ISSN: 2473-4901
v12 n6823

Auto-Grading UML Activity Diagrams With AI Tools A Controlled Pilot Demonstration

Stuart L. Wolthuis
Stuart.wolthuis@byuh.edu
Brigham Young University–Hawaii
St, Laie, HI 96762

Abstract

This effort evaluates and shares the possibility of using Generative AI (GenAI) as an auto-grader for Unified Modeling Language (UML) diagrams, specifically activity diagrams in a narrow pilot demonstration. Providing useful feedback to students is an important factor in quality teaching. Creating consistent, meaningful feedback to students is complicated by large class size and the experience of the grader. This work demonstrates the possibility of speeding up the feedback loop with the use of GenAI. Activity diagrams provide an abstract, functional view of a use case or business process. Providing students with actionable, consistent feedback is essential for a solid foundation in Object Oriented Programming (OOP) since functional models form the foundation to build structural, behavioral, DB, HCI, and Physical Architecture layer models. Findings demonstrate that GenAI is efficient and generally fast, it can also miss significant errors that should be included in feedback to students. Can we rely on GenAI to provide accurate, timely, and consistent auto-grading for UML activity diagrams?

Keywords: Auto-grade, AI, activity diagram, UML, assignment feedback, OOP

Auto-Grading UML Activity Diagrams With AI Tools
A Controlled Pilot Demonstration

Stuart L. Wolthuis

1. INTRODUCTION

Computer Science, Software Engineering, Information Systems and Information Technology students typically study Object Oriented Programming (OOP) using Java, Python, C++, or C# as reported by S3 CORP (2026). As a core class, systems engineering with OOP is also taught to provide future system analysts with the skills they need to document their analysis and design work (Dennis, et al., 2021).

A critical aspect of the learning process is timely and meaningful feedback so that students can hone their skills and understanding of important concepts. Timely grading of student assignments affords insight into improvement actions while they are still fresh in their minds. The author adopted this practice in 2008, expecting students to correct assignments (homework, projects, group projects) and turn them in again with the understanding that they could earn up to half the points they missed on their original artifact.

This work demonstrates and evaluates the use of GenAI as an auto-grader for an initial UML assignment in a systems engineering class. Students are taught the basic syntax for an activity diagram (process and data flow), then provided with feedback on their work regarding correct UML syntax and logic flow.

As enrollment has increased the time commitment to create and give meaningful and accurate feedback on assignments has become exhausting. With teaching assistants or graders, the scope of this responsibility is diminished but the guarantee of consistent evaluation and feedback is uncertain.

To solve both these problems, timely and consistent feedback with large class sizes, a microcosm of one scenario is addressed in this study. The trajectory and trust associated with AI tools have risen exponentially in the past 2 years (Kokol, 2024), (Ucar, et al., 2024). We are at an inflection point where AI tools have the capability to manage, determine, and predict with accuracy many of the systems used in industry, medicine, and finance. Given adequate prompts, should we expect AI tools to provide excellent results for timely and consistent feedback on assignments?

2. HISTORY OF GenAI AS AN AUTO-GRADER

According to researchers evaluating automated grading systems for STEM, another major capability provided by GenAI is the ability to scale, a university class currently offering 3 sections this semester with 6 teaching assistants could have no teaching assistants next semester; GenAI auto-grading would be a viable solution for this problem. Also, the ability to "evaluate multiple dimensions of student understanding" to include schematics (like activity diagrams) and logic represented in visual diagrams makes a strong case for the use of GenAI in auto-grading (Tan, et al., 2025).

Tan further points out that grading an entire exam was cut from 36 hours to 9.2 hours, a strong argument supporting the use of a GenAI auto-grader from an efficiency standpoint. The group also found that auto-grading systems achieved a 95% agreement with manual experts doing the same evaluation while reducing evaluation time over 90%.

Striewe and Goedicke (2014) developed an automated process to evaluate UML diagrams using trace generation. Once a key document was evaluated, a student submission would be compared using the sequence alignment. This dynamic checking allowed different but equally viable solutions compared to the answer key model.

