**تست واحد**

تست واحد یا micro level پایین‌ترین سطح تست است. هر کد تست واحد، یک قطعه کد یا یک تابع (متد) خاص را تست می‌کند. این تست نیاز به دانش در مورد طراحی و نحوه عملکرد داخلی تابع یا قطعه کد دارد. توسط برنامه نویس (و نه تست کننده) انجام می‌شود.

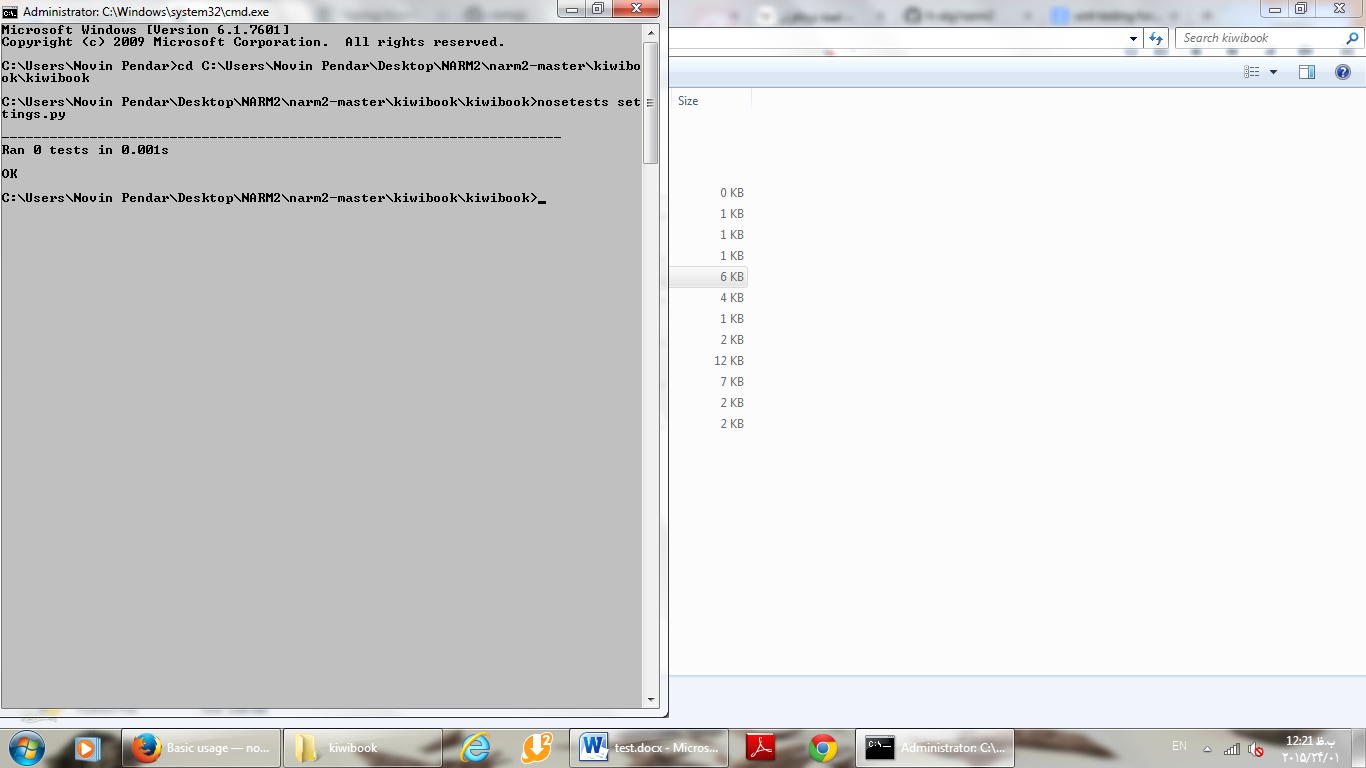
یک روش استفاده از ماژول unittest در جنگو است:

from django.utils import unittest

روش دیگر:

**nose extends unittest to make testing easier**

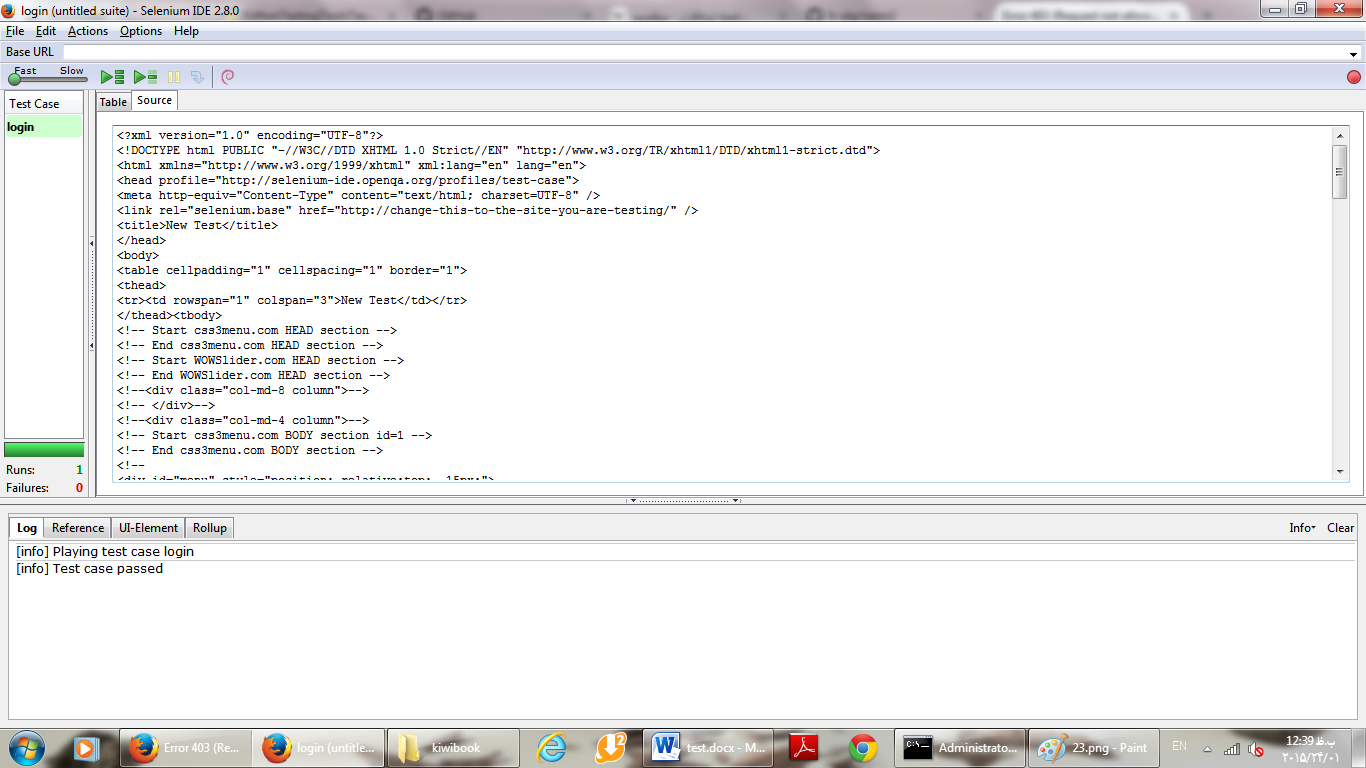
به منظور تست واحد، nose را که شکل کاملتر unittest است نصب کردیم که خروجی آن به شکل زیر است:



