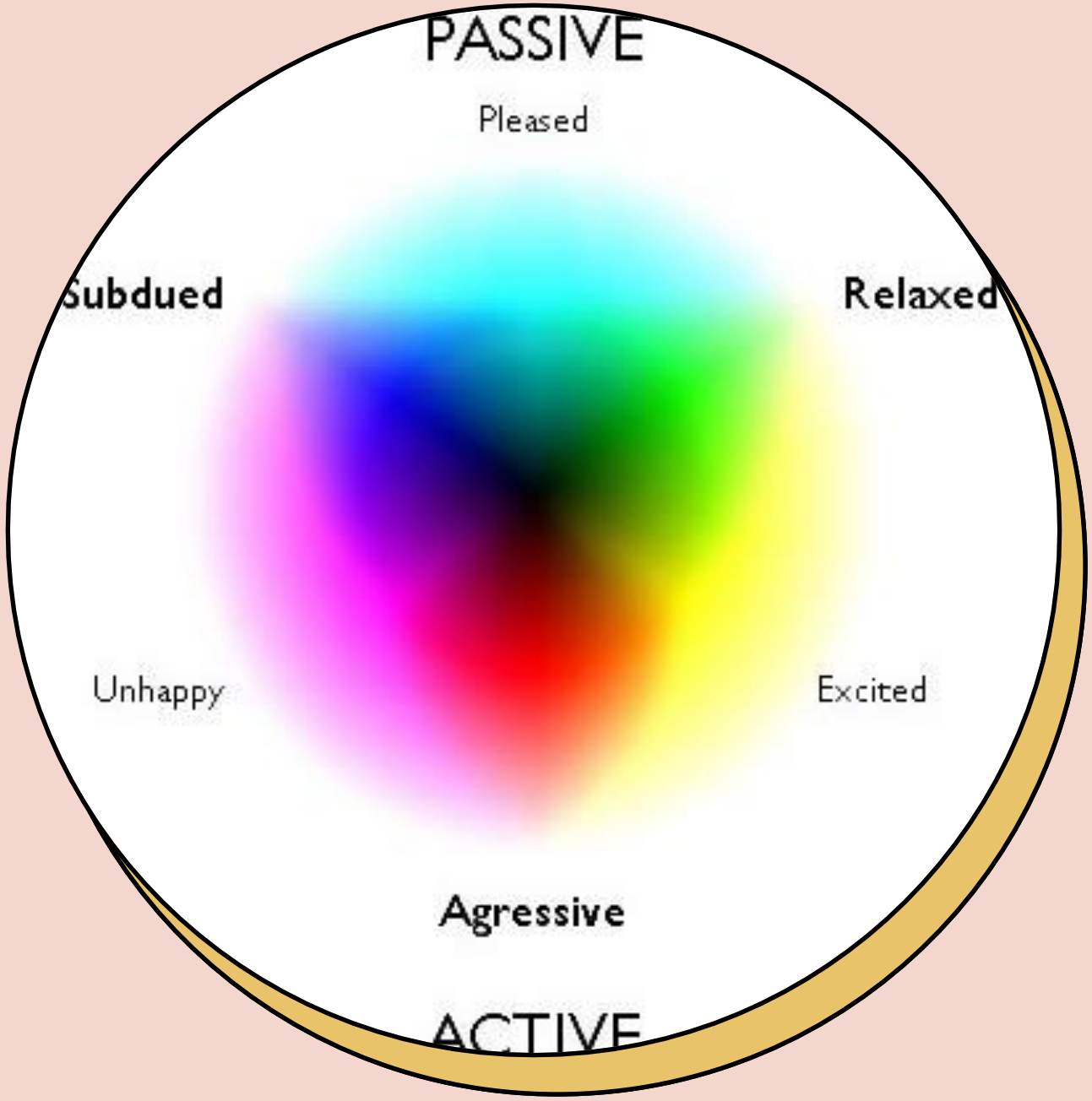
Mood Tracker

Capgemini NGT Programming Assignment



A lightweight, fully-tested CLI app to log your daily mood, view mood history, and generate distribution & streak reports.

