1. Use Case Realization & Operation Specification

Use Case: Register User

Goal: Create a new user account.

Main Flow:

1. User submits username, email, and password.
2. System checks if username and email are unique.
3. System hashes the password.
4. System stores the new user in the Users table.
5. System returns the new user_id.

Operations:

- registerUser(username, email, password) → user_id or error
- isEmailAvailable(email) → true/false
- isUsernameAvailable(username) → true/false
- hashPassword(password) → password_hash

Use Case: Login User

Goal: Authenticate a user.

Main Flow:

1. User submits email and password.
2. System retrieves user by email.
3. System verifies password hash.
4. System issues authentication token.

Operations:

- loginUser(email, password) → authToken or error
- getUserByEmail(email) → user record
- verifyPassword(password, password_hash) → true/false

Use Case: Create Mood Post

Goal: User records a mood entry.

Main Flow:

1. User submits mood_text and optional mood_score/mood_category.
2. System validates input.
3. System inserts a new post.
4. System triggers AI analysis.
5. System returns the created post.

Operations:

- createPost(user_id, mood_text, mood_score, mood_category) → post_id
- getPostById(post_id) → post
- triggerAnalysis(post_id) → analysis_id

Use Case: Add Comment

Goal: User comments on a post.

Main Flow:

1. User submits comment_text.
2. System validates user and post.
3. System stores comment.
4. System returns comment_id.

Operations:

- addComment(user_id, post_id, comment_text) → comment_id
- getCommentsForPost(post_id) → list of comments

Use Case: View Feed

Goal: User views their mood history and AI support.

Main Flow:

1. User requests feed.
2. System retrieves posts.
3. System retrieves analysis and recommendations for each post.
4. System returns combined feed.

Operations:

- getUserFeed(user_id) → list of posts with AI data

- getAnalysisByPost(post_id) → analysis
- getRecommendationsByAnalysis(analysis_id) → recommendations

Use Case: Analyze Mood

Primary Actor

User

Description

The system analyzes user-submitted mood text and generates a personalized wellness recommendation.

Preconditions

- User is logged in
- User submits mood text

Postconditions

- Sentiment classification returned
- Recommendation category determined
- Wellness recommendation generated

System Flow

1. User enters mood text
2. System sends text to SentimentService
3. SentimentService returns label + confidence
4. RecommendationEngine maps label to category

5. System returns final response

Operation Specifications

Operation 1

Class: SentimentService
Method: analyze(text: String)

Input: mood text
Output: sentiment label + confidence

Process:

- Call HuggingFace model
- Extract label and score
- Return results

Operation 2

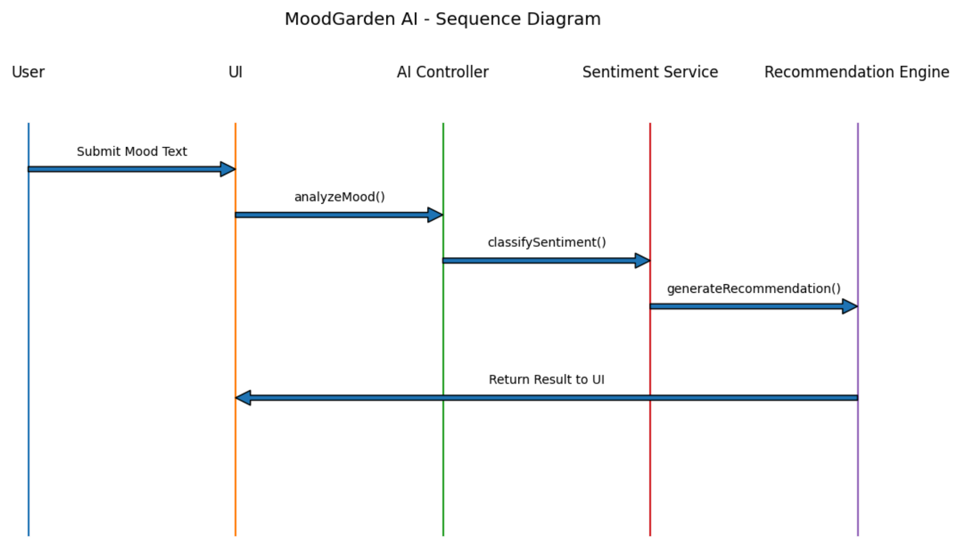
Class: RecommendationEngine
Method: generate(sentiment: String, confidence: double)

