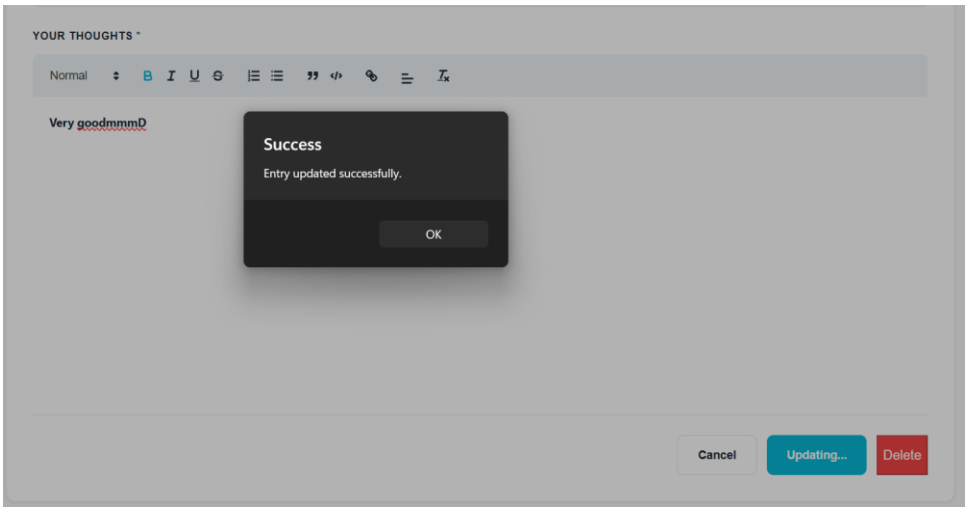

Figure 43:Testing- Successful message of Updated

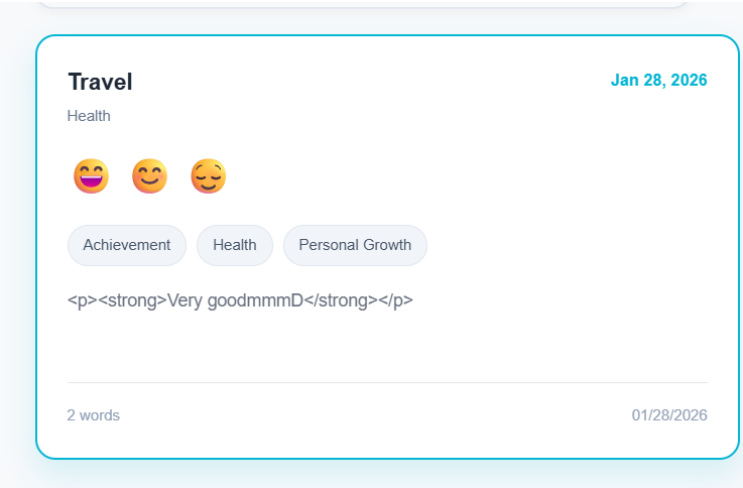

Figure 44: Testing-Recent Journal Updated Successfully

Removing Only Recent added Journal Entry.