Time consumption, subjectivity and bias, and scalability form three spokes that challenge traditional grading. GenAI, combined with ML for rubric based scoring creates a technical blueprint for a strong UML auto-grader (Deepshikha, 2025) that can address all three of these challenges. Other advantages shared in Deepshikha's research are the fact that GenAI auto-graders can now provide personalized feedback and allows educators to focus on teaching over manual grading. Furthermore, AI more reliably grades fully objective content compared to creative/subjective work such as activity diagrams given students can insert their own valid but varied design.

Educators often substitute a multiple-choice question midterm or final exam for a more insightful evaluation of student understanding simply because it is easier to grade and constraints do not allow the intense time commitment to evaluate essay or diagram answers. "AI-enabled systems can efficiently process massive volumes of student work" in a fraction of the time required by manual grading methods (Gnanaprakasam and Lourdusamy, 2024). These two researchers also note AI auto-graders are also impartial, consistent, and do not tire even after thousands of evaluations.

A study at the University of British Columbia evaluated 8 auto-graders and their efficacy across several disciplines. They specifically analyzed a UML auto-grader and noted their system used a similarity score between the answer key and a student's UML diagram. This group also uses AI to auto-grade code to evaluate whether a student follows the correct process taught in class (Niu, et al., 2025). These researchers, all computer scientists, acknowledge that advanced tools are in place to improve auto-grading and future auto-graders will become "increasingly adaptive to students' learning."

3. METHODOLOGY FOR THE STUDY

This study employs an e-commerce use case, a simple online ordering system for skateboards. This is the first UML model in the Systems Engineering class and serves as a platform to help students learn both UML syntax and correct implementation of logic. Decisions based on simple Boolean logic and guard conditions need to be understood and properly drawn.

Class sizes for Systems Engineering at this private university, with about 3,000 students, have increased in size, from 15 to 30. With this welcome influx of more students, new methods of grading need to be examined to achieve increased efficiencies (Fadhil, et al., 2025) and timely feedback.

The e-commerce use case reflects a very popular mode of transportation for students in this area where the climate is temperate all year round, making skateboards very popular.

A nine step process is given as the use case scenario, and students follow the instructor's directions shown on several large screens. The nine steps are as follows:

Business Process for North Shore Eco Ride (NSER)

e-commerce Skateboard

1. A user goes to "NorthShoreEcoRide.com"
2. User logs in or registers (inputs contact, shipping, & credit card info)
3. User selects a type of skateboard: Woody, Kick, Rail
4. User selects deck finish: NSER, Grip, Smooth, Custom
5. User selects truck type (the hardware that holds the wheels): Gullwing, Tensor, Turtle
6. Verify selections, redo 3, 4, or 5 if they want
7. User selects delivery type: USPS, FedEx / Expedite or Not
8. User selects payment option: PayPal, Visa, MC, Discover
9. User verifies total, completes "NorthShoreEcoRide" purchasing experience

For this study, the instructor provided AI with a correct version (see appendix) of the UML Activity Diagram, correct in both UML syntax and logic flow. Ten additional files were created with varying anomalies reflecting common mistakes made by students.

The ten additional files with common mistakes (and one correct file) are delineated as follows:

Key – correct solution
- 1 – missing frame
- 2 – missing initial node
- 3 – missing final node
- 4 – missing object (data flow) node
- 5 – missing decision nodes
- 6 – missing merge node

7 – correct, matches key
8 – logic error
9 – missing action
10 – missing text in decision node

ChatGPT 5.5 was given the following prompt:

The following homework assignment is used to teach students in systems engineering how to create a correct activity diagram using UML syntax: Business Process e-commerce Skateboard