همچنین از مفسر آنلاین codepad برای مشاهده خروجی استفاده کردیم.

**تست تطبیق با مرورگرهای مختلف و تست پلتفرم(system testing):**

تست نرم افزار با مرورگر فایرفاکس با نصب Selenium IDE انجام شده است.



**splinter - python acceptance testing for web applications**

برای تست اجرای سایت توسط مرورگرهای کروم،فایرفاکس از splinter نیز استفاده کردیم:

from splinter.browser import Browser

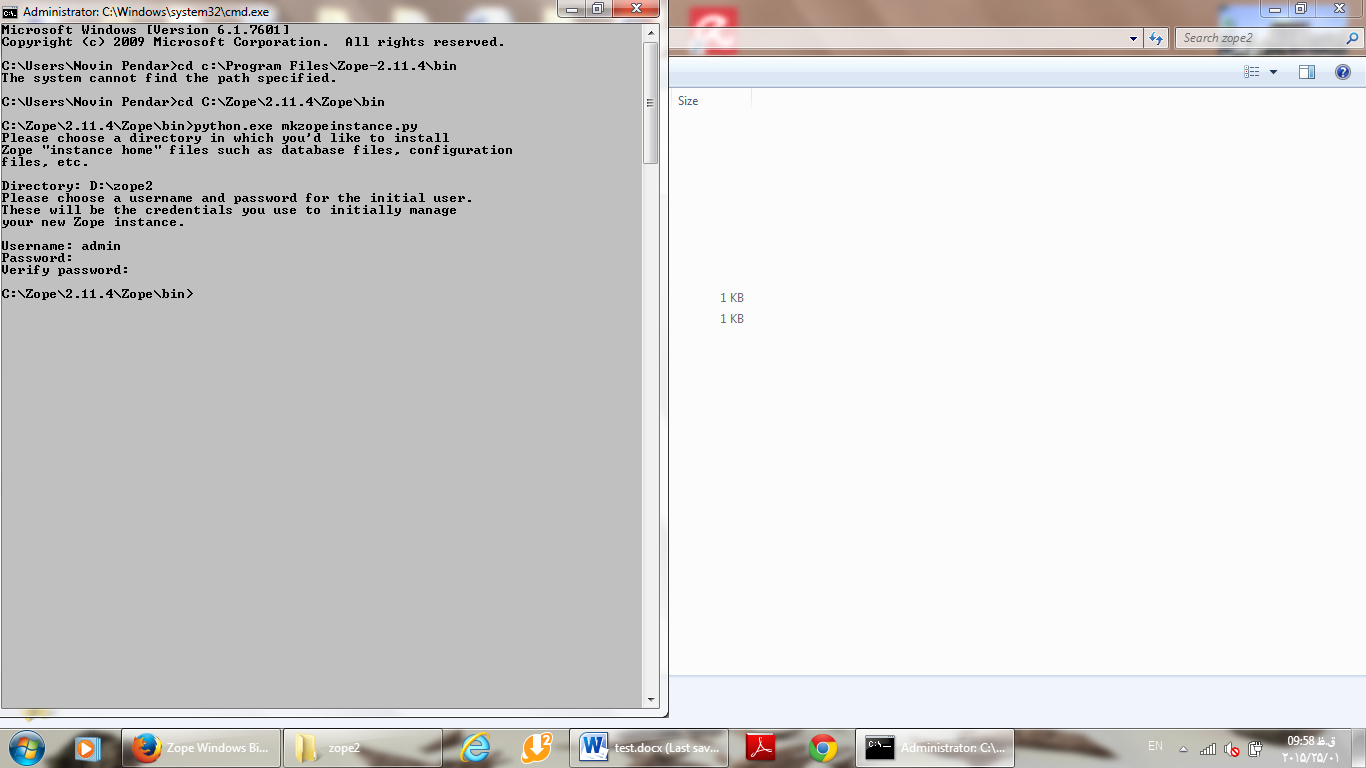
browser = Browser()

browser = Browser('webdriver.chrome')