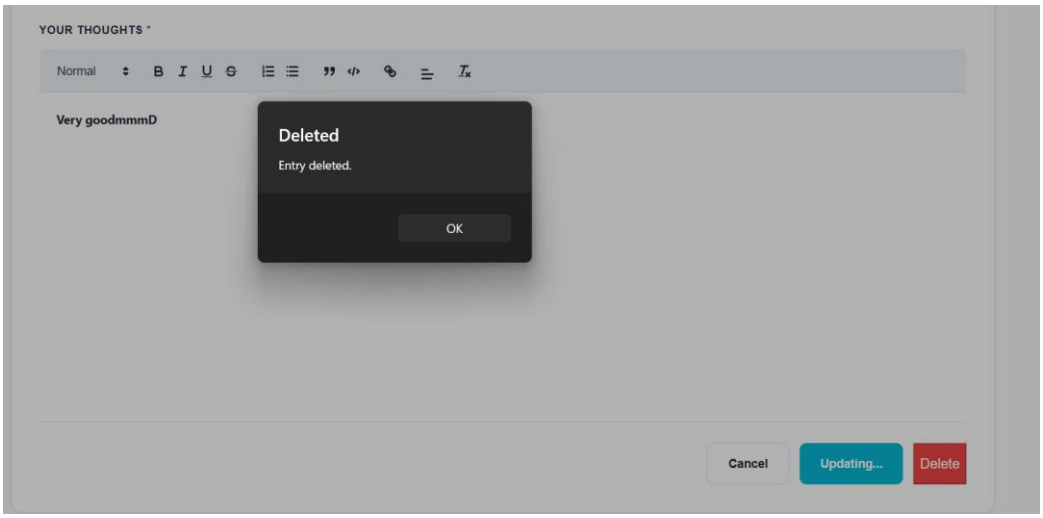

Figure 45: Testing-Removing Recent Journal

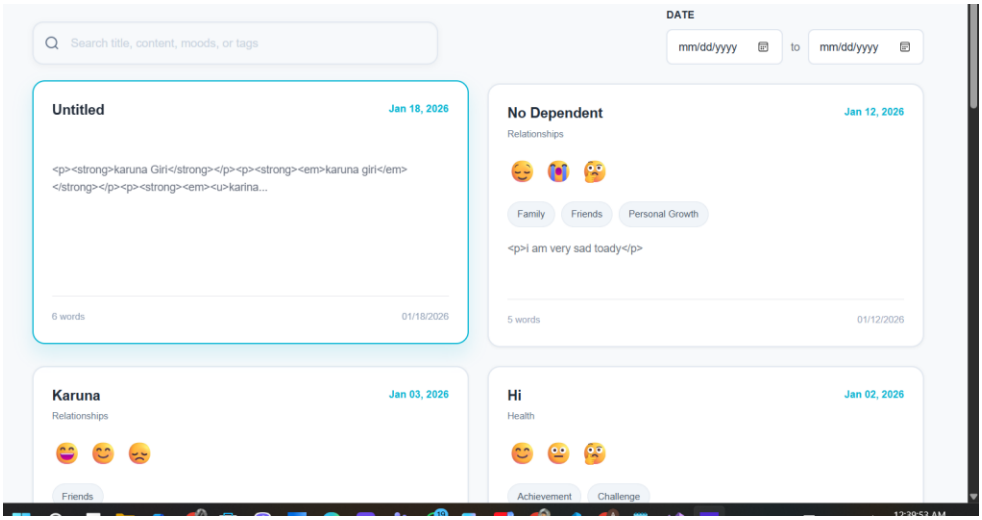

Figure 46: Testing-Journal Removed Successfully

	Id	EntryDate	Title	Category	Mood	
	Fil...	Filter	Filter	Filter	Filter	Filter
1	1	2026-01-02 00:00:00	Hi	Health	Happy,Neutral,Confused	Achievement,Cha
2	3	2026-01-02 00:00:00	sdkjf	Goals	Very Happy,Very Sad,Angry	Gratitude,Healt
3	4	2025-12-25 00:00:00	Christmas Day	Personal	Very Happy,Excited,Grateful	Family,Gratitud
4	5	2025-12-26 00:00:00	Boxing Day Reflections	Personal	Happy,Calm	Reflection,Fami
5	6	2025-12-27 00:00:00	Back to Routine	Work	Neutral,Tired	Work,Goals
6	7	2025-12-28 00:00:00	Gym Day	Health	Happy,Excited	Health,Achievem
7	8	2025-12-29 00:00:00	Year End Thoughts	Mindfulness	Calm,Confused	Reflection,Grat
8	9	2025-12-30 00:00:00	Final Day of 2025	Personal	Happy,Nostalgic	Reflection,Frie
9	10	2025-12-31 00:00:00	New Year Eve!	Goals	Very Happy,Excited,Confident	Goals,Personal
10	11	2026-01-01 00:00:00	Happy New Year 2026!	Personal	Very Happy,Excited	Goals,Gratitude
11	12	2026-01-02 00:00:00	haha	Goals	Very Happy,Neutral,Sad	Challenge
12	13	2026-01-03 00:00:00	Karuna	Relationships	Very Happy,Happy,Disappointed	Friends
13	14	2026-01-12 00:00:00	No Dependent	Relationships	Calm,Very Sad,Confused	Family,Friends,

Figure 47: Testing -Successful Removed from Database

3.4.2 Rich Text/Markdown Writing

Table 3: Test-2: Rich Text/Markdown Writing

Feature	Journal Entry Management
Test Case	Creating journal with different format.
Expected Result	Different format are allowed to use while writing the journal.
Actual Result	User used different format while writing new journal.
Conclusion	Test was successful.

Evidence:

User can select different font for entering journal.

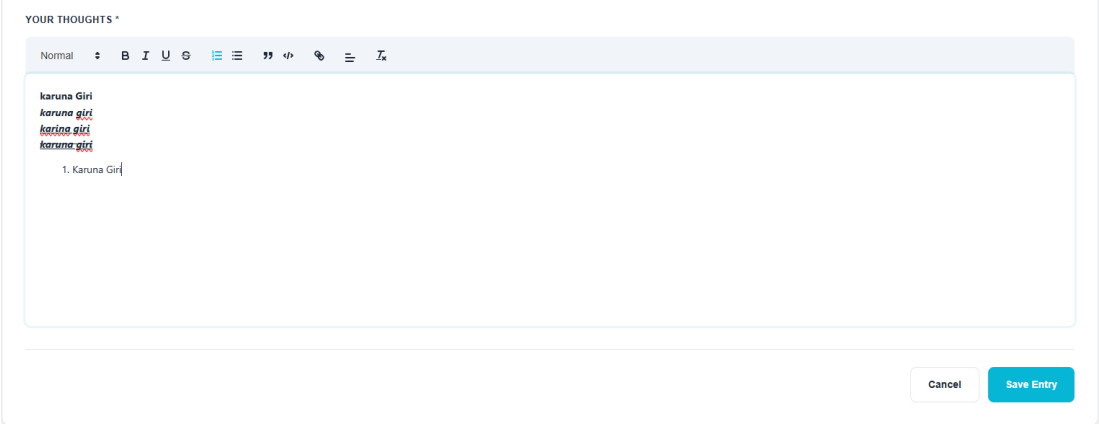

Figure 48: Figure of Testing of selecting different format

3.4.3 Mood Tracking

Table 4: Test-3: Mood Tracking

Feature	Journal Entry Management
Test Case	Select primary and secondary moods while creating new entry.
Expected Result	Three mood are allows to use while creating new entry.
Actual Result	User used up to three moods while writing new journal.
Conclusion	Test was successful.

Evidence:

User can select up to 3 moods only at a time.

