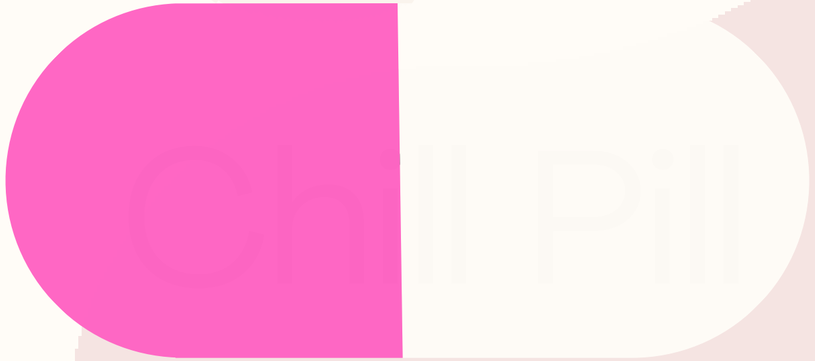


The go-to wellness journal!

Chill Pill

The go-to wellness journal!

Agenda

1. Aims
2. User Interface
3. Backend Server
4. Database
5. Demo

Aims

UI

A harmonic, user-friendly interface.

Journal Management

CRUD features to manage the entire journaling process.

Sentiment analysis and more!

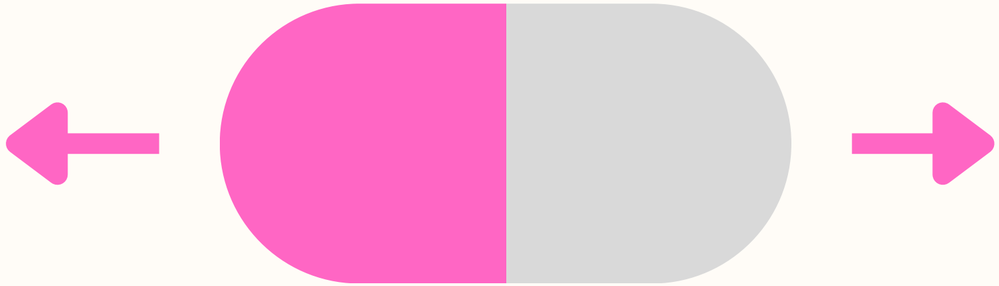
Sentiment API to analyse the user's journal entries and draw insights from.

User Interface

```
<!-- navigation bar. -->
<div class="navbar">
    <a href="http://127.0.0.1:5000">  New Entry</a>
    <a href="entries">  All Entries</a>
    <a href="moodtracker">  Mood Tracker</a>
    <a href="team">  The Team</a>
</div>

<!-- journal page. -->
<div class="journal-page">
    <form id="journal" action="http://127.0.0.1:5000/journal" method="POST">

        <!-- running timestamp. -->
        <div id="label" for="entry">
            <strong>Journal Entry</strong>
            <p id="timestamp"></p>
        </div>

        <textarea id="entry" name="entry" placeholder="How are you doing?
        "></textarea>
        <br><br>
        <button type="submit">Submit</button>
    </form>
    <br><br>

    <!--- amends web page to show a confirmation / error message. -->
    {% if result %}
    <p><strong>
        {{ result }}</strong>
        <br>
        {% if id %}
            entry_id = {{ id }}
        {% endif %}
    </p>
    {% endif %}
</div>
```

1. Entry area
2. Save button
3. Timestamp
4. Navigation bar
5. Entry body
6. Personalised affirmations

UI: Animation



CHILL PILL

```
@keyframes left-pill-animation {
    0% {
        margin-right: 0px;
    }
    100% {
        margin-right: 125px;
    }
}
```

```
#pill-left {
    background-color: #a61266;
    border-top-left-radius: 50px;
    border-bottom-left-radius: 50px;
    animation: left-pill-animation;
    animation-duration: 1s;
    animation-fill-mode: forwards;
}

#pill-right {
    background-color: #ffffec;
    border-top-right-radius: 50px;
    border-bottom-right-radius: 50px;
}
```

Backend Server: OOP & APIs

```python
class JournalEntry:
  def __init__(self, body, sentiment):
    """
    An instance of a journal entry.
    """

    self.body = body
    self.sentiment = sentiment
    self.timestamp = dt.now().strftime("%d/%m/%Y %H:%M:%S")


class JournalManager(DataManager):
    """
    Inherits from DataManager for the journal CRUD functionalities.
    """
    def __init__(self, dbconnection):

        # get the collection from the DB connection.
        self.collection = dbconnection.get_collection()

    def create(self, body, sentiment):
        """
        Creates a journal entry and sends it to the collection.
        """

        # initialise a journal entry as an object.
        entry = JournalEntry(body, sentiment)

        # add entry to mongo db collection.
        submission = self.collection.insert_one({
        'body': entry.body,
        'sentiment': entry.sentiment,
        'timestamp': entry.timestamp
        })
```

Modularising the journal

Using SOLID OOP principles.

