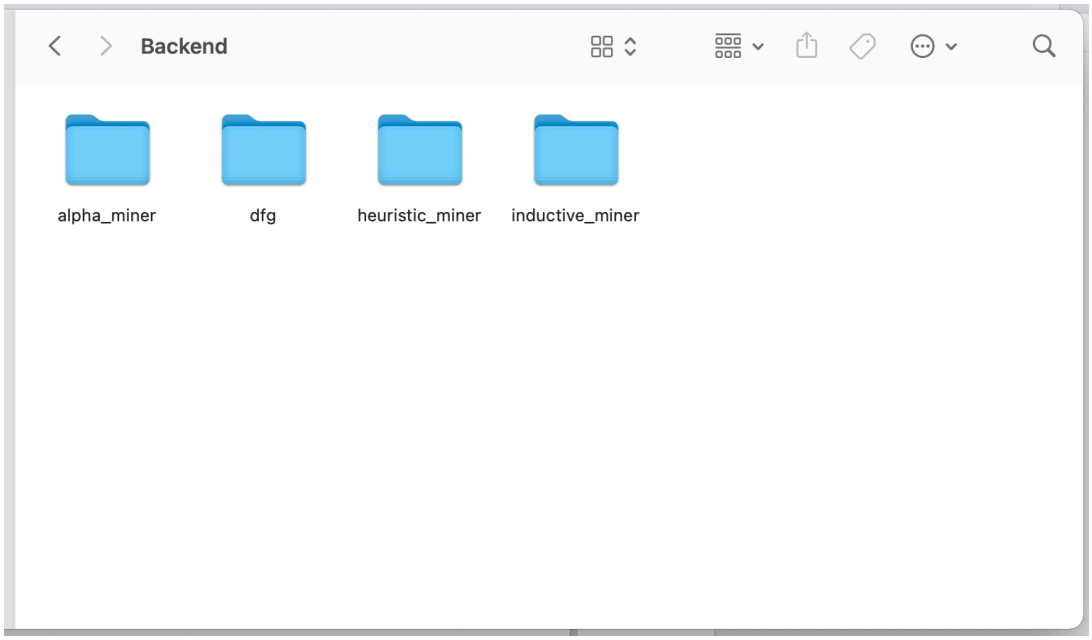
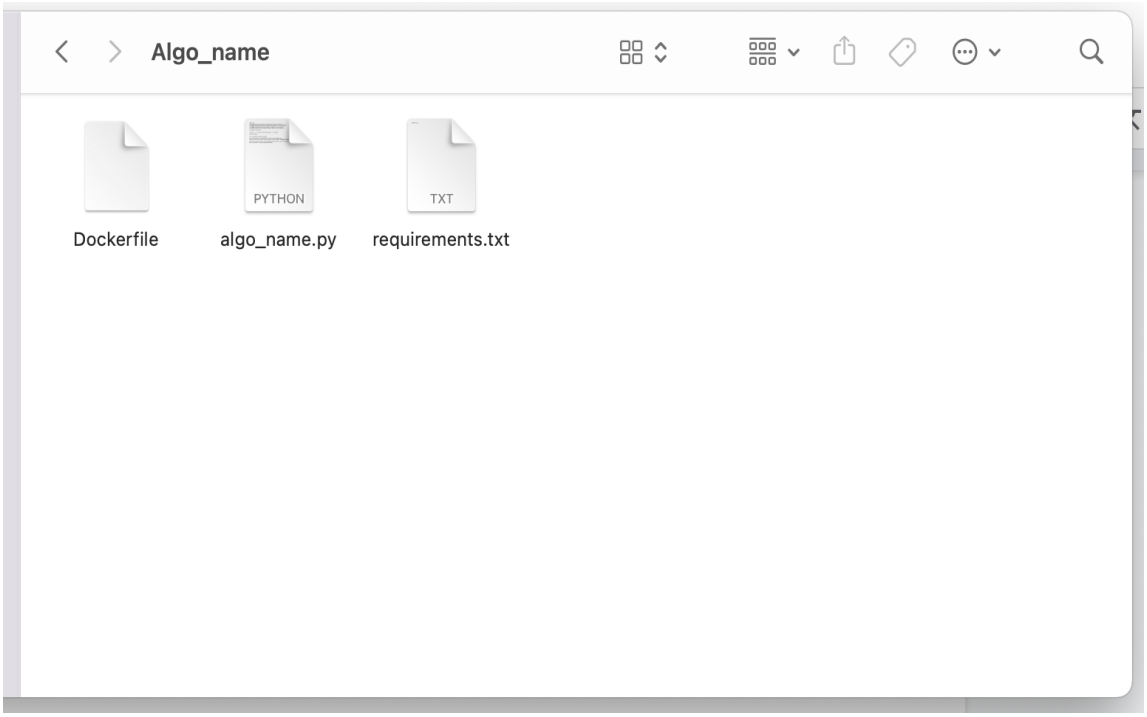
To Upload an Algorithm User has to Run the Following Commands.