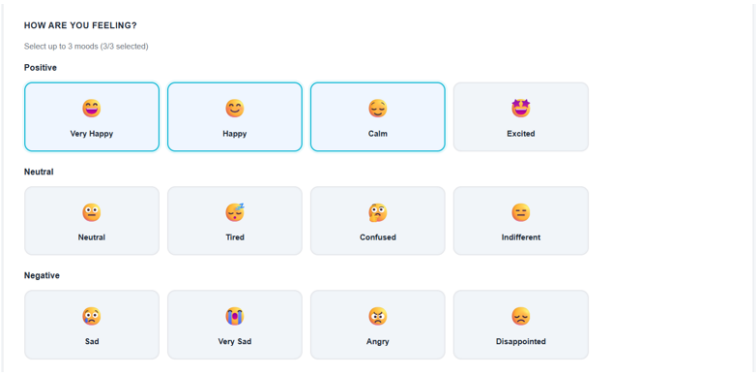

Figure 49: Figure of Testing Mood Selection

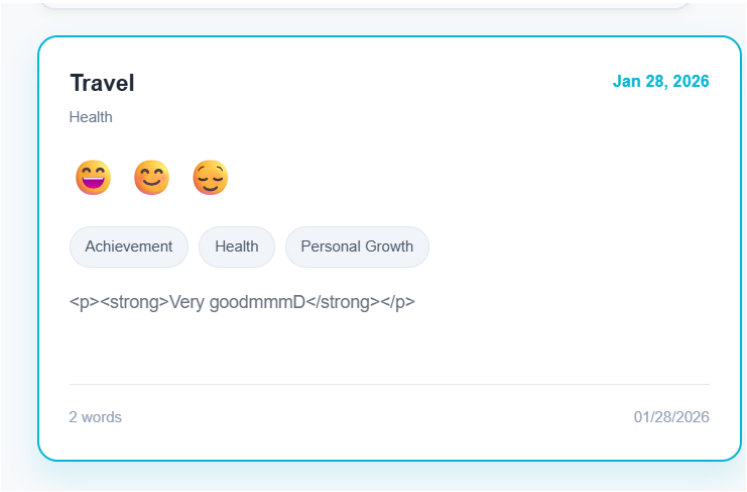

Figure 50: Testing Successfull Mood Selection

3.4.4 Tagging System

Table 5: Test-4: Tagging System

Feature	Journal Entry Management
Test Case	Select different tags while creating new entry.
Expected Result	Different tags are allowed to use while creating new entry.
Actual Result	User different tags while writing new journal.
Conclusion	Test was successful.

Evidence:

User can used different predefined tags as well.

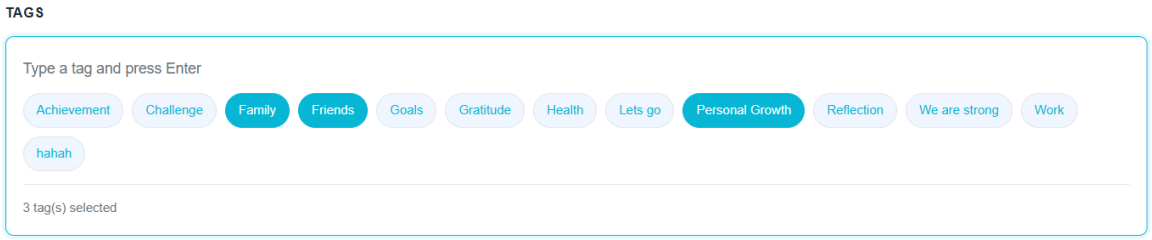

Figure 51: Figure of Testing predefined tags

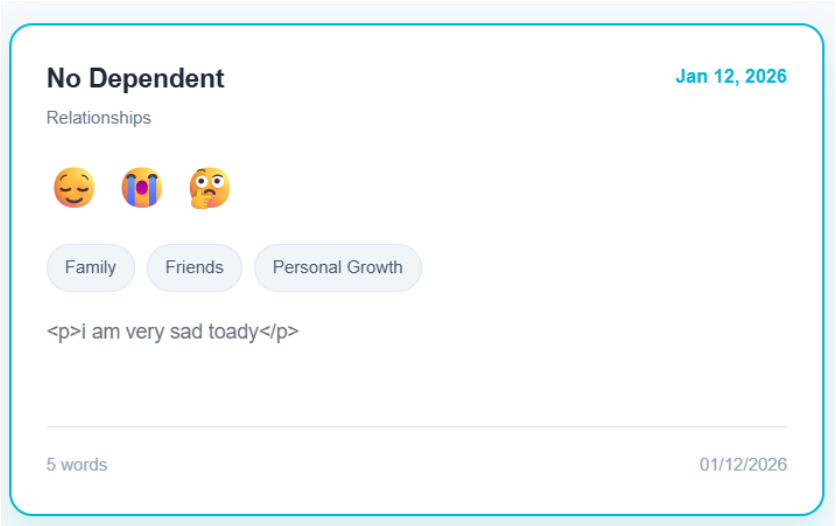

Figure 52: Testing successfully attach in Journal

3.4.5 Calendar Navigation

Table 6: Test-5: Calendar Navigation

Feature	Journal Entry Management
Test Case	Select date from calendar which has one or more entries.
Expected Result	Entries for selected date should be displayed.
Actual Result	User can see date of entries in the calendar.
Conclusion	Test was successful.

Evidence:

It shows the days in which journal entries had done with blue tick.

