**Review Report Sprint 1**

**26/05/2025 – 02/06/2025**

Group ID: **05**

Project Name: **SoulNote**

Prepared by: **Huỳnh Văn Sinh**

Team members:

23127262- **Lý Quốc Thạnh** *Team Leader – Implementor – Business Analyst*

23127109- **Nguyễn Lê Quang** *Implementor – Business Analyst – Tester*

23127109- **Huỳnh Văn Sinh** *Implementor – Designer – Tester*

23127485- **Phạm Quang Thịnh** *Implementor – Business Analyst – Designer*

23127515- **Nguyễn Tấn Văn** *Implementor – Designer – Tester*

1. **What went well**

* Team members proactively shared their progress and supported one another.
* The Use-case Model and detailed specifications were completed on time.
* The sprint helped improve everyone’s understanding of the system flow and SoulNote’s structure.
* Task management tools (Google Sheet, Zalo, GitHub) were used more effectively than in the previous sprint.

1. **What went wrong**

* Some members were not familiar with how to describe use-cases in detail, leading to repeated revisions.
* Initial task distribution was unbalanced, causing workload issues for certain members.

1. **Problems and causes**

* **Problem:** Miscommunication led to incorrect task execution.  
  **Cause:** Lack of proper meeting notes and confirmation after discussions.
* **Problem:** Delays in some documentations.  
  **Cause:** Some members were not familiar with the formatting templates and had to learn and revise several times.

1. **What Can Be Done Differently**

* Clearly define task objectives and expected outcomes, and provide template references at the start.
* Conduct short daily check-ins (via group chat) to share individual progress.
* Allocate tasks more evenly to avoid overloading certain members.

1. **Lessons Learned**

* Each team member must be more proactive in asking, clarifying, and confirming tasks.
* Agreeing on common formats early helps reduce wasted time later.
* Completing use-case specifications early supports smoother development in UI, database, and backend phases.

**Review Report Sprint 2**

**02/06/2025 – 15/06/2025**

Group ID: **05**

Project Name: **SoulNote**

Prepared by: **Nguyễn Lê Quang**

Team members:

23127262- **Lý Quốc Thạnh** *Team Leader – Implementor – Business Analyst*

23127109- **Nguyễn Lê Quang** *Implementor – Business Analyst – Tester*

23127109- **Huỳnh Văn Sinh** *Implementor – Designer – Tester*

23127485- **Phạm Quang Thịnh** *Implementor – Business Analyst – Designer*

23127515- **Nguyễn Tấn Văn** *Implementor – Designer – Tester*

1. **What went well**

* Revised Vision and Project Plan documents were completed with highlighted updates and revision history as required.
* Sprint 2 backlog was clearly planned and evenly assigned to all members with estimated effort.
* Use-case diagrams were effectively designed using draw.io, and aligned closely with the functional requirements.
* Team collaboration improved, with daily check-ins being implemented successfully.
* Members showed better understanding and fluency in writing use-case specifications based on the feedback from Sprint 1.

1. **What went wrong**

* Some estimated efforts in the Sprint 2 backlog were not realistic and had to be adjusted mid-sprint.
* Diagram formatting inconsistencies led to rework in the use-case specification document.
* A few members struggled with breaking down complex flows into alternate flows properly.
* Some proposed feature ideas were interesting but lacked clarity on how to implement them during this sprint.

1. **Problems and causes**

* Problem: Task estimation inaccuracies.

Cause: Members lacked past references to estimate technical tasks precisely.

* Problem: Use-case diagram formatting had to be redone.

Cause: Team did not align on a unified design standard before implementation.

* Problem: Some use-case specifications lacked clarity in alternate flows.

Cause: Limited experience with writing complex conditional logic.

* Problem: Some proposed features could not be implemented immediately.

Cause: Technical requirements were not clearly defined, and the team lacked sufficient knowledge to implement them within a short sprint timeframe.

1. **What Can Be Done Differently**

* Review and discuss estimation strategies in advance to improve accuracy.
* Set clear visual design guidelines for diagrams at the beginning of the sprint.
* Conduct peer reviews for all use-case specifications before submission.
* Maintain a shared example library for templates and completed use-cases.

1. **Lessons Learned**

* Having a common structure and formatting agreement early saves time later.
* Estimation becomes easier with more shared knowledge and task history.
* Peer review is a powerful tool to detect issues early in documentation.
* Visual tools and written specs should be reviewed together to ensure consistency.

**Review Report Sprint 3**

**16/06/2025 – 29/06/2025**

Group ID: **05**

Project Name: **SoulNote**

Prepared by: **Nguyễn Lê Quang**

Team members:

23127262- **Lý Quốc Thạnh** *Team Leader – Implementor – Business Analyst*

23127109- **Nguyễn Lê Quang** *Implementor – Business Analyst – Tester*

23127109- **Huỳnh Văn Sinh** *Implementor – Designer – Tester*

23127485- **Phạm Quang Thịnh** *Implementor – Business Analyst – Designer*

23127515- **Nguyễn Tấn Văn** *Implementor – Designer – Tester*

1. **What went well**

* Use-case specification was revised and extended based on TA’s feedback, including new use-cases for recently clarified requirements.
* The team successfully drafted the software architecture document, clearly describing key components and their organization using the MVC model.
* Class diagrams were created in alignment with the architecture, showing main classes, attributes, operations, and relationships.
* The database design was completed with an ER diagram that accurately reflects current data requirements.
* Collaboration continued to improve, and task distribution was more balanced compared to previous sprints.
* Revision history was properly maintained across all submitted documents.