Input: sentiment + confidence
Output: recommendation string

Process:

- If POSITIVE → MAINTAIN
- If NEGATIVE → CALM
- If low confidence → NEUTRAL
- Return predefined recommendation

2. Interaction Diagram (Sequence Diagram)



3. ERD

Backend version:

Entities:

Users

- user_id (PK)
- username
- email
- password_hash
- created_at
- updated_at

Posts

- post_id (PK)

- user_id (FK → Users)
- mood_text
- mood_score
- mood_category
- created_at
- updated_at

Analysis

- analysis_id (PK)
- post_id (FK → Posts)
- sentiment_label
- sentiment_score
- emotion_tags
- summary
- created_at

Recommendations

- recommendation_id (PK)
- analysis_id (FK → Analysis)
- quote_text
- image_prompt
- image_url
- resource_links
- created_at

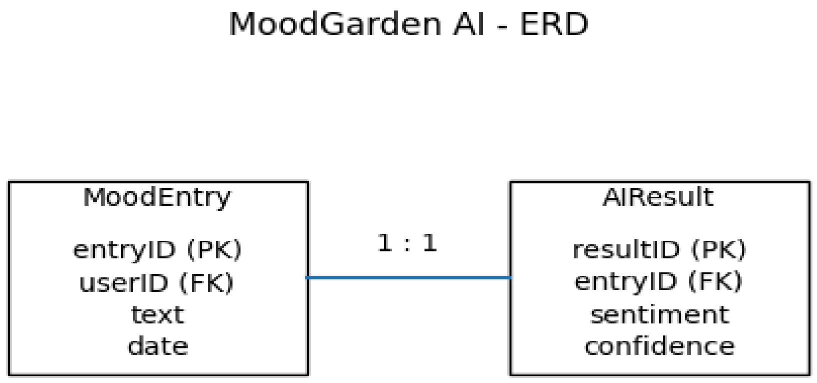
Comments

- comment_id (PK)
- post_id (FK → Posts)
- user_id (FK → Users)
- comment_text
- created_at

Relationships:

- One user has many posts.
- One post has one analysis.
- One analysis has many recommendations.
- One post has many comments.
- One user has many comments.

(AI Portion)



