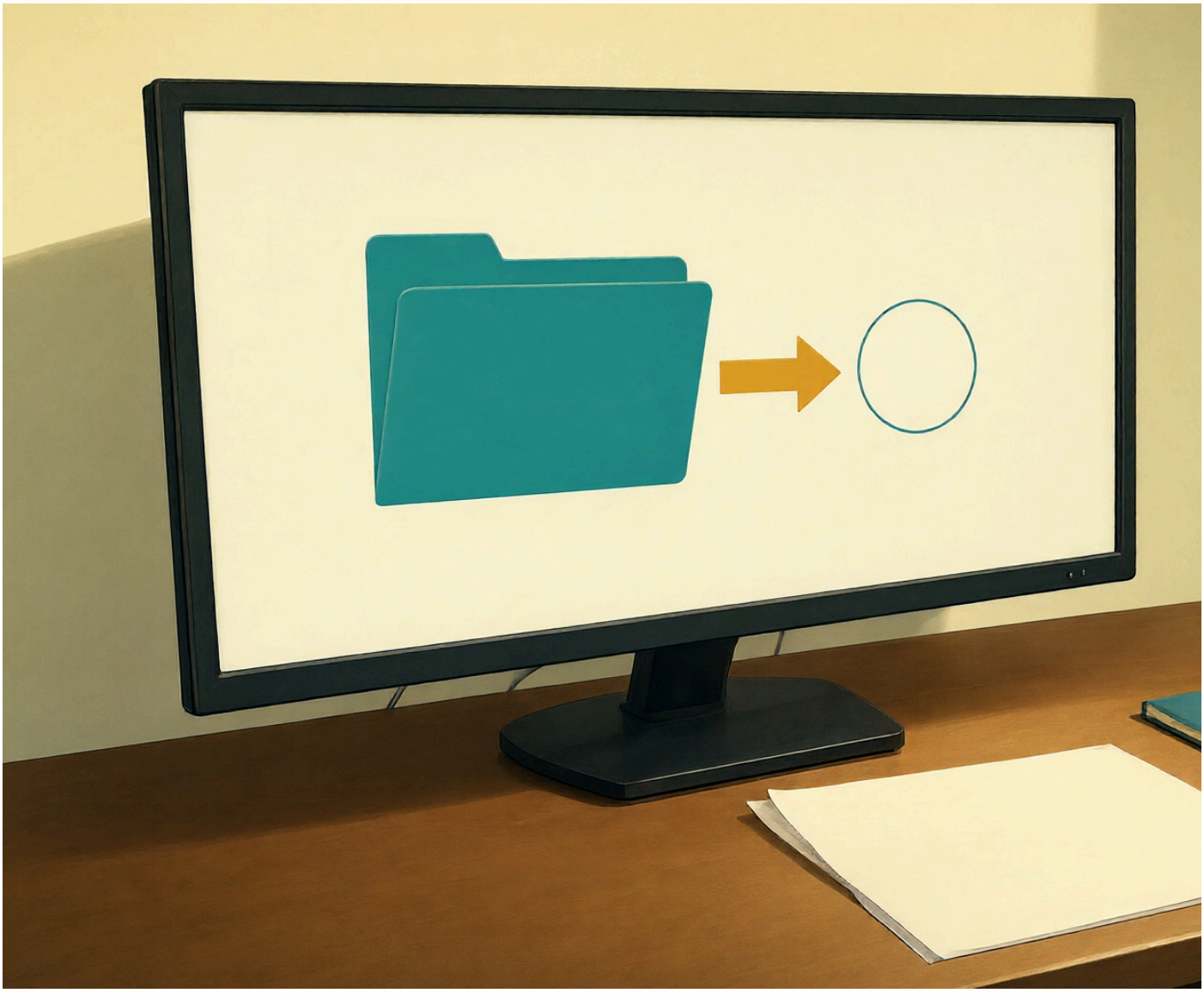
The Version That Entered the World



مدارج



On Monday morning, seventeen-year-old computing student Kai opened the small task organizer he had been building for two years. He kept moving it between folders and adding hopeful words to its filename, but nobody else had tried it. A message arrived from his classmate Ziad, who was also seventeen: could he test a copy on Friday? Kai read it twice. Ziad had not asked for a famous application or a presentation before the whole class. He wanted to try his friend's work. Yet the short distance between their computers suddenly felt surprisingly difficult to cross.

