**TOOLS & TECHNOLOGIES**

*Software / Server Backend-Frontend*

**Design Considerations**

There were a plethora of options and path we could take in order to achieve connectivity with the Hardware and store/manipulate/present the data in a user-friendly way for the prototype.

However, due to time constraint and limitations for the prototype, we had to make an informed decision taking those two factors into account.

When it comes to Software / Backend Server and Frontend client, here were the 3 short listed candidates:

*- ASP.NET Core / Blazor Web Server with Microsoft SQL Database (Full-stack)*

*Xamarin framework for cross-platform mobile client app*

*Language used:* ***C#***

This is a great enterprise-grade platform tied with the Microsoft ecosystem.

The C# language also has a lot of great libraries for this type of project.

Leveraging cloud services hosting like Azure App Service and security like Single Sign-On with the MS 365 platform, it would’ve been a very seamless integration with our product, and used by many corporations across the world.

However this is not a free nor cost-effective platform and caters for bigger deployments and corporations that have already made profit from their products.

- *Node.JS with PostgreSQL (Backend) and Angular (Frontend)*

*React Native for the cross-platform mobile client app*

Language used: **JavaScript**

This combination is the most popular/cost-effective choice for a fast backend/frontend server solution and web development.

All these frameworks use JavaScript, and integrates very well from one to another.

From a simple Google search to surveys on website such as StackOverflow or Github, they are amongt the top 5 choices for Frontend and Backend development.

However there is a small learning curve if not familiar with JavaScript.

The cost here is non-existent as these platforms are free to use/develop.

*- Django with MySQL (Backend) and Vue.JS (Frontend)*

*Flutter for cross-platform mobile client app*

*Language used:* ***Python****,* ***HTML/Javascript****,* ***Dart***

Going this route mixes a lot of different platforms that are known to be very fast to develop/prototype on.

According to Statisticsanddata.org - March 2021, Django is the second most popular Backend framework after Laravel.

It is a very intuitive, fast and secure backend platform that leverages the world’s most popular/used language Python.

Being popular, Python also has an incredible amount of libraries that include AI and Machine Learning.

Although Django offers full-stack posibilities with its templates frontend, it is not the most popular/eye-candy of the frontends.

Vue.JS was picked as it is a very fast/modern Javascript platform to work on with plenty of templates to build upon.

It is also a very popular choice among Frontend developers, and it is praised for its ease of use/features and agility.

Static websites made from this framework are very lean and perform very well in all browsers.

Flutter platform complements this group well as it is aimed at fast development of cross-platform mobile apps with great and beautiful frontend capabilities. Made by Google, it is a fairly new player but is already been used by big corporations such as Google themselves, eBay, BMW and Alibaba group (AliExpress).

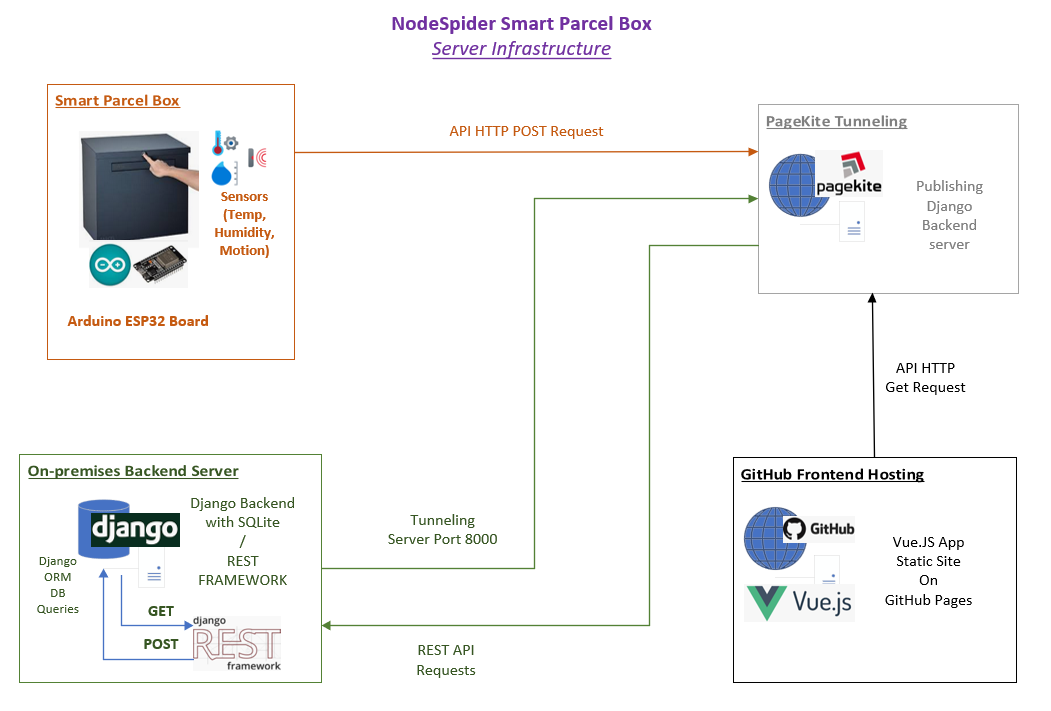
It uses the Dart language which is syntatically very similar to C++ and is focussed on being client-optimized with native performance on any platform.

