Mood Tracker System Report

By Dhara Joshi

Sardar Vallabhbhai Patel Institute of Technology

Master of Computer Application

SAP Educate to Employ (E2E) - Software Development Track

Submitted on: July 29, 2025

Contents

1. Project Overview

1.1 Introduction

The Mood Tracker System is a cloud-ready, ABAP RAP-based application developed on the SAP BTP ABAP Environment. It is designed to capture, manage, and analyze users' mood entries, providing a seamless and modern solution for tracking emotional well-being. The system leverages OData V4 services and SAP Fiori Elements for an intuitive user interface, ensuring robust backend logic and clean-core compliance.

1.2 Objectives

The primary objectives of the Mood Tracker System are:

- To enable users to record and manage mood entries with associated notes and timestamps.
- To provide a structured database schema for storing mood types and entries.
- To deliver analytical insights through a user-friendly dashboard.
- To ensure scalability and extensibility using modern SAP development practices.

1.3 Key Features

- **Mood Entry Management**: Users can create, update, and delete mood entries with associated mood types and notes.
- **Mood Type Catalog**: A predefined set of mood types (e.g., Happy, Sad, Anxious) for consistent categorization.
- **Audit Logging**: Tracks all create, update, and delete operations for data integrity.
- **Fiori UI Integration**: Provides a modern, responsive interface using SAP Fiori Elements.
- **Analytics Dashboard**: Displays trends and patterns in mood entries for better insights.

1.4 Benefits

- Streamlined mood tracking with an intuitive interface.
- Enhanced data management through a structured persistence layer.
- Scalable architecture compliant with SAP's clean-core principles.
- Actionable insights through analytical dashboards.

2. Database Schema (Persistence Layer)

The Mood Tracker System utilizes two primary database tables to manage mood-related data efficiently.

2.1 Table: ZMOOD_ENTRY - Mood Entries

Field Name	Data Type	Description
client	abap.clnt	Client Key (Mandatory)
mood_id	abap.char(10)	Unique Mood Entry ID
mood_type_id	abap.char(5)	Linked Mood Type ID
mood_date	abap.dats	Date of Mood Entry
note	abap.char(255)	Additional Notes
created_by	abp_creation_user	Created By
last_changed_on	abp_lastchange_utcl	Changed Timestamp
changed_by	abp_lastchange_user	Changed By

Table 1: ZMOOD_ENTRY Table Structure

2.2 Table: ZMOOD_TYPE - Mood Types

Field Name	Data Type	Description
client	abap.clnt	Client Key (Mandatory)
mood_type_id	abap.char(5)	Unique Mood Type ID
mood_text	abap.char(50)	Mood Description (e.g., Happy, Sad)
created_by	abp_creation_user	Created By
last_changed_on	abp_lastchange_utcl	Changed Timestamp
changed_by	abp_lastchange_user	Changed By

Table 2: ZMOOD_TYPE Table Structure

3. Core Functionality (ABAP Classes)

3.1 Class: ZCL_MOOD_DEMO

This class handles the core logic for retrieving and displaying mood entries, integrating with the database tables.

```
CLASS zcl_mood_demo DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC.

  PUBLIC SECTION.
    INTERFACES if_oo_adt_classrun.
ENDCLASS.

CLASS zcl_mood_demo IMPLEMENTATION.
  METHOD if_oo_adt_classrun~main.
    DATA: lt_entries TYPE STANDARD TABLE OF zmood_entry,
          ls_entry   TYPE zmood_entry,
          lt_types   TYPE STANDARD TABLE OF zmood_type,
          ls_type    TYPE zmood_type,
          lv_text    TYPE string.

