AI Tool Evaluation Form: QA and Testing (Week 11)

Student: Sena Nur Pekgöz

Role: QA and Testing (V&V)

Date: 10 December

Tools Evaluated: OpenAI (for BeePlan), Deepseek (for KidTask)

Part 1: Tool Analysis - OpenAI (BeePlan)

Project Context: Python-based GUI, Scheduling Algorithms, Constraint Satisfaction.

Quality Area	Rating (1-5)	Observation / Justification
Output Quality	**5**	OpenAI excelled at generating complex logical test cases. It correctly identified the "Lab vs. Theory" sequencing violation and provided an optimized $O(n)$ solution for conflict detection. The generated test data covered edge cases (e.g., Friday 13:20 exam blocks) accurately.
Usability	**4**	Very easy to interact with via natural language. However, pasting large chunks of Python code for context sometimes hit token limits or required re-prompting to focus on specific functions.
Output Trustworthiness	**4**	The suggested code fixes for the algorithm were logically sound. However, manual verification was still required for the JSON export function, as the first suggestion missed a specific formatting requirement.

Key Takeaway: OpenAI is highly effective for algorithmic debugging and logic verification in Python.

Part 2: Tool Analysis - Deepseek (KidTask)

Project Context: Java GUI (Swing/JavaFX), OOP, Data Persistence.

Quality Area	Rating (1-5)	Observation / Justification
Output Quality	**4**	Deepseek generated excellent boilerplate code for JUnit tests and GUI event listeners. It successfully detected the lack of input validation in the "Points" field. However, some GUI layout suggestions required manual tweaking to align with the panel requirements.
Usability	**5**	Deepseek (specifically the "Coder" model) integrated well with code contexts. It felt very responsive for Java syntax and quickly identified missing imports or syntax errors in the Swing components.
Output Trustworthiness	**3**	While syntax was correct, one logic fix for the "Level Update" feature initially caused an infinite loop in the update() method. This required careful human review before committing to the repository.

Key Takeaway: Deepseek is a strong coding assistant for Java syntax and boilerplate, but complex state-management logic requires strict human review.

Part 3: Comparative Evaluation (Task 3)

1. Comparison of Capability

- **Algorithm vs. Structure: OpenAI** demonstrated superior understanding of abstract constraints (BeePlan's scheduling rules), making it better for the "Algorithm" focus of Project 1. **Deepseek** was faster and more precise with strict syntax and GUI structure, fitting the "Object-Oriented" nature of Project 2 (KidTask).
- **Error Handling:** OpenAI provided detailed explanations of *why* an error occurred (educational value), whereas Deepseek focused on providing the *code fix* immediately (speed value).

2. Trustworthiness Verdict

- **Winner: OpenAI**. In the context of "Verification & Validation" (V&V), OpenAI provided fewer "hallucinated" code logic errors compared to Deepseek during this week's testing.

3. Final Recommendation

- Use **OpenAI** for logic-heavy debugging (algorithms, complex rules).
- Use **Deepseek** for rapid test case generation and syntax fixing (GUI, standard CRUD operations).