DEVELOPMENT OF A WEB PLATFORM FOR AIR QUALITY MONITORING

R. Sta. Ana[1 *], K. A. Panlilio[1], R.V. Ramos[1,2], R. A. B. Torres[1], B.A. B. Recto[1]

[1] Training Center for Applied Geodesy and Photogrammetry, University of the Philippines, Diliman, Quezon City, Philippines – rcstaana1@alum.up.edu.ph, kppanlilio@gmail.com, rvramos@up.edu.ph, rbtorres@alum.up.edu.ph, bearecto15@gmail.com

[2] Department of Geodetic Engineering, University of the Philippines, Diliman, Quezon City, Philippines

KEYWORDS: Air Quality, WebGIS, Dashboard, Automation, Web Development

ABSTRACT:

Access to information on the quality of the air we breathe is important as it has a huge impact on our health. Also, with the recent Covid pandemic, maintaining our respiratory health has been more important than ever. With knowledge on the air quality in our surroundings, we can make better-informed decisions concerning our health. While geospatial datasets are available, they are often not in an easily digestible format. The tools commonly used in the analysis of these datasets usually require technical knowledge, which limits their accessibility. The study hopes to address these issues by scoping currently available technologies to develop a framework for consolidating the datasets into a single data hub, and presenting the data in way that's easily understandable without requiring a technical background. Easy-to-use processing and automation tools are also being developed to make performing analysis on the datasets simpler and more accessible. The work done in this project is also intended to serve as a template for future studies in similar fields.

1. INTRODUCTION

The study acknowledges the existence of other available sources online where we can see the Air Quality Indices in Metro Manila. Examples include the NCR-EMB's dashboard (NCR-EMB, 2023) and IQAir's dashboard (IQAir, 2023). This study aims to develop a similar dashboard that will not compete with, but rather supplement the existing dashboards with tools and information not currently available on existing platforms. This includes remotely sensed datasets, simulation model outputs, data from available low-cost sensors, among others. Certain GIS capabilities will also be included in the dashboard which researchers may utilize to perform spatial analyses and generate reports.

The consolidation of datasets from various sources can provide a broader view of the air quality in the area. Data from low-cost sensors on the other hand, can be used in analysis at a smaller-scale for a more localized and detailed evaluation of air quality. Integration of remotely sensed data can enhance the accuracy of the air quality models and can provide valuable insights which may not have been possible otherwise. The simulation models can be used in forecasting and in analyzing air quality trends.

The GIS capabilities included in the dashboard are designed to provide the user a way to perform simple analysis, and to export their findings in an easily comprehensible format such as charts, and maps. This feature is intended to assist the user, especially researchers and policymakers, in data-driven decision-making.

The dashboard was designed with these considerations in mind, aiming to provide a platform accessible to users of varying backgrounds, such as regular citizens, researchers, and policymakers.

2. METHODOLOGY

In developing the web platform, the workflows and products are divided into three main components: (1) the frontend user interface, (2) the backend and database management, and (3) the automation and processing tools.

2.1 Frontend

The frontend user interface was designed to be intuitive and user-friendly in order to make the dashboard more accessible to a broader audience. The users will include people of diverse backgrounds and so the dashboard must be able to cater to varying levels of technical expertise. A design that is simple to navigate also encourages people to use the platform more often. It can be split into 3 main pages, the Landing Page, Data Hub, and Map.

2.1.1 Landing Page: The landing page (in Figure 1) is where the user will be initially directed when they access the website.

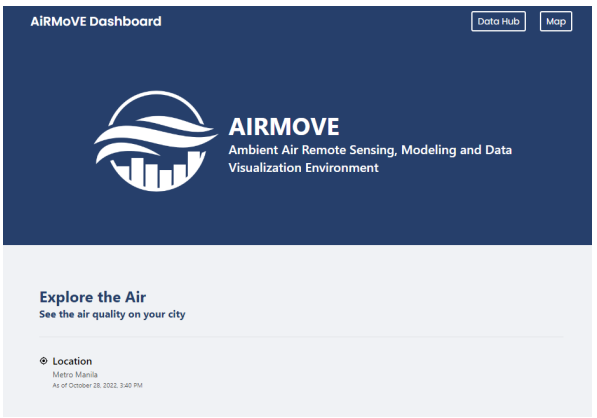

Figure 1. Dashboard Landing Page

* Corresponding author

This page introduces AiRMoVE, the Ambient Air Remote Sensing, Modeling and Visualization Environment (AiRMoVE) project developing the dashboard, and provides general information about air quality indices, which users unfamiliar with the topic may need to know to more effectively utilize the dashboard. Ways to contact the project are also provided on this page.