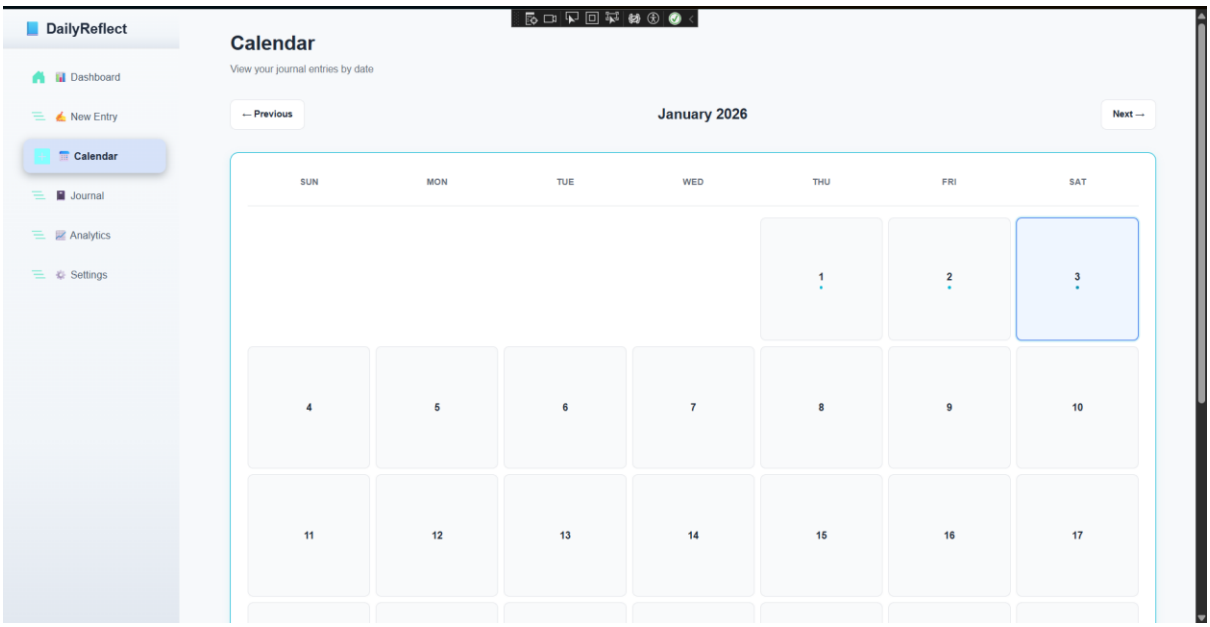

Figure 53: Figure of Testing calendar navigation for journal entry

3.4.6 Paginated Journal View

Table 7: Test-6: Paginated Journal View

Feature	Journal Entry Management
Test Case	Open journal list and navigate through pages using Next/Previous control.
Expected Result	Entries should be changed according to selected control.
Actual Result	User can see different entries according to Next/Previous control button.
Conclusion	Test was successful.

Evidence:

User can adjust entries per page.

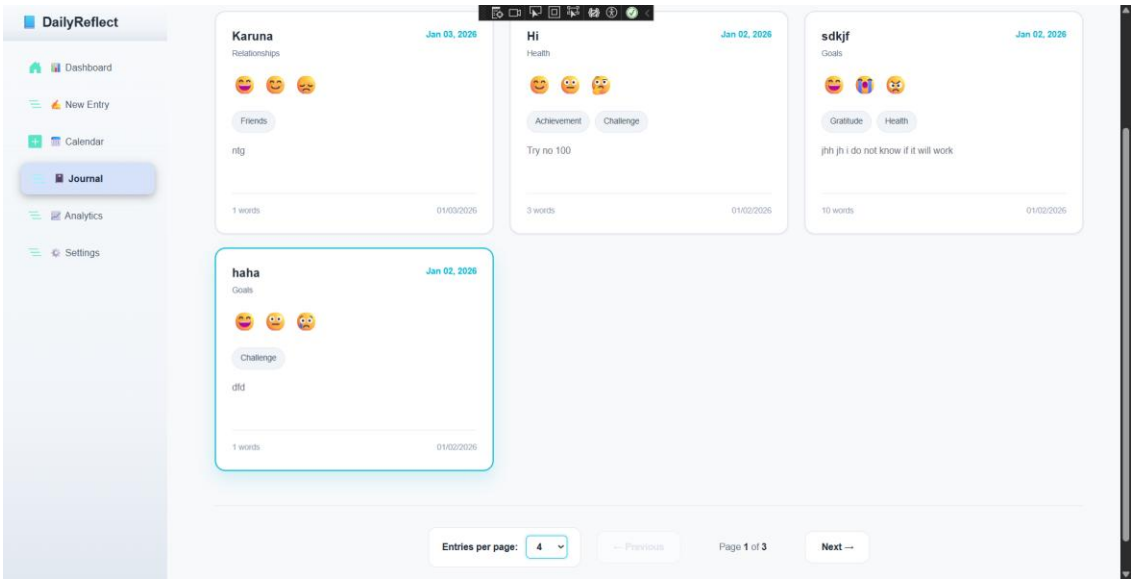

Figure 54: Figure of Testing Paginated Journal View

3.4.7 Search & Filter

Table 8: Test-7: Search & Filter

Feature	Journal Entry Management
Test Case	User search entries by title, tag and date range.
Expected Result	Only matching entries should be displayed according to selected filter.
Actual Result	Search and Filter returned the matches entries only.
Conclusion	Test was successful.

compare certain time or mood.

Evidence:

User can search journal by using journal title and data as well.

