

Week 11 Output: Versions and Test Cases

Student: Sena Nur Pekgöz

Role: QA and Testing (V&V)

Date: 10 December

Part 1: BeePlan (Python) Testing

AI Tool Used: OpenAI (ChatGPT)

Focus: Algorithm verification, conflict detection, and input validation.

1.1 Test Cases Generated by OpenAI

The following test cases were generated to verify the "Scheduling Rules" defined in the BeePlan requirements.

Test ID	Requirement	Test Input	Expected Outcome	Status
BP-01	Friday Exam Block No courses between 13:20-15:10 on Fridays.	Input a course scheduled for Friday at 14:00.	System should flag a "Time Conflict" error or automatically move the slot.	Pass
BP-02	Lecturer Load Max 4 hours of theory per day per lecturer.	Assign "Dr. Smith" to 5 hours of theory courses on Monday.	System should return a validation error: "Instructor load exceeded (>4h)".	Pass

BP-03	Lab Sequence Lab sessions must follow theory hours.	Schedule "Physics Lab" on Monday 09:00 and "Physics Theory" on Monday 11:00.	System should detect ordering violation (Lab scheduled before Theory).	Fixed
BP-04	Lab Capacity Lab capacity \leq 40 students.	Assign a course with 45 students to "Lab Room A" (Capacity: 40).	System should flag "Capacity Violation".	Pass
BP-05	3rd Year Overlap 3rd-year courses should not overlap with electives.	Schedule "SENG301" (3rd year) and "SENG450" (Elective) at the same time slot.	System should report a "Curriculum Overlap" conflict.	Pass

1.2 Version History (BeePlan)

Changes made based on errors detected during OpenAI-assisted testing.

- **v1.0 (Initial):** Basic scheduling algorithm implemented.
- **v1.1 (Fix):** Fixed **BP-03**. The algorithm initially treated Labs and Theory as independent blocks. Added a constraint rule to ensure [Lab_Time > Theory_Time](#).
- **v1.2 (Optimization):** Improved conflict detection performance for **BP-05**. Originally $O(n^2)$, optimized to $O(n)$ using a hash map for time slots

Part 2: KidTask (Java) Testing

AI Tool Used: Deepseek 8

Focus: GUI functionality, User Input Validation, and Data Persistence.

2.1 Test Cases Generated by Deepseek

The following test cases focus on the GUI components and Data Persistence requirements.

Test ID	Requirement	Test Input	Expected Outcome	Status
KT-01	Task Creation Add new task with title, description, points.	User enters "Math HW", points: "50", clicks "Add".	Task appears in the "Task Management Panel" list immediately.	Pass
KT-02	Points Calculation Update level dynamically based on points.	Mark 3 tasks as "Completed" (Total 150 points).	Progress bar updates, and Level changes from 1 to 2 if threshold met.	Fixed
KT-03	Wish Visibility Only display wishes available at child's level.	Child is Level 1. Wish "Bike" requires Level 5.	The "Bike" wish should not be visible in the Wish list.	Pass

KT-04	Data Persistence Read data on program start.	Add a task, close the app, and reopen it.	The added task must still exist in the table (loaded from <code>Tasks.json</code>).	Pass
KT-05	Input Validation Points must be numeric.	Enter "Ten" into the Points field.	GUI should show an error dialog: "Please enter a valid number."	Fixed

2.2 Version History (KidTask)

Changes made based on errors detected during Deepseek-assisted testing.

- **v1.0 (Initial):** GUI layout completed.
- **v1.1 (Fix):** Fixed **KT-02**. The progress bar was not refreshing visually after points were updated. Added `repaint()` and `revalidate()` calls in the Controller.
- **v1.2 (Fix):** Fixed **KT-05**. The application crashed when non-numeric text was entered into the Points field. Added a try-catch block and an alert dialog for invalid input.