2.1.2 Data Hub: The data hub page provides users with download links to non-spatial products of the project. This includes toolkits, technical documentations and manuals on their use, and publication materials.

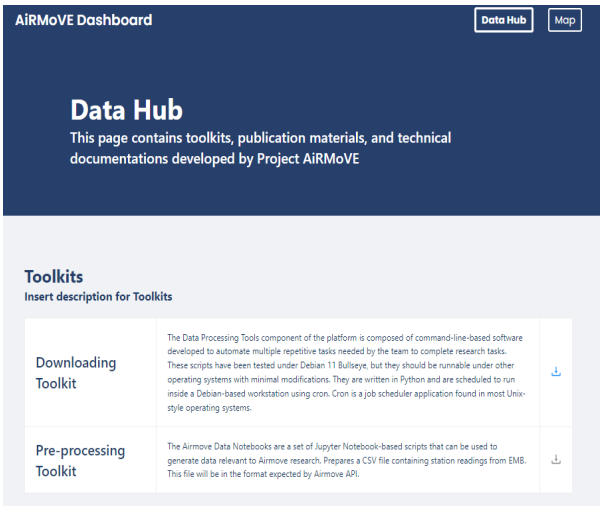

Figure 2. Data Hub Page

The toolkits currently include an automated downloader of datasets used by the project in their research, and another one for processing station readings into a format which can be easily imported and displayed in the dashboard. The automated downloader is used to minimize the time spent on repetitive tasks, which can be spent instead more meaningfully on conducting analysis or research. It also consolidates the datasets into a single storage making accessing and managing them more organized. The tool downloads satellite imagery from various sources which makes it also useful for other projects making use of similar datasets, and for users who want to conduct their own analysis. The processing tool is mainly developed for managing the datasets displayed on the map and so its intended users would be the system administrators of the dashboard, and other administrators using a similar setup.

Included also in the data hub page are the technical documentations on the design and development process of the dashboard components, and manuals on the use of the toolkits. The documentations are provided as a guide for other projects who wish to develop a similar dashboard, while the manuals enable users to fully access and utilize the tools and features.

Lastly, the publication materials of the project (social media posts, infographics, etc) are compiled and are displayed in the data hub page.

2.1.3 Map: The map page (in Figure 3) is mainly for visualizing the spatial datasets compiled and produced by the project, and for allowing users to download these datasets for their own use. The page consists of a map visualization area, and a sidebar for selecting what datasets to display or download.

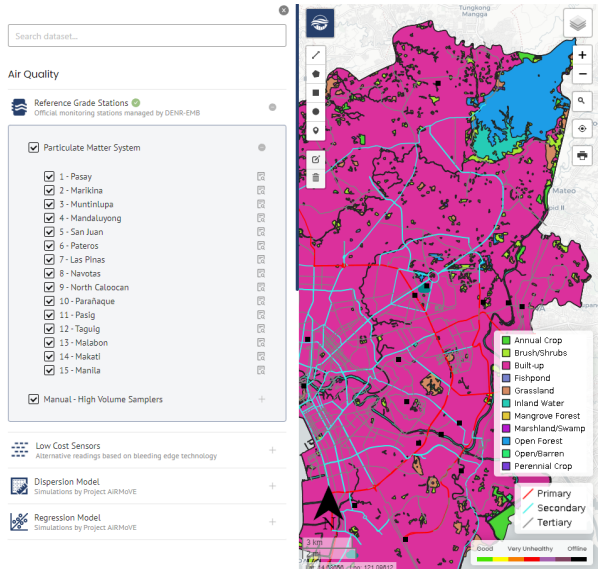

Figure 3. Map Page with Sidebar

The map visualization area also provides drawing tools (in Figure 4) for annotating the display in the creation of maps. These drawing tools can also be used to do basic spatial analysis, such as getting the length of a path or area of a polygon. The annotated maps can then be printed if needed.

