Smart Journaling

Journaling has long been recognized as a valuable tool for self-expression and personal reflection. It provides individuals with a space to record their thoughts, emotions, and experiences, helping them process their daily lives and reflect on their personal growth.

> *"Journaling is a way of capturing your life one day at a time. It is a wonderful way to be able to reflect on your past whether in good times or bad. It is really easy to forget how wonderful your life is in certain times, and a quick look back allows you to see the people, places visited, and get some perspective."*
>
> — George Bonelli

In today's busy world, many individuals find it difficult to keep track of their daily thoughts, feelings, and experiences. Despite the therapeutic benefits of journaling, people often struggle to maintain a consistent journaling habit or capture the full depth of their emotions. As a result, important memories and reflections can easily be forgotten, leaving individuals disconnected from their past experiences and their emotional journey.

Additionally, mood swings and emotional shifts are often hard to track without a clear record of one's daily state of mind. Without the ability to reflect on their mood patterns over time, people may miss valuable insights into the causes of their emotional fluctuations. This lack of awareness can prevent individuals from recognizing early signs of stress, anxiety, or other mental health concerns, making it harder to address these issues proactively.

Project Introduction

The Smart Journaling project aims to tackle the challenges of emotional awareness and personal reflection by leveraging technology to provide a smart, accessible journaling platform. Through features such as mood tracking, AI-driven prompts, and personalized insights, Smart Journaling seeks to help users better understand their emotional patterns, reflect on their experiences, and foster personal growth. By offering an innovative way to engage with one's emotions and memories, this project aligns with the goals of improving mental health and well-being (SDG 3).

Ultimately, the Smart Journaling platform is designed to empower individuals with the tools they need for self-reflection, emotional regulation, and personal development. By promoting regular journaling and emotional awareness, this project strives to create a positive impact on mental health, contributing to a healthier and more informed society.

At the end of this project, you will also learn to:

1. Call API request in Java
2. Integrate existing AI models into your project using API

Since API calls are not the primary focus of this course, the Java methods handling the API requests will be provided for you. Your main task will be to follow the guide in the Appendix and concentrate on extracting relevant values from the API responses.

What You Will Need To Do

We separated our project functionalities into basic features and extra features. The basic features are the main basic functionalities that our project must have in order to make it workable, the extra features on the other hand will be the additional functionalities which improve the project overall as a whole, which includes scalability and accessibility.

Note that GUI (Graphical User Interface) is considered an extra feature, you may decide to use the CLI (Command-Line Interface) as the interface for the end users to interact with the software.

Basic Features (8 Marks)

User Account & Login / Registration Page [1 mark]
Each user will have their own user account. Therefore, you will need to create a User class which contains the following required fields:

- Email Address
- Password
- Display Name

We will need a sign-up page for the users to register themselves and a login page for users to log into their accounts from any location. To ease yourself in logic management, the user will log in only by using a valid email, instead of the option to log in with a username.

Here, I have created 2 mock user accounts for you to test your program out, you can create more accounts if you want but these 2 accounts have to be in your project for this section's presentation.

`UserData.txt`

```
s100201@student.fop
Foo Bar
pw-Stud#1

s100202@student.fop
John Doe
pw-Stud#2
```

The list above is in the format of EmailAddress, DisplayName, Password, separated by a "New Line" or `\n`. Copy and paste the list above in `UserData.txt` file, meaning that the content of that file should look **EXACTLY** as the content shown above.

Data Storage [1 mark]
Throughout the project, we expect various data generated like user details, user's current state and many more. So, we will need to store all related data in data storage so that the data remains saved even after the program is terminated. For basic features, you may save each user's data in CSV, TXT or BAT format.

Note that using external data storage or databases *(refer to Extra Features)* are counted as extra features. However, you are still required to have **AT LEAST ONE** file I/O application in order to get this mark.

Welcome Page & Menu [1 mark]
After logging in, the user will be greeted with a welcome message on the landing page. The welcome message should start with the following greeting based on the current time, followed by the user's display name. To keep things simple, here are the rules:

Time of the day (GMT+8)	**Greetings**
00:00 - 11:59	Good Morning
12:00 - 16:59	Good Afternoon
17:00 - 23:59	Good Evening

This smart journal project only needs 2 main functionalities:

1. Create, Edit & View Journals
2. View Weekly Mood Summary

Journal Page [2 marks]

Upon selecting the option to create & view journals, users will see a list of past journal dates, extending up to the current day.