1. A user goes to "NorthShoreEcoRide.com"
2. User logs in or registers (inputs contact, shipping, & credit card info)
3. User selects a type of skateboard: Woody, Kick, Rail
4. User selects deck finish: NSER, Grip, Smooth, Custom
5. User selects truck type (the hardware that holds the wheels): Gullwing, Tensor, Turtle
6. Verify selections, redo 3, 4, or 5 if they want 7. User selects delivery type: USPS, FedEx / Expedite or Not
8. User selects payment option: PayPal, Visa, MC, Discover
9. User verifies total, completes "NorthShoreEcoRide" purchasing experience The assignment is instructor directed and done with the students. Please correct and compare the key svg file to the 10 other SVG files in the following directory: AA-AutoGradeUML.AD

SVG files are ideal because they retain "crisp text and lines at any zoom level, allowing AI vision models to accurately parse text and structural connections without pixelation … as they are more accurate in modeling." (Diyawanna Group, 2026)

The 10 files were uploaded to AI tools as a zip file for processing. Loading the files individually did not work as a copy and paste of 10 files only pasted the first file.

For this study, the following AI tools were used: ChatGPT 5.5, Claude Sonnet 5, and Gemini 3.6 Flash.

4. RESULTS OF THE AI EVALUATION

ChatGPT 5.5 provided feedback for the 10 files in 1 minute 30 seconds. Each file was evaluated, here are the results:

"I checked the 10 SVGs in Archive.zip against the key file. No file is an exact match to the key, but the closest ones are AD.NSER.2.drawio.svg, AD.NSER.3.drawio.svg, and AD.NSER.7.drawio.svg. They keep the full same flow and title; the only noticeable mismatch is the wording Verify Choices versus the key's VerifyChoices."

Ironically, number 7 was correct, but a space was left out when it was converted to the SVG format.

"The key diagram itself is structurally sound for the assignment: it has the login/register branch, skateboard choice steps, the verification loop, delivery, payment, and final completion."

The specific details provided by the AI tool are in the appendices, here are the deficiencies in the evaluation:

The following observations summarize the AI tool's evaluation of the 10 UML Activity Diagram files. Reference to the file numbers delineated in section 3 is used (we will ignore the repeated error of the space missing in Verify Choices):

File 1, the AI tool correctly identified that the title was missing, but did not mention the frame, a required syntax for activity diagrams.

File 2, the AI tool missed the absence of the initial node.

File 3, the AI tool missed the absence of the final node.

File 4, the AI tool correctly identified the missing data flows (object flows)

File 5, the AI tool correctly found there was a missing decision node, and stated it interrupted the logic flow

File 6, the AI tool correctly found that the merge node was missing, also a logic flow problem

File 7, the AI tool found this file to be correct compared to the key file

File 8, the AI tool correctly identified the guard conditions were flipped (the YES / NO on the first decision node), a logic error

File 9, the AI tool correctly noted that an action step was missing

File 10, the AI tool correctly identified that "New User" was missing from the first decision node, but missed that "Verify Selections" was missing from the second decision node

Within this study, ChatGPT 5.5 correctly identified seven of the intentionally introduced defects (70%), establishing promising performance while also revealing limitations in detecting several fundamental UML syntax elements.

Specifically, it missed the following syntax items: In File 1 it missed the required frame, in File 2 and 3 it overlooked the initial and final nodes, respectively, and in File 10 it only found a problem with one guard condition in one decision node, but there was a second decision node that was also missing its guard condition.

The second AI tool exercised in this study was Claude Sonnet 5. This intelligent software correctly graded all ten UML activity diagrams. Furthermore, Sonnet 5 provided suggested teaching points to help students better understand the correct use and placement of guard conditions in the activity diagram.

Sonnet 5 also offered to create a grading rubric with points assigned for each assessment item. This AI tool took over 5 minutes to open, render, and evaluate the UML files. This AI tool found all the errors in the incorrect submission files compared to the key file and correctly determined that file 7 was correct.

The third and final AI tool used in this study was Gemini 3.6 Flash. The evaluation time completed in under a minute. Gemini also correctly graded the incorrect submitted files and correctly identified that file 7 was correct.

