GSOC Proposal: PyAfipWs windows automation and packaging

Organization: Python
Sub-organization: Python Argentina (PyAr)
Project: PyAfipWs

A little about me

Hello, my name is Tambe Hanslett. I am from Cameroon. After high school, I decided to pursue software development full-time. I've been freelancing for three years.
Throughout my time freelancing, I've had the opportunity to use some technologies and work on a variety of projects. Some of the python libraries I used include Pyqt, Flask, Django, Pywebview, pymysql. I do have a tech stack I usually use which is Flask, PostgresSql, Javascript, Vue.js, pywebview.
Here are my contact details

- Name: Tambe Hanslett
- Email: HanslettTheDev@gmail.com
- Phone Number: +237653877406
- Github: HanslettTheDev
- Time Zone: GMT+1 (Cameroon, Douala)

Code contribution

I opened a pull request to solve this issue "pip install dependencies in setup" to add all the module dependencies in the setup.py. Here is the link to my Pull request #112

Project Project(with PyAfipWs)

PyAfipWs is a vibrant community building unique interfaces, tools, and apps for Argentina's gov't. web services (soap, com/dll simil-ocx, pdf, dbf, xml, json, etc.). Currently, they have support using GitHub actions to automatically build installations when a pull request gets merged with the main branch. But it needs improvements in building a one-click NSIS(Nullsoft Scriptable Install System) Installer when a pull request merges. In addition, I'll work on improving the unit test coverage and packaging of the module to PyPi. And lastly, If I manage my time well, I'll happily provide a web service for testing.

Detailed description

1. **Improve the windows installation packaging with Github Actions**
   Currently, the GitHub action workflow to build the module to a windows installation package works well but it doesn't come with a one-file installer like most windows installers do. There are many one file installer tools out like sourceforge but based on the requirements from the pull request, the best option was NSIS script. Since nsis provides more flexibility, that's precisely what I will use. My Job will be to improve the workflow to automatically generate a nsis.py script which will create a one-file installer once a pull request has been merged with the main branch. This workflow also has the role to publish the newly generated installer as a new release on the GitHub repository. I plan on adding a helpful readme.md file in the workflows section to help future contributors how to improve whatever functionality is already there and have a full grasp of what is going on.

2. Writing Unit tests for 3 different issues

PyAfipWs has many sub-modules and though the overall modules have good test coverage, some of the submodules need some test coverage. The 3 different issues I'm referring to are Isuses[#107, #90, #91]. The three listed issues require test coverage for various functions. My job will be to ensure they are test cases that cover these issues and that help to catch potential bugs that may occur before it's being merged with the main branch. All of my steps to accomplish this will be well documented to give the next contributors and the overall community a full understanding of the solutions that I will provide

3. Publishing Packages for PyPI

Currently, now, It's only possible install the PyAfipWs module from the command line using pip and the github url. If the package were published, it could be used directly via pip. I plan on helping in this area by providing a GitHub workflow that will automatically upload the package on PyPi whenever there is a merged pull request on the merged branch. This will be done in collaboration with the mentors whom I'll discuss with time how this can be implemented successfully. Once the workflow is up and running, there could be a helpful wiki or README file underlying the steps to make the workflow a success to ensure a consistent pattern of design is maintained for any future changes.

Weekly timeline

Below is an approximation of how I'll organize myself to work on the GSOC(PyAfipWs) community. This timetable is subject to adjustments. Some of the deadlines and milestones will sometimes be heavily dependent on the outcome of discussions with my mentors. I've made sure to indicate what I'll be able to produce as results each week. But this is not absolute as requirements and ideas can change the scope of the milestone but certainly, there will be results each week

Week	Activities	Deliverables
Community Bonding (< May 29)	• Familiarize myself with PyAfipWs tools and build processes • Discuss with the mentors on how to accomplish the project proposals • Get final feedback on mentors on project proposals and planning executions	I'll be documenting my overall understanding of how PyAfipWs tools work and having a good understanding of how to build the tools packaged with it
Week 1 (June 1-7)	• Get hands on experience with github workflows and the underlying workflow for building windows installer	Discuss with mentors on the github workflow and perhaps documenting the workflow in detail to have a full understanding of how it works
Week 2 (June 14)	• Begin implementation of the pyinstaller github workflow	Pull Requests with initial concept
Week 3 (June 21)	• Continue improving the workflow	Pull Requests with major changes
Week 4 (June 28)	• Finish implementing the pyinstaller github workflow	Pull Requests with the new and improved workflow

Week 5 (July 5)	• Document the workflow setup process and how it works for new contributors • Discuss with mentors on the different Unit test coverages and understand the overall setup	Add some helpful documentations perhaps in the wiki or a REAME file to assist new contributors Full scope of the unit test coverage understood and taking note of where improvements can be made
Week 6 (July 12)	• Start working on Issue #107, #90, #91 on improving Unit test coverage	Pushing series of commits on the different issues and later on creating a pull request after confirming the test coverages are working well Improve documentation on the improved testcases if possible
Week 7 (July 19)	• Learn more about PyPI and how it's packaging system works • Solve the Issue #105 to get default dependencies	Pull request on a solution to the issue #105 Discuss with the mentors on how the PyPI Solution can be achieved using github workflows and any further requirements
Week 8 (July 26)	• Start working on the github workflow for deployment • Complete the PyAfipWs deployment on PyPi	Working github workflow for deploying the whole package on PyPI
Week 9 (August 2)	• Preparing documentation report of the scope of work done	Final documentation report in the mentors' preferred format.
Week 10 (August 9)	• Time slot in case of any unpredictable delay.	…
Week 11 (August 16)	• Time slot in case of any unpredictable delay.	…
Week 12 [Final Week] (August 23)	• Time slot in case of any unpredictable delay.	…

Other commitments

I'm a freelancer and I have some clients who might pop up around that period. But It won't affect my commitment to PyAr Community. Completing this proposal is a priority. So my other side projects and clients won't be a hindrance

.