□□□ विकास को स्वचालित परीक्षण और उपकरणों के माध्यम से बेहतर बनाना

यह ब्लॉग पोस्ट □□□□□□□-4□ की सहायता से तैयार किया गया है।

यूनिट टेस्टिंग का महत्व

□□□□□□□□□ में, हमने प्रोजेक्ट के शुरुआती चरण में ही यूनिट टेस्टिंग लागू कर दी थी, जो कि बहुत मूल्यवान साबित हुई है। हर पुल रिक्वेस्ट (□□) □□□□□□□ पर यूनिट टेस्ट को ट्रिगर करती है, और हमारा कवरेज लक्ष्य लगभग 80% है। टेस्ट लिखने के दो मुख्य परिदृश्य हैं: नए इंटरफेस को वैलिडेट करना और बग्स को रिप्रोड्यूस और फिक्स करना। जितने अधिक टेस्ट जमा होते हैं, हमारा कोडबेस उतना ही मजबूत होता जाता है। ऑटोमेटेड टेस्टिंग हमें कोड को रिलीज़ और रिफैक्टर करने के लिए आत्मविश्वास से भर देती है, बिना मैन्युअल वैलिडेशन के।

परीक्षण प्रक्रिया और वास्तविक अनुप्रयोग

यहां कुछ उदाहरण दिए गए हैं कि कैसे यूनिट टेस्ट हमारी मदद कर सकते हैं:

टेस्ट प्रक्रिया 1: कुछ उपयोगकर्ताओं ने रिपोर्ट किया कि जब वे विवरण कुंजी (□□□□□□□□□□□□ □□□) वाले ऑब्जेक्ट को सेव करते हैं, तो एक त्रुटि होती है। मैंने इस समस्या को पुन: उत्पन्न करने के लिए एक टेस्ट लिखा, समस्या का पता लगाया और उसे ठीक किया। फिर भविष्य में सत्यापन के लिए इस टेस्ट को संरक्षित रखा।

टेस्ट प्रक्रिया 2: नए इंटरफेस को विकसित करते समय, मैंने कोड को लागू करने के बाद संबंधित टेस्ट लिखे, ताकि यह सुनिश्चित किया जा सके कि कोड सही ढंग से काम कर रहा है।

टेस्ट प्रक्रिया 3: `AVObject.m` कोड को संशोधित करने के बाद, मैंने `AVObjectTest.m` टेस्ट चलाया, यह जांचने के लिए कि क्या किसी भी टेस्ट में असफलता हुई है।

टेस्ट प्रक्रिया 4: □□ सबमिट करने पर □□□□□□□ पर स्वचालित टेस्ट ट्रिगर होगा।

यूनिट टेस्ट लिखने के फायदे

- **मैन्युअल सत्यापन कम करना:** यूनिट टेस्ट मैन्युअल जांच को खत्म करके समय बचाते हैं।
- **त्रुटि पहचान:** कोड में बदलाव के कारण होने वाली समस्याओं को पहले ही पहचान लेना, ताकि त्रुटियां प्रोजेक्ट के अन्य हिस्सों को प्रभावित न करें।
- **सहयोगी प्रोजेक्ट:** एकाधिक डेवलपर्स वाले प्रोजेक्ट में, यूनिट टेस्ट सुनिश्चित करते हैं कि प्रोजेक्ट में स्थिरता और विश्वसनीयता बनी रहे, चाहे प्रोजेक्ट किसी और को सौंप दिया जाए।
- **उच्च गुणवत्ता वाले ओपन सोर्स प्रोजेक्ट:** लोकप्रिय ओपन सोर्स प्रोजेक्ट में आमतौर पर व्यापक यूनिट टेस्ट होते हैं, जो उनकी विश्वसनीयता और लोकप्रियता में मदद करते हैं।

प्रभावी यूनिट टेस्ट कैसे लिखें

- **मॉड्यूलर कोड:** डेटा लेयर और □□ लेयर को अलग करें ताकि टेस्टिंग आसान हो।
- **कवरेज को अधिकतम करें:** न्यूनतम टेस्ट कोड का उपयोग करके अधिकतम कवरेज प्राप्त करें।
- **असिंक्रोनस प्रोसेसिंग:** सुनिश्चित करें कि टेस्ट असिंक्रोनस ऑपरेशन को हैंडल कर सकता है।
- **फ्रेमवर्क चयन:** अपनी आवश्यकताओं के अनुसार उपयुक्त टेस्ट फ्रेमवर्क चुनें।
- **कवरेज रिपोर्ट:** कवरेज रिपोर्ट का उपयोग करके यह जानें कि कोड के कौन से हिस्से टेस्ट किए गए हैं।

परीक्षण फ्रेमवर्क का मूल्यांकन

हमने कई फ्रेमवर्क का मूल्यांकन किया: - □□□□□□□□: `expect(error).not.beNil()` - □□□□□□□: `describe("") it("")` - □□□□□: `describe("") it("")` - □□□ और □□□ फ्रेमवर्क में कुछ सीमाएँ हैं, जैसे □□□□□□ के साथ खराब एकीकरण, कोई परीक्षण बटन नहीं, और साइडबार में सभी यूनिट टेस्ट सूचीबद्ध नहीं हैं।

एसिंक्रोनस टेस्टिंग को संभालना

एसिंक्रोनस टेस्टिंग उन ऑपरेशन्स के लिए महत्वपूर्ण है जो तुरंत पूरे नहीं होते हैं। सुनिश्चित करें कि आपका फ्रेमवर्क एसिंक्रोनस टेस्टिंग को प्रभावी ढंग से सपोर्ट करता है। उदाहरण के लिए, □□□□□□□ में □□□□□□□□□□□□□□ का उपयोग करके एसिंक्रोनस ऑपरेशन के पूरा होने का इंतजार करें, और फिर □□□□□□□□□□ करें।