By Harshita Thota

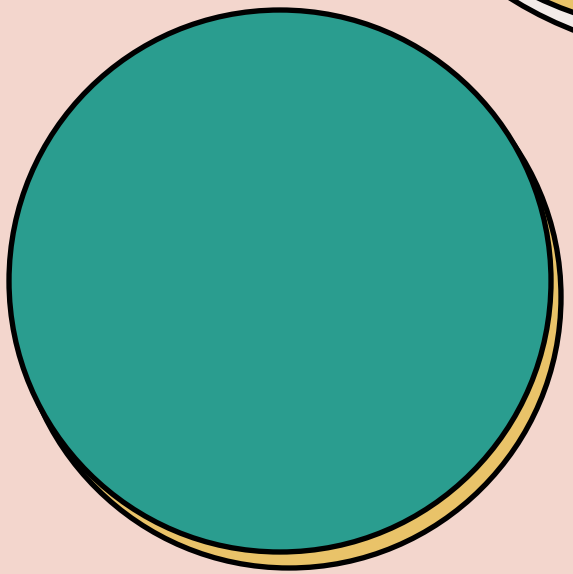
Context & Motivation

Purpose of a Mood Tracker?

- **Self-awareness**: helps users notice emotional patterns over days or weeks
- **Early warnings**: spot stretches of negative moods (anxiety, sadness) before they escalate
- **Data-driven reflection**: quantify how activities, sleep, or social time impact well-being
- **Simple habit**: logging 10 seconds a day builds mindfulness and mental health habit

Requirements and Scope

- Functional requirements:
  - Log a mood entry (date, mood, note)
  - View mood log by date, and mood type
  - Reports short-term and long-term state of mood and mood swings via distribution, streak, trend
- Non-functional constraints:
  - Data kept in memory or file
  - Public GitHub repo + short presentation
  - Repo link: Click here to go to Repo!



Architecture & Layers

[User Interface]

- CLI commands

↓

[Controller Layer]

- Parses user input, calls service

↓

[Service / Analytics (Business Logic)]

- use-case functions (add_entry, get_distribution)
- distribution, streak

↓

[Repository Layer]

- load/save entries from entries.json

Tech Stack & Rationale

- Python 3.11
  - Batteries-included standard library
  - Universally available, easy to package
- Click for CLI
  - Simple decorators for commands/options
  - Automatic help text & validation
- JSON storage (entries.json)
  - No external DB dependency
  - Human-readable & version-controllable
  - Repository abstraction makes adding SQLite/Postgres trivial
- Pytest
  - Fast, expressive tests for each layer
  - Easy coverage reporting (`pytest-cov`)
- Bash (`run.sh`)
  - One-liner demo for reviewers
  - Runs all reports automatically