API server

Using Flask and the journal and mood tracker models.

Configurating server

Connecting it to the MongoDB.

```python
def sentiment_analysis(text):
    """
    Sentiment analysis of a piece of text. Returns the positivity value on a scale
    """

    url = 'http://text-processing.com/api/sentiment/'
    textblob = TextBlob(text)
    subj = textblob.sentiment.subjectivity

    # subjectivity analysis to exclude objective entries.
    if subj > 0.2:

        # API has an 80k char limit.
        if len(text) < 80000:

            try:
                req = requests.post(url, data={"text": text}, timeout=10)

                # perform sentiment analysis on OK response.
                if req.status_code == 200:
                    data = req.json()
                    pos = data.get('probability', {}).get('pos')

                    # only return pos value is numeric.
                    if isinstance(pos, int) or isinstance(pos, float):
                        return format(pos * 10, '.2f')

                    # otherwise return no data.
                    return

                # otherwise return no data.
                return

            except Exception as e:
                print("Please see error below.")
                print(e)

                # return no data if exception.
                return

        # return no data if subjectivity <= 0.2.
        return
```

Backend Server: APIs & Affirmations

Sentiment API

- Text-Processing sentiment API
- Represent sentiment numerically from 0-10 (negative to positive)

Subjectivity Analysis

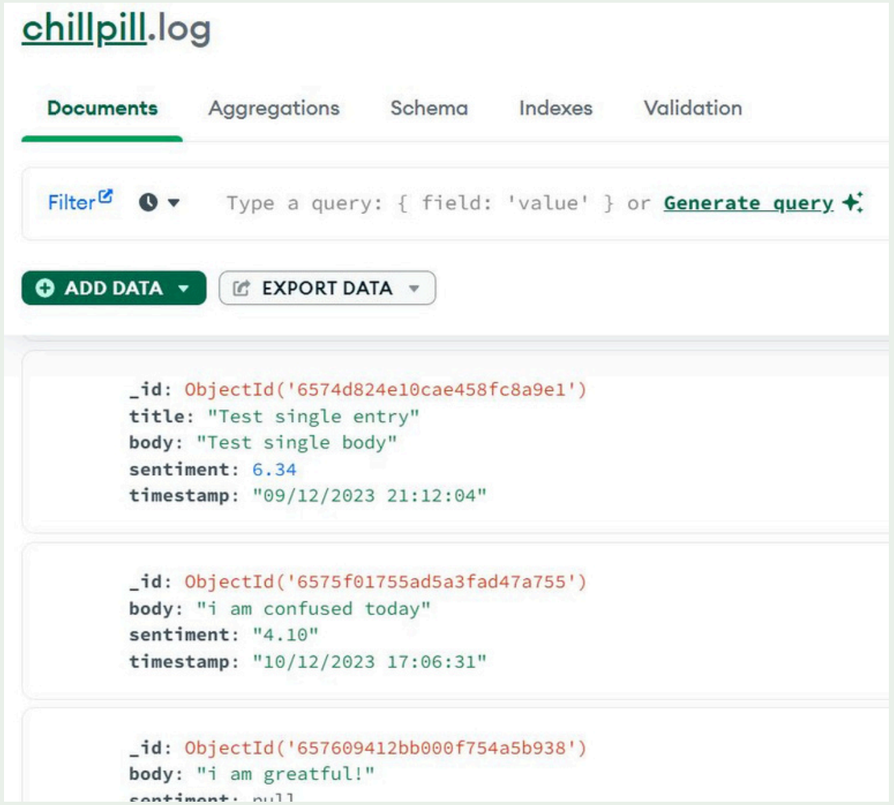
- TextBlob subjectivity tool to analyse subjectivity of entries
- Filter out objective by removing entries with subjectivity values less than 0.2.

Affirmation Generator

- Classify sentiment as keys within a dictionary
- Associate each key with a list of various affirmations.
- Generate random affirmation based on entry

Database

We used MongoDB to store input data. Tailored to storing data efficiently, provides flexible schema.