कवरेज रिपोर्ट

□□□□□ 7 ने अंतर्निहित कवरेज रिपोर्टिंग सुविधा पेश की है। इसे सक्षम करने के चरण निम्नलिखित हैं: 1. स्कीम सेटिंग्स में `Gather Coverage Data` को सक्षम करें। 2. टेस्ट टार्गेट के बजाय ऐप टार्गेट के लिए टेस्ट करें।

यह सुविधा डेवलपर्स को यह देखने की अनुमति देती है कि कौन सी कोड लाइनें टेस्ट की गई हैं, जिससे अनटेस्टेड कोड के हिस्सों की पहचान करने में मदद मिलती है। अधिक जानकारी के लिए □□□ □□□□ □□□□□ के ब्लॉग पर जाएं।

□□□□□□□□ का उपयोग करके रिमोट ऑटोमेशन टेस्टिंग

□□□□□□□□ को ऑटोमेशन टेस्टिंग के लिए सेट करने में कई चरण शामिल हैं: 1. □□□□□□□□ **इंस्टॉल करें:** अपने लोकल मशीन या डेटा सेंटर सर्वर पर □□□□□□□□ सेट करें। 2. □□□□□□□ **इंटीग्रेशन:** □□□□□□□ □□ बिल्ड प्लगइन का उपयोग करके पुल रिक्वेस्ट सबमिट होने पर टेस्ट ट्रिगर करें। - वेबहुक्स को कॉन्फ़िगर करें ताकि इवेंट्स □□□□□□□□ को भेजे जा सकें। - सुनिश्चित करें कि □□□□□□□□ पुल रिक्वेस्ट के नवीनतम कोड तक पहुंच सकता है। 3. **टेस्ट स्क्रिप्ट:** □□□□□□□□ में टेस्ट स्क्रिप्ट सेट करें ताकि टेस्ट प्रक्रिया स्वचालित हो सके। - सुनिश्चित करें कि □□□□□□□□ □□□□□□□ को टेस्ट परिणामों की सूचना दे सकता है। - टेस्ट फेल होने पर □□□□□□ या ईमेल नोटिफिकेशन कॉन्फ़िगर करें।

□□□□□□□□ का उपयोग करके रिमोट ऑटोमेशन टेस्टिंग करने से ऑटोमेशन टेस्टिंग के सभी लाभ मिलते हैं, जो स्थानीय टेस्टिंग से बेहतर होता है, क्योंकि यह एक स्वच्छ और नियंत्रित वातावरण में टेस्ट चलाता है।

रिमोट पैकेजिंग और डिप्लॉयमेंट

हालांकि सभी प्रोजेक्ट्स को रिमोट पैकेजिंग की आवश्यकता नहीं होती है, लेकिन यह □□□ और अन्य पुन: प्रयोज्य कंपोनेंट्स के डिप्लॉयमेंट प्रक्रिया को सरल बना सकता है। इसके चरणों में शामिल हैं: - □□□□□□□□ को कोड पढ़ने के लिए कॉन्फ़िगर करना। - रिलीज़ वर्ज़न को पढ़ना। - साइनिंग सर्टिफिकेट तक पहुंचने के लिए कमांड लाइन में कीचेन को अनलॉक करना।

अतिरिक्त टूल्स और टिप्स

- □ □□□□□□**:** जेलब्रेक किए गए □□□□□□ पर किसी भी ऐप के □□ इंटरफ़ेस का विश्लेषण करें।
- □ □□□□**:** जेलब्रेक किए गए □□□□□□ पर नेटवर्क रिक्वेस्ट्स, □□, लोकल फ़ाइलें, □□□□□□□□□□□□□□□ और लॉग्स का विश्लेषण करें।
- □ □□□ **प्रबंधन:** लोकल □□□ का उपयोग, उन्नत □□□□□□□□ कॉन्फ़िगरेशन और □□□□ को प्रकाशित करना।
- □ **फ्रेमवर्क निर्माण:** डायनामिक लाइब्रेरी और स्टेटिक लाइब्रेरी के बीच अंतर, और सिम्युलेटर और डिवाइस दोनों के लिए उपयुक्त फ्रेमवर्क को कैसे पैकेज करें।
- □ □□□□□ **टिप्स:** उपयोगी शॉर्टकट, जैसे □□□□□ + □□□□□□□□ + □ नेविगेटर में फ़ाइल दिखाने के लिए, □□□□□ + □□□□□□□□ + □ फ़ाइल को तेज़ी से खोलने के लिए।

निष्कर्ष

स्वचालित परीक्षण और उपयुक्त उपकरणों ने विकास प्रक्रिया को काफी बढ़ावा दिया है। यूनिट टेस्टिंग को शुरुआती चरण में शामिल करके, एसिंक्रोनस प्रोसेसिंग का उपयोग करके और कवरेज रिपोर्ट का उपयोग करके, हम अधिक विश्वसनीय और रखरखाव योग्य एप्लिकेशन बना सकते हैं। □□□□□□□□ जैसे □□/□□ उपकरण और □□□□□ डेवलपमेंट टूल्स को एक मजबूत परीक्षण रणनीति के साथ जोड़कर, उच्च गुणवत्ता वाले सॉफ़्टवेयर डिलीवरी सुनिश्चित की जा सकती है।

धन्यवाद

□□□□□□□□□□ टीम और हमारे परीक्षण प्रक्रिया में योगदान देने वाले सभी लोगों को विशेष धन्यवाद।