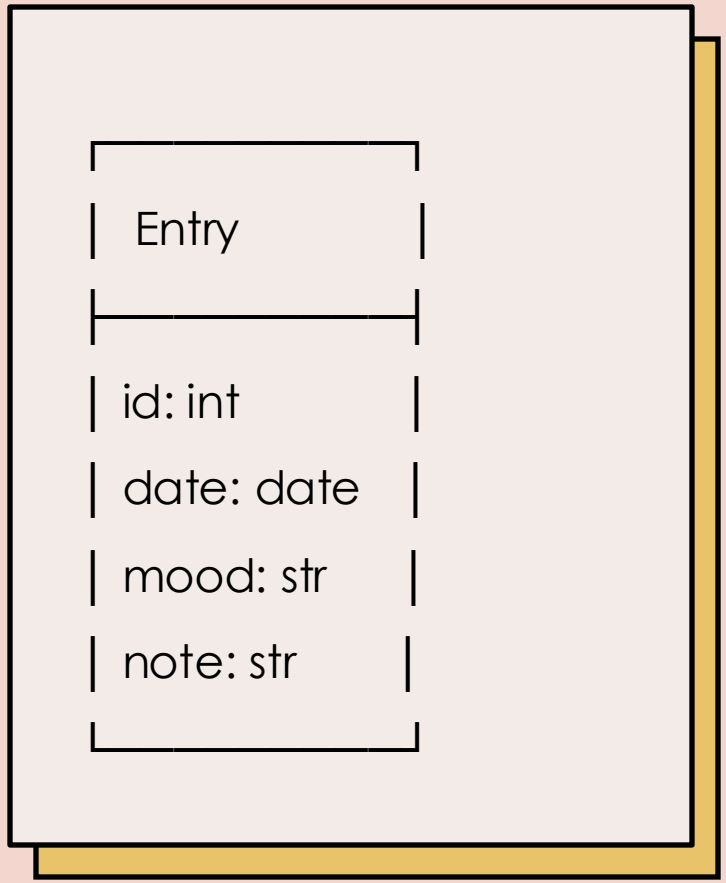
```
(.venv) (base) harshitathota@Harshitas-MacBook-Air Mood-Tracker % pytest --maxfail=1 --disable-warnings -q
pytcst --cov=app
.............                                                            [100%]
13 passed in 0.05s
============================================== test session starts ==============================================
platform darwin -- Python 3.11.7, pytest-7.4.0, pluggy-1.6.0
rootdir: /Users/harshitathota/Mood-Tracker
plugins: cov-6.2.1, Faker-36.1.1, anyio-4.2.0
collected 13 items

tests/test_analytics.py .........                                        [ 76%]
tests/test_repository.py ...                                             [100%]

================================================ tests coverage ================================================
_____________________________________ coverage: platform darwin, python 3.11.7-final-0 _____________________________________

Name                Stmts   Miss  Cover
---------------------------------------
app/__init__.py         0      0   100%
app/analytics.py       31      0   100%
app/cli.py             51     51     0%
app/models.py           9      0   100%
app/repository.py      18      0   100%
app/service.py         22     22     0%
---------------------------------------
TOTAL                 131     73    44%
============================================== 13 passed in 0.10s ==============================================
```