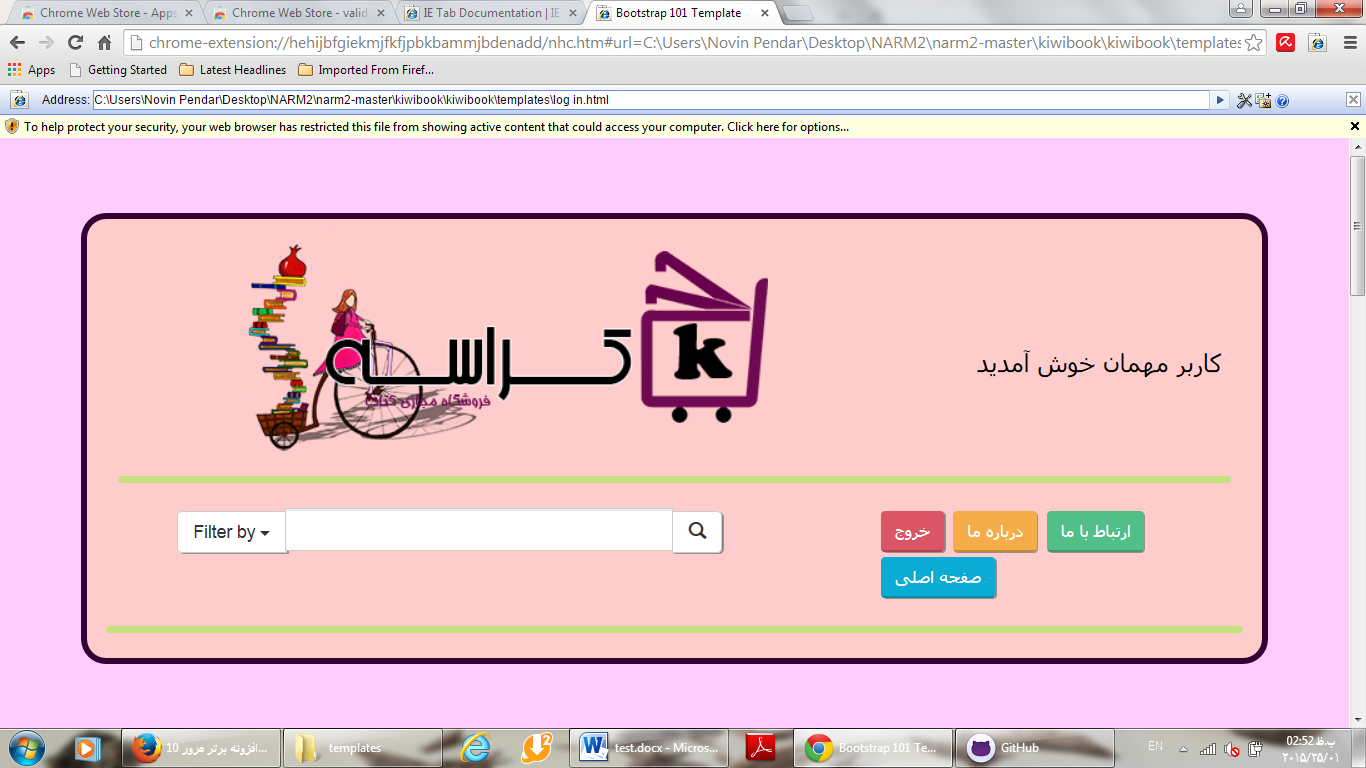
browser = Browser('webdriver.firefox')

browser = Browser('zope.testbrowser')

zope.testbrowser برای تست جعبه سیاه است که برای تست هر وب سایتی کاربرد دارد:



تست سایت را در مرورگر کروم با افزونه validity و در IE با افزونه IE Tab در همان مرورگر انجام می دهیم:

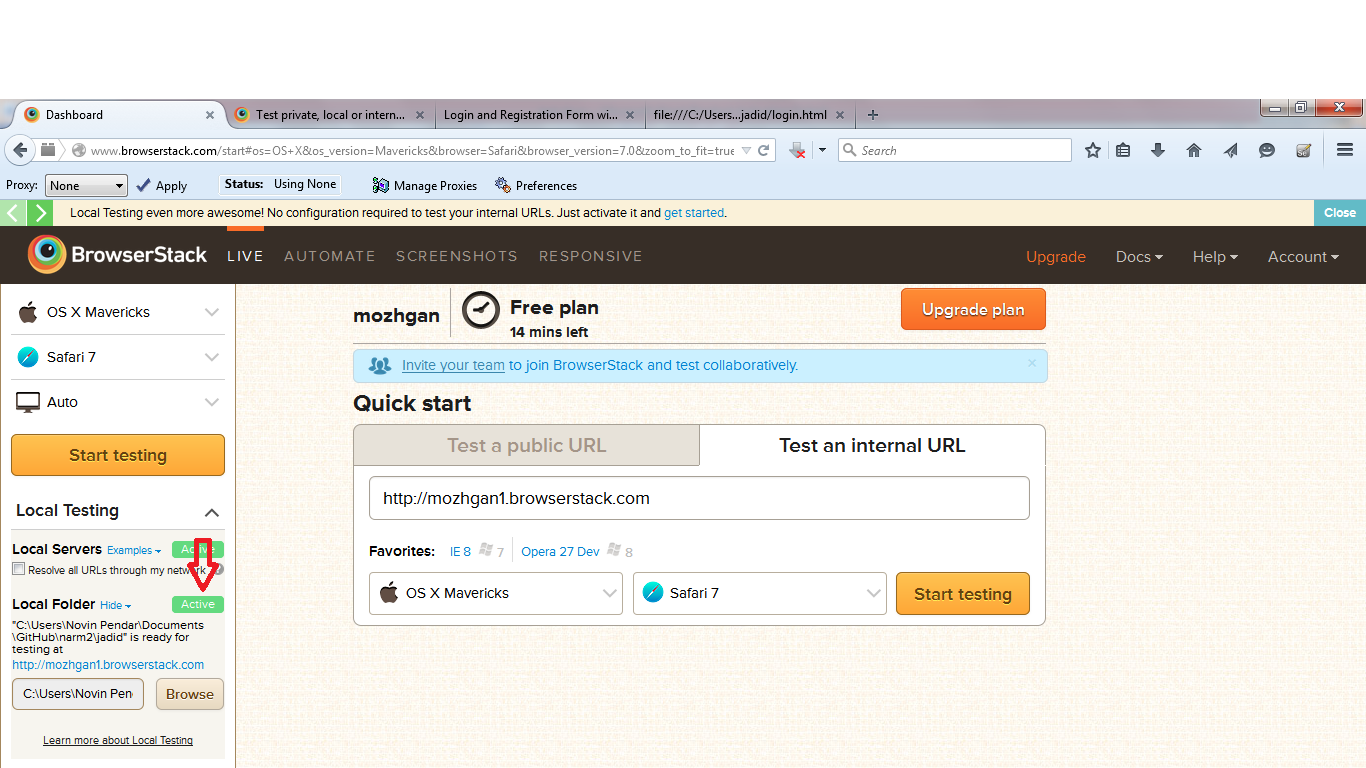


البته برای تست آنلاین وبسایت با public URL ابزارهای فراوانی نظیر [Browserling](https://browserling.com/" \t "_blank)،[Browsershots](http://browsershots.org/" \t "_blank)،[CrossBrowserTesting](http://crossbrowsertesting.com/" \t "_blank)