5. CONCLUSIONS

This study examined whether several contemporary LLMs, ChatGPT 5.5, Claude Sonnet 5, and Gemini 3.6 Flash, could generate timely and consistent feedback on a Systems Engineering class's UML Activity Diagram. This work was motivated by the real-world pressures facing contemporary computing education: growing class sizes and a parallel demand for rapid, high-quality feedback that supports student learning without imposing an unsustainable burden on instructional staff.

Findings indicate that GenAI holds a sizeable promise as an instructional support tool but needs improvement before it replaces human grading. ChatGPT 5.5 correctly identified seven of the ten diagram errors intentionally introduced into the evaluation set, generating its assessment in approximately ninety seconds. The model reliably found a range of structural and logical errors, including absent decision and merge nodes, missing object flows, incorrect guard conditions, and omitted actions. On the other hand, the Claude Sonnet 5 and Gemini 3.6 Flash both correctly detected all 9 defects and the 1 correct diagram. These results suggest that full reliance on GenAI is still in need of improvement but an excellent first pass evaluation tool for educators.

Taking all this into consideration, GenAI is positioned as a strong assistant to instructors, but is not ready to fully work autonomously as a grading system. Human oversight is still essential, AI-based tools can certainly improve the feedback delivery loop, provide excellent direction for unbiased evaluation, and guide instructional improvement. As AI models continue to mature and improve, they will increase their position as valuable partners in formative assessment.

6. FUTURE STUDIES

There are many opportunities for future studies.

First, future studies could evaluate AI performance using significantly larger datasets containing actual student submissions from multiple semesters providing a longitudinal study. Reliable measures from larger data sets of grading accuracy would identify additional error patterns commonly found in practice.

Second, research should examine if prompt engineering can improve grading performance by the AI tool. This study employed a single prompt and a reference solution. More detailed prompts that explicitly

instruct the AI to examine every expected UML syntax and logic flow would help produce a more accurate evaluation.

Third, future work should compare several AI models, including ChatGPT versions, Claude, CODEX, Gemini, and other open-source vision-language models, to establish whether differences exist in their capability to recognize semantic correctness and UML syntax.

Fourth, additional UML models should be evaluated. Activity diagrams represent only one type of functional model. Additional studies could be done for structural models (class, object), behavioral models (sequence, communication, behavioral state machine), deployment diagrams, database, and HCI models. This expanded study would provide a broader understanding of AI's capabilities for grading and providing timely and consistent feedback for systems analysis and software engineering education.

Fifth, additional future research should examine alignment with learning outcomes for specific courses and majors. Accuracy alone will not necessarily assess student learning, and the contribution AI can make to the learning process, student retention, and student confidence in the skill they attempt to learn. Side by side studies comparing instructor feedback and AI feedback would also provide useful insight.

Finally, from ChatGPT 5.5: "future studies should explore hybrid assessment methodologies that combine AI with rule-based validation or conventional UML parsers. Such systems may leverage the strengths of AI in providing natural-language explanations while using formal syntax checking to eliminate errors that current language models occasionally overlook. This combination has the potential to produce grading systems that are both highly accurate and pedagogically valuable."

7. REFERENCES

Deepshikha, D. (2025). A comprehensive review of AI-powered grading and tailored feedback in universities. *Discover Artificial Intelligence, 5*, 251. https://doi.org/10.1007/s44163-025-00517-0

Dennis, A., Wixom, B., & Tegarden, D. (2021). *Systems analysis and design: An object-oriented approach with UML*. John Wiley & Sons.