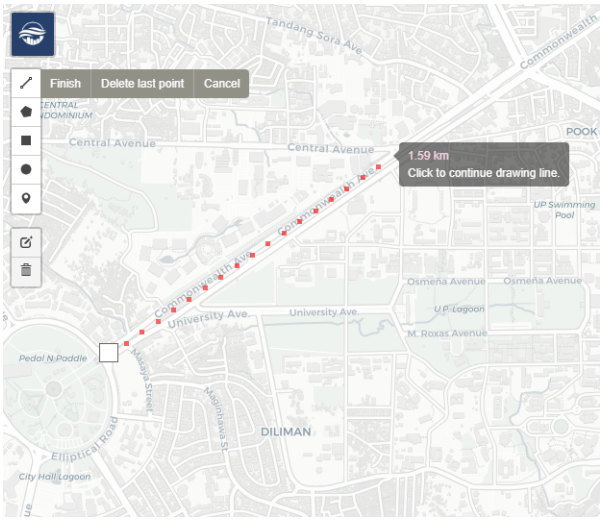

Figure 4. Drawing and Measurment tools

Typical map controls such as zoom, and pan are also available for navigating the map display.

The sidebar, shown in Figure 5, contains a list of the available datasets which can be toggled on or off to select which layers to display on the map. A search bar on top is also available when looking for a specific layer. Downloadable files will also have an icon beside their name which the user can click. This will then open another sidebar where the user can filter which datasets to download. The

user can also choose to display here tables or charts of the filtered datasets showing its statistics (average, min, max, standard deviation). Both tables and charts are downloadable by the user with the tables being exported in .csv and the charts in .png format.

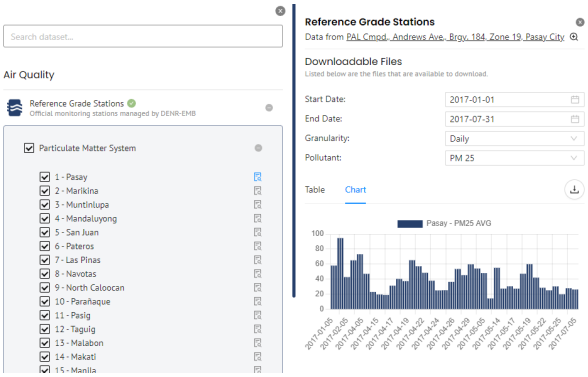

Figure 5. Sidebar with filtered data and chart

The initial design phase was conducted in Figma (Figma, 2023). Users were then asked to test the web page and provide feedback on its features, and design. Useful comments were incorporated into the design and the process was iterated until it was finalized. The frontend component was then built using ReactJS (Meta, 2023).

2.2 Backend

2.2.1 API Server: The second component is the backend which also includes database management. The API was built using Django (Django Software Foundation, 2023) with the Django REST framework plugin, which allows developers to quickly create REST endpoints. (Django REST, 2023). The API handles the communication and transfer of data between the frontend and the database. An admin account was also created which can be used to manage the data in the database remotely. Here the admin can add or remove entries from the database, and also generate statistical reports (min, max, mean, standard deviation) on the datasets available. The user interface is shown in Figure 6.

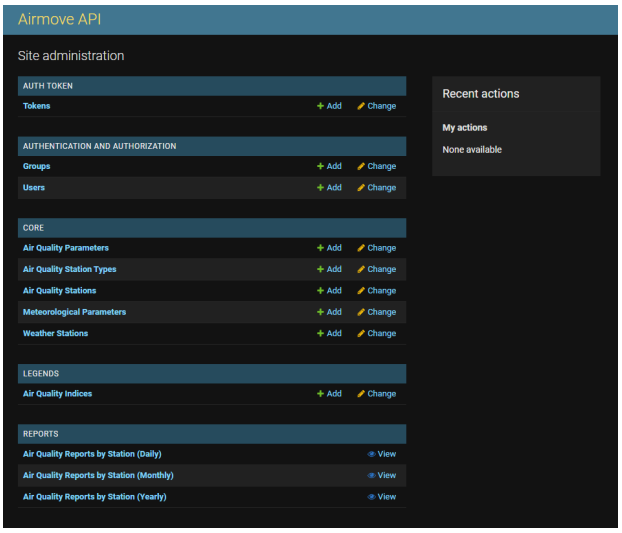

Figure 6. Web-based Admin Interface

The API provides 7 endpoints (in Figure 7) each returning a ".json" response which is then read or displayed in the frontend.