1. **What went wrong**

* Some components in the architecture document lacked technical depth in the first draft and required major revisions.
* Misalignment was observed between some class diagrams and the actual system design discussed in meetings.
* Designing the ER model took longer than expected due to changing data requirements and uncertainty over relationships between entities.
* Some team members had limited familiarity with architectural patterns, causing delays in document writing.

1. **Problems and causes**

* Problem: Architectural components were initially too generic or vague.

Cause: Lack of technical discussion early in the sprint to agree on key design decisions.

* Problem: Class diagrams and backend structure did not match perfectly.

Cause: Not all updates during implementation were reflected in the documentation immediately.

* Problem: Slow progress in ER diagram design.

Cause: Some entities and relationships were only clarified mid-sprint.

* Problem: Difficulty applying MVC concepts.

Cause: Team had limited experience with architectural styles, especially mapping to real implementation.

1. **What Can Be Done Differently**

* Schedule dedicated design sessions at the start of the sprint to clarify architecture and class responsibilities.
* Maintain real-time updates to documentation as code and design evolve.
* Allocate more time and research effort into understanding architectural patterns in advance.
* Use diagrams collaboratively (e.g., shared draw.io) to reduce miscommunication.

1. **Lessons Learned**

* Early alignment on technical design significantly improves document quality and saves time.
* Diagrams and documentation should be developed in parallel with coding to ensure consistency.
* Even partial familiarity with patterns like MVC can be leveraged effectively with clear team communication.
* Logging revision history and highlighting changes improves transparency and TA feedback processing.
* Several use-case names were too long and lacked clarity; naming conventions need to be more concise.
* Relationships between use-cases (e.g., <<include>>, <<extend>>) were not analyzed carefully, leading to unclear grouping of actions.
* Some main and alternative flows in use-case specifications were too vague, lacking detailed user-system interactions and system responses.
* Redundancies existed in use-cases such as 'Browse memory' and 'Browse memories using multiple filters' or 'Delete memory' and 'Delete emotion statistic'.

**Review Report Sprint 4**

**27/06/2025 – 03/08/2025**

Group ID: **05**

Project Name: **SoulNote**

Prepared by: **Nguyễn Lê Quang**

Team members:

23127262- **Lý Quốc Thạnh** *Team Leader – Implementor – Business Analyst*

23127109- **Nguyễn Lê Quang** *Implementor – Business Analyst – Tester*

23127109- **Huỳnh Văn Sinh** *Implementor – Designer – Tester*

23127485- **Phạm Quang Thịnh** *Implementor – Business Analyst – Designer*

23127515- **Nguyễn Tấn Văn** *Implementor – Designer – Tester*

1. **What went well**

* A working version of the software was successfully deployed, implementing two main use-cases: "Create Memories” and “Report Emotion
* The test plan was standardized, including test environment descriptions, tools used, and clearly assigned human resources.
* At least 15 test cases were written for two use-cases: Create Memories and View Emotion Report.
* Test cases were executed on multiple environments (Chrome, Firefox, Android devices, macOS), and specific bugs were recorded where applicable.
* The user interface (UI) was revised and finalized based on feedback from Sprint 2, resulting in a consistent and approved design.
* Folder structure for source code and the deployment diagram were completed and properly described in the updated SAD.
* Revision history and feedback from previous sprints were well-documented and preserved

1. **What went wrong**

* Some components in the architecture document lacked technical depth in the first draft and required major revisions.
* Misalignment was observed between some class diagrams and the actual system design discussed in meetings.
* Designing the ER model took longer than expected due to changing data requirements and uncertainty over relationships between entities.
* Some team members had limited familiarity with architectural patterns, causing delays in document writing.

1. **Problems and causes**

* Problem: Architectural components were initially too generic or vague.

Cause: Lack of technical discussion early in the sprint to agree on key design decisions.

* Problem: Class diagrams and backend structure did not match perfectly.

Cause: Not all updates during implementation were reflected in the documentation immediately.

* Problem: Slow progress in ER diagram design.

Cause: Some entities and relationships were only clarified mid-sprint.

* Problem: Difficulty applying MVC concepts.

Cause: Team had limited experience with architectural styles, especially mapping to real implementation.

1. **What Can Be Done Differently**

* Schedule dedicated design sessions at the start of the sprint to clarify architecture and class responsibilities.
* Maintain real-time updates to documentation as code and design evolve.
* Allocate more time and research effort into understanding architectural patterns in advance.
* Use diagrams collaboratively (e.g., shared draw.io) to reduce miscommunication.

1. **Lessons Learned**

* Early alignment on technical design significantly improves document quality and saves time.
* Diagrams and documentation should be developed in parallel with coding to ensure consistency.
* Even partial familiarity with patterns like MVC can be leveraged effectively with clear team communication.
* Logging revision history and highlighting changes improves transparency and TA feedback processing.
* Several use-case names were too long and lacked clarity; naming conventions need to be more concise.
* Relationships between use-cases (e.g., <<include>>, <<extend>>) were not analyzed carefully, leading to unclear grouping of actions.
* Some main and alternative flows in use-case specifications were too vague, lacking detailed user-system interactions and system responses.
* Redundancies existed in use-cases such as 'Browse memory' and 'Browse memories using multiple filters' or 'Delete memory' and 'Delete emotion statistic'.