Diyawanna Group of Companies. (2026). *Home*. https://diyawanna.com/, Retrieved 14 Jul 2026

Fadhil, A.A., Albayati, A.H. (2025). Automatic evaluation of the student's activity diagrams in the educational process. Ingénierie des Systèmes d'Information, Vol. 30, No. 8, pp. 2149-2156. https://doi.org/10.18280/isi.300820

Gnanaprakasam, J., & Lourdusamy, R. (2024). The role of AI in automating grading: Enhancing feedback and efficiency. IntechOpen. https://doi.org/10.5772/intechopen.1005025

Kokol, P. (2024). The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of the Recent Research Literature. *Information*, *15*(6), 354. https://doi.org/10.3390/info15060354

Niu, J., Wong, J., Lake, C., Rahardjo, J., Zarkoob, H., Ola, O., ... & Wolfman, S. (2025, February). Expanding the Horizons of Autograding: Innovative Questions at UBC. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (pp. 868-874). https://doi.org/10.1145/3641554.3701892

S3CORP. (2025, December 8). *Top 10 object-oriented programming languages in 2026*. https://www.s3corp.com.vn/insights/top-oop-languages, Retrieved 14 Jul 2026

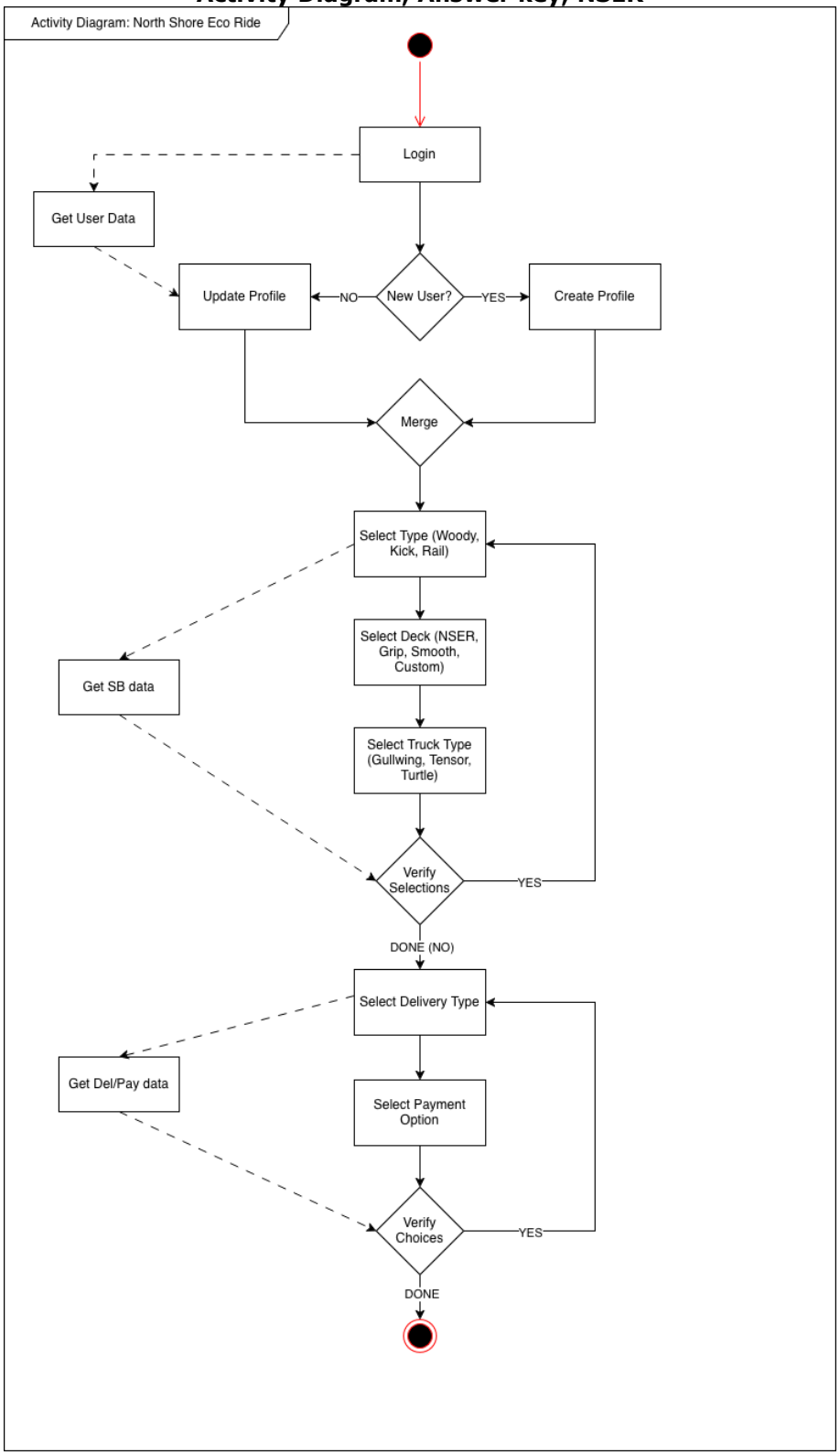
Striewe, M., & Goedicke, M. (2014). *Automated assessment of UML activity diagrams*. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14)* (p. 336). Association for Computing Machinery. https://doi.org/10.1145/2591708.2602657