Kai opened the task screen. The sample entries were familiar: review a lecture, submit an assignment, organize files. He moved the pointer toward the screenshot button, then opened the font settings instead. He enlarged a heading, reduced it again, and compared two almost identical colors. Ziad's message remained unanswered. Choosing a color did not

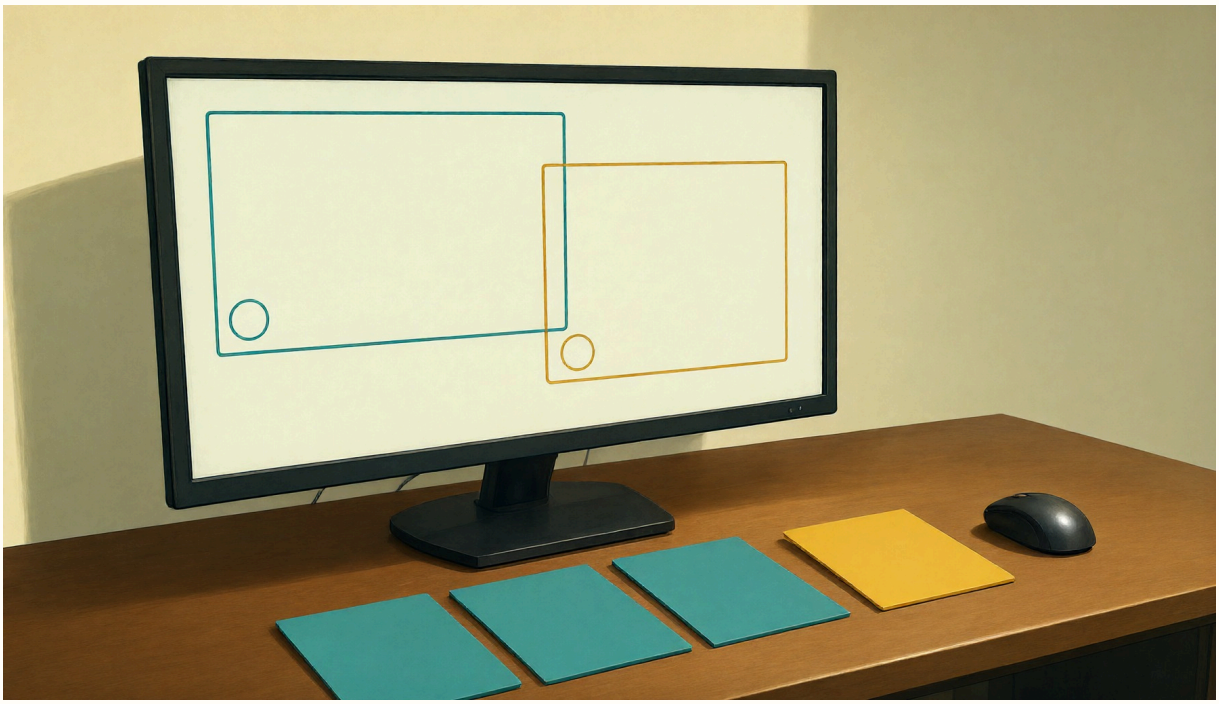
require so much attention, but it gave Kai's hands something harmless to do while avoiding the step that troubled him. He closed the settings and looked again at the tasks that only he had ever tested.



He drafted a reply saying the version needed improvements, then stopped before sending it. Which improvements? He knew about some small defects, but he could not explain what would make the program ready for a trial. He imagined Ziad pausing at an unfamiliar button and asking a question he had missed for two years. Kai put a sheet beside the computer and wrote down his fear: someone would evaluate his work before he understood what the trial required. At last, his worry described a situation he could ask about.

Kai brought his sheet to the institute on Tuesday morning. He nearly told his instructor, Sami, that he delayed everything. Instead, he asked the narrower question he had written down: how could he tell whether a

trial copy was ready? Forty-year-old Sami taught them to build small programs. He asked what Ziad needed to do, rather than everything Kai hoped to add. Kai began listing reminders, colors, and synchronization between devices. Sami pointed to the first line. Which features did Friday's trial actually require? Kai paused, then crossed out half his list.