4. Database Design

```
CREATE TABLE Users (
   user_id        UUID PRIMARY KEY,
   username       VARCHAR(50) UNIQUE NOT NULL,
   email          VARCHAR(255) UNIQUE NOT NULL,
   password_hash  VARCHAR(255) NOT NULL,
   created_at     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Posts (
    post_id        UUID PRIMARY KEY,
    user_id        UUID NOT NULL,
    mood_text      TEXT NOT NULL,
    mood_score     INT,
    mood_category  VARCHAR(50),
    created_at     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_posts_user
        FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE Analysis (
    analysis_id      UUID PRIMARY KEY,
    post_id          UUID NOT NULL,
    sentiment_label  VARCHAR(50),
    sentiment_score  FLOAT,
    emotion_tags     JSON,
    summary          TEXT,
    created_at       TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_analysis_post
        FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);

CREATE TABLE Comments (
    comment_id    UUID PRIMARY KEY,
    post_id       UUID NOT NULL,
    user_id       UUID NOT NULL,
    comment_text  TEXT NOT NULL,
    created_at    TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_comments_post
        FOREIGN KEY (post_id) REFERENCES Posts(post_id),
    CONSTRAINT fk_comments_user
        FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

Table: MoodEntry

Field	Type
entryID	int
userID	int
text	varchar
date	datetime

Table: AIResult

Field	Type
resultID	int
entryID	int
sentiment	varchar
confidence	decimal
category	varchar
recommendation	text

5. Test Cases (Unit + Integration)

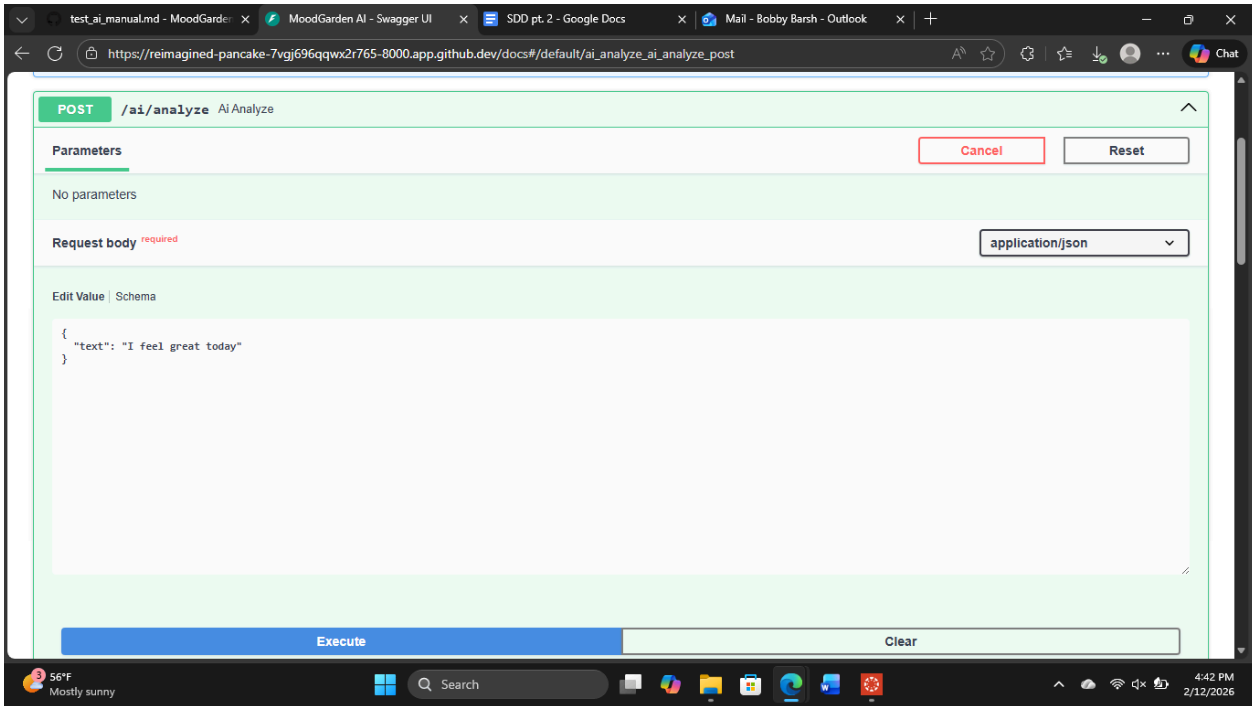
Test Case 1 – Positive Input

Input:
"I feel great today"

Expected:

- sentiment = POSITIVE
- category = MAINTAIN
- recommendation returned

Actual Result:


POST `/ai/analyze` Ai Analyze

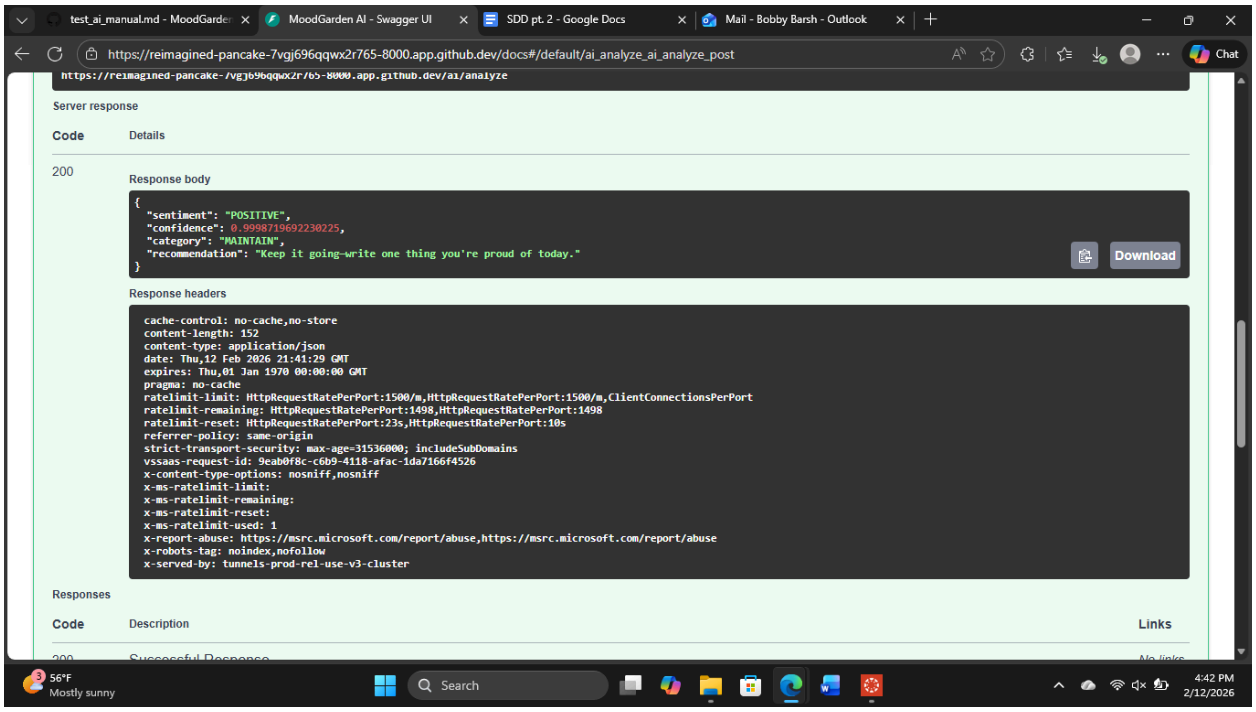
Parameters

No parameters

Request body required — application/json

Edit Value | Schema

