Application of Python Automated Testing in Intelligent Parking Area

Xiao YU¹,∗ and Peng XIA²

¹Foshan Polytechnic, Foshan, China
²Foshan Nanhai Guangdong Technology University CNC Equipment Cooperative Innovation Institute, Foshan, China

∗Corresponding author

Keywords: Intelligent parking, Automatic testing, Python.

Abstract. At present, with the rapid development of the economy and the rapid growth of private vehicles, intelligent parking lots are rapidly appearing in major shopping malls and other public places. Intelligent parking involves many options, such as unattended, automatic license plate recognition, efficient entry and exit, business volume, coupons, VIP, fast-track payment, temporary car and other unlicensed car code sweeping, etc. It puts forward a test to the process control of intelligent parking, and how to ensure the design of intelligent parking system and control equipment in various fields. The sample application scenarios and options can quickly and accurately execute the process, control equipment, complete payment and other functions. The large-scale manual scenario testing by testers alone cannot meet the requirements. Therefore, this paper presents and designs the software and method of intelligent parking system process automation test based on Python and Jenkins integrated tools, which can test all application scenarios and options quickly, accurate and complete, and ensure the accuracy, integrity and efficiency of intelligent parking system.

Introduction

With the rapid growth of private cars, parking problems began to trouble the public. How to quickly and conveniently park and pay has become an urgent problem for the public to solve. In this context, smart parking becomes the preferred method. In daily life and travel, intelligent parking has shown more and more high use value and social value. At present, the intelligent parking lot can realize automatic license plate photo recognition, automatic car entry and exit, automatic line payment by WeChat or alipay or self-service payment by payment machine, on-site remote call for help and other functions. Intelligent parking system involves VIP, merchant coupons, car rental, temporary car, unlicensed car, various preferential billing rules, free parking time, multiple license plate binding, a variety of voice and display tips, automatic pole, automatic in and out of the car, and many other front-end options. However, after designing an intelligent parking system that can cover a series of processes such as various application scenarios, various vehicle types, and various parking billing rules, traditional manual testing cannot guarantee a complete, accurate and rapid test of all processes. Therefore, this paper proposes and designs an automated test technology based on Pyhon and Jenkins integrated tools, which can test all application scenarios and options quickly, accurate and complete, and ensure the accuracy, integrity and efficiency of intelligent parking system.

Intelligent Parking Automation Test Design

At present, intelligent parking lot is generally composed of web intelligent parking system, camera gun, entrance and exit all-in-one machine control equipment, gateway and other online payment APP, WeChat public number entrance, etc. The functions of the parking system include billing, gate control, voice display, online payment, on-site payment, self-service payment machine payment, WeChat access reminder, coupon collection, etc.

If the full coverage automated test is to be completed for the intelligent parking system meeting the above functions, the black box test is needed for the intelligent parking system of web. Because all the
pre-options, billing rules, benefits, payments, data, control commands are in there. This paper defines the web parking system in the name of Zhibo Cloud, and uses python to simulate the external equipment and interface environment. The devices and interface environment to be simulated include: entrance and exit all-in-one machine equipment, simulated external web operation interface, small program push interface, third-party open platform interface and fee-end interface. These are written uniformly in python and run test cases for various scenarios and process rules for automated testing. In order to ensure that the test case data and the simulated all-in-one machine, external interface and simulation small program can interconnect and intercommunicate in a platform, the system adopts the general MQTT tool as the intermediate platform to forward and realize data interconnection and intercommunication. This not only ensures that the process data is intercommunicated according to the protocol, but also ensures that the data instructions of test cases can be quickly distributed to the all-in-one equipment. Finally, the test results returned by each device in the test case are received through MQTT and used as a criterion to judge whether the test passes, as shown in figure 1.

The testing process is as follows: integrate the Python test cases into the Jenkins tool to automate the timing, automate the tests according to the sequence of test cases, and present the test reports in XML format according to Jenkins’s junit rules.

The flow of automated testing is described below as a normal in-and-out process. The process is as follows: vehicle approach, snapshot, all-in-one machine prompt, parking system processing, record parking records, timing, car out, all-in-one machine prompt, billing, payment, all-in-one machine prompt, car out complete. The python automated test processing flow based Jenkins is shown in figure 2.

![Figure 1. Intelligent parking automated test data trend chart.](image)

![Figure 2. Parking automation test process based on Jenkins.](image)
Integrated Python Automated Test Cases in Jenkins

Test Case Writing Based on Python

Python is now widely used in emerging fields such as artificial intelligence, automated testing, and web crawlers. Python has the title of "glue language", and various support libraries are quite rich. It develops rapidly and can quickly build the platform and realize the required functions. Therefore, Python3.6 is adopted as the development platform in this paper. The corresponding library requires requests, ws4py (websocket push), time, urllib, websocket, json and so on.

The development of Python is based on the window environment, and the development tool is programmed and run by Pycharm, which is convenient and quick. It can be ported to the Linux environment for running after the development. The simulation equipment and environment of the system are run in centos, and test case integration runs on Jenkins which installed in centos.

MQTT uses the general open source EMQ software to send, receive and forward the data. It is quite convenient and easy to install and use. The operating environment is centos6 system.

Integration Setup for Jenkins

Jenkins is an open source software project. It is a continuous integration tool based on Java development that monitors repetitive work over time. It can provide an open and easy-to-use software platform that enables continuous integration of software.

Jenkins supports shell scripts, BAT batch processing and Python scripts for automated testing. Test reports support the Junit specification and generate reports in XML format. You only need to configure the relevant configuration in the configuration interface when building the test Project, as shown in figure 3.

```
#!/usr/bin/python
import('This is a python script')
import time
import sys

request_path = '/jenkin/workspace/test.shocks/test_case_xml/evt/test/test_case.xml

for request_path in validate_request

from external include import ClosePush

import test_case

def test_suite():
    print('test case start')
    def test_case(xml):
        print('test case after')
        return

        def test_xml(xml):
            test_case = TestCase(xmlesar"
            return test_case

        if _name_ == '_main_
        test = unittest.TestSuite()
        suite = test.addTest(test_case)
        suite = test.addTest(test_case)
        test = test.addTest(test_case)
        test = test.addTest(test_case)
        test = test.addTest(test_case)

Figure 3. Python test case setup based on Jenkins.

Automated Test Verification

Import the Python test case into Jenkins, run the simulated Python device, and automate the test. You can manually click on Jenkins to build the test, run the Python use case, or set a schedule in the configuration to automatically test at a certain time. After the test, Jenkins will automatically generate an XML test report, and the developer will be able to know the test results and problems in detail, so as to update the iteration quickly. The process of testing is shown in figure 4.
Conclusion and Prospect

With the development of tools like Python and Jenkins, more and more complex software process tests will rely more on automated tests. In the future, we can also try to write test cases through appropriate simulation equipment, and carry out accurate and efficient process automation test in other fields.

Acknowledgement

This research was financially supported by scientific research project foundation from Foshan Polytechnic.

References


