Unit Test project

Using Unit test, Integration test and Acceptance test to take the features of Blogful project as:

- Add entry
- Edit entry
- View entry
- Update entry
- Delete entry
- Correct login
- Incorrect login

Examples for test cases as below.

Unit Test (test_filter.py)

```
import os
import unittest
import datetime

# Configure your app to use the testing configuration
if not "CONFIG_PATH" in os.environ:
    os.environ["CONFIG_PATH"] = "blog.config.TestingConfig"

import blog
from blog.filters import *

class FilterTests(unittest.TestCase):
    def test_date_format(self):
        # Tonight we're gonna party...
        date = datetime.date(1999, 12, 31)
        formatted = dateformat(date, "%y/%m/%d")
        self.assertEqual(formatted, "99/12/31")
```

```
if __name__ == "__main__":
    unittest.main()
```

Integration Test (test_view_integration.py)

```
import os
import unittest
from urllib.parse import urlparse

from werkzeug.security import generate_password_hash

# Configure your app to use the testing database
os.environ["CONFIG_PATH"] = "blog.config.TestingConfig"

from blog import app
from blog.database import Base, engine, session, User, Entry

class TestViews(unittest.TestCase):
    def setUp(self):
        """ Test setup """
        self.client = app.test_client()

        # Set up the tables in the database
        Base.metadata.create_all(engine)

        # Create an example user
        self.user = User(name="Alice", email="alice@example.com",
                         password=generate_password_hash("test"))
        session.add(self.user)
        session.commit()

    def tearDown(self):
        """ Test teardown """
        session.close()
        # Remove the tables and their data from the database
        Base.metadata.drop_all(engine)

    def test_add_entry(self):
```

```
            pass # put code here


if __name__ == "__main__":
    unittest.main()
```

Acceptance Test (test_view_acceptance.py)

```
import os
import unittest
import multiprocessing
import time
from urllib.parse import urlparse

from werkzeug.security import generate_password_hash
from splinter import Browser

# Configure your app to use the testing database
os.environ["CONFIG_PATH"] = "blog.config.TestingConfig"

from blog import app
from blog.database import Base, engine, session, User

class TestViews(unittest.TestCase):
    def setUp(self):
        """ Test setup """
        self.browser = Browser("phantomjs")

        # Set up the tables in the database
        Base.metadata.create_all(engine)

        # Create an example user
        self.user = User(name="Alice", email="alice@example.com",
                         password=generate_password_hash("test"))
        session.add(self.user)
        session.commit()

        self.process = multiprocessing.Process(target=app.run,
```

```
                                                kwargs={"port": 8080})
        self.process.start()
        time.sleep(1)


    def tearDown(self):
        """ Test teardown """
        # Remove the tables and their data from the database
        self.process.terminate()
        session.close()
        engine.dispose()
        Base.metadata.drop_all(engine)
        self.browser.quit()

    def test_login_correct(self):
        pass # put code here

    def test_login_incorrect(self):
        pass # put code here


if __name__ == "__main__":
    unittest.main()
```