```
{
  "text": "I feel great today"
}
```


```
https://reimagined-pancake-7vgj696qqwx2r765-8000.app.github.dev/ai/analyze
```

Server response

Code: 200

Response body

```
{
  "sentiment": "POSITIVE",
  "confidence": 0.9998719692230225,
  "category": "MAINTAIN",
  "recommendation": "Keep it going—write one thing you're proud of today."
}
```

Response headers

```
cache-control: no-cache,no-store
content-length: 152
content-type: application/json
date: Thu,12 Feb 2026 21:41:29 GMT
expires: Thu,01 Jan 1970 00:00:00 GMT
pragma: no-cache
ratelimit-limit: HttpRequestRatePerPort:1500/m,HttpRequestRatePerPort:1500/m,ClientConnectionsPerPort
ratelimit-remaining: HttpRequestRatePerPort:1498,HttpRequestRatePerPort:1498
ratelimit-reset: HttpRequestRatePerPort:23s,HttpRequestRatePerPort:10s
referrer-policy: same-origin
strict-transport-security: max-age=31536000; includeSubDomains
vssaas-request-id: 9eab0f8c-c6b9-4118-afac-1da7166f4526
x-content-type-options: nosniff,nosniff
x-ms-ratelimit-limit:
x-ms-ratelimit-remaining:
x-ms-ratelimit-reset:
x-ms-ratelimit-used: 1
x-report-abuse: https://msrc.microsoft.com/report/abuse,https://msrc.microsoft.com/report/abuse
x-robots-tag: noindex,nofollow
x-served-by: tunnels-prod-rel-use-v3-cluster
```

Status: PASS

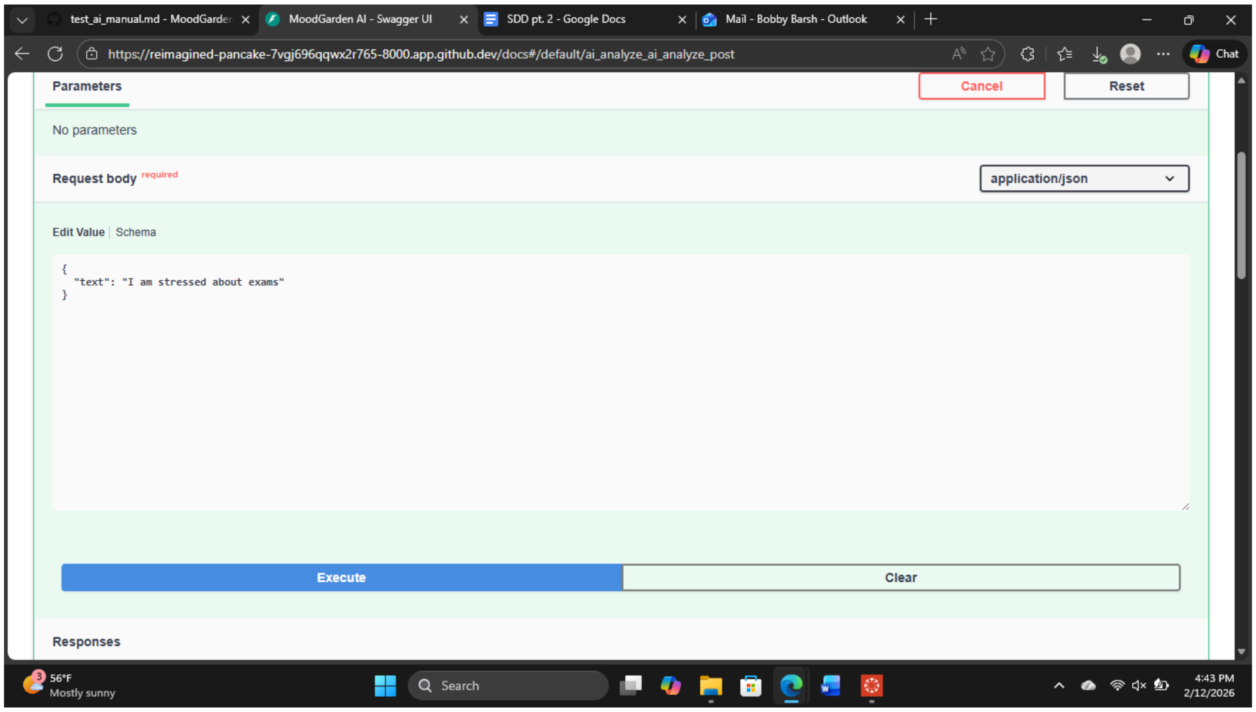
Test Case 2 – Negative Input

Input:
"I am stressed about exams"

Expected:

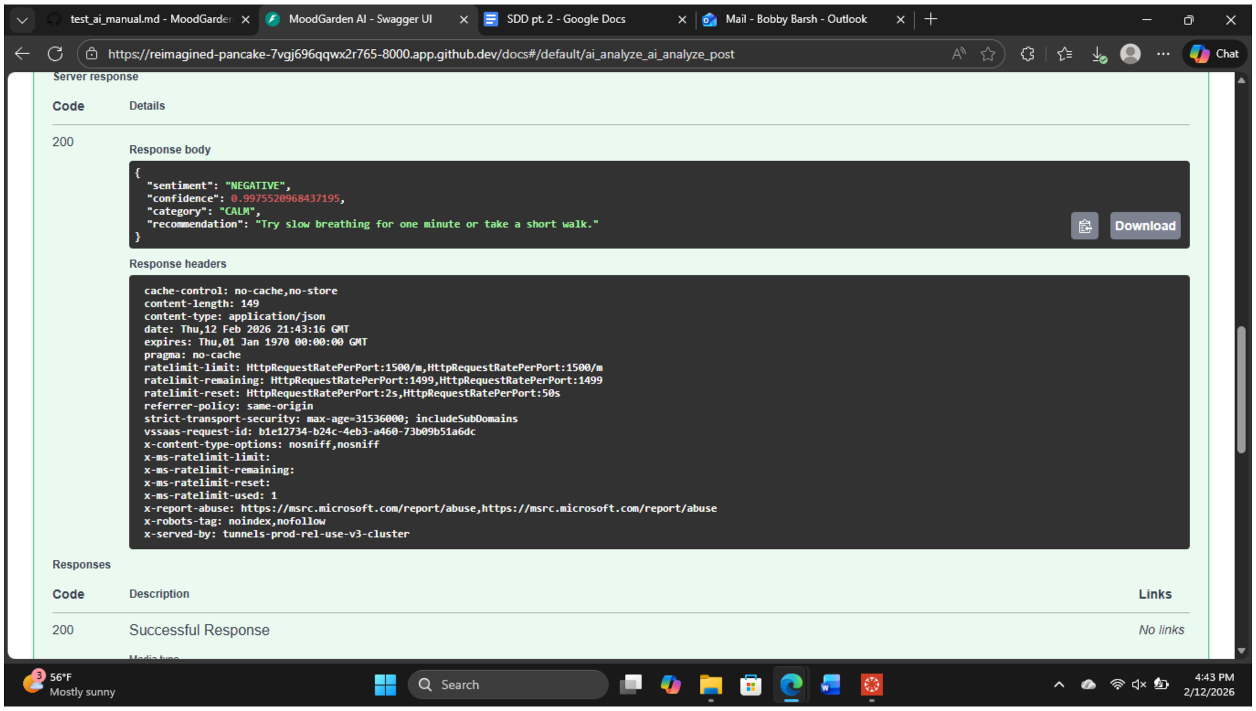
- sentiment = NEGATIVE
- category = CALM
- recommendation returned

Actual Result:



test_ai_manual.md - MoodGarden
MoodGarden AI - Swagger UI
SDD pt. 2 - Google Docs
Mail - Bobby Barsh - Outlook
https://reimagined-pancake-7vgj696qqwx2r765-8000.app.github.dev/docs#/default/ai_analyze_ai_analyze_post
Parameters
Cancel
Reset
No parameters
Request body required
application/json
Edit Value | Schema
Execute
Clear
Responses
56°F Mostly sunny
Search
4:43 PM 2/12/2026