Example: Journals Page

```
=== Journal Dates ===
1. 2025-10-08
2. 2025-10-09
3. 2025-10-10
4. 2025-10-11 (Today)

Select a date to view journal, or create a new journal for today:
> 1
```

- For any past date displayed, users have the option to view the journal entry written on that specific day.
- For the current date, the following options are available:
    1. If no journal entry has been recorded for the current date, the user will be prompted to "Create Journal."
    2. Once a journal entry has been created, the user can then choose to "View Journal" or "Edit Journal."

Example: View a Journal Entry

```
=== Journal Entry for 2025-10-10 ===
I had a great day at the park with my friends.

Press Enter to go back.
>
```

When a new journal entry is initiated, a message will appear, prompting the user to begin typing their journal. An example of this process is shown below.

Example: Create a New Journal Entry

```
Select a date to view journal, or create a new journal for today:
> 4

Enter your journal entry for 2025-10-11:
> Today I learned how to create a simple journal app!

Journal saved successfully!

Would you like to:
1. View Journal
2. Edit Journal
3. Back to Dates

> 3
```

Example: Edit Today's Journal Entry

```
Select a date to view journal, or edit the journal for today:
> 4

Edit your journal entry for 2025-10-11:
> Today I learned how to create a simple terminal journal app!
```

Note that when no journal entry exists for the current day, the instructions should prompt the user to "**create a new journal** for today." Once a journal entry has been created, the prompt should change to offer the option to "**edit the journal** for today."

You may rephrase these instructions as you see fit, but the logic must be clear to prevent users from becoming confused about why they can create a new journal if one already exists for the current day.

Weather Recording [1 mark]

Traditional diaries require users to manually note down the weather for the day (given that the user wants to track the weather). In this project, we would automatically retrieve the weather data based on the user's current location.

In order to do so, we will need to use an API to retrieve the current weather data. To understand more about API, you may refer to this article or explore online. But for this project, the API can be considered like a Java method that returns a result based on input parameters.

To ease the process, we will use the following API from Malaysia's official open data portal, which uses the first type of API calling — GET. I have also created a Java class that does the API calling, in which the code can be found in the Appendix.

Sample Weather API Response (Formatted)


```
[
  {
    "location": {
      "location_id": "St009",
      "location_name": "WP Kuala Lumpur"
    },
    "date": "2025-10-11",
    "morning_forecast": "Ribut petir di beberapa tempat",
    "afternoon_forecast": "Ribut petir di beberapa tempat",
    "night_forecast": "Tiada hujan",
    "summary_forecast": "Ribut petir di beberapa tempat",
    "summary_when": "Pagi dan Petang",
    "min_temp": 23,
    "max_temp": 34
  }
]
```


This API returns the weather data and weather forecast of the current date, and for additional simplicity, we will only extract the value of the `summary_forecast` field for the current date. So if the current date is 11 October 2025, the weather would be "Ribut petir di beberapa tempat". Finally, link the weather to the journal entry for that day.

This part focuses more on testing your knowledge on integrating methods from other Java classes and also value extraction from String objects. You may also play with different API to obtain the weather data.

Sentiment Analysis [1 mark]
Sometimes writing down your mood is difficult as it is too abstract to describe your mood, thus this project uses sentiment analysis model to classify your mood sentiment based on the input journal sentence.

Similarly, we will be using API calling from Hugging Face's model. By leveraging existing open sourced models, we are able to incorporate AI functionalities into our system and make our project "smarter". Here we will use DistilBERT base uncased finetuned SST-2 model to provide sentiment classification on the journal sentence.

This part will introduce you to the second kind of API call — POST. Read more on the implementation and code examples in the Appendix.

Sample Model API Response (Formatted)


```
[
    [
        {
            "label":"POSITIVE",
            "score":0.9998701810836792
        },
        {
            "label":"NEGATIVE",
            "score":0.00012976663128938526
        }
    ]
]
```


From the sample API response, we see that the response includes 2 labels each with a score. The score can be considered as the likelihood of the label being true for the input journal sentence, thus we will only use the label with the higher score. Link the label as mood to the journal entry for that day.