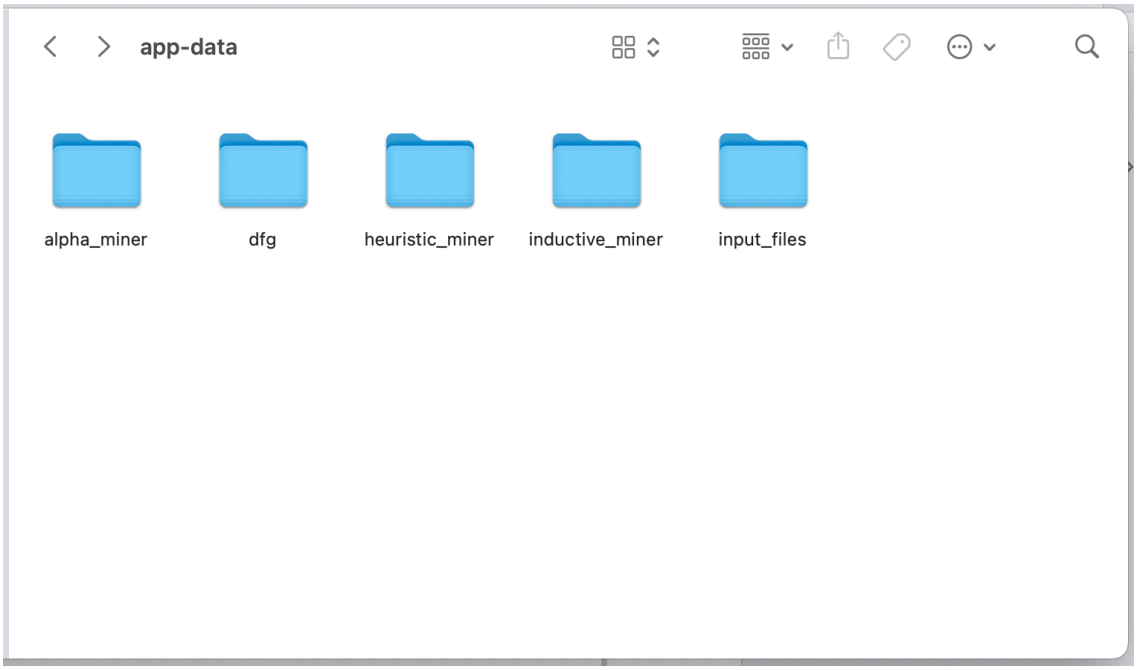
1. First of All User has to go in the backend Folder of the App and have to create a new folder with the name of the algorithm.



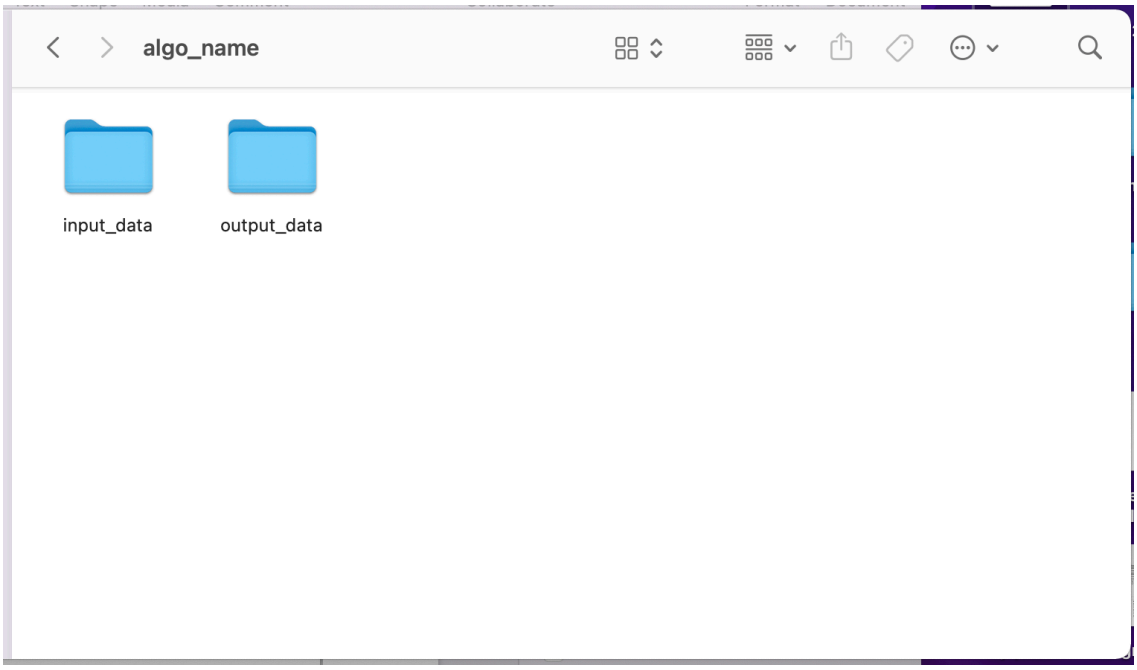
2. User has to put official algo file, docker file and requirements file in the algo_name folder