```
{
  "text": "I am stressed about exams"
}
```


Server response

Code Details

200

Response body

```
{
  "sentiment": "NEGATIVE",
  "confidence": 0.9975520968437195,
  "category": "CALM",
  "recommendation": "Try slow breathing for one minute or take a short walk."
}
```

Response headers

```
cache-control: no-cache,no-store
content-length: 149
content-type: application/json
date: Thu,12 Feb 2026 21:43:16 GMT
expires: Thu,01 Jan 1970 00:00:00 GMT
pragma: no-cache
ratelimit-limit: HttpRequestRatePerPort:1500/m,HttpRequestRatePerPort:1500/m
ratelimit-remaining: HttpRequestRatePerPort:1499,HttpRequestRatePerPort:1499
ratelimit-reset: HttpRequestRatePerPort:2s,HttpRequestRatePerPort:50s
referrer-policy: same-origin
strict-transport-security: max-age=31536000; includeSubDomains
vssaas-request-id: b1e12734-b24c-4eb3-a460-73b09b51a6dc
x-content-type-options: nosniff,nosniff
x-ms-ratelimit-limit:
x-ms-ratelimit-remaining:
x-ms-ratelimit-reset:
x-ms-ratelimit-used: 1
x-report-abuse: https://msrc.microsoft.com/report/abuse,https://msrc.microsoft.com/report/abuse
