Name: Tristan Ancelet
Course: SDV4116-O
Term: C202303
Section: 01

Technical Research & Development Project

For my project I will be creating a Markup Language Conversion Application. The technology that will be involved (at the start) is the following: Python and Markup Languages (ML). Python will be used to facilitate the conversion from one ML to another. The MLs are used to store data into a format that is easily transferable and parsible between different software and systems. To begin with, everything is just being used to create software that will be able to convert one ML to another. The end goal for the next iteration of this project however is to be able to setup a simple API server that would allow for one type of request to come in and be converted into another (REST/JSON -> SOAP/XML).

Using Kivy and Python I am able to create templated and functional UI that limits the ammount of actual code I need to type overall. This is due to kivy being a fully functional UI library with the option to template UI (like using XML in android studio) using Kivy's kv language. This optional language allows for templating the UI without needing to spend time creating UI elements using python code. Although creating actual python classes can extend the control the developer has on the way the UI functions and behaves. Another feature of this project would be the markup language conversion library. It is a library, written in python, that will be designed to be as extendable and pluggable as possible. This would allow the library to be used regardless of what it is interfacing with. Lastly another feature of the application would the persistant storage of the users history. This would allow them to be able to recall and recover previous states of the the conversion. Much like how git keeps tracks of changes in development projects.

Nearly all industries and software are connected to the internet in some way or form. Many of these systems use API to tranfer data between one another. These API often format data in different ways. Two of the most well know API are SOAP (which used XML) and REST (which primarily uses JSON). Each type of API has their strong points, but often integrations between one another can be difficult to implement due to the different ways that these API format data to be transfered.

Currently there are only a few features built into the project so far. Firstly the actual UI has been implemented in a rudimentary way, it is at the moment very simple and without any complex system integrations for event handling. This was done using Kivy and it's kv templating language. Which functions similarly to how android development uses XML files, which allows for widgits or views to be templated and reused throughtout the application regardless of placement. Next we have the Conversion library. This library will be modular and will be used and implemented via the Stratagey Design Pattern. This will allow it to function via regardless of the UI type it is meant to be integrate with. Lastly I have been implementing a Data Object and View Object for

each part of the application that requires data to function. This is being implemented for both the user provided data in the editor view and for persistant data stored and retrieved from the local SQLite database. As this will allow for much easier development and more understandable code.

Many things went as expected during the development of this project. Kivy and Python allowed for very quick development of UI that is cross-platform for nearly all operating systems with the exception of iOS in most cases. Additionally with python there are already libraries/modules that preform many of the simple conversions from and to most markup languages. So on that front all that is needed is to use the builtin modules and download others that I do now currently have. These will allow me to develop the code necessary for me tie all of the different parts of the project together into a useful tool.

Thankfully this early into the project I have not had too many things go wrong yet. The most trouble I've experienced is the fickle nature of Kivy templated UI using the kv language. As when templating a element/widget that will be used multiple times there can be issues that may occur. One of the primary issues happens when specifiying a background color for the templated widget without the class being extended in python code, as it will only be applied to the last object that was created in the templated view.

During the development and research of this project, the end goal of the next iteration has definately changed. As I learn more about what each type of API supports, it makes the way I will implement the API server idea. Although in respect to this iteration, not much will change. As with the utilities that python offers allows for the brunt of this project to be handled easily.

After my experiences developing this project so far I believe that this project is still very possible. As the technology around my project is both very modular and can be implemented reliably using OOP. This because Python allows for a great degree of freedom when designing software, as it is almost universally supported across all operating systems and can even be compiled for extra speed and resource efficency with the help of projects like PyPy or CPython.

References

- GeeksforGeeks. (2022, August 29). Difference between REST API and SOAP API. https://www.geeksforgeeks.org/difference-between-rest-api-and-soap-api/
- Wikipedia contributors. (2023, February 21). Markup language. Wikipedia. https://en.wikipedia.org/wiki/Markup_language
- Kv language — Kivy 2.1.0 documentation. (n.d.). https://kivy.org/doc/stable/guide/lang.html