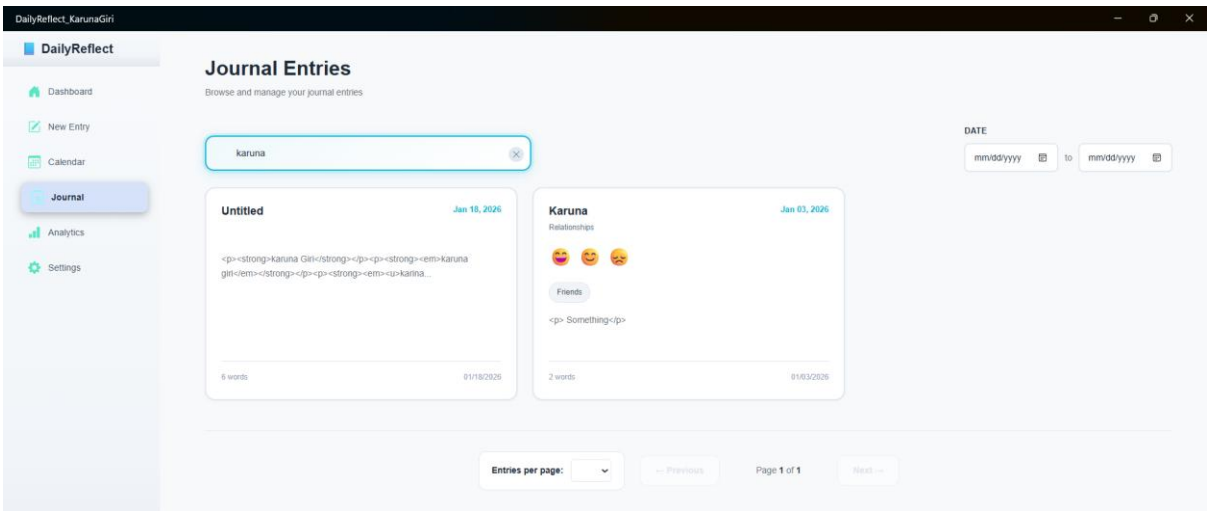

Figure 55: Testing-Searching Journal by Title and Date

3.4.8 Streak Tracking

Table 9: Test-8: Streak Tracking

Feature	Journal Entry Management
Test Case	User can track their streak of entries
Expected Result	Streak count should be displayed as per user entry per day if miss then reset.
Actual Result	Streak updated in dashboard and calendar.
Conclusion	Test was successful.

Evidence:

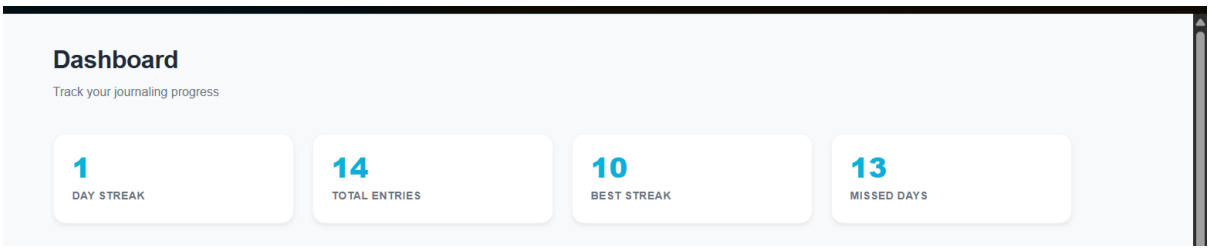

Figure 56: Testing-Figure of Streak Track

3.4.9 Theme Customization

Table 10: Test-9: Theme Customization

Feature	Journal Entry Management
Test Case	User can track their streak of entries
Expected Result	Streak count should be displayed as per user entry per day if miss then reset.
Actual Result	Streak updated in dashboard and calendar.
Conclusion	Test was successful.

Evidence:

User can customize theme as white and black for better visual UI appearance.