Tan, L. Y., Hu, S., Yeo, D. J., & Cheong, K. H. (2025). A comprehensive review on automated grading systems in STEM using AI techniques. *Mathematics*, *13*(17), 2828. https://doi.org/10.3390/math13172828

Ucar, A., Karakose, M., & Kırımça, N. (2024). Artificial intelligence for predictive maintenance applications: Key components, trustworthiness, and future trends. Applied Sciences, 14(2), 898. https://doi.org/10.3390/app14020898

ISSN: 2473-4901
v12 n6823

APPENDIX A
Activity Diagram, Answer key, NSER



APPENDIX B
ChatGPT 5.5 Evaluation

File	Match level	What is different
AD.NSER.1.drawio.svg	Close	Same flow, but missing the title frame text; also uses Verify Choices instead of VerifyChoices.
AD.NSER.2.drawio.svg	Very close	Same flow; only the Verify Choices wording differs from the key.
AD.NSER.3.drawio.svg	Very close	Same as file 2.
AD.NSER.4.drawio.svg	Broken	Missing Get User Data, Merge, Get SB data, and Get Del/Pay data.
AD.NSER.5.drawio.svg	Broken	Missing the New User? decision, so login flows straight into profile creation/update.
AD.NSER.6.drawio.svg	Broken	Missing the Merge node, so the create/update branches never rejoin properly.
AD.NSER.7.drawio.svg	Very close	Same flow; only the Verify Choices wording differs from the key.
AD.NSER.8.drawio.svg	Close	Same flow, but the first YES / NO labels are flipped compared with the key.
AD.NSER.9.drawio.svg	Broken	Missing the Select Deck step.
AD.NSER.10.drawio.svg	Broken	Missing the New User? decision and several connecting edges, so the flow is incomplete.

APPENDIX C
ChatGPT 5.5 Evaluation, whether the UML flow is basically correct for the assignment

File	Grade	Correction checklist
AD.NSER.1.drawio.svg	Pass with minor edits	Add/restore the title styling from the key; change Verify Choices to VerifyChoices if exact match is required.
AD.NSER.2.drawio.svg	Pass with minor edits	Only the wording differs: Verify Choices vs. VerifyChoices.
AD.NSER.3.drawio.svg	Pass with minor edits	Only the wording differs: Verify Choices vs. VerifyChoices.
AD.NSER.4.drawio.svg	Fail	Restore the missing flow pieces: Get User Data, Merge, Get SB data, and Get Del/Pay data.
AD.NSER.5.drawio.svg	Fail	Restore the New User? decision branch so login does not go straight into profile creation/update.
AD.NSER.6.drawio.svg	Fail	Add the missing Merge node so the create/update branches rejoin correctly.
AD.NSER.7.drawio.svg	Pass with minor edits	Only the wording differs: Verify Choices vs. VerifyChoices.
AD.NSER.8.drawio.svg	Pass with minor edits	The flow is correct, but the YES / NO labels on the first decision are reversed compared with the key.
AD.NSER.9.drawio.svg	Fail	Add the missing Select Deck step.
AD.NSER.10.drawio.svg	Fail	Restore the missing New User? decision and the missing connecting edges.