You may also experiment with different available models in Hugging Face or even try an entirely different model API.

p.s. I realized that the label that has the highest score will display first, this finding could perhaps help you out in value extraction.

Weekly Summary Page [1 mark]
Lastly, the weekly summary page provides an overview of the weather and mood from the past seven days, allowing users to review the weather changes over the week and also their mood fluctuations throughout the week.

Suggested Extra Features (4 Marks)

Marks at this section are given based on the amount of impact or significance that the suggested features or extra features by students has towards the project as a whole. It is SUBJECTED to the demonstrator or the lecturer giving the marks to decide how much marks should be awarded for each extra feature.

Graphical User Interface (1 - 3 Marks)
A graphical user interface (GUI) is a digital interface in which a user interacts with graphical components such as icons, buttons, and menus. In a GUI, the visuals displayed in the user interface convey information relevant to the user, as well as actions that they can take. A nice-looking and user-friendly GUI will give the user a better experience using the software. You may choose to use JavaFX or Spring Boot with other technologies to do so.

Relational Database (1 - 2 Marks)
A relational database is a collection of information that organizes data in predefined relationships where data is stored in one or more tables (or "relations") of columns and rows, making it easy to see and understand how different data structures relate to each other. Relationships are a logical connection between different tables, established on the basis of interaction among these tables. You may use Oracle Database, MySQL, PostgreSQL, Firestore or other relational databases to do so.

Note that the parts where you are required to copy and paste the content in this assignment question into a text file like the `UserData.txt`, you may load them into your database instead of writing it in a txt file. Just to remind that you are still required to have **AT LEAST ONE** file I/O application in order to get the mark for Data Storage at Basic Feature.

Password Hashing (1 Mark)

Storing the raw passwords of our users in any kind of storage violates the basic privacy of the users. According to the United Kingdom's *Article 5 of Regulation (EU) 2016/679* of the European Parliament and of the Council, personal data shall be processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures ('integrity and confidentiality'). You may use hashing, Caesar cipher, or any other encryption algorithm to tackle this matter. However, you must be able to justify the algorithm during the presentation. Plus, you should also show that your database or the text file's password section stores the hashed version of the password.

Expanding Functionalities on Existing Features (1 - 3 Marks)

Since this project is super basic compared to existing systems like Day One or Reflection, you are highly encouraged to add some functionalities to each page or existing features. One simple example is to create a mapping rule (if-else) to map different possible values of the `weather_forecast` field to simpler English terms like "sunny", "rainy" etc.

Note that the impact of your added functionalities affect the marks awarded as extra features, i.e. more impactful or significant functionality rather than a small add-on grants you more marks.

Tips For This Assignment

To help you complete the assignment easily, here are some tips from our experiences in doing project-based assignments.

Modularity

This project is modular and can be separated into a few parts:

- Login / User Registration, User Class creation and Data Storage
- Welcome Page and Summary Page
- Journal Page
- Weather Value Extraction
- Mood Classification Value Extraction

This eases your team to delegate the tasks among the team members to effectively construct each functionality separately and compile together after testing has been done. Note that the separation above is just a suggestion and not compulsory to follow while your team distributes tasks and responsibilities.

Version Control

We encourage you to utilize Git versioning and the GitHub platform while collaborating with your team to complete this project. It also serves the purpose of tracking the contribution of each team member in this project and avoiding “free-riders”.

You may read more about Git and GitHub in the following links:

- What is Git?
- Getting Started on GitHub
- Creating Pull Requests on GitHub
- Resolving Merge Conflicts on GitHub

Relative Path

Since there are many File I/O involved in this assignment, there is a tendency that developers use their local absolute file path in their code. Here's a sample difference between absolute path and relative path:

- Absolute Path:
  `C:\Users\Documents\WIX1002\Assignment\SampleInput.txt`

- Relative Path:
  `./SampleInput.txt` (given that the project root ~ is at the `\Assignment\` folder as shown in the Absolute Path)

We strongly advise you to NOT use the absolute path but use the relative path since the file path in your PC will NOT necessarily be the same for your other project collaborators, but the file hierarchy in the project should be the same for all of your project collaborators.

Contact Me

If you have any questions or need clarifications about the assignment, please contact me, Lim Jun Yi, using either of the following methods:

- WhatsApp me at +60123681620
- Email me at 22004811@siswa.um.edu.my

I will try my best to answer your questions as soon as possible. Hope you enjoy this assignment!

