International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

Making Cross Platform Web Applications using Electron Js and Python

Dr. Sonakshi Vij[1], Akash Pandey[2], Harshit Jain[3], Himanshu Tyagi[4], Harshita Sinha[5]

[1]Assistant Professor, Department of Computer Science and Engineering, Krishna Engineering College, Ghaziabad, Uttar Pradesh, India
[2,3,4,5] Final Year Students, Department of Computer Science and Engineering, Krishna Engineering College, Ghaziabad, Uttar Pradesh, India

---***---

Abstract - *Electron is used to make cross platform web applications using web frameworks or languages as javascript, HTML or CSS. Shifting from tkinter to electron was a very good idea for this project. We made a fully deployable electron application that is running our face authentication software fully written on python initially. Generally electron uses Node.js to communicate from the backend server, so here python3 is used as the backend for communication. This makes the User Interface of this application more appealing and intuitive without reducing any functionality of the system.*

***Key Words*: Electron-Python Web Application, Face Authentication, ZerorPc, Electron GUI**

1. INTRODUCTION

Electron as a framework is a new concept for making desktop applications programmed on web technologies. Many Web apps shifted their codebase to electron with added functionalities and the ease of coding in web frameworks.

It's easier to maintain and the main codebase is used and fully deployed as a desktop application used by any operating system. Electron with python as a backend is an innovative idea of deploying python applications as web apps running on a flask server or an RPC That can communicate between the python code running on Web server with the server's backend in node JS. We found Zerorpc as the best RPC in this use case as flask is a heavy and resource hungry. In our project the old tkinter GUI looked very less appealing and basic. The problems we had with the previous UI and Application are:

- Issues with deployment
- UI Looked Non intuitive
- Device must have python and all the other libraries installed in the computer to use the application.
- Tkinter is not as progressive as electron.
- Needs a separate package manager and IDE
- Lacks the full flexibility of a website application

For this cross communication between python and electron we actually need an inter-process communication (IPC) mechanism.

It is unavoidable unless Python and JavaScript have direct foreign function interface for each other. HTTP is a very popular way to establish inter process communication, also we have various options like we can use sockets of communication. Then, for which, we want an abstract our messaging layer that could be implemented with zeroMQ which is one of the best messaging library. We need to establish a way to synthesize raw data that can be used by zerorpc to communicate between backend and frontend. that is an RPC used for server communication. The best part is our RPC Zeror supports both node.js and python which suits our use case.

2. PROPOSED SYSTEM

The best feature about using Electron as a GUI for Python is that you get to use cutting edge web tech. also you don't have to learn old , not well maintained GUI toolkits, and the best part is that you can use web languages and frameworks to code your frontend with the added functionalities of native javascript and itd frameworks. The backend code written in python needs to be connected to a server running the code either on flask or any existing RPC software. Zerorpc is Open source FOSS RPC project used for communicating between server and the application. We found out zeror to be more effective than eel and less resource hungry.

```
start
 |
 V
+------------+
|            | start
|            +-------------> +-------------------+
|  electron  | sub process   |                   |
|            |               | python web server |
| (basically |     http      |                   |
|  browser)  | <-----------> | (business logic)  |
|            | communication |                   |
|            |               | (all html/css/js) |
|            |               |                   |
+------------+               +-------------------+
```

Fig 1.1: Architecture of the system

There are 4 major components in the whole architecture of this system

2.1 MAIN PROCESS

The purpose of main process in electron is to create and manage the windows in the browser of the application running as a block. Every browser window individually run and have its own UI or process.

The backdrop and functionality both of using the main process in electron is that there can only be one main process in the block at any given time in the browser window.

2.2 RENDERER PROCESS 1 (THAT IS VISIBLE)

The main process creates different browser windows that divides into different renderer processes in the system, Or in easy words each browser window is a renderer process in itself running individually. Its main purpose is to render or show the webpage in the application window. Its main functionality is to render the graphics ui in real time.

2.3 RENDERER PROCESS 2 (THAT IS HIDDEN)

This is a tab or browser window that is hidden or abstracted from the user using the service. Main function of this layer is to block the incoming threads and it does not render the User Interface of the application.

To basically achieve the thread blocking in the window we will simply create a hidden window, complete the given task under it which is running in the background. Then when the task gets completed we stop the hidden process and release the memory without causing any difficulty to the main process. The idea is to make and release memory of a hidden window in the application progressively.

2.4 PYTHON SCRIPT

We make a Node package manager package to run the application using python, which is the python-shell. Using the python shell allows us to run external scripts written in python within our node js backend application. As the node.js features are allowed inside the electron app so we can effectively communicate using python scripts from the node package manager.

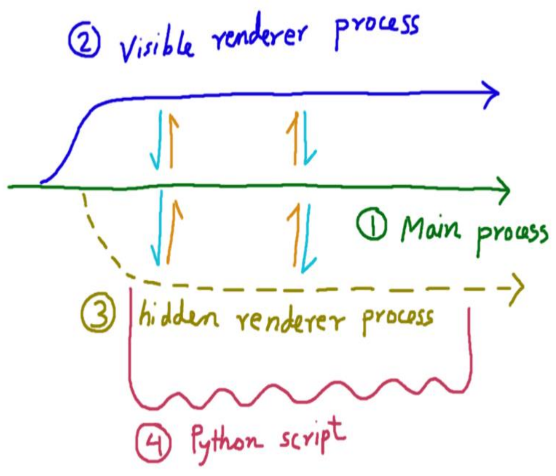

Fig 2.1: Components of Application

2.5 COMMUNICATION:

This part involves the communication between all these 4 components. This involves socket communications between the hidden render process, main render process and the main process layer where the python codebase communicates using python shell scripts to establish a connection to the running server and effectively communicate with the backend server. This communication involves the use of the existing RPC Software and is very efficient to use in this technique. This cross communication between python and node js using javascript is a new technology and still some communication gaps which needs to be resolved.