x-robots-tag: noindex,nofollow
x-served-by: tunnels-prod-rel-use-v3-cluster
```

Responses

Code	Description	Links
200	Successful Response	No links

test_ai_manual.md - MoodGarden
MoodGarden AI - Swagger UI
SDD pt. 2 - Google Docs
Mail - Bobby Barsh - Outlook
https://reimagined-pancake-7vgj696qqwx2r765-8000.app.github.dev/docs#/default/ai_analyze_ai_analyze_post
Chat
Download
56°F
Mostly sunny
Search
4:43 PM
2/12/2026

Status: PASS

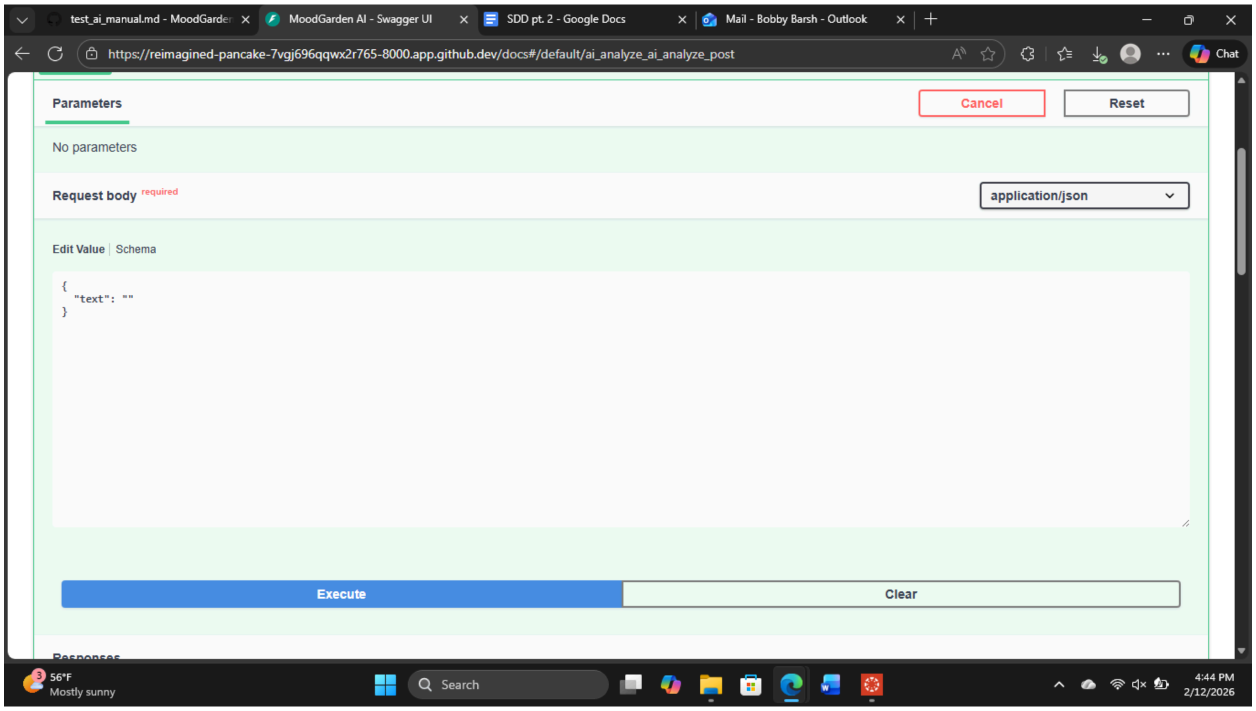
Test Case 3 – Empty Input

Input:
""

Expected:

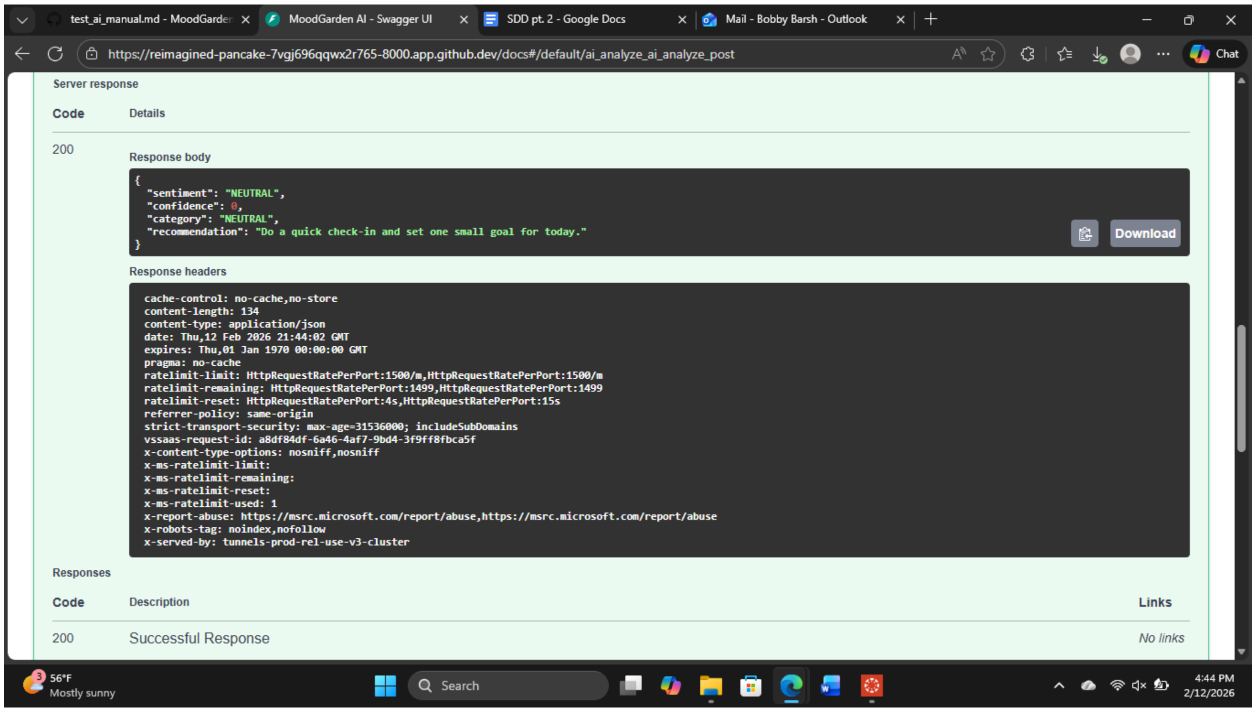
- sentiment = NEUTRAL
- category = NEUTRAL
- fallback recommendation

Actual Result:


test_ai_manual.md - MoodGarden
MoodGarden AI - Swagger UI
SDD pt. 2 - Google Docs
Mail - Bobby Barsh - Outlook
https://reimagined-pancake-7vgj696qqwx2r765-8000.app.github.dev/docs#/default/ai_analyze_ai_analyze_post
Parameters
Cancel
Reset
Request body required
application/json
Edit Value | Schema
Execute
Clear
Responses
56°F Mostly sunny
Search
4:44 PM 2/12/2026

No parameters