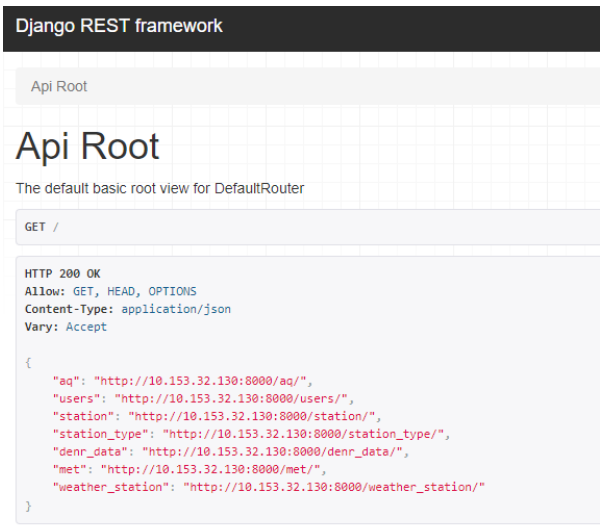

Figure 7. API Endpoints

Incoming requests from clients are handled by Nginx (Nginx, 2023), which serves as a reverse proxy server, hiding the internal structure of the application for added security. The requests are forwarded to a uWSGI server (Unbit, 2023) which processes these requests and communicates them to the Django application. The responses from the application are then sent back to the uWSGI server which in turn sends them back to Nginx. Nginx also handles serving static files, such as CSS, and Javascript, offloading this task from the uWSGI server to improve performance. The typical flow of requests are shown in Figure 8.

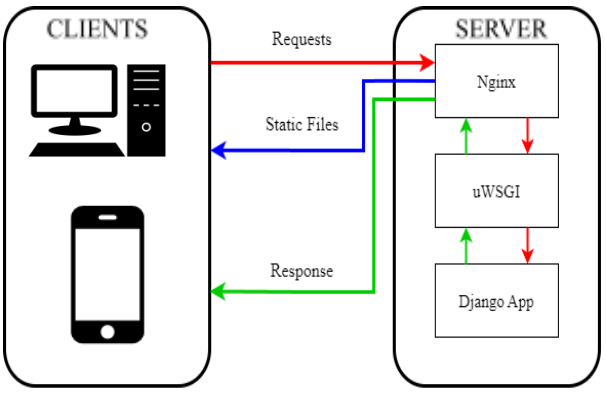

Figure 8. Typical flow of requests in the server

2.2.2 Database and Geoserver: The database is based on PostgreSQL (PostgreSQL Global Development Group, 2023) along with its PostGIS extension. Datasets which can be in a table format are stored here. This includes air quality parameters, meteorological parameters, and locations of air quality and weather sensors. Automatically generated outputs are also stored here, such as statistical reports on air quality.

For other geospatial datasets like rasters and vectors, a GeoServer (Open Source Geospatial Foundation, 2023) was set up to manage them. As it is expected that more datasets

are going to be added in the future, the importer extension for geoserver was also included. This allows batch uploading of datasets which makes future uploads easier and minimizes the need for manual work. Its interface is shown in Figure 9.

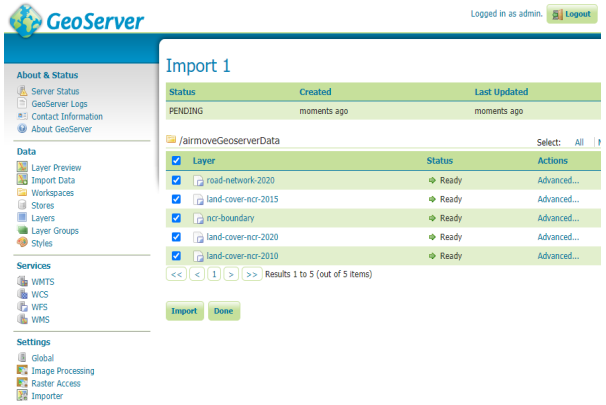

Figure 9. Geoserver Importer interface

A folder was also created where all future datasets to be uploaded will be copied to. This is done to keep the files organized and to avoid duplicate entries.

