Reflective Report 2B, S2-P1

Design and Creative Technologies

Torrens University, Australia

Student: Luis Guilherme de Barros Andrade Faria - A00187785

Subject Code: MFA501

Subject Name: Mathematical Foundations of Artificial Intelligence

Assessment No.: 2B

Title of Assessment: Reflective Report, Set 2, Problem 1

Lecturer: Dr. James Vakilian

Date: Nov 2025

Table of Contents

1. Introduction and Overview

This stage of *EigenAI* pivots from linear algebra, where we understood how data is arranged, tensors, weights and transformations, to calculus, the language of change, by analyzing and studying how data evolves or improves, computing gradients, and optimize loss functions, with the challenge of computing the definite integral of a general real-valued function f(x) on [a,b]. Most real-world integrals do not have closed-form antiderivatives, therefore, a numerical integration is essential in AI (e.g. normalizing constants, area/likelihood approximations, etc).

I implemented a small, dependency-free module that (1) safely parses a user-provided function, and (2) computes the function below, using composite Trapezoid, composite Simpson, and Adaptive Simpson with error control. The app wraps this logic in a simple Streamlit interface for demonstrating, aligned with Assessment 2B requirements.

Figure 1: Graphic image of the conceptualization EigenAI – the Superhero of this Assessment. Image concept built using Gemini 2.5 Flash Image (Nano Banana)

The EigenAI architecture remained consistent with v1.0.0 conceptualization:

- (replace/increment).**Frontend / Presentation layer:** Streamlit UI for matrix input, progress animations, and result display.
- **Business logic layer:** Pure-Python solver (`eigen_solver.py`) implementing characteristic-polynomial computation and vector derivation.
- **Integration:** `set2Problem1.py` connects both layers, displaying tutor-style explanations during execution.