Data Model & Class Diagram



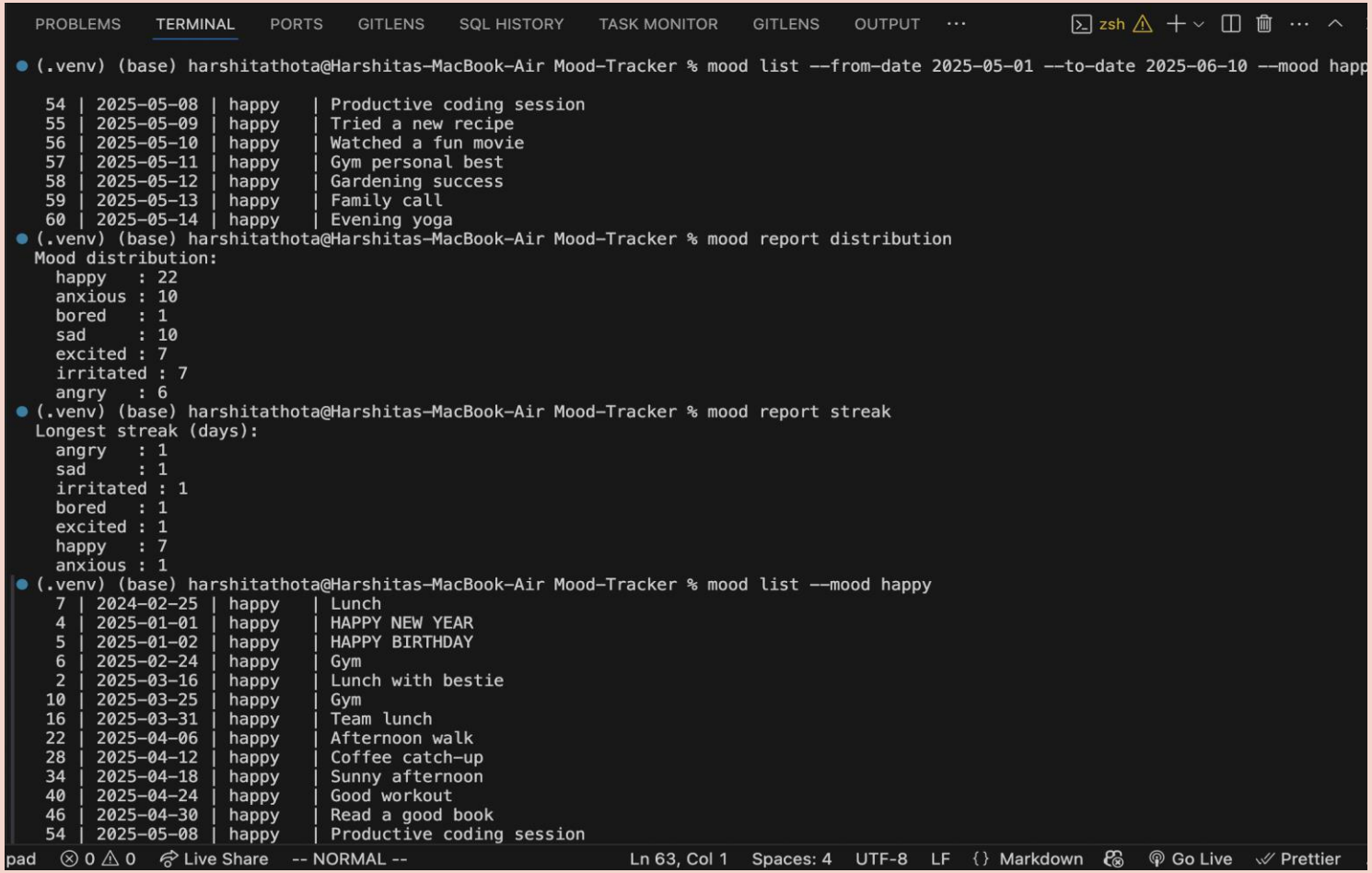
Entry holds one log record

Stored as a JSON array of objects

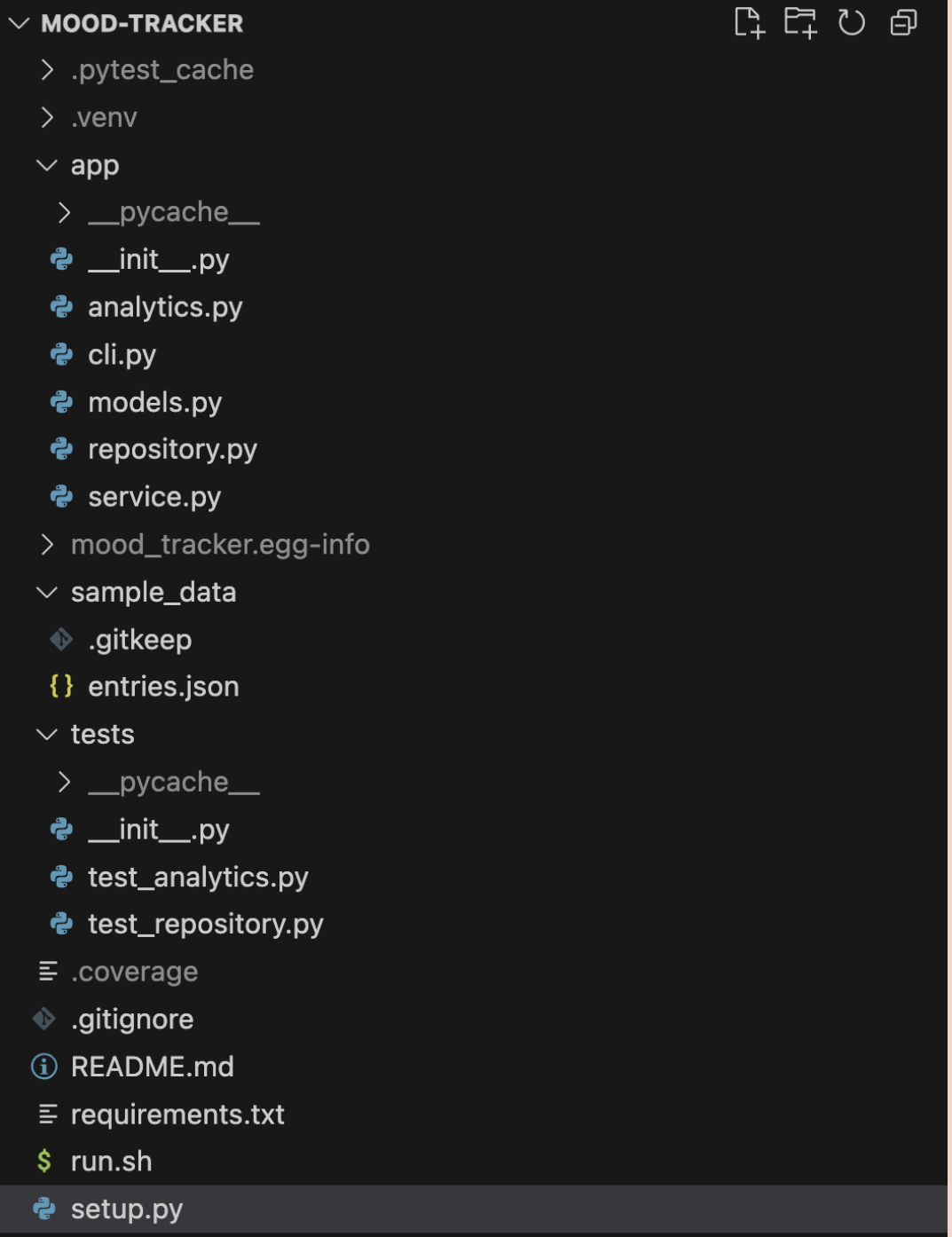

```
[
  {
    "id": 1,
    "mood": "happy",
    "date": "2025-06-26",
    "note": "Lunch with bestie"
  },
  {
    "id": 2,
    "mood": "happy",
    "date": "2025-03-16",
    "note": "Lunch with bestie"
  },
  {
    "id": 3,
    "mood": "sad",
    "date": "2025-03-17",
    "note": "F on exam"
  },
```

