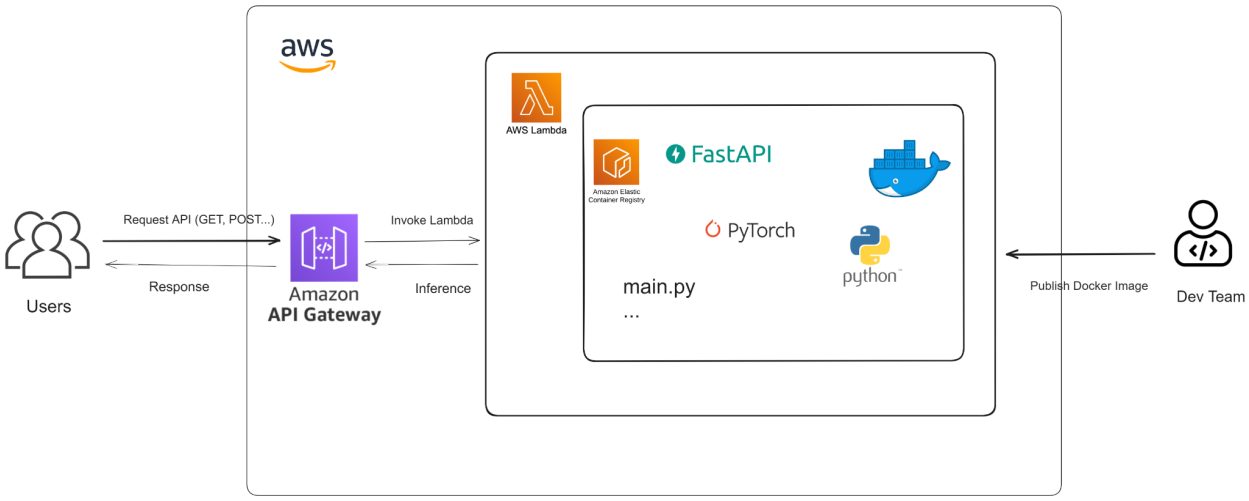
I. Back End Structure

The backend of TeethSeg is a RESTful API developed using Python FastAPI, ensuring robust functionality and reliability. For deployment we used the AWS Cloud using the Amazon Elastic Container Registry (ECR), Lambda and API Gateway services. Here's the architecture of the backend in the AWS Cloud.



Backend deployment in aws lambda

Note: You can deploy the backend on your own server by following the installation instructions provided below.

II. Project Structure

.

├── model/ : Stores pre-trained MeshSegNet models.

├── temp/ : Temporary file storage during processing.

├── output/ : Stores files generated by the model for API responses.

├── config.py: Configuration and model loading.

├── helper.py: Utility functions for APIs.

├── main.py: Defines API endpoints.

├── meshsegnet.py: Encapsulates MeshSegNet's architecture and methods.

├── model.py: Contains `predict` and `predict_alpha` functions for 3D segmentation.

├── setup.py:Installs required packages and configures the workspace.
├── install-requirements.ps1: Configures the project on Windows.
├── install-requirements.sh: Configures the project on Linux/macOS.
├── Dockerfile: Used for deployment to Render hosting.
├── requirements.txt: Lists backend package dependencies.
├── README: Provides installation, configuration, and project information.
├── venv: Isolated environment for Python projects.
└── License: Project licensed under MIT License.

IV. Installation

To set up and run the backend locally:

1. You can run the setup.py script to configure you're project.

```
$ python setup.py
```

Or by following these instruction:

2. Create a new venv.

```
$ py -m venv venv
```

3. Activate venv (Windows)

```
$ ./venv/Scripts/activate
```

4. Install dependencies

```
$ pip install -r requirements.txt
$ pip install pygco
```

5. Run server

After configuring your project now you can run the app using uvicorn.

```
$ uvicorn main:app --reload
```

V. API Endpoints

Here's the available endpoints and the methods to call them:

- "/": Description of the API and its routes
- "/api/v1/predict": Without post-proccessing (this will not give a good result)
- "/api/v1/predict/post_processing": With post-proccessing (The segmentation with post proccessing is more precise)

Go to http://localhost:8000/ if you see a message like {"message": "Hi 3DSF Interns!"} everything is working correctly.

Make sure to call api/v1/predict and api/v1/predict/post_processing endpoints with the POST method.