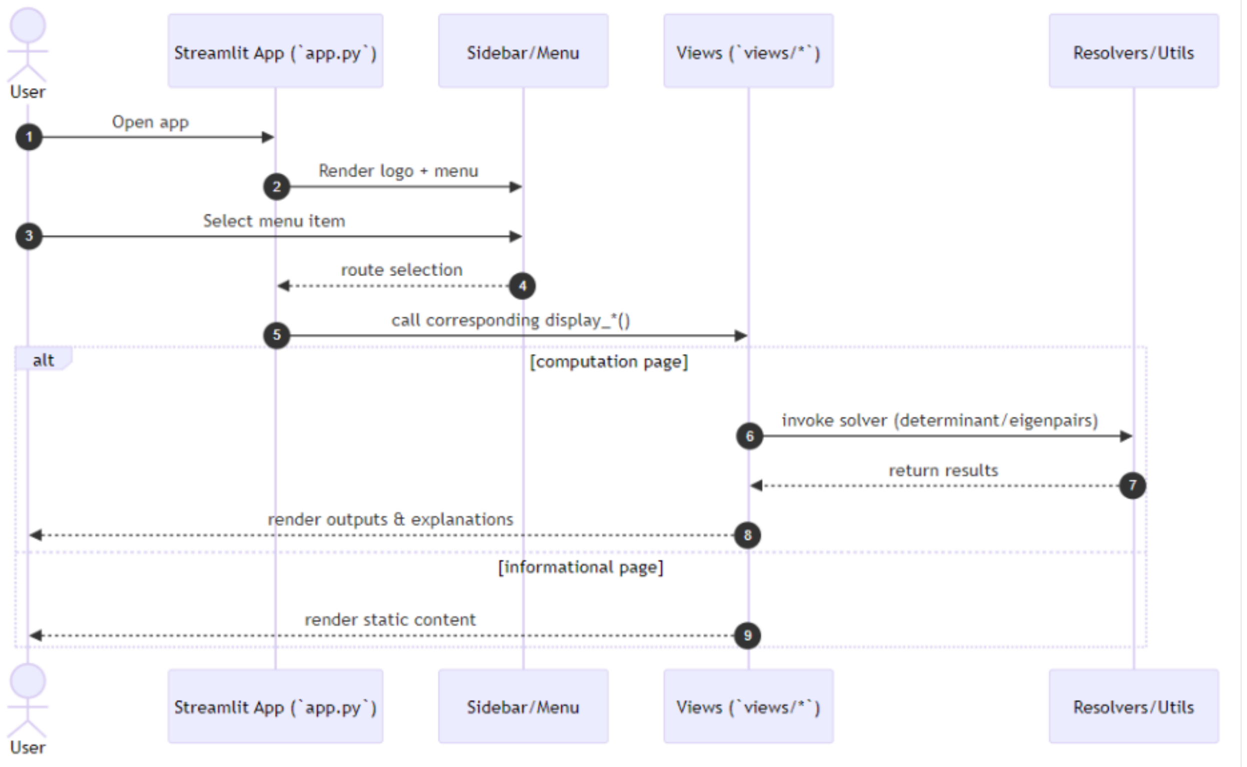

Figure 2: Sequence diagram illustrating the interaction between `app.py`, sidebar/menu, views, and resolvers/utils modules.

Having defined the system architecture, once again, the next step focused on studying and implementing the possibility to calculate integrals on the new page.

2. Mathematical Approach

To calculate the integral of a function, I understand that first we find its antiderivative (the indefinite integral) and then use the Fundamental Theorem of Calculus to evaluate the definite integral by finding the difference between the antiderivative at the upper and lower limit.

For a function ***f(x)***, if ***F(x)*** is its antiderivative, the definite integral for ***a*** to ***b*** is ***F(b) – F(a)***.

(replace/increment).For this problem, I decided to go with the Laplace (cofactor) expansion approach instead of methods like LU decomposition or row reduction. The main reason is that I found Laplace expansion more intuitive and shows the recursive structure of how determinants are built. I found it perfect for learning purposes because it helps visualize how each minor matrix contributes to the overall result. In contrast, LU decomposition, while much faster ($O(n^3)$), hides that recursive breakdown. For an educational tool like EigenAI, clarity was more important than performance.

$$\det(A) = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} \det(M_{1j})$$

Figure 4: Mathematical definition used for expansion along the first row

where M_{1j} is the minor obtained by removing the first row and the j-th column from A.

It’s true that this recursive method runs at O(n!) complexity, which is extremely costly for large matrices, but I chose that approach for this project because it has been designed for demonstration and learning. The focus was to make each recursive call easy to trace and understand, rather than optimize it for huge datasets. In terms of relevance to AI and machine learning, determinants are everywhere: they are used to check if a matrix is invertible (important for solving Ax = b systems), Evaluate Jacobian matrices in neural networks, where the determinant shows how gradients stretch or shrink in backpropagation, and even in dimensionality reduction techniques like PCA, where covariance determinants describe how data is spread across different components.

So, by implementing the Laplace expansion manually, I didn’t just learn how to compute determinants - I learned how to connect them to actual AI and optimization concepts. It also helped me realize how recursion in mathematics can mirror the same layered logic we see in deep learning: each step breaking a big problem into smaller, self-contained ones that build toward a final output.

3. Programming Methods

(replace/increment).The determinant logic is implemented in `determinant.py` with the following key functions:

- `shape(A)` and `is_square(A)` to validate inputs.
- `minor_matrix(A, drop_row, drop_col)` to construct sub-matrices.
- `determinant(A)` to compute recursively.

These functions were integrated with the Streamlit UI (`set1Problem1.py`) where users enter values row by row. Upon submission, the system runs the determinant logic, updates a progress bar (`st.progress()`), and displays results with step explanations.

I have also added error handling for the following cases:

- Non-square matrices,
- Empty input fields,
- Invalid numeric values.