3. In the app-data folder of the application, the user has to create the same folder as the name of the algorithm.



4. User has to create input_data and output_data folder in that algo_name folder.



5. In the docker file of the algorithm user has to Copy algo_name file in /app and also have to put algo_name in Volume and algo_name file in EntryPoint.

Dockerfile

```
# syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip3 install -r requirements.txt

COPY alpha_miner.py /app

VOLUME ["/alpha_miner"]

ENTRYPOINT [ "python3", "alpha_miner.py"]
```

6. In the official file of the algorithm user has to set input and output path as per the folder created in app_data folder of the respective algorithm.

alpha_miner.py — Edited

```
import sys
from datetime import datetime
from pm4py.objects.log.importer.xes import importer as xes_importer
from pm4py.algo.discovery.alpha import algorithm as alpha_miner
from pm4py.objects.petri_net.exporter import exporter as pnml_exporter


file_name = sys.argv[1]

file_path = '/../alpha_miner/input_data/' + file_name

print(file_path)

log = xes_importer.apply(file_path)

net, initial_marking, final_marking = alpha_miner.apply(log)

date = datetime.now().strftime("%d-%m-%Y-%H-%M-%S")

pnml_exporter.apply(net=net, initial_marking=initial_marking,final_marking=final_marking,
output_filename="/../alpha_miner/output_data/output_petrinet_{date}.pnml".format(date=date))
```

7. Now in the frontend folder of the app, user has to set path in config.json file as per the saved location in the system and have to set the variable as per the operating system in all files in that directory and along with that user have to define the algo_name along with its input, output and description in config.json file.

config.json

```
{
    "PATHS":{
        "VOLUME_PATH_WINDOWS":"//f/process-mining-lab/Desktop-Application-Framework/app-data/",
        "VOLUME_PATH_MAC":"/Users/kumarmukand/Pictures/Desktop-Application-Framework/app-data/",
        "IMAGE_BUILD_PATH_WINDOWS":"F:\\process-mining-lab\\Desktop-Application-Framework\\Backend\\",
        "IMAGE_BUILD_PATH_MAC":"/Users/kumarmukand/Pictures/Desktop-Application-Framework/Backend/"
    },
```

config.json

```
            ]
        },
        "TOKEN_BASED_REPLAY": {
            "INPUT" : ["Log file(.xes file format)","Petri Net File(.pnml file
format)"],
            "INPUT_FORMAT" : [".xes",".pnml"],
            "OUTPUT" : "Resultant PDF(.pdf file format)",
            "DESCRIPTION" : ["can be used with any Petri net model with unique
visible transitions and hidden transitions. When a visible transition needs to be
fired and \n",
                "not all places in the preset are provided with the correct number
of tokens, starting from the current marking it is checked if for some place there
is a \n",
                "sequence of hidden transitions that could be fired in order to
enable the visible transition. The hidden transitions are then fired and a marking
that permits to enable the visible transition is reached."
            ]
        }


    }

}
```

8. In the command prompt, the first user has to go to the same directory in the backend folder of the algorithm and have to built the image in the docker.

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
(processlab) 192:Backend kumarmukand$ docker build -t algo_name .
```

9. User has to the tag and push that repository into the docker.

```
TERMINAL    PROBLEMS    OUTPUT    DEBUG CONSOLE
(processlab) 192:Backend kumarmukand$ docker tag alog_name localhostt:5000/algo_name
```

```
TERMINAL    PROBLEMS    OUTPUT    DEBUG CONSOLE
(processlab) 192:Backend kumarmukand$ docker push localhostt:5000/algo_name
```

10. To manually run and check user can just pass the file name and follow this command to run the algorithm.