2.3 Automation and Processing tools

The third component consists of software tools developed to assist the project in conducting their research and in the management of the platform. As the datasets required for the analysis and simulations come from various sources, a tool was developed to automate downloading of these datasets from Google Earth Engine. The tool allows the researchers of the project to download the relevant datasets without the need for manually navigating different data repositories. It also automates the initial processing steps such as clipping an image to the relevant area. A script was created for each dataset to be downloaded. Table 1 shows a list of the datasets currently included in the tool.

Satellite	Dataset
ERA 5	• Mean 2m Air Temperature • Total Precipitation • 10m u-component of wind • 10m v-component of wind
Landsat 8	• Cloud Mask
MODIS	• Aerosol optical depth (blue band) • Aerosol optical depth (green band)
Sentinel 5P	• NO_2 total column density

Table 1. Automated Downloader Datasets

The list is currently comprised only of the datasets used by the researchers of the project. However, other datasets can be easily added by developing a script for that dataset using the currently available ones as a template.

The script works by first selecting the dataset and performing the initial processing in Google Earth Engine (Google Earth Engine, 2023). This is done through the Earth Engine API with a Google service account created specifically for this. Datasets are filtered by date and location to only include those needed by the project. This means that only the satellite images covering the study area of the project (Metro Manila) are downloaded. Raster bands which are not needed for the research are also excluded. This further reduces the download time and storage needed for the datasets.

Some datasets require initial processing which can be done in Google Earth Engine. An example is the Landsat 8 cloud mask. The Earth Engine API provides some basic GIS tools such as raster manipulation which is used here to generate a cloud mask from the Landsat 8 QA band. Doing the processing in Google Earth Engine is faster as it is being done remotely on Google's servers instead of relying on the local machine's hardware.

The dataset is then exported into the Google Drive of the service account and then downloaded by a separate script into the local machine. This extra step of exporting to Google Drive is necessary as Google limits the allowed size of any dataset downloaded directly to a local machine. This has the added benefit of having a backup of preprocessed datasets in Google Drive in case it is lost in the local machine.

The tool also checks for new available datasets at a scheduled time every day. Upon detection of new datasets, the tool will automatically start the download process. This will ensure that the researchers will always have access to the most up-to-date information. This scheduling for the downloader was implemented through the use of the cron utility. This has been configured to run the downloader scripts at a specified time every day. It also automatically provides the required parameters such as date range of datasets to be downloaded, download folder destination, and access credentials for the service account. Currently, it is set to download only the datasets released on the current day, but it can be easily set to download a specific date range instead if historical data is needed.

The tool will be available for download from the project's website that it may also be used by other researchers requiring the same datasets. Other researchers may also use it as a template for developing download scripts for other datasets.

Processing tools were also developed to automate the management of the platform. Having multiple data sources means that sometimes these datasets will have different formats. In order to effectively integrate them, we will have to convert them into a standard format. However, doing this conversion for each dataset, and all upcoming datasets as well, is impractical. This conversion process is then a prime target for automation. One example of this is the conversion of weather data from different stations into a specific format readable by the Django application.

Other processing tools were developed not for automation, but to make them more accessible. Certain analysis done by the researchers of the project require technical expertise and the use of certain proprietary GIS software. By developing tools to do this analysis for us, the steps can be simplified and more accessible to other researchers. This still requires the

user to have a basic understanding of the tool and its features though, which will be discussed in the included manuals.

These tools were developed using Python (Python Software Foundation, 2023), which is a high-level programming language widely used for its simplicity and code readability. Python also has an extensive collection of libraries and modules for processing and GIS purposes. Having access to these libraries minimizes the need to develop certain scripts from scratch.

2.4 Other Technologies Used

GitHub was used for version control during the development process. GitHub allows developers to make changes to different parts of the code base simultaneously through the use of branches and merging options, resulting in higher productivity and less bottlenecks in the development workflow. It also provides a centralized repository online which is easily accessible by anyone with the right credentials. This also makes deploying the system simpler as all the required files are in one place and accessible online. Having a centralized repository in the cloud also provides the developers a reliable backup in case of data loss.

Deployment of the system consists of multiple steps such as installing the required packages, setting up configuration files, and copying of data. To make it simpler, the developers used Ansible (Ansible Community, 2023) to automate these steps. Ansible is an open-source task automation tool running on a host node, in this case the developer's own laptop, sending commands through SSH to a target control node, the server where the system is being deployed. It does this by reading through a list of tasks, called a playbook, and executing each one by one on the target node. Some examples of these tasks are installing a specific version of Python on the target node, or migrating a database. Ansible can also be given access to a GitHub repository which in this project is used to store the files needed for the deployment. This also means that deployment can be done fully remotely.

