Journal - Part 2

1. Contribution of each member

For M6: Amod Ghimire (15 hours) - Updated documentation, Project Presentation slides 3 & 6, Addressed issues from M5 grading, performed demo screen recording, Completed project video #2. Kevin Li (10 hours) - Updated documentation, Project Presentation slides 5 & 8, Completed project video #3. Nyi Nyi Oo (10 hours) - Updated documentation, Project Presentation slides 4 & 9, Fixed chatbot, Completed project video #4. Christine Jiang (20 hours) - Updated documentation, Project Presentation slides 5 & 8, Addressed issues from M5 grading, Completed project video #5, Edit video #1.

2. Github Repository Info

Link to your GitHub repository: CPEN 321 Journal Project Commit SHA of your final released version: (on the main branch) `06ce5bc2f9ba0e14e8d8298d3f9259bd619abcb2`

3. Location of front-end and back-end tests

Type of Tests	Location
General Backend mocked tests	`/Backend/tests/mocked`
General Backend unmocked tests	`/Backend/rasa_api/__tests__`
General RASA chatbot tests	`/Backend/tests/unmocked`
Performance (Response Time)	`/Backend/tests/*` (All tests for api endpoints)
Encryption of Entries	`/Backend/tests/unmocked/journal.unmocked.test.ts#L266`
Usability (Frontend)	`Frontend/app/src/androidTest/java/com/example/cpen321project/Nonfunctional_clicks_test.kt`
General Frontend tests	`Frontend/app/src/androidTest/java/com/example/cpen321project`

4. Information about Physical Device tested

Manufacturer: Samsung Model: S10

5. Information about Back-end Server

Public IP of Back-end server: `ec2-35-183-201-213.ca-central-1.compute.amazonaws.com` Domain name: `https://cpen321project-journal.duckdns.org` Rasa Action Server (NodeJSWrapper): `https://54.234.28.190:3001/api/chat` RASA Action Server (EC2): `https://ec2-54-234-28-190.compute-1.amazonaws.com:5055/webhook`

6. Neccessary User Accounts to run our App

To run our app, the following accounts and associated credentials or tokens are required:

- Firebase Account:
    - Used for authentication and managing user data (e.g., profile info, FCM tokens).
    - Required: Firebase Admin SDK JSON file (cpen321project-c324e-firebase-adminsdk.json).
- Stripe Account:
    - Used for handling payment processing when users upgrade to the Premium account.
    - Required: Stripe Secret Key and Publishable Key (cpen321project-stripe-secret.txt).
- Google Account:
    - Used for user login via OAuth.
    - Required: Google OAuth credentials including Web Client ID, Google Numeric ID. An updated google token is only required if you are testing this app.
- OpenAI Account:
    - Used for processing journal content through the GPT-based AI therapy bot.
    - Required: Valid OpenAI API Key (OPEN_API_KEY).
- Server Secret:
    - Required for encrypting FCM tokens and ensuring secure operations on the backend.
    - File: severSecret.txt (must be requested from the team).

7. 'Complex' Use Cases

Our core complexity is implemented in the weekly trend analysis feature within the sentiment analytics system:

Use Case: Display Sentiment Trends in Analytics

- How It Works:
    - When the user accesses the analytics dashboard, we fetch their activity and emotion logs stored over the past week.
    - These logs include weights and metadata produced via the OpenAI LLM and saved during journal entry submission.
    - The analysis pipeline:
        - Parses emotion and activity scores from the last 7 days.
        - Computes trends by evaluating gradients (rise or fall) in data over time.
        - Applies filtering logic to ignore minor fluctuations (defined dynamically based on average activity frequency).
        - Matches emotion and activity trends chronologically, using a sliding time window to infer correlations.
        - Categorizes these as trend patterns like ++, --, +-, -+.
- Why This is Complex:
    - Implementing meaningful gradient detection requires nuanced tuning of thresholds.
    - Cross-matching multiple trends in sequence with noise filtering and categorization is non-trivial.
    - We had to build and maintain a data structure that dynamically classifies and tracks trend types and outputs summaries.
    - The system balances performance with insightfulness, which required iterations of optimization and testing.

8. Humblebrag

We're especially proud of our end-to-end encrypted journaling system and secure access control, both of which required close coordination between the frontend, backend, and database layers:

- AES Encryption of Journal Entries Each journal entry is encrypted before being stored in MongoDB. We implemented AES-based encryption and ensured that no plain text is saved. Entries are decrypted only after successful user authentication. This protects sensitive mental health data even if the database were compromised.
- Strict Google-Based Auth Enforcement All data operations require Google token validation. This adds a strong layer of access control, letting users manage their data securely without traditional password-based login.
- FCM Token Encryption Users' FCM tokens (for notifications) are encrypted using a server-stored secret before saving to the database. This protects against misuse of tokens and safeguards users from spam notification attacks.
- Being able to highlight the date in the main view: This is able to let the user know the dates, they have completed and are yet to complete through the highlight. This is also good because the user can always know if they successfully saved the journal.