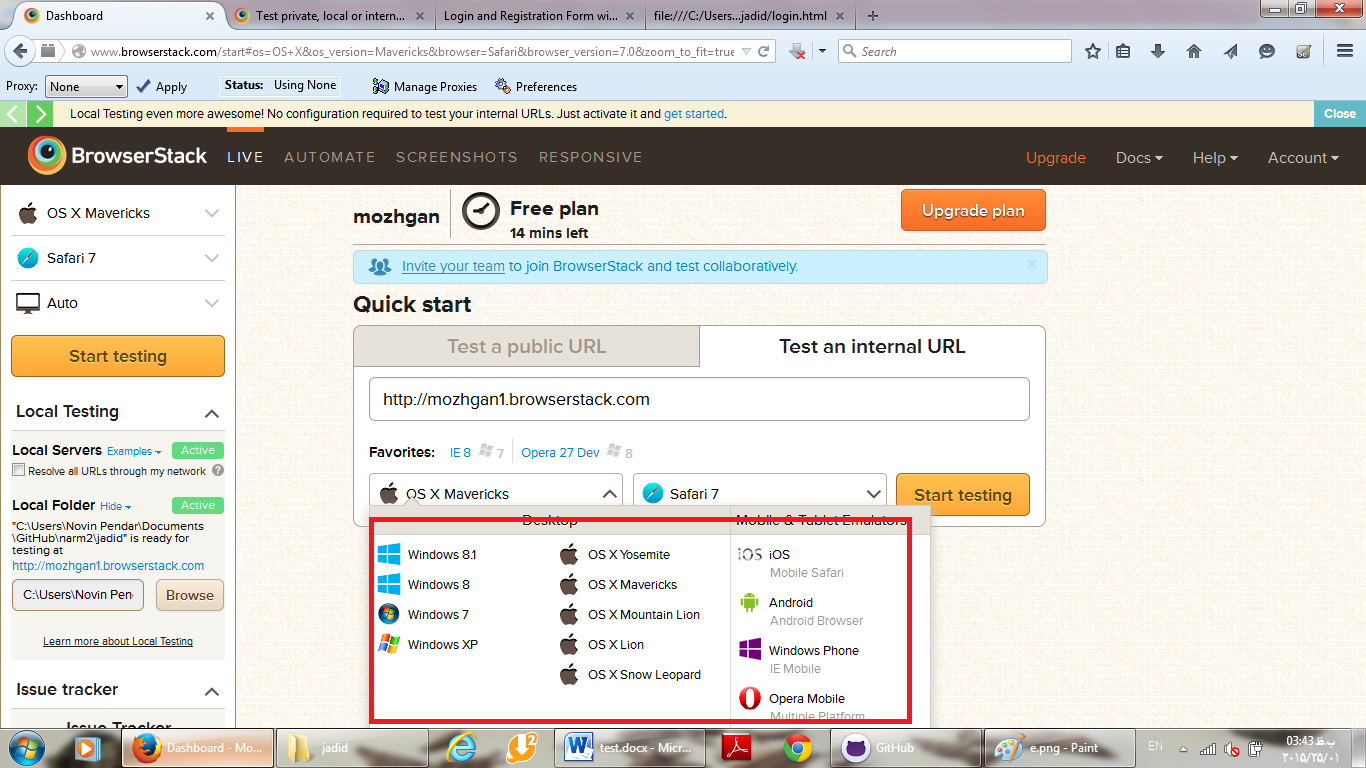
وجود دارند که به ما اجازه تست وب سایتمان با مرورگرهای مختلف بدون نیاز به نصب آنها را می دهند و خروجی یا اسکرین شات هایی از خروجی را نمایش می دهند.

از آنجا که ما نیاز به تست internal URL داریم از سایت browserstack به این منظور استفاده کردیم که امکان تست سایت را هم روی مرورگرهای مختلف و هم روی پلتفرم های مختلف به ما میدهد.