Four playbooks for Ansible were created. One each for installing basic requirements, frontend deployment, backend deployment, and geoserver deployment.

The playbook for installing the basic requirements is always run first. The tasks here include generating an ssh key which will be used to give the target node access to the project's GitHub repository. It also updates the apt repository and installs packages which will be used in the other playbooks.

The frontend deployment playbook copies the files needed for frontend deployment from the repository into the target node. It also installs NodeJS which is used as the frontend server.

The backend deployment playbook has the most tasks as it manages both the API and the database deployment. It installs the requirements for the backend such as Python, uWSGI, Nginx, and sets up the configuration files for each. It also installs the database server and performs migrations and initial processing of datasets such as generation of statistical reports.

The last playbook installs the geoserver along with its required packages in the target node. It also installs the importer extension and creates a directory where future datasets to be uploaded to the geoserver will be placed.

3. RESULTS AND DISCUSSION

A web platform was developed using the technologies mentioned. During development, the dashboard and the database were deployed in a Linux instance hosted in the National Engineering Center of the University of the Philippines Diliman. There are also plans to obtain the https://airmove.tcagp.upd.edu.ph domain name to make the dashboard more accessible. Once the platform has been turned over to the stakeholders, it will be deployed in their own servers to also make it easier for them to maintain.

The use of a Django, uWSGI, Nginx stack is effective at developing a secure and scalable backend for a web application. Although the use of Django requires the added installation of another application server, uWSGI, its simplicity and readability of code more than makes up for it.

4. CONCLUSION

The AiRMoVE dashboard is one of the project's technical outputs that utilizes existing technologies in developing a web-based platform accessible to users of varying technical expertise and provides the researchers an efficient way of disseminating the research outputs such as maps, charts and datasets to the general public.

For future development, a visits counter could be integrated to gauge the effectiveness of the platform. A page where the users can provide feedback could also be developed.

Ansible can also be used to create playbooks specifically for the management of the system. This can include tasks like migrating the database to a new schema, batch uploading of new data, and restarting the system in case of an error. This allows the system admin to make changes to the platform remotely.

Another tool which can be developed is a script for automating the management of the service account's Google Drive. As the automated downloader is constantly running, new datasets will keep being downloaded into the Google Drive. Eventually these files will need to be deleted to make space for newer datasets. It should also be checked if the to-be-deleted files have backup copies in the local machine. A tool to automate this repetitive process would be useful, and would also minimize the probability of human error.

ACKNOWLEDGEMENT

This study is done by Ambient Air Remote Sensing, Modeling and Visualization Environment (AiRMoVE), a research project funded and monitored by the Department of Science and Technology Philippine Council for Industry, Energy and Emerging Technology Research and Development (DOST PCIEERD).

REFERENCES

Ansible Community, 2023. Red Hat Ansible Automation Platform. https://www.ansible.com/

Django REST, 2023. Home – Django REST framework. https://www.django-rest-framework.org/. (4 October 2023).

Django Software Foundation, 2023. Django. The web framework for perfectionists with deadlines. https://www.djangoproject.com. (9 August 2023).

Figma, 2023. Free Online Wireframe Tool | Figma. https://www.figma.com/wireframe-tool/ (4 October 2023).

Google Earth Engine, 2023. https://earthengine.google.com/ (4 October 2023).

IQAir, 2023. Philippines Air Quality Index (AQI) and Air Pollution information. https://www.iqair.com/philippines. (5 October 2023).

Meta, 2023. React. https://react.dev (16 March 2023).

NCR-EMB, 2023. Real Time Air Monitoring. https://ncr.emb.gov.ph/realtimeairmonitoring/ (5 October 2023)

Nginx, 2023. https://www.nginx.com/. (4 October 2023).

Open Source Geospatial Foundation, 2023. GeoServer. https://geoserver.org. (9 August 2023).

PostgreSQL Global Development Group, 2023. PostgreSQL. https://www.postgresql.org. (4 October 2023).

Python Software Foundation, 2023. Python. https://www.python.org (4 October 2023)

Unbit, 2023. The uWSGI Project. https://uwsgi-docs.readthedocs.io/en/latest/. (4 October 2023).