While we acknowledge the limitation in our salt design (use of a static serverSecret), we've laid a strong foundation for integrating per-user dynamic salts or key derivation enhancements in future work.

9. Limitations of Our Project

A list of limitations of your project.

While we're proud of our app, it has a few limitations:

- No Offline Support
  - The app requires an internet connection for nearly all actions: login, journaling, analytics, etc.
- Image-Only Media Support:
  - Users can only attach static images (JPG, PNG, JPEG) to their entries. Support for audio or video journaling is not yet implemented.
- No Undo for Deleted Entries:
  - Once a journal entry is deleted, recovery is not possible.
- Sentiment Analysis Sensitivity
  - The accuracy of our emotional trend analysis depends on how descriptive the user's text is. Sparse entries may lead to weak insights.
- Token Expiry for Tests
  - Google tokens used for testing expire quickly (~1–2 hours), making automation and CI/CD pipelines harder to maintain.
- Single Language
  - The app is primarily designed with English prompts and UI. the underlying NLP (via OpenAI) is capable of handling multiple languages, the app's reflection prompts from the chatbot and the interface does not support languages other than English.

10. Reflections on Generative AI Technologies

1. Which Generative AI technologies did you use in the scope of this course?
   - Throughout the course, our team primarily used ChatGPT as our main generative AI assistant, and occasionally explored DeepSeek for more technical code-generation support. ChatGPT was used extensively for brainstorming, refining documentation, debugging, and assisting with test writing, while DeepSeek was used in specific scenarios where we wanted code output with greater technical focus or alternatives. While most of the team gravitated toward ChatGPT for its general flexibility and natural language support, DeepSeek was tested as a complementary tool for its code-completion strengths.
2. For which phases of the overall Software Engineering process and which concrete tasks in these phases was Generative AI the most helpful and why? Discuss phases such as requirements elicitation and documentation, design, implementation, code review, testing, and project refinement (for the final release).
   - Generative AI was especially helpful during the requirements elicitation, design, and documentation phases. It played a key role in helping us clarify functional and non-functional requirements, brainstorm and refine user flows, and structure

technical documents efficiently. In the design phase, AI assisted us in rapidly drafting function headers, data structure outlines, and system architecture ideas. This accelerated alignment across our team by providing consistent baselines that we could refine collaboratively.

- During documentation tasks — such as writing milestone reports, user-facing instructions, and API descriptions — AI tools were extremely helpful for polishing tone, correcting grammar, and generating content quickly. We also found AI particularly helpful during the testing phase, especially for generating test skeletons (e.g., Espresso tests, mocking interfaces in Jest) and suggesting edge cases. In these stages, where repetitive patterns and boilerplate dominate, AI tools saved time and effort, and helped us think more clearly about what we were doing — especially when we already understood the context.

3. For which phases of the overall Software Engineering process and which concrete tasks in these phases was Generative AI the least helpful and why? Discuss phases such as requirements elicitation and documentation, design, implementation, code review, testing, and project refinement (for the final release).

- Generative AI was least helpful during the implementation and project refinement phases, especially when dealing with project-specific logic or frameworks (like Android, Firebase, or Stripe). While AI could generate code snippets, these were often syntactically incorrect, outdated, or incompatible with our existing setup. For example, AI-generated test code would frequently use deprecated methods or incorrect libraries, leading to more time spent debugging or rewriting than if we had written the code ourselves from scratch.

- Additionally, AI struggled to understand context-specific constraints, such as dependencies across files, project structure, or user flow logic. When we asked for help with mid-to-late implementation tasks — like integrating payment APIs or resolving framework-specific errors — the answers often hallucinated methods, assumed wrong structures, or lacked depth. Without strong prior knowledge, the AI outputs were more misleading than helpful. We learned to use AI effectively only when we already knew what we wanted and needed help getting unstuck.

4. Is there anything else you would like to share about AI's impact on your software engineering tasks?

- Overall, AI was a valuable companion . It was most effective when used as a support tool for brainstorming, accelerating repetitive tasks, and cleaning up documentation and being available 24/7. However, we quickly learned that blind reliance led to frustration, especially when AI produced overly confident but incorrect answers. The hallucinations and lack of awareness about project-specific context meant we often had to double-check everything it produced.