Appendix

I have created the necessary files required for API calling in [this GitHub repository](). Inside the repository you should see 2 Java Files and 1 `.gitignore` file:

- `API.java`: Main class containing API calling functions and sample usage
- `EnvLoader.java`: Custom class to read sensitive information from `.env` file
- `.gitignore`: To prevent sensitive information in `.env` from being exposed

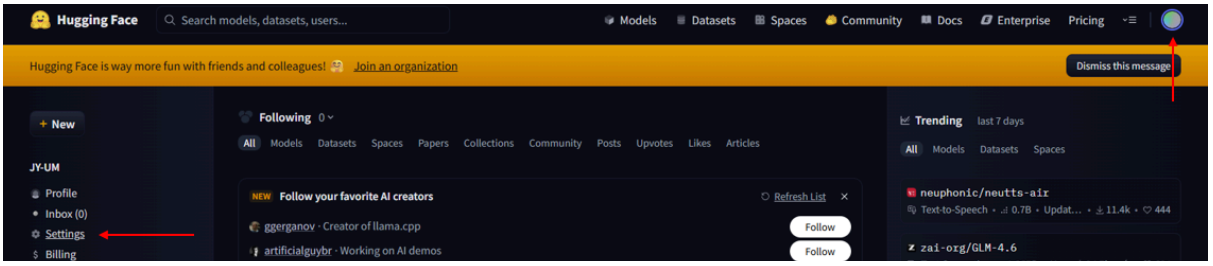
Make sure you copy these 3 files in your project folder.

Getting Access Token for Hugging Face Model API Calling

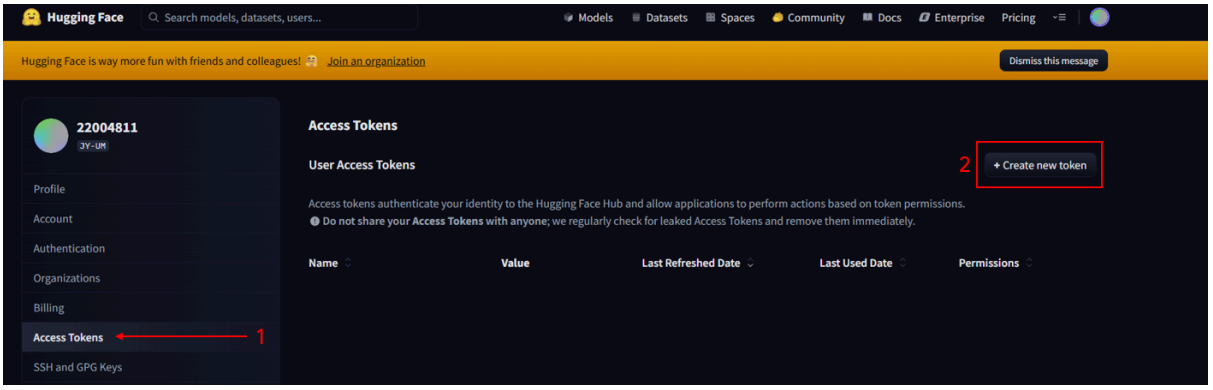
> **IMPORTANT DISCLAIMER:**
> Your Hugging Face access token is **private** and **sensitive information**. Do **NOT** share or expose your token publicly or with others. Each user should use their own token stored securely in a `.env` file. Make sure your `.env` file is included in `.gitignore` to prevent accidental sharing of this sensitive information, especially when using version control like Git. This helps keep your account and data secure.

For hugging face model API calling, we will need to first get an access token from the Hugging Face Website. The steps are listed as follows:

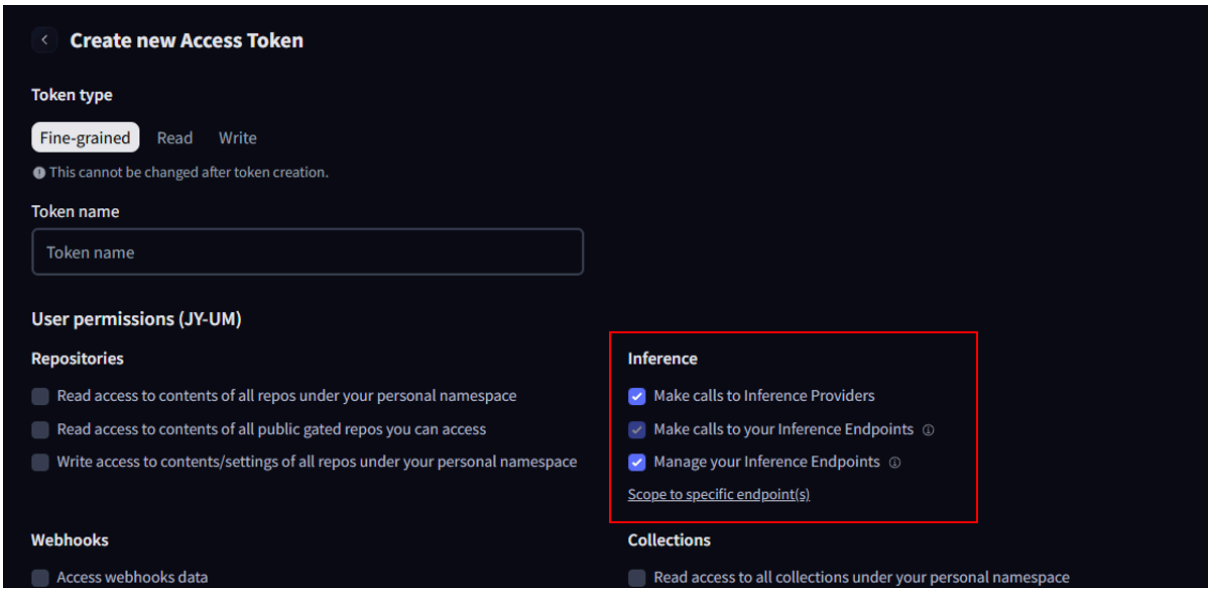
1. Create a new account (unless you already have one) at [HuggingFace]().
2. After logging in, find “Settings” in the side bar or click your profile icon at the top right of the menu bar > “Settings”.



3. Click “Access Tokens” > “Create New Token”



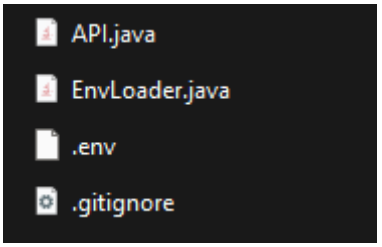
4. Select “Fine-grained”, give this token any name, and check all boxes in the Inference section.



5. Scroll to the bottom and click “Create token”.
6. A pop-up dialog titled “Save your Access Token” will show up, copy your token (“hf_xxxxxxxxxxxxxxxx”).
7. In your project repository, create a file called `.env` and paste your access token similar to below in the file:

.env
BEARER_TOKEN=hf_xxxxxxxxxxxxxxxx

As of now, your project folder should at least have these files:



How to Use API Calling

In `API.java`, look for the comment `// Example Usage`. It contains the sample usage of API calling for GET weather data and POST to obtain mood prediction.

For GET weather data, you will use the same API URL (getURL) as given in the code, and use `api.get()` method to obtain the response string as shown in "Sample Weather API Response (Formatted)". Then, use your knowledge in value extraction to extract the value needed.

For POST to obtain mood prediction, you will also use the same API URL (postURL) as given in the code, ensure that you have the access token in your `.env` file, and initialize the `journalInput` variable with the user's journal input sentence (e.g. "Today I learned how to create a simple terminal journal app!"). This sentence is being used in the `jsonBody` variable which would be "posted" to the model API as model input, and the model API will return the predicted score as shown in "Sample Model API Response (Formatted)" via the `api.post()` method. Then, use your knowledge in value extraction to extract the value needed.

Note: If you want to experiment with different models instead of the one provided (distilbert/distilbert-base-uncased-finetuned-sst-2-english), you may change it to your preferred model as follows:

To use tabularisai/multilingual-sentiment-analysis model, the API URL would be https://router.huggingface.co/hf-inference/models/tabularisai/multilingual-sentiment-analysis

However, please check if the model supports this way of API calling (with URL):

1. In the model page, click "Deploy" and see if there is an option called "Inference Providers".
2. If there is, the model supports this way of API calling.
3. If there isn't, try looking for other models.