```
{
  "text": ""
}
```


test_ai_manual.md - MoodGarden
MoodGarden AI - Swagger UI
SDD pt. 2 - Google Docs
Mail - Bobby Barsh - Outlook
https://reimagined-pancake-7vgj696qqwx2r765-8000.app.github.dev/docs#/default/ai_analyze_ai_analyze_post
Server response
Code
Details
Response body
Download
Response headers
Responses
Code
Description
Links
56°F Mostly sunny
Search
4:44 PM 2/12/2026

200

```
{
  "sentiment": "NEUTRAL",
  "confidence": 0,
  "category": "NEUTRAL",
  "recommendation": "Do a quick check-in and set one small goal for today."
}
```

```
cache-control: no-cache,no-store
content-length: 134
content-type: application/json
date: Thu,12 Feb 2026 21:44:02 GMT
expires: Thu,01 Jan 1970 00:00:00 GMT
pragma: no-cache
ratelimit-limit: HttpRequestRatePerPort:1500/m,HttpRequestRatePerPort:1500/m
ratelimit-remaining: HttpRequestRatePerPort:1499,HttpRequestRatePerPort:1499
ratelimit-reset: HttpRequestRatePerPort:4s,HttpRequestRatePerPort:15s
referrer-policy: same-origin
strict-transport-security: max-age=31536000; includeSubDomains
vssaas-request-id: a8df84df-6a46-4af7-9bd4-3f9ff8fbca5f
x-content-type-options: nosniff,nosniff
x-ms-ratelimit-limit:
x-ms-ratelimit-remaining:
x-ms-ratelimit-reset:
x-ms-ratelimit-used: 1
x-report-abuse: https://msrc.microsoft.com/report/abuse,https://msrc.microsoft.com/report/abuse
x-robots-tag: noindex,nofollow
x-served-by: tunnels-prod-rel-use-v3-cluster
```

200 Successful Response *No links*

Status: PASS

Sequence: Register and Login

Register:

1. User sends registerUser request.
2. API checks uniqueness.
3. API hashes password.
4. API stores user.
5. API returns user_id.

Login:

1. User sends loginUser request.
2. API retrieves user by email.
3. API verifies password.
4. API returns auth token.

Sequence: View Feed

1. User requests feed.
2. API retrieves posts for user.
3. API retrieves analysis for each post.
4. API retrieves recommendations for each analysis.
5. API returns full feed.

6. Test Cases

Unit Test Cases – Black Box

Registration Tests:

- Valid registration → success.
- Duplicate email → error.
- Weak password → validation error.

Post Creation Tests:

- Valid mood_text → post created.
- Empty mood_text → error.

Comment Tests:

- Valid comment → success.
- Invalid post_id → error.

Unit Test Cases – White Box

registerUser():

- Path where username and email are unique → insert executed.
- Path where email exists → error returned.
- Path where username exists → error returned.

analyzePost():

- Post exists → analysis and recommendations created.
- Post missing → error path.
- AI returns low confidence → default sentiment branch.

6. Integration Test Case

1. Black Box Unit Test Cases (Table Format)

Black Box Unit Tests – Sentiment & Recommendation

Test ID	**Input**	**Expected Output**	**Actual Result**	**Status**
BB-01	“I feel great today”	Positive, Maintain	(see screenshot)	PASS
BB-02	“I am stressed about exams”	Negative, Calm	(see screenshot)	PASS
BB-03	“”	Neutral fallback	(see screenshot)	PASS

2. White Box Unit Test Cases

White Box Unit Tests – Recommendation Logic

Test ID	**Condition Tested**	**Expected**
WB-01	confidence >= 0.60 and POSITIVE	category = MAINTAIN
WB-02	confidence >= 0.60 and NEGATIVE	category = CALM
WB-03	confidence < 0.60	category = NEUTRAL

3. Integration Test Case

This proves full flow works.

Integration Test Case – End-to-End AI Flow

Test Scenario:
User submits mood → API → Sentiment model → Recommendation engine → Response returned.

Evidence:
Swagger execution screenshots.

Status: PASS

Sequence: Register and Login

Register:

6. User sends registerUser request.
7. API checks uniqueness.
8. API hashes password.
9. API stores user.
10. API returns user_id.

Login:

5. User sends loginUser request.

6. API retrieves user by email.
7. API verifies password.
8. API returns auth token.

Sequence: View Feed

6. User requests feed.
7. API retrieves posts for user.
8. API retrieves analysis for each post.
9. API retrieves recommendations for each analysis.
10. API returns full feed.