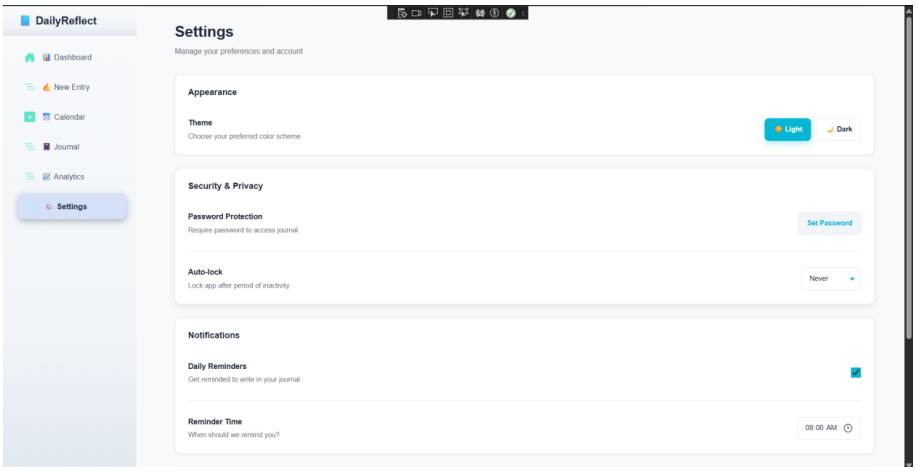

Figure 57: UI with Light Theme

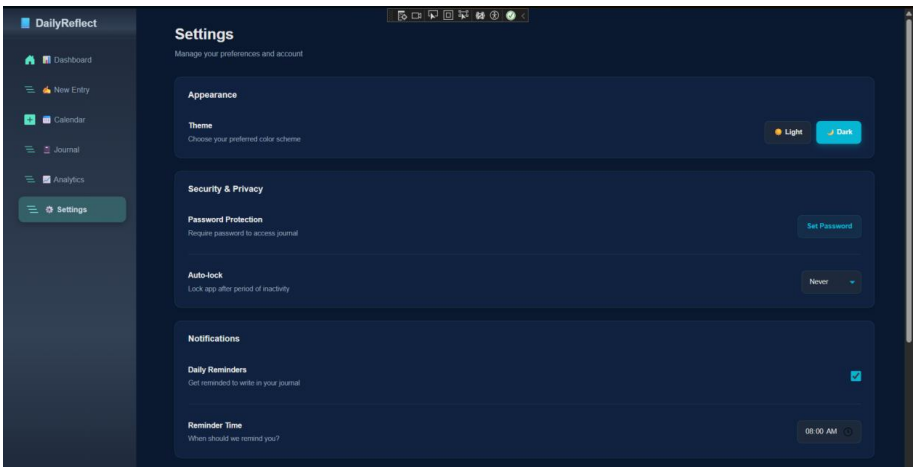

Figure 58: UI with Dark Theme

3.4.10 Dashboard Analytics & Insights

Table 11: Test-10: Dashboard Analytics & Insights

Feature	Journal Entry Management
Test Case	Add multiple entries with different moods and tags, then open the dashboard.
Expected Result	Charts and summary cards should update to reflect mood distribution, counts and trends.
Actual Result	Dashboard analytics refreshed correctly with accurate statistics and visual charts.
Conclusion	Test was successful.

Evidence

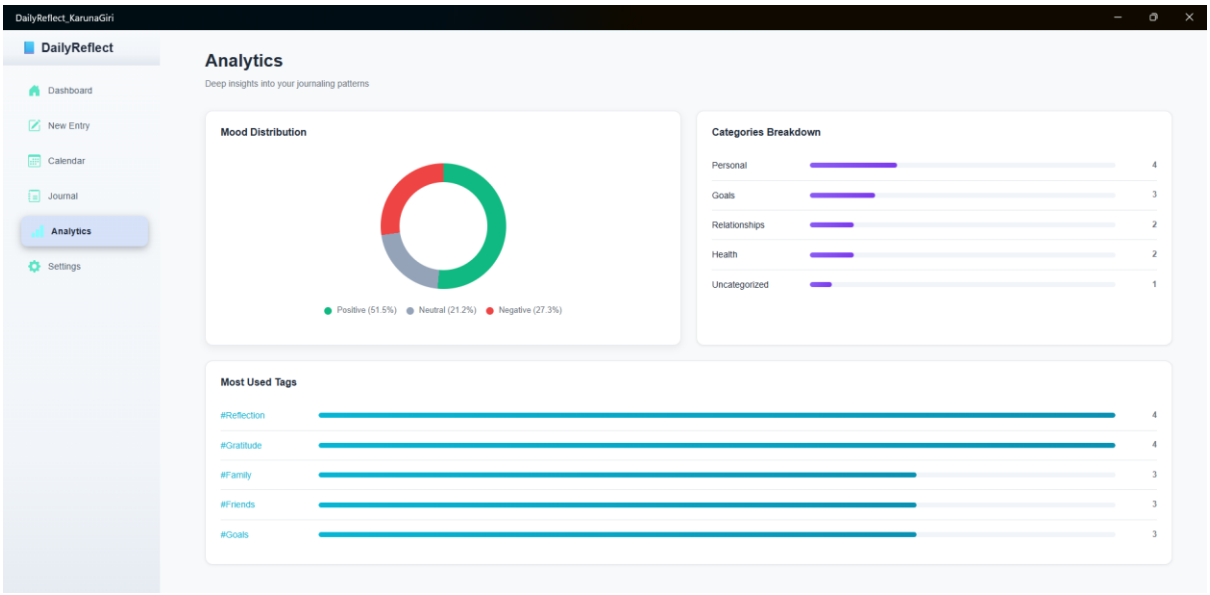

Figure 59: Testing-Figure of Analytics and Insights

3.4.11 Security and Privacy

Table 12: Test-11: Security and Privacy

Feature	Journal Entry Management
Test Case	Attempt to access protected pages with wrong PIN/credentials, then with valid details
Expected Result	Access should be denied for protected pages without pin.
Actual Result	Unauthorized access was blocked and only visible after valid pin.
Conclusion	Test was successful.

Evidence:

Setting the pin for Journal Protection.

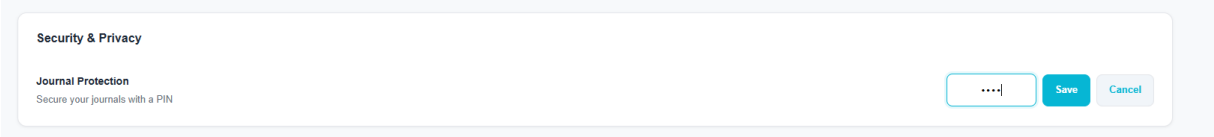

Figure 60: Testing-Figure of setting pin

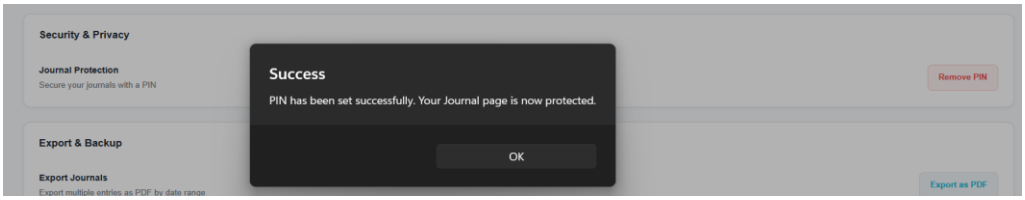

Figure 61: Testing-Successfully setting pin

Accessing the Journal by entering the pin.