They kept three functions: add a task, mark it complete, and remove it. Instead of writing “excellent” beside them, Kai asked what a tester should be able to observe. They chose two examples with invented names, requiring nobody's personal files or information. In the first, the tester would add “Arrange the shelf,” mark it complete, close the program, and reopen it to check whether the mark remained. In the second, he would add “Draw a square,” delete it, and check that it disappeared from both the full list and the completed list. Kai left a space beside each step for its actual result.

He suggested waiting until he added reminders. Would their absence prevent either example? No. He moved them to a later list and fixed Friday afternoon as the delivery time. He sent Ziad the three functions and asked him to test them on Wednesday. The program stayed unattached: Kai still needed evidence that his promises worked. He placed the sheet beside his computer. Tonight's question would concern those steps, rather than his worth as a programmer.

On Wednesday, Ziad placed the test sheet between them and asked to control the mouse. He added “Arrange the shelf” and marked it complete. The mark appeared correctly. Kai leaned forward, ready to record a success, but Ziad pointed to the next instruction: close and reopen the program. The task's name remained, yet its completion mark disappeared. Kai reached toward the mouse, then left it with his friend. They repeated the steps with a new task, “Sort the pencils.” The same thing happened. Kai could no longer dismiss it as a possible wrong click.



They moved to deletion. Ziad added “Draw a square” and marked it complete first, making it visible in both lists. He deleted it from the full list, then opened the completed list. Its name was still there. Kai recorded the four steps, the expected result, and what they actually observed. He did not claim to know the cause: he still needed to trace how information was saved and the lists were refreshed. They repeated the example under another name. Again, the deleted task remained in the completed view. They now had two cases he could reproduce.

After Ziad left, Kai opened the font settings almost automatically. He paused over the list, then closed it. Better lettering would not preserve a completion mark or remove a stranded task. Before changing the code, he copied the trial file and saved the test notes beside it. Both defects remained when he finished that evening. They were unpleasant to see, but now he had observable results to work from on Thursday.

On Thursday morning, Kai reopened the copy he had kept before editing and reproduced the missing completion mark. He traced the saving routine and found that it recorded the task's name while leaving its previous state unchanged. He corrected that specific step, added “Arrange the shelf,” marked it complete, and restarted the program. Both the name and the mark remained. Before recording a success, he repeated the test with “Sort the pencils” and checked that an unfinished task did not become complete by mistake. He wrote down the results beside Wednesday's observations.