This is also very cost-effective

**High level design**

For our Smart Parcel project, we have decided to go with Django/Vue.JS/Flutter combo as we needed to use modern and agile platforms that could prototype our ideas really quickly, which is perfect for this assignment.

It also gives a bit of challenge for the developer (Richard) as it uses 3 different programming languages.



Django backend server handles all of the queries/data manipulation with its SQLite database. REST APIs have been implemented so the Arduino board can send all of the data from its sensors.

We’ve set up 3 data models, which are Temperature (degrees)/Humidity (percentage)/Motion(true/false triggers). They have been serialized to be used as APIs and turned into endpoints.

Token based authentication has also been added for increased security, so only the board loaded with the Token can do a POST request.

For the purpose of this assignment, and as we do not have any funds for this project, the Django server is hosted on one of Richard’s computer, and using Pagekite.me service to broadcast the server online.

Pagekite is a tunneling forwarder solution which “publishes” a localhost server port to the web. It is free to use for 30 days but also has a small fee of $3 per month to keep the service active.

For the frontend, we are publishing our Vue.JS landing page on Github Pages.

We could’ve leveraged Django’s full stack approach and use Django templates to design the frontend, however it doesn’t have as many resources as Javascript based frameworks and lacks flair. This would also mean that the frontend is accessed through PageKite’s URL, which is not desirable.

As we also have to publish a group website for this assignment, doing our frontend through Vue.JS would serve two purpose:

- Present the data sent from the Arduino as a proof of concept to showcase our Smart Parcel

- Introduce our Group/Team and the Assignment’s goals in a modern/sleek web page

Vue.JS supports API GET requests using its Axios module, and can effectively retrieve the desired data from Django backend and present it on the static page.

The page is then published on Github Pages using a Shell script for Git Deployment.

The cross-platform app made on Flutter follows the same logic as the frontend website:

On launch it does a connectivity check to make sure the mobile phone is connected to the internet, then it proceed to connect to Django backend URL and starts doing the GET request to retrieve the desired data and display on the app.

This type of set up means that we are seperating the backend server from the static frontend, which helps with security and also helps us leverage the free GitHub pages hosting platform in order to display our Group Website with live data from our prototype as a 2-in-1 solution.

**Things that can be achieved past the Prototype stage**

If we were to get sufficient funding/interest in the project to go past the prototype stage, the first thing we could do is to look at Cloud hosting for our backend/database in a service like Google Firebase-App Engine/Amazon AWS Cloud/Azure App Service so we wouldn’t need to worry about maintaining an on-premises solution.

Having that type of service also means that we could have a subscription model where we could charge our users for data retention, or additional features with the data (Eg. save 12 months worth of data/webcam footage from the motion detections)

A full log in system should be implemented so Smart Parcel users can have their own page that links to their Smart Parcel boxes.

As we would have more sensors on the Arduino boards, we could leverage Python’s incredible AI/Machine Learning libraries in order to build prediction models on the Temperature/Humidity and seeing trends on the amount of mail/parcels you are getting for that time of the year.

Powerful graphs could be generated/built from that data on data analytics platforms like Microsoft PowerBI or Tableau.

This would get more insights on the data coming from the Smart Parcel.