```
TERMINAL    PROBLEMS    OUTPUT    DEBUG CONSOLE                                   bash - Backend
(processlab) 192:Backend kumarmukand$ docker run  -v /Users/kumarmukand/Music/process-mining-lab/Desktop-Application-Framework/app-data/algo_name:/algo_name algo_name r
unning-example.xes
```

11. User can also create all the images directly by putting it in config.json file located in Front-end directory and then just have to run algorithms_installer.py and this will create all the images and it will push it in the local registry.

```
TERMINAL    PROBLEMS 6    OUTPUT    DEBUG CONSOLE
(processlab) 192:Frontend kumarmukand$ python algorithms_installer.py
```

Changes which to be made in Frontend Part

12. If the user adds a new algorithm,following changes are required for the UI:
    Navigate to the file “DesktopApp.py” in Frontend folder; in the method ‘def ExitApp(self) add a new 2 lines for the new algorithm as mentioned below:
    Update the NEW_ALGO_NAME with the name of the algorithm created.

```
filelist_output_data_x = glob.glob(os.path.join(config["PATHS"]
            ["VOLUME_PATH_MAC"]+'NEW_ALGO_NAME/output_data/', "*"))
for f in filelist_output_data_x:
            os.remove(f)

filelist_input_data = glob.glob(os.path.join(config["PATHS"]["VOLUME_PATH_MAC"]
+'NEW_ALGO_NAME/input_data/', "*"))
for f in filelist_input_data_x:
            os.remove(f)
```

```python
def ExitApp(self):
    with open('config.json', 'r') as f:
        config = json.load(f)

    filelist_INPUT = glob.glob(os.path.join(self.input_path, "*"))
    for f in filelist_INPUT:
        os.remove(f)

    filelist_output_data = glob.glob(os.path.join(config["PATHS"]["VOLUME_PATH_MAC"]+'alpha_miner/output_data/', "*"))
    for f in filelist_output_data:
        os.remove(f)
    filelist_input_data = glob.glob(os.path.join(config["PATHS"]["VOLUME_PATH_MAC"]+'alpha_miner/input_data/', "*"))
    for f in filelist_input_data:
        os.remove(f)
```

13. Navigate to the file "**DesktopApp.py**" in Frontend folder; in the method 'def `algo_item_clicked(self)` add an elif condition for the new algorithm created: Update the `NEW_ALGO_NAME` with the name of the algorithm created.

```python
elif self.algo_name_selected == "NEW_ALGO_NAME":
        self.label_showAlgoSummary.setText("            Item Selected is Token Based Replay \n\n"+"INPUT PARAMETERS:\n"+"".join(config['ALGORITHMS']['NEW_ALGO_NAME']['INPUT'])+"\nOUTPUT PARAMETERS:\n"+"".join(config['ALGORITHMS']['NEW_ALGO_NAME']['OUTPUT'])+"\nDESCRIPTION:\n"+"".join(config['ALGORITHMS']['NEW_ALGO_NAME']['DESCRIPTION']))
```

14. Navigate to the file "**DesktopApp.py**" in Frontend folder; in the method 'def `disable_algorithm(self)` , if the new algorithm takes two inputs ,update the method with the new algorithm name in a new variable item3.

```python
def disable_algorithm(self):

    if len(self.total_items)==1:
        for x in range(self.listAlgorithm.count()):
            self.listAlgorithm.item(x).setFlags(Qt.ItemIsEnabled)

        items2 = self.listAlgorithm.findItems("token_based_replay",Qt.MatchContains)
        if len(items2) > 0:

            for item2 in items2:
                print("row number of found item =",self.listAlgorithm.row(item2))
                self.checkrow2= self.listAlgorithm.row(item2)
                print("text of found item =",item2.text() )

        for x in range(self.listAlgorithm.count()):
            if(x==self.checkrow2):
                self.listAlgorithm.item(x).setFlags(Qt.NoItemFlags)
            else:
                print("Alpha Miner Disabled")

    elif len(self.total_items)==2:
        for x in range(self.listAlgorithm.count()):
            self.listAlgorithm.item(x).setFlags(Qt.ItemIsEnabled)

        items = self.listAlgorithm.findItems("token_based_replay",Qt.MatchContains)
        if len(items) > 0:

            for item in items:
                print("row number of found item =",self.listAlgorithm.row(item))
                self.checkrow= self.listAlgorithm.row(item)
                print("text of found item =",item.text() )

        for x in range(self.listAlgorithm.count()):
            if(x==self.checkrow):
                print("Alpha Miner Enabled")
            else:
                self.listAlgorithm.item(x).setFlags(Qt.NoItemFlags)
```