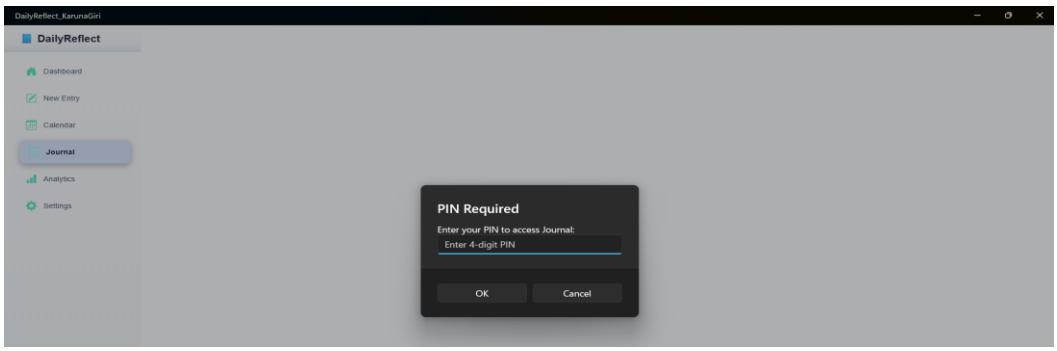

Figure 62: Testing-Figure of user entering pin

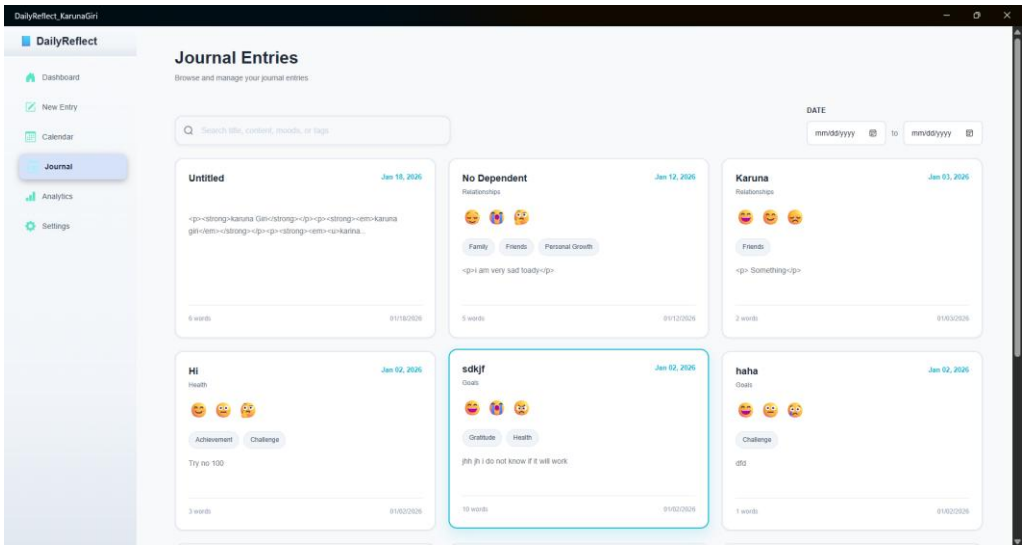

Figure 63:Testing- Successfully accessing journal

3.4.12 Export Journals

Table 13: Test-12: Export Journals

Feature	Journal Entry Management
Test Case	Select a date range, export journals to PDF, and open the downloaded file.
Expected Result	PDF should be generated according to date range.
Actual Result	Exported PDF downloaded according to the date range.
Conclusion	Test was successful.

Evidence:

User can select journal as per also for generating pdf.