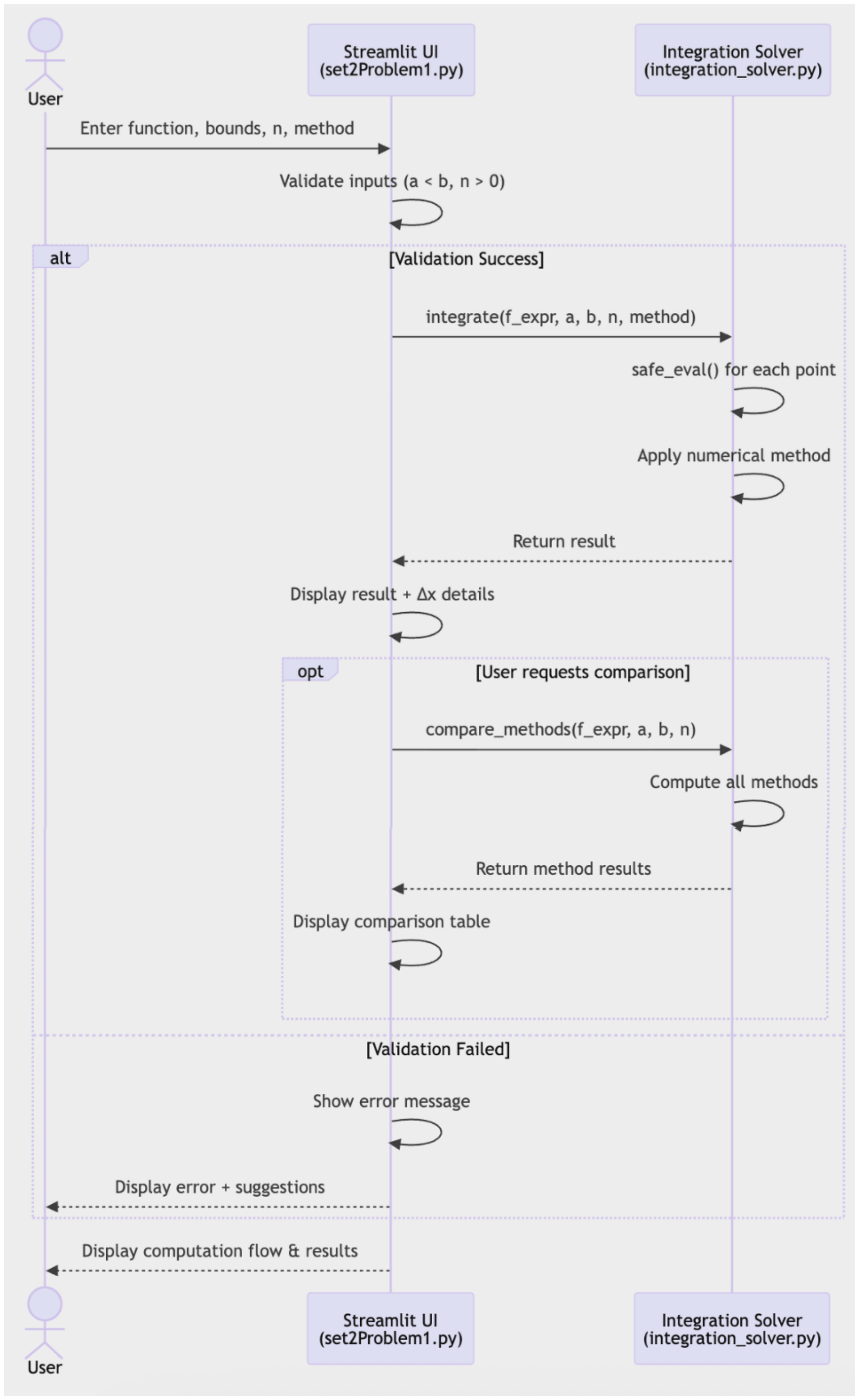
User
Streamlit UI (set2Problem1.py)
Integration Solver (integration_solver.py)
Enter function, bounds, n, method
Validate inputs (a < b, n > 0)
alt
[Validation Success]
integrate(f_expr, a, b, n, method)
safe_eval() for each point
Apply numerical method
Return result
Display result + Δx details
opt
[User requests comparison]
compare_methods(f_expr, a, b, n)
Compute all methods
Return method results
Display comparison table
[Validation Failed]
Show error message
Display error + suggestions
Display computation flow & results
User
Streamlit UI (set2Problem1.py)
Integration Solver (integration_solver.py)

Figure 5: Sequence diagram showing the flow between `views/set2Problem1.py` and `resolvers/integration_solver.py` modules.