UI Design: CLI Usage

- The first command filters the log to only happy” entries within that date range, printing each entry’s ID, date, mood, and note.
- mood report distribution aggregates and prints total counts for each mood type across the entire dataset.
- mood report streak calculates and displays your longest consecutive-day streak for each mood.
- mood list --mood happy shows all “happy” entries in reverse-chron order, demonstrating flexible single-parameter filtering.



Meaning this application allows users to view the mood log by date and mood type, providing insight into the short-term and long-term state of their mood and mood swings.



Repo Structure & Key Files

```
mood-tracker/
├── app/
│   ├── models.py       # Entry dataclass
│   ├── repository.py   # JSON persistence
│   ├── analytics.py    # distribution, streak, trend
│   ├── service.py      # business logic
│   └── cli.py          # Click CLI (controller)
├── tests/              # pytest suites (repo, analytics, service, cli)
├── run.sh              # demo script
├── requirements.txt
└── README.md
```

- **Tests:** ensure each layer works in isolation, plus CLI integration

All this information plus additional information on the features of the application are listed in the README.md file!!

How to Run, Test & Verify

1. Setup:

```
1. Clone & enter
git clone https://github.com/HarshitaThota/Mood-Tracker.git
cd Mood-Tracker

2. Create & activate venv
python3 -m venv .venv
source .venv/bin/activate      # macOS/Linux
.venv\Scripts\activate         # Windows

# 3. Install dependencies
pip install -r requirements.txt
```

2. Seed/demo data: `bash run.sh`

- I ship a `run.sh` script that seeds three example moods (`happy`, `sad`, `anxious`) into `entries.json` so you see real data on first run.

```
# Seed via CLI
echo "⏳ Adding demo entries"
mood add --mood happy   --date 2025-06-25 --note "Gym"
mood add --mood sad     --date 2025-06-26 --note "Rainy day"      You,
mood add --mood anxious --date 2025-06-27 --note "Project deadline"
```

3. Test data:

```
pytest --maxfail=1 --disable-warnings -q
pytest --cov=app
```

- Unit-tests live in `tests/` to validate distribution, streak, and date-filtering logic

Conclusion & Next Steps

- In Summary:
  - Delivered a layered, fully-tested CLI app
  - Covers all Capgemini requirements: logging, viewing, distribution, streaks
- Future Work:
  - Can swap JSON → Postgres/SQLite for real-world scale
  - Add update/delete operations
  - Build out Flask front-end with live charts
  - NLP on notes for sentiment trends