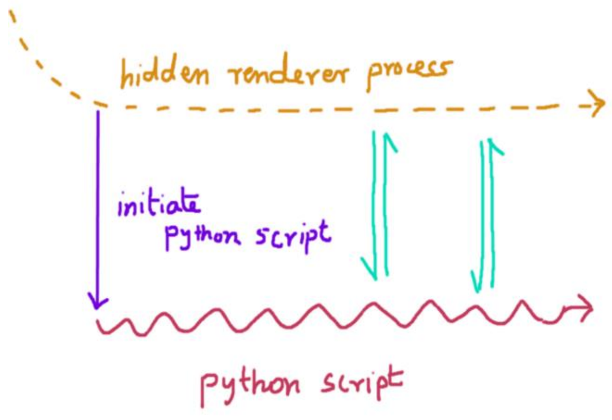

Fig 2.2: Communication Between layers

3. ANALYSIS & COMPARISONS

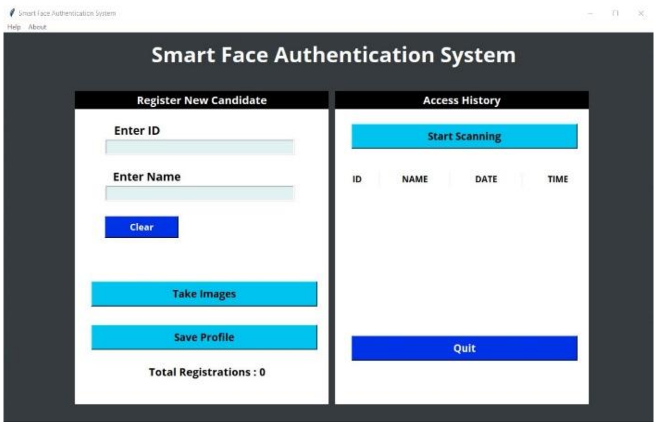

Fig 3.1 : Previous GUI Made using Tkinter [Python]

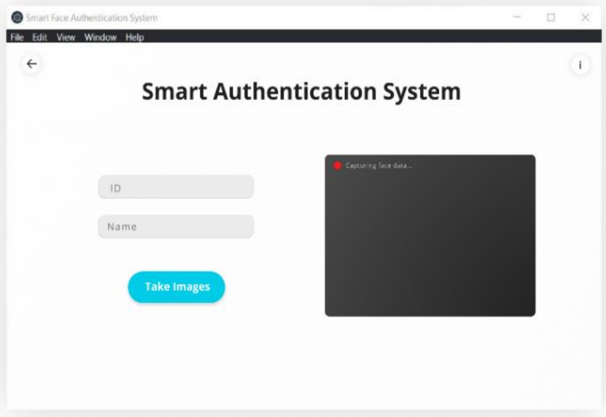
Smart Face Authentication System
File Edit View Window Help
Smart Authentication System
ID
Name
Take Images
Capturing face data...

Fig 3.2: Revised Application made on Electron

This process lets us not only make a very nice Graphical interface but also gives us the robustness of the electron and flexibility of web applications using javascript frameworks.

- Native menus and notifications
- Packaged installers, automatic updates to your application
- the large community and ecosystem of Electron
- debugging and profiling is easier
- Google and thousands of developers keep working on improving this Electron Toolkit each and every day to give the best version possible with minimum amount of bugs.
- This makes us able to use multiple Javascript plugins and frameworks or JS UI widgets and controls e.g. JQuery, Charts, Frameworks and the possibilities of using electron are limitless.
- The full functionality of NodeJs and its large Package repository is a addon.
- ES6 syntax that is a clean javascript syntax with classes, no semicolons and modules can be used.

Electron browser window with python in backend is a piece of code where we use Electron (nodejs with chromium (browser)) as a GUI communicating with Python as the backend via IPC connectors as Eel or zerorpc that is very capable and let us use many functionalities of the web frameworks with the robustness of python as the backend and the zeror lets the seamless transfer of data between all these layers. Also the system may or may not have chrome it doesn't matter as it comes in packaged with the electron itself.

4. CONCLUSION

This paper presents an effective solution to build deployable electron-python web apps which run on different operating systems and web and is robust and more effective than tkinter. The improvements in working of the web app and the redesigned UI makes it a sure shot solution above tkinter. The flexibility of using any web framework to design beautiful looking Graphical user interfaces and the backend process using python makes it very versatile.

REFERENCES

[1] https://medium.com/heuristics/electron-react-python-part-2-architecture-d49634521efd

[2] https://medium.com/@abulka/electron-python-4e8c807bfa5e

[3] https://efficientcoder.net/connect-python-3-electron-nodejs-build-desktop-apps/

[4] https://www.skcript.com/svr/how-to-execute-python-scripts-in-electron-and-nodejs/

[5] https://Zerorpc.io

[6] https://www.fyears.org/2017/02/electron-as-gui-of-python-apps.html