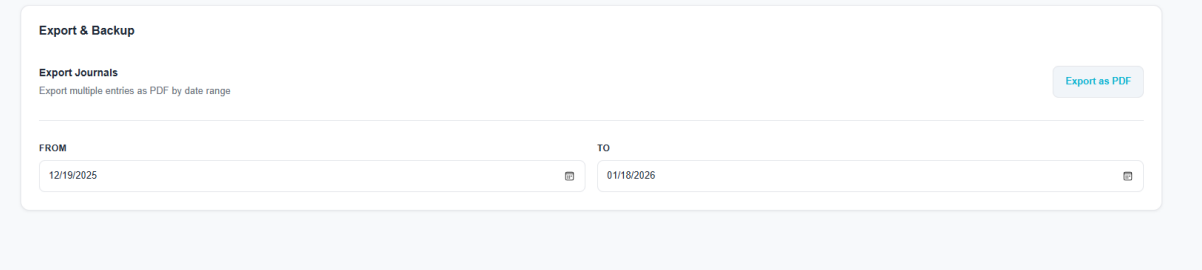

Figure 64:Testing- Figure of Selecting Journal by Date for PDF

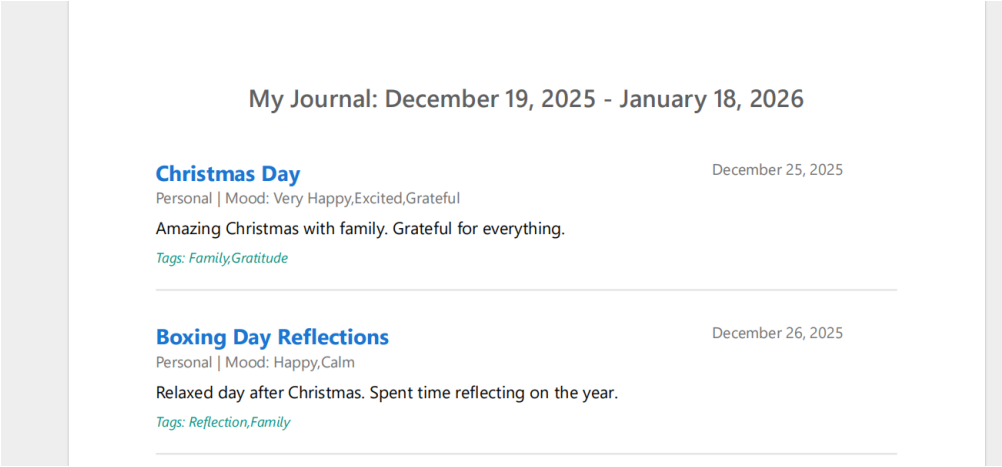

Figure 65: Testing-Successfully exporting in PDF

4 Individual Reflection

The creation of this journaling application reflects the design of an end-to-end design of a structured cross-platform solution based on the use of .NET MAUI, MVVM, and SQLite. The system is not merely a progression of a simple journal entry manager to a full-fledged product that encompasses mood tracking, streak counting, navigation through the calendar, a full text editor, analytical functionality, the ability to switch between themes, and protection by PIN without making the architecture overly complex. The design choices around separation of views, view-models and data services, centralisation of filtering and pagination functions, and the imposition of local persistence without doubt demonstrate knowledge of the concept of software engineering and its effects on future extensibility.

The UX, performance, and robustness are observed during the project through the management of search and filtering, responsive dashboards, thoughtful error management, and organization of twelve features of the project testing. The practical work experience developed during the solutions of the data binding, navigation flow and security issues is a very good foundation and reference point in future work on applications development. Continued improvements like encrypted storage, cloud synchronisation, reminders and enhanced mood analytics can easily improve this journaling product into a production-grade personal wellbeing app.

Another field that has been brought out by the project is the necessity of regulated project management and documentation within application development. Well defined feature identification, test tables and test user manuals ensure that the stakeholders will know the system behaviour even without reading the code. Regular version control, purposeful commit history, and visual artefacts like ER diagrams, use-case diagrams and screen shots of the Ui are also indicative of a professional working process that is well aligned with the industry expectations of a maintainable software project that is auditable.

5 Conclusion

The DailyReflect journaling application will be the final successful product out of the CS6004NT Application Development coursework- an operating C#.NET desktop prototype of an application that modernizes personal reflection practices. By carefully fulfilling all the three milestones such as Gits repository setup with wireframes (Week 9), core functionality development (Week 11) and final features delivery (Week 13), all the 12 core features have been met such as the daily entry CRUD functions, extensive feeling tracking in up to Positive/Neutral/Negative moods, smart tagging, calendar navigation, paginated views, sophisticated search/filtering, streaks tracking, dashboard analytics, customizing themes, PDF export, and strong PIN access control.

The technology stack choice here was very successful, as .NET MAUI Blazor Hybrid allowed consistency of the UI cross-platform, SQLite local persistence with security, Quest PDF journal export, Quill.js rich-text editing, and the Bootstrap responsive design. Such a combination of strategic options fulfilled all the non-functional requirements such as performance, scalability, usability and security and exhibited code quality excellence via modularity, error handling, version control practices, intuitive UX design all of which achieved the entire 90 marks of code quality and 10 marks of documentation quality.

In the future, DailyReflect will have a solid platform on which production can be deployed with potential to improve, including cloud syncing across platforms, mood detection by devices using artificial intelligence on natural language data and extensions of the mobile app. The process of development enhanced the competence in the latest tooling's of the .NET ecosystems, database architecture, the agile milestone project, and professional documentation methods greatly and converted the theoretical material of courses into the practical and consumer friendly application which had the potential to expand in real world scenarios to trace the personal growth.

6 References

getbootstrap. (2026) *Get started with Bootstrap* [Online]. Available from: https://getbootstrap.com/docs/5.3/getting-started/introduction/ [Accessed 16 January 2026].

learn.microsoft. (2026) *Build a.NET MAUI Blazor Hybrid app with a Blazor Web App* [Online]. Available from: https://learn.microsoft.com/en-us/aspnet/core/blazor/hybrid/tutorials/maui-blazor-web-app?view=aspnetcore-9.0 [Accessed 16 January 2026].

microsoft. (2021) *Entity Data Model* [Online]. Available from: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/entity-data-model [Accessed 19 December 2025].

microsoft. (2025).*NET Multi-platform App UI* [Online]. Available from: https://dotnet.microsoft.com/en-us/apps/maui [Accessed 19 December 2025].

quilljs. (2026) *Quickstart* [Online]. Available from: https://quilljs.com/docs/quickstart [Accessed 16 January 2026].

sqlite. (2025) *About SQLite* [Online]. Available from: https://sqlite.org/about.html [Accessed 19 December 2025].

Staff, C. (2025) *coursera* [Online]. Available from: https://www.coursera.org/articles/ui-design [Accessed 17 December 2025].

Szymanowski, M. (2025) *PDF Generation in C#/.NET Using QuestPDF* [Online]. Available from: https://pdfbolt.com/blog/generate-pdf-csharp-questpdf [Accessed 16 January 2026].