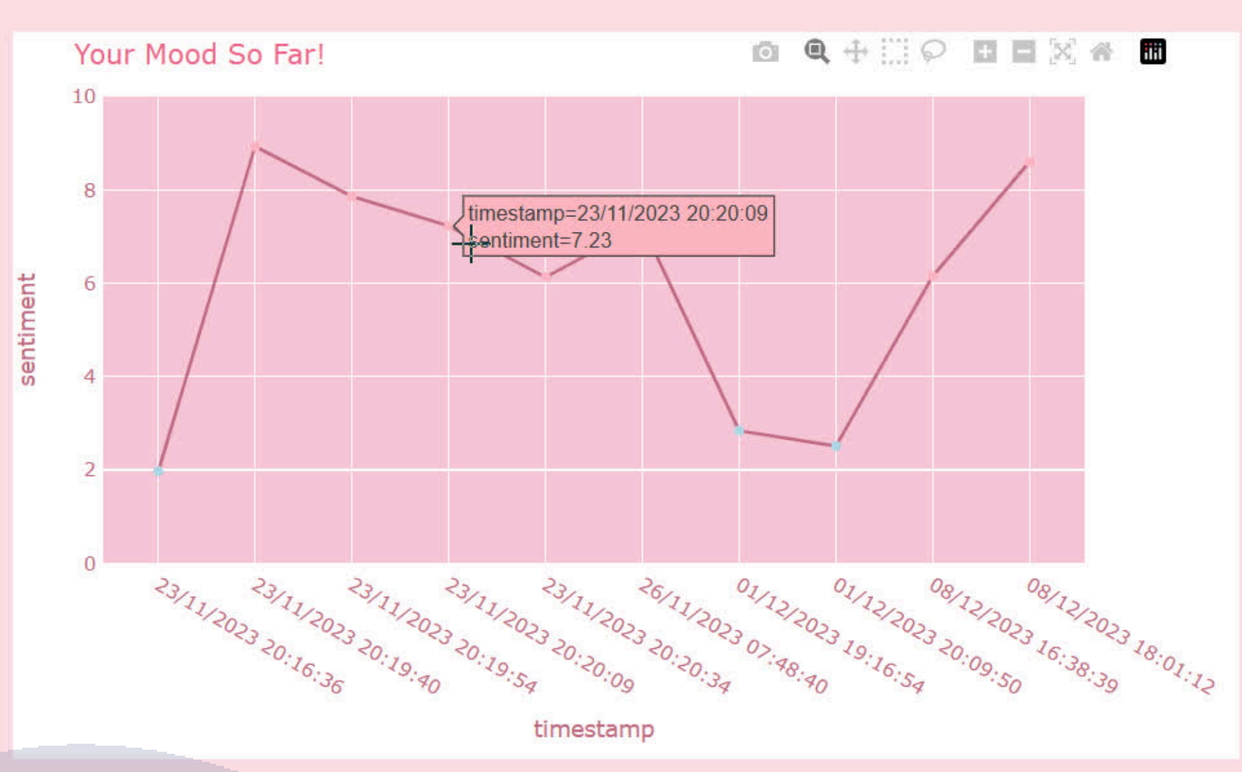
Endpoints in the application:

- **UPDATE:** processes a POST request to delete a specific entry by its entry_id.
- **DELETE:** processes a POST request to modify an entry's content, conducts sentiment analysis

```python
@app.route('/delete/<entry_id>', methods=['POST'])
def delete_entry(entry_id):
    """
    Deletes an entry, redirects to the entries page.
    """
    journal.manager.delete(entry_id)

    # return to entries page.
    entries_data = journal.manager.read_all()

    if isinstance(entries_data, dict):
        entries_data = [entries_data]

    return redirect(url_for('entries', entries_data=entries_data))
```

```python
@app.route('/update/<entry_id>', methods=['POST'])
def update_entry(entry_id):
    """
    Updates an entry, redirects to the entries page.
    """

    try:
        # get the updated details.
        entry = request.form.get('entry')

        if not entry:
            # otherwise, throw an error template and return to the og entry.
            result = '404 Error: Entry body not found.'
            entry_data = journal.manager.read_one(entry_id)

            return render_template('entry.html', entry_data=entry_data, result=result)

        time = dt.now().strftime("%d/%m/%Y %H:%M:%S")

        # update sentiment analysis on updated entry.
        sentiment = sentiment_analysis(entry)

        update_data = {
            'body': entry,
            'sentiment': sentiment,
            'last timestamp': time
        }

        journal.manager.update(entry_id, update_data)
```