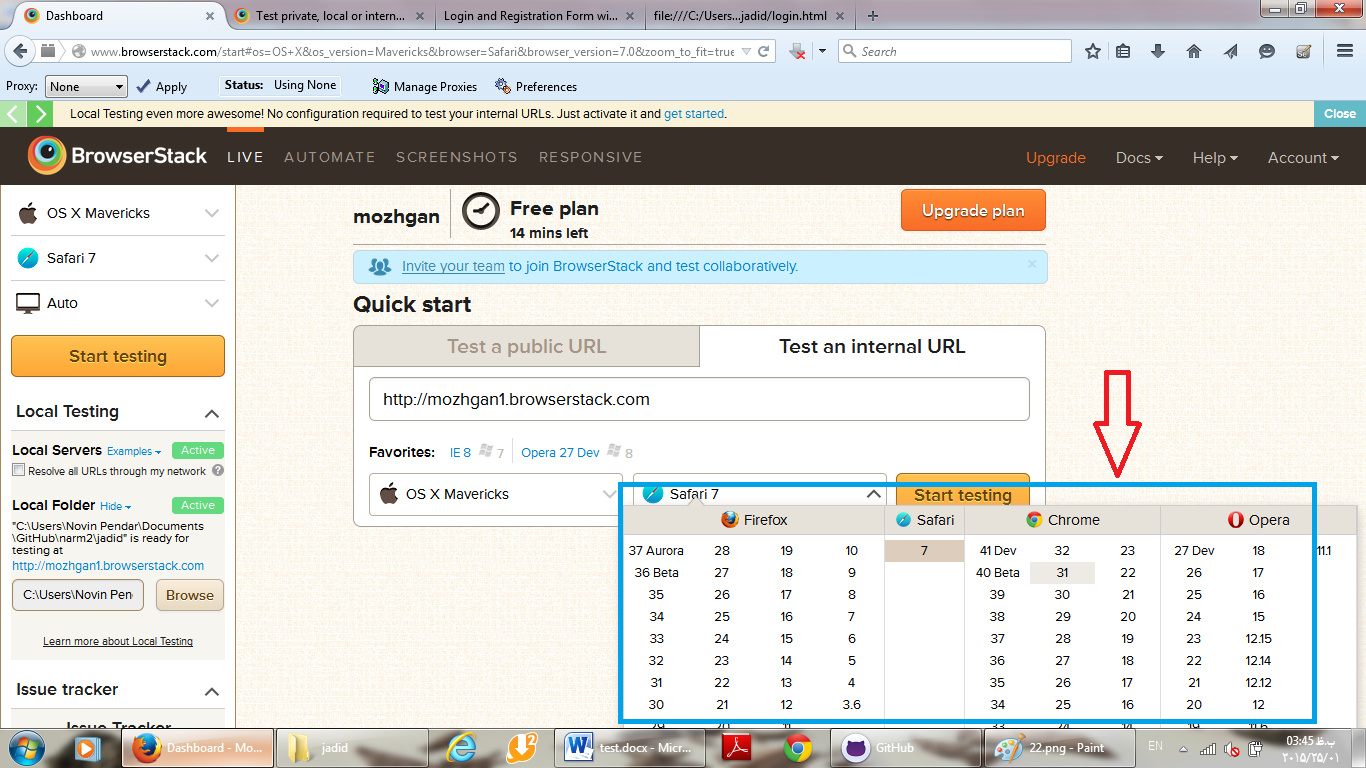
برای استفاده از این سایت بعد از لاگین کردن نیاز داریم localhost را اکتیو کنیم:



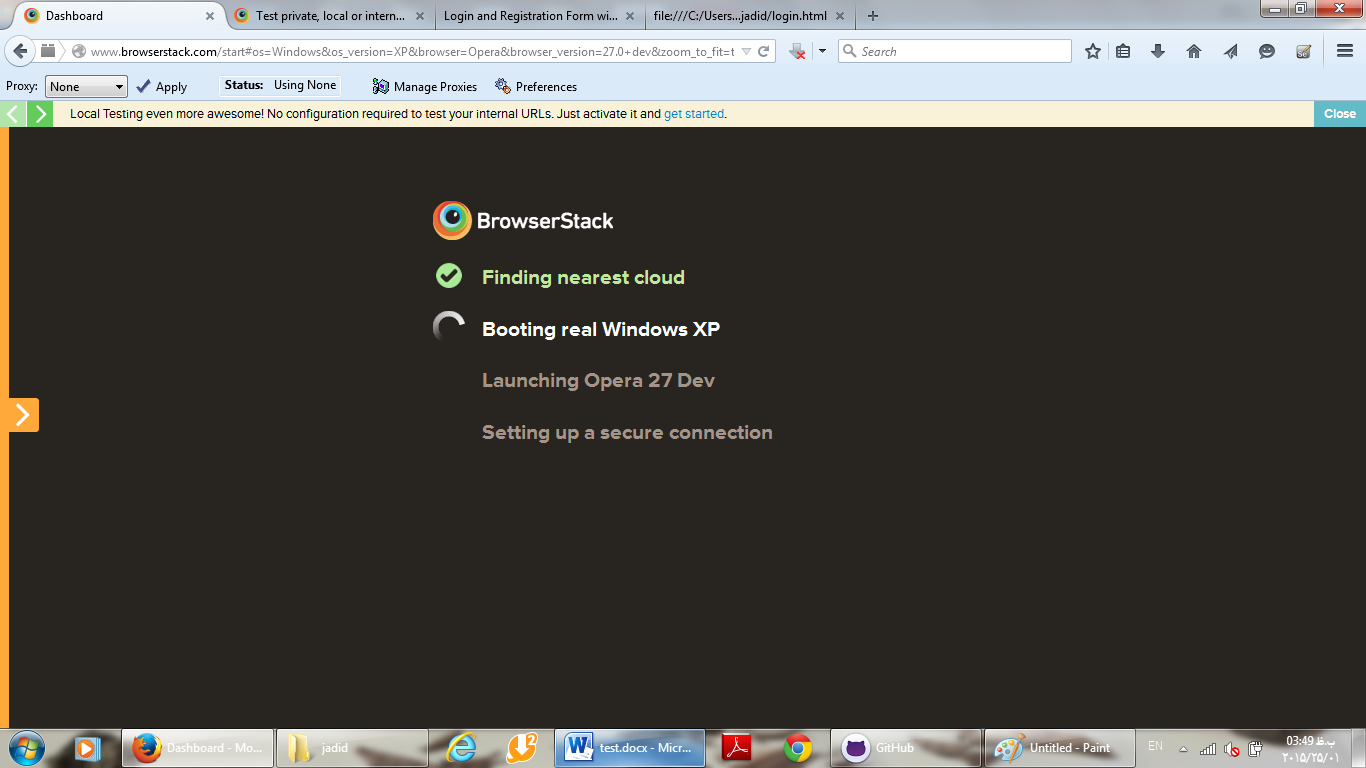
همانطور که مشاهده می شود پلتفرم های مختلفی تعریف شده:



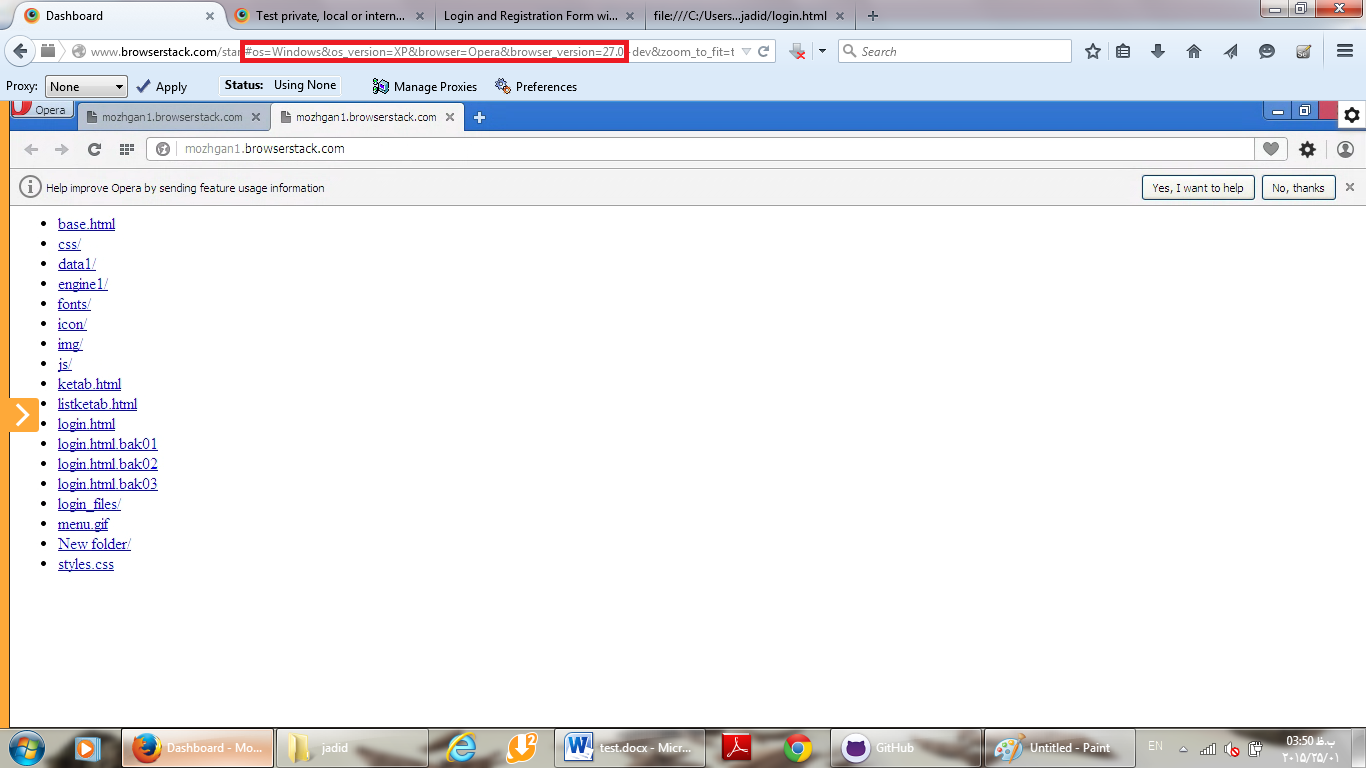
و نیز مرورگرهای متنوع(multiple browsers)



بعد از انتخاب پلتفرم و مرورگر مورد نظر:



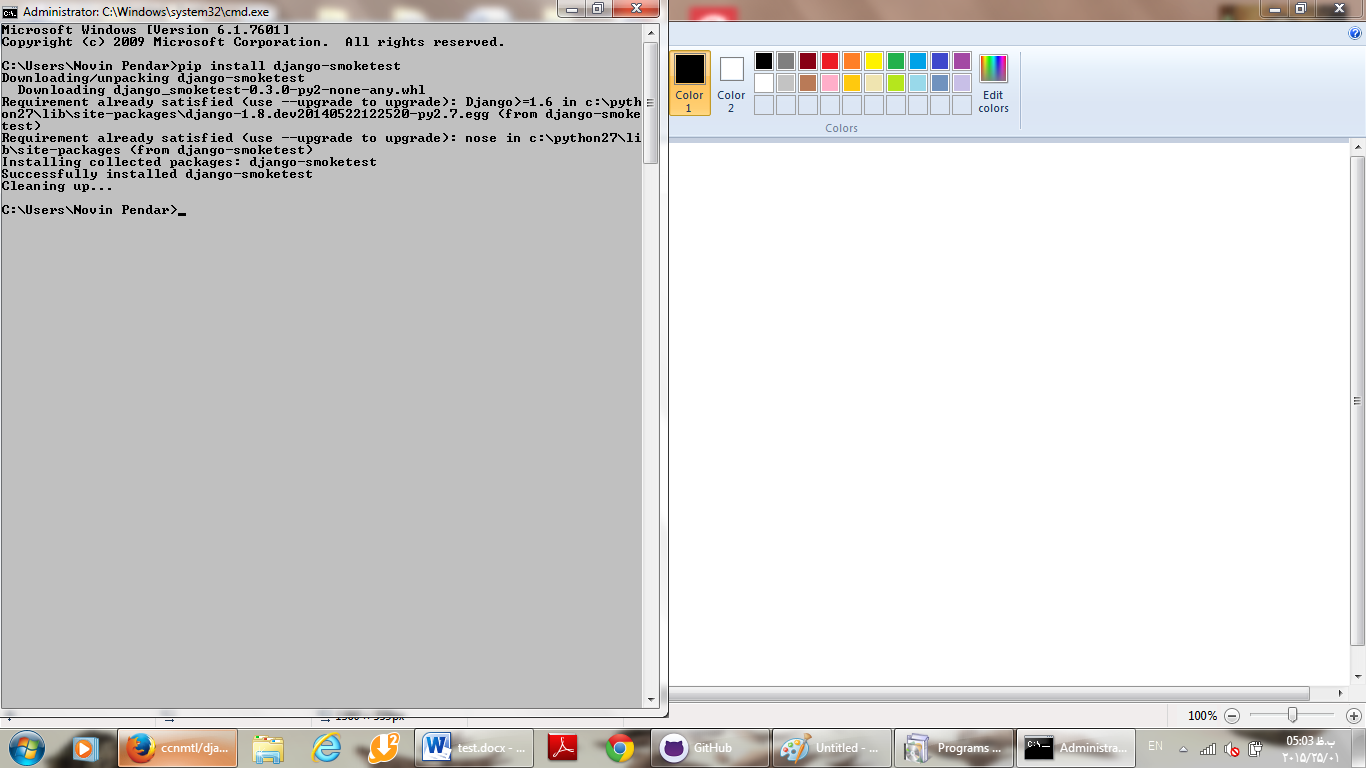
به عنوان مثال خروجی ویندوز xp و مرورگر اپرا به شکل زیر است:



**روش های مختلف تست با استراتژی جعبه سیاه:**

خروجی های سیستم را با ورودی های مختلف چک کرده و مقادیر مرزی متغیرها را بررسی کردیم.