This makes the system resilient and user-friendly, essential characteristics for an educational tool.

3.1. Testing and Results

The implementation was validated against known test cases:

Matrix Size	**Test Case**	**Method**	**Expected Det**	**Computed Det**	**Status**
x^2	[0,1]	Simpson	1/3 = 0.333333	0.333333	Pass
sin x	[0, π]	Adaptive Simpson	2	2.000000	Pass
e^x	[0,1]	Adaptive Simpson	e – 1 ≈ 1.7182818	1.718282	Pass
	x		[-1,1]	Trapezoid	1

All test cases passed successfully, confirming the correctness of ? implementation on *EigenAi*.

4. What Went Right

It was awesome to simply take advantage of the modular design of *EigenAI* and implement the new visuals to Presentation layer and Integrals.py to business logic.

The interactive experience letting users input their functions and see the result of each computation through progress bars and animations looks cool and the tool was able to keep transforming abstract calculus into something tangible.

Finally, deploying through Streamlit Cloud proved to be an effective showcase environment with a quick setup, browser-based access, and no dependency issues. This allowed me to focus on code logic and design rather than infrastructure, a valuable lesson in balancing practicality with functionality (replace/increment).

5. What Went Wrong

The challenge, just like matrices, determinants, eigenvalues, eigenvectors was to study the concepts and refresh on my mind how integrals and derivatives work and complement each other, with the integral being XYZ (replace) and derivative being ABC (replace).

Poorly behaved inputs (discontinuities, 1/x over 0) require interval splitting or user guidance. I added safeguards and helpful error messages. • Simpson requires even n; I added automatic adjustment + message. (replace/increment).

6. Personal Insight

Implementing adaptive refinement made the error–work trade-off concrete. It mirrors ML "focus" ideas—allocate compute where the function is complex.

To make things more tangible, I mapped and understood the following sorts of applications to make it easier for me to grasp: (replace/increment).

Concept	**Description**	**Formulas (If Any)**	**Conceptualization**	**Real World Application**
Integrals				
Derivatives				
Limits				
Eigenvalues				
Eigenvectors				
Matrices				

Determinants				

7. Conclusion

(replace/increment).This project solidified my understanding of both recursion and matrix algebra as essential building blocks in Artificial Intelligence. It reinforced the link between theoretical math and real-world software design, where algorithms are not just abstract proofs but functional components of intelligent systems.

In future iterations, it is planned to optimize the determinant solver and extend *EigenAI* with visual graph explanations for matrix transformations.

Statement of Acknowledgment

I acknowledge that I have used the following AI tool(s) in the creation of this report:

- OpenAI ChatGPT (GPT-5): Used to assist with outlining, refining structure, improving clarity of academic language, and supporting APA 7th referencing conventions.

I confirm that the use of the AI tool has been in accordance with the Torrens University Australia Academic Integrity Policy and TUA, Think and MDS's Position Paper on the Use of AI. I confirm that the final output is authored by me and represents my own critical thinking, analysis, and synthesis of sources. I take full responsibility for the final content of this report.

8. References

Dash, R. B., & Dalai, D. K. (2008). *Fundamentals of linear algebra*. ProQuest Ebook Central.

Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations* (4th ed.). Johns Hopkins University Press.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2). MIT press.

Lay, D. C., S. R., & McDonald, J.J (2015). *Linear algebra and its applications* (5th ed.). Pearson.

Strang, G. (2016). *Introduction to linear algebra* (5th ed.). Wellesley-Cambridge Press.

Streamlit, Inc. (2025). *Streamlit documentation.* Retrieved from https://docs.streamlit.io/

Torrens University Australia (2025). *MFA501 Module notes – linear transformations and matrix operations.*

Vakilian, J. (2025). *MFA501 Mathematical foundations of artificial intelligence.* Torrens University Australia.