```

```
    SELECT * FROM zmood_entry INTO TABLE @lt_entries.
    SELECT * FROM zmood_type INTO TABLE @lt_types.

    LOOP AT lt_entries INTO ls_entry.
      READ TABLE lt_types INTO ls_type WITH KEY mood_type_id =
         ls_entry-mood_type_id.
      IF sy-subrc = 0.
        lv_text = ls_type-mood_text.
      ELSE.
        lv_text = '-'.
      ENDIF.
      out->write( |{ ls_entry-mood_id } { ls_entry-mood_type_id }
         { lv_text } { ls_entry-mood_date } { ls_entry-note }| ).
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.
```

4. CDS Views (Data Model Layer)

4.1 Interface Views (ZR_)

4.1.1 ZR_MOOD_ENTRY

Provides raw data access to the ZMOOD_ENTRY table.

```
@AccessControl.authorizationCheck: #CHECK
@Metadata.allowExtensions: true
@EndUserText.label: 'Mood Entry Interface View'
@ObjectModel.sapObjectNodeType.name: 'ZMOOD_ENTRY'
define root view entity ZR_MOOD_ENTRY
  as select from zmood_entry
{
  key mood_id as MoodId,
  mood_type_id as MoodTypeId,
  mood_date as MoodDate,
  note as Note,
  @Semantics.user.createdBy: true
  created_by as CreatedBy,
  @Semantics.systemDateTime.lastChangedAt: true
  last_changed_on as LastChangedOn,
  @Semantics.user.lastChangedBy: true
  changed_by as ChangedBy
}
```

4.1.2 ZR_MOOD_TYPE

Provides raw data access to the ZMOOD_TYPE table.

```
@AccessControl.authorizationCheck: #CHECK
@Metadata.allowExtensions: true
@EndUserText.label: 'Mood Type Interface View'
```

```
@ObjectModel.sapObjectNodeType.name: 'ZMOOD_TYPE'
define root view entity ZR_MOOD_TYPE
  as select from zmood_type
{
  key mood_type_id as MoodTypeId,
  mood_text as MoodText,
  @Semantics.user.createdBy: true
  created_by as CreatedBy,
  @Semantics.systemDateTime.lastChangedAt: true
  last_changed_on as LastChangedOn,
  @Semantics.user.lastChangedBy: true
  changed_by as ChangedBy
}
```

4.2 Projection Views (ZC_)

4.2.1 ZC_MOOD_ENTRY

Projects mood entry data for UI consumption.

```
@Metadata.allowExtensions: true
@EndUserText.label: 'Mood Entry Projection View'
@AccessControl.authorizationCheck: #CHECK
@ObjectModel.sapObjectNodeType.name: 'ZMOOD_ENTRY'
define root view entity ZC_MOOD_ENTRY
  provider contract TRANSACTIONAL_QUERY
  as projection on ZR_MOOD_ENTRY
{
  key MoodId,
  MoodTypeId,
  MoodDate,
  Note,
  CreatedBy,
  LastChangedOn,
  ChangedBy
}
```

4.2.2 ZC_MOOD_TYPE

Projects mood type data for UI consumption.

```
@Metadata.allowExtensions: true
@EndUserText.label: 'Mood Type Projection View'
@AccessControl.authorizationCheck: #CHECK
@ObjectModel.sapObjectNodeType.name: 'ZMOOD_TYPE'
define root view entity ZC_MOOD_TYPE
  provider contract TRANSACTIONAL_QUERY
  as projection on ZR_MOOD_TYPE
{
  key MoodTypeId,
  MoodText,
  CreatedBy,
```

```
12    LastChangedOn,
13    ChangedBy
14  }
```

5. Service Definition and Binding

- **Service Definition**: ZS_MOOD_SRV
- **Service Binding**: ZB_MOOD_BINDING
- **Type**: OData V4
- **Exposed Entities**:
  - /MoodEntries
  - /MoodTypes

6. RESTful Implementation

The RESTful API exposes the following entities:

- **MoodEntries**: CRUD operations for mood entry management.
- **MoodTypes**: Read and manage mood type catalog.

7. Final Outcome

The Mood Tracker System provides a robust and user-friendly platform for managing mood entries. It leverages SAP RAP for clean-core compliance, ensuring scalability and maintainability. The integration with SAP Fiori Elements delivers a modern, responsive UI, enhancing user experience.

8. Conclusion

The Mood Tracker System, developed using the ABAP RESTful Application Programming Model (RAP) on SAP BTP, offers a comprehensive solution for tracking and analyzing emotional well-being. By utilizing CDS views, OData V4 services, and Fiori Elements, the system ensures seamless integration, a modern UI, and efficient backend processing. The clean-core architecture supports future extensibility, making it a scalable solution for personal and institutional use. This project successfully delivers a user-centric, data-driven application aligned with SAP's enterprise-grade standards.