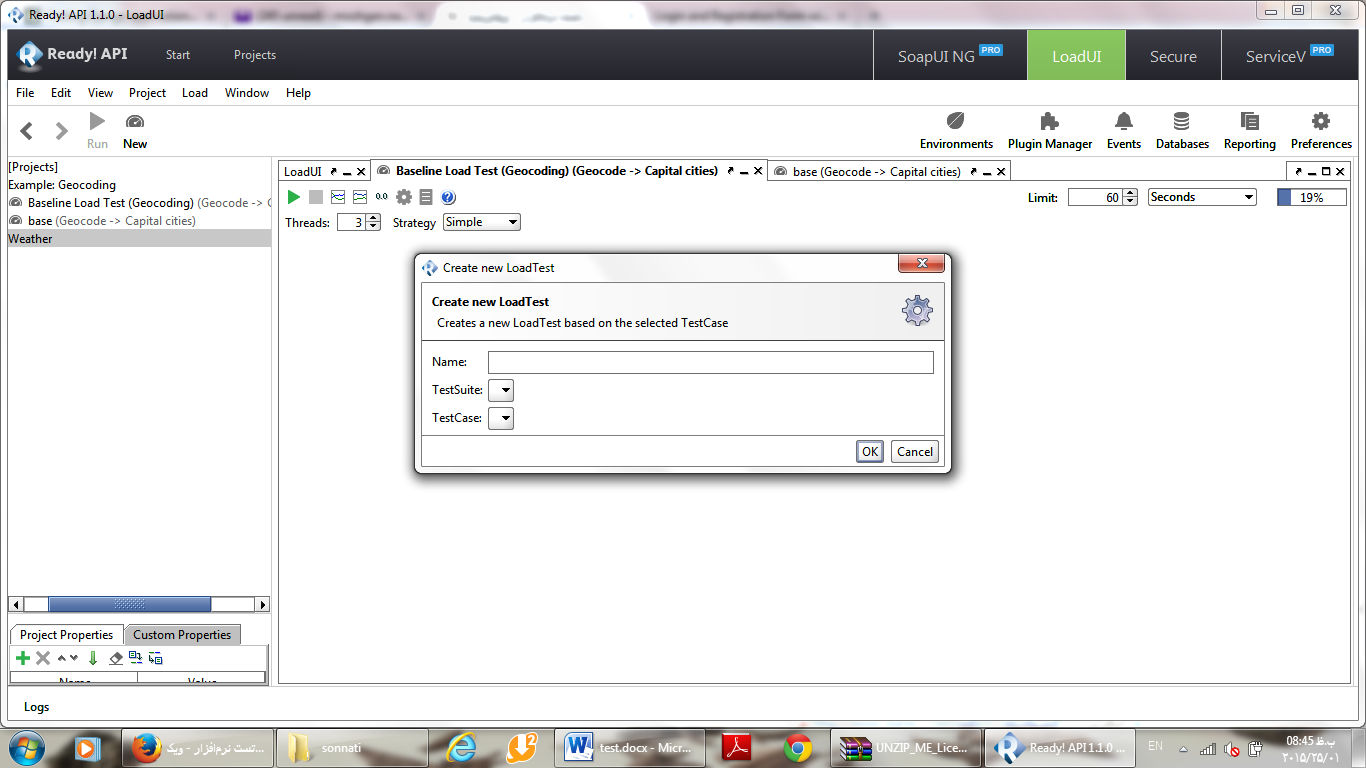
از جمله این روش ها smoke test است که برای استفاده با فریم ورک جنگو آن را نصب می کنیم و به لیست INSTALLED\_APPLICATIONS اضافه می کنیم.



و به urls.py:

('smoketest/', include('smoketest.urls'))

برای load test و stress test نرم افزارهایی نظیر SoapUI یا Neoload به کار برده می شوند:



**روش های مختلف تست با استراتژی جعبه سفید:**

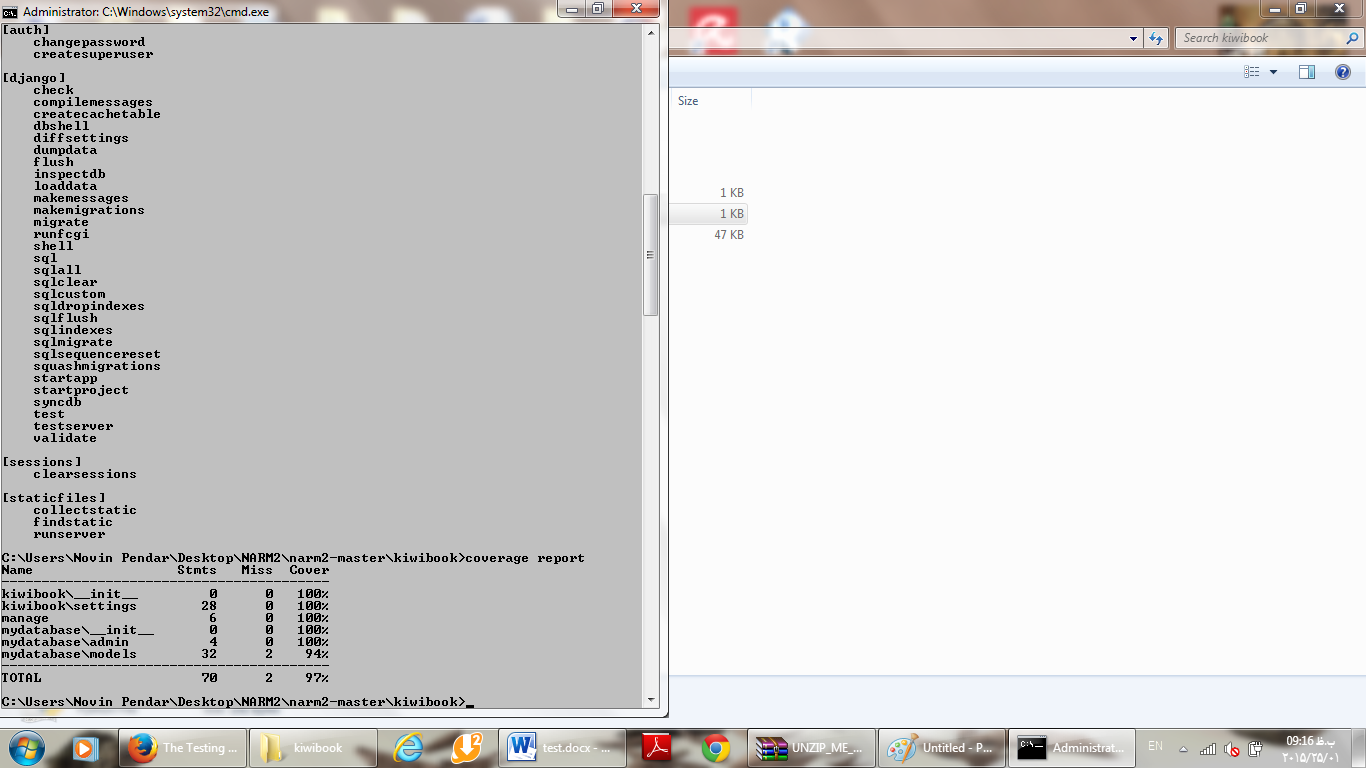
حلقه ها و انشعاب ها به دقت بررسی شده و خطاها مدیریت می شوند.

Code coverage: بخش مهمی از تست اپلیکیشن است و بیان می کند که چه مقدار از سورس کد تست شده است.جنگو ابزاری برای اندازه گیری این معیار دارد.ابتدا coverage.py را نصب کرده و کد زیر را اجرا می کنیم:

coverage run --source='.' manage.py test myapp

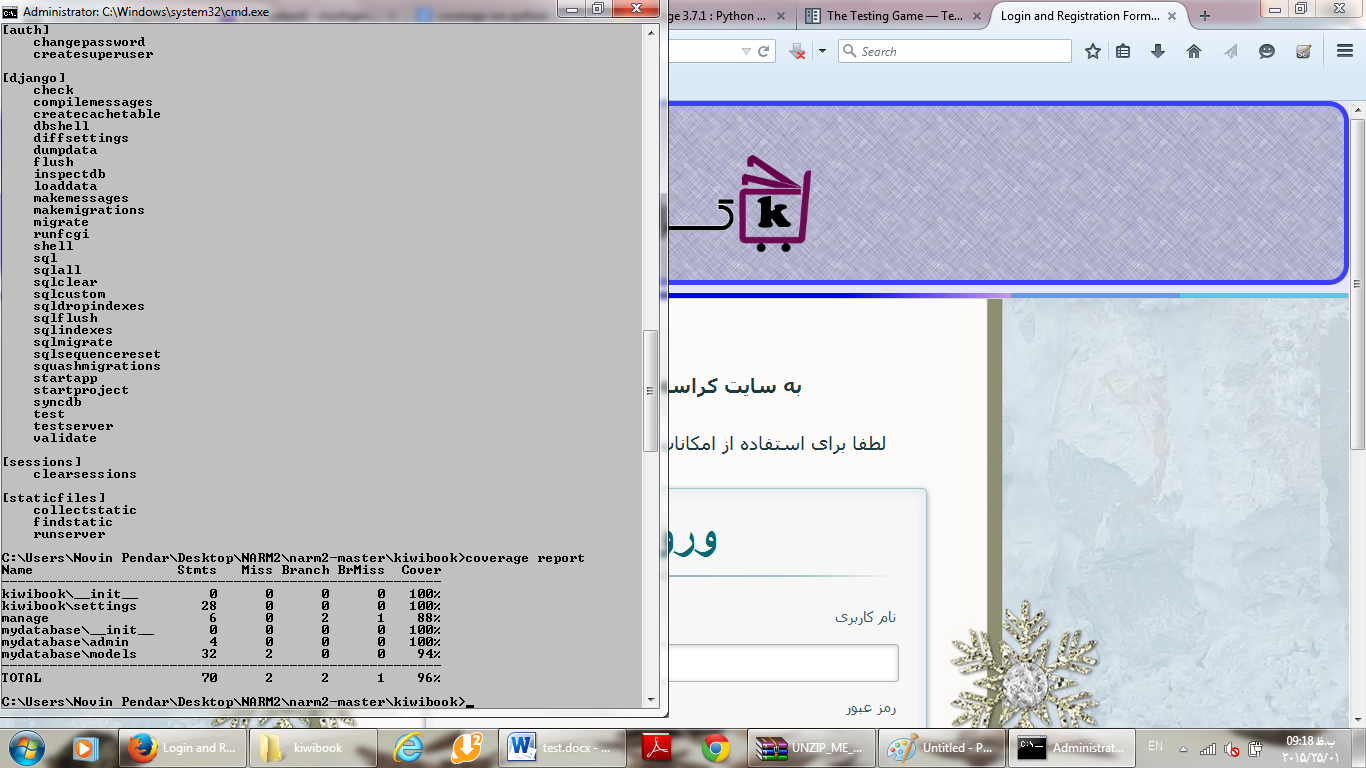
و می توانیم گزارش آن را از طریق coverage report دریافت کنیم.

یا اینکه مستقیما pip install coverage را اجرا می کنیم که نتیجه ی تست coverage در شکل زیر مشاهده می شود:



همچنین جهت تست branch coverage این دستور را تایپ می کنیم:

coverage run --branch



تست های دیگر نظیر تست پذیرش نیازمند تعامل مستقیم با کاربران و گزارش خطاها توسط آنان است.

لود در برابر زمان اجرا