The Mood Tracker



Sentiment analysis is colour-coded based on the score:

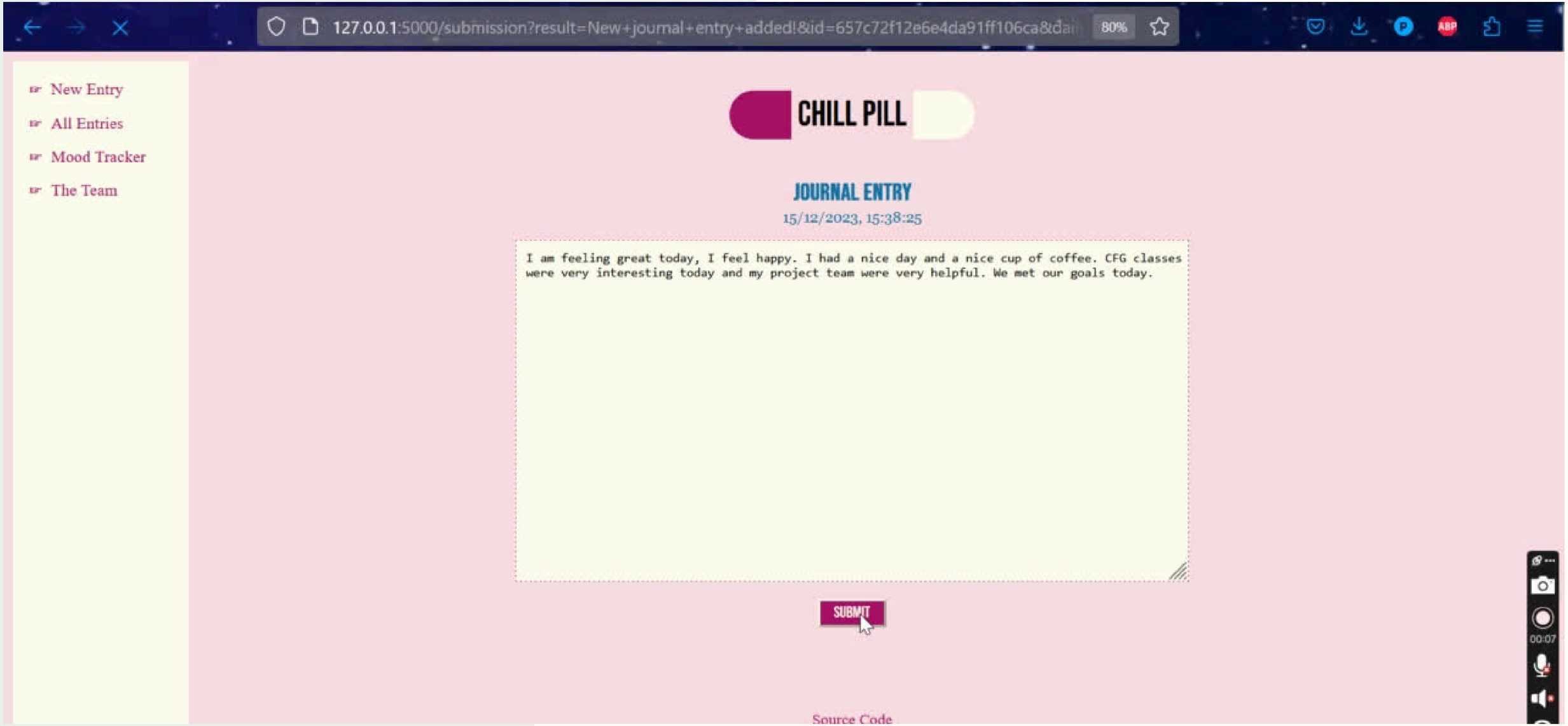
- pink for scores above 6
- purple for 4-6
- blue for scores below 4

Data analysis for latest statistics:

- highest and lowest sentiment scores
- average score for the last 7 entries

Demo

Live Demo brought to you by Group 5!


127.0.0.1:5000/submission?result=New+journal+entry+added!&id=657c72f12e6e4da91ff106ca&da
80%
New Entry
All Entries
Mood Tracker
The Team
CHILL PILL
JOURNAL ENTRY
15/12/2023, 15:38:25
I am feeling great today, I feel happy. I had a nice day and a nice cup of coffee. CFG classes were very interesting today and my project team were very helpful. We met our goals today.
SUBMIT
Source Code
00:07

Thank you!