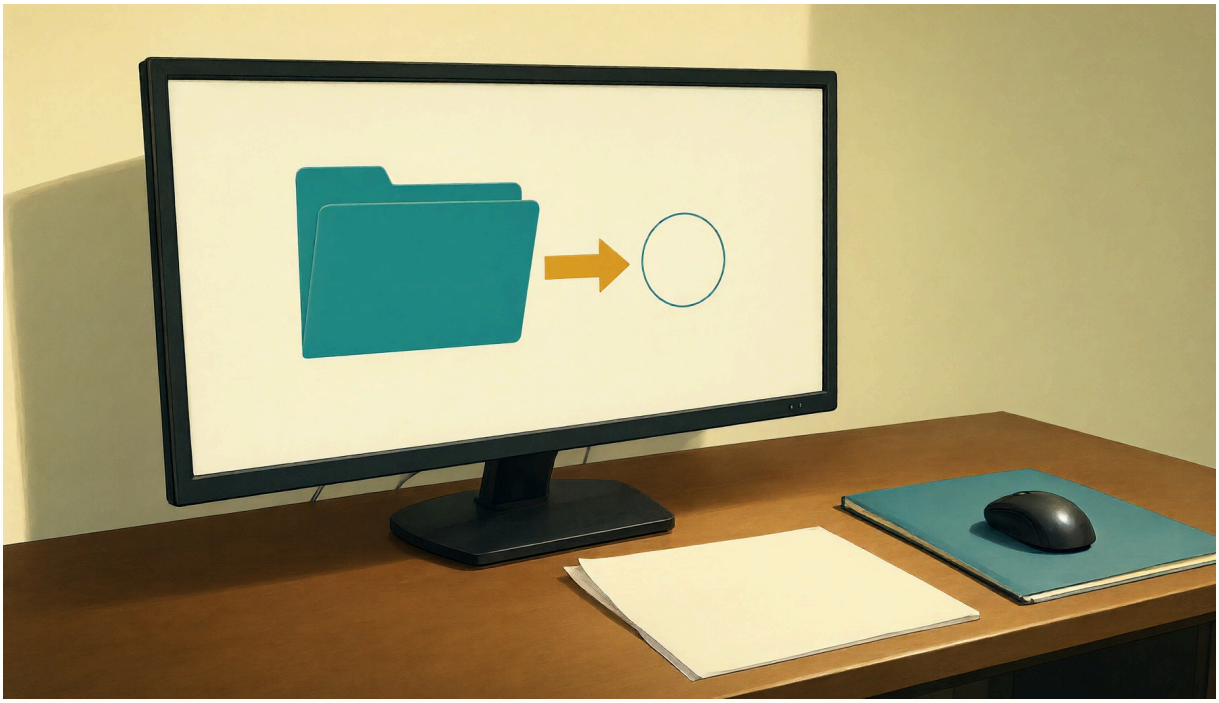


He then traced the deletion problem. The full list changed, while the completed view still displayed an older list. He made both views read the current task state. He repeated “Draw a square”: add, complete, delete, inspect both lists. The name appeared in neither. He repeated the sequence with another task and restarted once more. These results did

not prove the application had no other defects. They showed that the two recorded problems no longer appeared through the same steps. He numbered the new version and kept the earlier copy separate.

Kai prepared brief instructions covering the three functions, startup, and limits: one device, invented task names, no synchronization or reminders. A reminder feature tempted him to open another file, but he put it on the next-version list. Ziad still needed to try the program without being guided to every button. Kai collected the tested version, instructions, and results in one folder. He did not send it on Thursday; Friday's final trial still came before delivery.

On Friday morning, Kai placed the folder before Ziad and moved away from the computer. He kept his hands back instead of pointing out buttons before his friend could find them. Ziad read the startup sheet, added “Arrange the shelf,” marked it complete, and restarted the program. This time, the mark remained. Kai recorded the result under Thursday's version number. Ziad then added “Draw a square,” marked it complete, deleted it, and checked both views. Neither contained it. He repeated the sequence with a name of his own while Kai watched without coaching.



Ziad asked how to request a reminder before a task was due. Kai wanted to open the code immediately. He could spend the whole morning beginning that addition. Instead, he asked Ziad to read the instructions, which explicitly excluded reminders. Their absence did not prevent adding, completing, or removing tasks, but the program would not yet suit someone who needed automatic alerts. Kai entered the request on the next-version list and kept that limitation in the trial description. Passing the tests did not make the limitation disappear, and a new suggestion did not cancel their delivery time.

Kai checked what the recipient would receive: a numbered version, short startup steps, both test results, and the same limits. He removed an older copy that had nearly entered the folder, then reopened the selected version to confirm its number. When Ziad asked about two devices, Kai

explained that the trial supported one only. They closed the application. A single checked folder remained before Kai. He had not sent it yet, but could now describe its contents accurately.

As Friday afternoon approached, Kai opened a message to Ziad. He first wrote, “This version is unfinished,” then deleted it. The description was too broad to help. Instead, he explained the three functions, attached the test steps and results, and stated the limits: one device, no reminders, no synchronization. He asked Ziad to use invented task names and report the steps preceding any problem, rather than simply saying that something did not work. Kai checked the attachments once more. They included the version he had actually tested. Before the agreed time, he pressed Send.

He waited longer than the message needed to travel. For a moment, he considered changing the heading color and sending another copy. He moved his hand away. Ziad's reply was ordinary: the folder had arrived, and he would begin with the instructions. It contained no verdict on Kai's future and no promise that everyone would like the program. Kai set aside time on Sunday evening to sort the comments: reproducible defects first, then improvements needing a new decision. He kept the delivered copy in a dated folder, separate from later edits.

His worry had not vanished. Yet sending no longer represented an unknown judgment on two years of work. He had delivered something bounded, tested, and described honestly. You can try the same approach privately. Choose an imaginary task, such as preparing a card about a geometric shape. Write down what you will deliver, an observable condition showing it is finished, and a definite delivery time. Add one improvement to leave for a later version. At that time, compare your result with your written condition.