□□ कोड एडिटर्स की अगली दिशा

हाल ही में, मैं □□□□□□ □□□□□□□□ में एक `xelatex` पाइपलाइन जोड़ने पर काम कर रहा था।

मुझे □□□□□□ फ्लो में `fontawesome5` पैकेज के साथ एक समस्या का सामना करना पड़ा। 4□-□□□□ द्वारा प्रदान किया गया समाधान (□□□ □□□□ 2021 इंस्टॉल करना और `tlmgr install fontawesome5` का उपयोग करना) मेरे लिए काम नहीं किया। हालांकि, 4□ ने एक बेहतर तरीका सुझाया: □□□ □□□□ 2023 में अपग्रेड करना और `fontawesome5` को इंस्टॉल करने के लिए `tlmgr` का उपयोग करना। हालांकि इससे समस्या पूरी तरह से ठीक नहीं हुई, लेकिन □□□ □□□□ 2023 पर स्विच करने से स्थिति में काफी सुधार हुआ।

मैंने समस्या को समझने में मदद के लिए □□□□□□□ का उपयोग किया। अधिक जानकारी के लिए, □□□□ □□□□□□□ □1 □□□ □□ □□□□ 4□-□□□□ □□□□□□ देखें।

इस समय, मैंने □□□□□□ या □□□□□□□□ जैसे एडिटर्स का उपयोग नहीं किया, हालांकि मैंने इन्हें एक अन्य प्रोजेक्ट में आज़माया था। इन कोड एडिटर्स की समस्या यह है कि ये केवल स्थानीय टेस्ट आउटपुट को कैप्चर करते हैं, जो क्लाउड वातावरण में उनकी कार्यक्षमता को सीमित कर देता है।

□□□□□□ □□□□□□□, □□□□□□□ □□□□, या किसी भी कोड डिप्लॉयमेंट या टेस्टिंग फ्लो जैसे वर्कफ़्लोज़ में, कोड एडिटर्स को बेहतर तरीके से इंटीग्रेट किया जाना चाहिए। उन्हें क्लाउड और □□/□□ प्रक्रियाओं के साथ सहज इंटरैक्शन प्रदान करना चाहिए।

यह एकीकरण अन्य सामग्री निर्माण टूल्स पर भी लागू होता है—चाहे वह टेक्स्ट, इमेज, ऑडियो, या वीडियो के लिए हो। इन टूल्स को □/□ टेस्टिंग सिस्टम के साथ एकीकृत किया जाना चाहिए। □□ टूल्स सामग्री उत्पन्न कर सकते हैं, और □/□ टेस्टिंग टूल्स फीडबैक प्रदान कर सकते हैं। यह डायनामिक □□□□□□□□□□□□□□ □□□□□□□□ □□□□ □□□□□ □□□□□□□□ (□□□□) के समान है, जहां □□ मॉडल वास्तविक दुनिया के फीडबैक के आधार पर समय के साथ सुधार करते हैं।

□□□□ (□□□□□□□□□□□□□□ □□□□□□□□ □□□□ □□□□□ □□□□□□□□) को केवल मॉडल आउटपुट्स तक सीमित रखने के बजाय इसे वास्तविक दुनिया के परीक्षण और तैनाती वातावरण में विस्तारित करने की यह अवधारणा, कोड एडिटर्स और □□-संचालति कंटेंट क्रिएशन टूल्स दोनों में सुधार के लिए एक आशाजनक दिशा प्रतीत होती है।

टेस्ट तात्कालिक या लंबा हो सकता है, और यह स्वचालित या मानव-सहायित हो सकता है। यदि टेस्ट स्वचालित हैं, जैसे कि □□ टूल के लिए यूजर □/□ टेस्टिंग, तो इसमें अभी भी मानव फीडबैक शामिल होता है, लेकिन प्रक्रिया स्वचालित होती है। उदाहरण के लिए, हम कंप्यूटर को □/□ टेस्टिंग के परिणामों के आधार पर हर दिन या हर घंटे परिणाम जांचने दे सकते हैं ताकि निर्माण प्रक्रिया में सुधार किया जा सके। इसी तरह, □□□□□□□□ या □□□□□□ □□□□□□□□ जॉब्स के लिए, हम कंप्यूटर को उनके कार्य पूरा होने के बाद जांच करने दे सकते हैं।

यदि मानव सहायता शामिल है, तो मशीन द्वारा प्रतिक्रिया को पूरी तरह से समझा नहीं जा सकता है और यह अक्सर कुछ हद तक अस्पष्ट होती है। उदाहरण के लिए, जब □□ टूल्स चित्र या वीडियो जैसी सामग्री बनाते हैं, तो मनुष्य यह बता सकते हैं कि सामग्री पर्याप्त मजेदार नहीं है, या किसी विशेष विवरण को सुधारा जाना चाहिए। मशीनों को अभी भी हर चीज को परफेक्ट बनाने में लंबा सफर तय करना है, और क्या कुछ "परफेक्ट" है यह अक्सर व्यक्तिपरक होता है, जो व्यक्तिगत पसंद पर निर्भर करता है। यह मानव प्रतिक्रिया ही है जो चीजों को बेहतर बनाने में मदद करती है।

सिद्धांत रूप में, सभी मानव-परिभाषित नियमों को प्रॉम्प्ट के रूप में लिखा जा सकता है। इसमें यूजर प्रॉम्प्ट और सिस्टम प्रॉम्प्ट शामिल हैं। हमें हर बार आउटपुट को ठीक करने के बजाय प्रॉम्प्ट को सुधारने पर ध्यान केंद्रित करना चाहिए।