APPENDIX D
Claude Sonnet 5, Evaluation

File	Verdict	Issue
AD.NSER.1	Minor	Missing only the title textbox ("Activity Diagram: North Shore Eco Ride"). Diagram logic itself is complete and correct.
AD.NSER.2	**Error**	Missing the **Initial node** (the filled black start circle). The activity has no defined starting point — a required UML activity-diagram element.
AD.NSER.3	**Error**	Missing the **Final node** (activity final). The "yes" branch of Verify Choices dead-ends with an arrow to nowhere instead of terminating the flow.
AD.NSER.4	**Major error**	Three whole action nodes are missing: *Get User Data*, *Get SB data*, *Get Del/Pay data* — along with all their edges. This breaks three separate paths: the parallel login→profile-update path, the flow into "Verify Selections," and the flow into "Verify Choices." Significant incompleteness.
AD.NSER.5	**Major error**	Missing the **"New User?" decision** diamond entirely. Login connects straight to Merge, and *Create Profile* / *Update Profile* become orphaned nodes with no incoming transition — they're drawn but unreachable. The core branching logic of the diagram is broken.
AD.NSER.6	**Major error**	Missing the **Merge node**. *Create Profile* and *Update Profile* both dead-end (arrows to nowhere), and *Select Type* has an incoming arrow from nowhere. The new-user/returning-user branches never rejoin the main flow.
AD.NSER.7	✅ Correct	Matches the reference model exactly — nodes, shapes, and every transition.
AD.NSER.8	Minor	Structurally identical to the key, but the **YES/NO labels next to the New User? decision are swapped** — as drawn, "NO" appears to point at Create Profile and "YES" at Update Profile, which is backwards logic.
AD.NSER.9	**Error**	Missing the **"Select Deck Finish" action** (step 4 of the spec: NSER/Grip/Smooth/Custom). Flow jumps straight from Select Type to Select Truck Type, skipping an entire required business step.
AD.NSER.10	Minor	Structurally correct, but the **"New User?" and "Verify Selections" decision diamonds are blank** — the guard condition/question text was deleted or never entered, so a reader can't tell what's being decided at those branch points.

Additional suggestions from Claude Sonnet 5: "A note that applies to all 10 (including the correct ones). None of the files put actual guard-condition text (e.g., [new user] / [existing user], [ok] / [redo]) directly on the decision's outgoing transition arrows — the "YES"/"NO"/"DONE" labels are floating text boxes placed nearby, not actually bound to any edge in the model. It works visually but isn't strictly correct UML activity-diagram syntax (guard conditions belong on the transition itself, in brackets). Worth a class-wide teaching point regardless of individual grades."

APPENDIX E
Gemini 3.6 Flash, Evaluation

Submission	Control Flow Correctness	UML Syntax (Nodes/Decisions)	Loop-Back Logic (Step 6)	Completeness (Steps 1–9)	Overall Alignment
AD.NSER.1	Correct	Proper Action/Initial/Final syntax	Correctly targets steps 3–5	Complete	Pass / Key Aligned
AD.NSER.2	Missing directional arrows	Mixed decision/action nodes	Missing guard labels on decision	Omitted step 4	Needs Revision
AD.NSER.3	Correct	Standard syntax	Loop goes back to Step 1 (incorrect)	Complete	Minor Correction Needed
AD.NSER.4	Correct	Standard syntax	Correct	Complete	Pass / Key Aligned
AD.NSER.5	Flow breaks at Step 6	Missing final decision node	Unclosed loop	Missing step 8	Needs Revision
AD.NSER.6	Correct	Standard syntax	Correct	Complete	Pass / Key Aligned
AD.NSER.7	Correct	Standard syntax	Correct	Complete	Pass / Key Aligned
AD.NSER.8	Fork/Join misuse	Non-standard activity shapes	Direct loop-back without decision node	Complete	Needs Revision
AD.NSER.9	Correct	Standard syntax	Correct	Complete	Pass / Key Aligned
AD.NSER.10	Correct	Standard syntax	Loop targets step 2 instead of 3	Complete	Minor Correction Needed