Validate and POST User Details

This Document outlines the process of validating the user details and subsequently posting the validated data

CONSTANTS.PY

Added three constants which are used throughout the project

```
 # ---------MOBILE VALIDATION-----------------
DATA = {'records':[]}
VALID_COUNTRY_CODE = ['91', '46', '16']
EXCLUDED_NUMBERS = [9036318984, 9945383132]
```

VALIDATE_UTILS.PY

Required constants and logging module is imported

```
from constants import *
from logger import *
```

is_valid_mobile function is used to validate the user mobile based on the length, excluded mobile number and country code.
If the length of mobile is less than 12, exception is raised
If the mobile number is in the constant EXCLUDED_NUMBERS, then function return's TRUE else it will validate further
If the country code in which is at the starting of the mobile[:2] is in the constant VALID_COUNTRY_CODE, then function return's TRUE, else exception is raised

NEW FUNCTION

```
def is_valid_mobile(mobile):
    if isinstance(mobile, int):
        mobile = str(mobile) # converting mobile to str
        if len(mobile) == 12:
            if int(mobile) in EXCLUDED_NUMBERS:
                return True
            if mobile[:2] in VALID_COUNTRY_CODE:
                logging.debug(f'Mobile Number verified Successfully')
                return True
            else:
                raise ValueError(f'Country code is not valid :-{mobile[:2]}')
        else:
            raise ValueError(f'Invalid Mobile Number length :- {len(mobile)}')
```

is_valid_name function is used to validate the name which is sent by user based length and checking if all the characters are alphabets or not.
Exception raised when the length of name is less than 3.
After removing the accepted special characters like dot (.), underscore (_) and space ( ), If the name contains only alphabets, then function returns TRUE, else Exception is raised.

NEW FUNCTION

```
def is_valid_name(name):
    if isinstance(name, str):
            clean_name = name.replace('.', '').replace('_', '').replace(' ', '')
            if len(clean_name) > 3:
                if clean_name.isalpha():
                    logging.debug(f'Name verified successfully')
                    return True
                else:
                    raise ValueError(f'User name must be alphabetic after removing spe
            else:
                raise ValueError(f'User name cannot be {len(clean_name)} characters, u
        else:
            raise ValueError(f'Name should be a string, received: {type(name).__name__
```

MAIN.PY

Imported required modules for logging and validating

```
from logger import *
from constants import DATA
from validate_utils import *
```

Defined a class **PostUserData**

```
class PostUserData():
```

Constructor is defined for logging that app has started

```
def __init__(self):
        logging.info("User details validation and posting started")
```

post_new_record method will take the argument ‘**record**’ and initially checks the key ‘**mobile**’ is available in that dict, If mobile key is available, then **is_valid_mobile** function is called for validating the mobile. else we will log the error
If the mobile is valid then ‘**name**’ key is verified using the **is_valid_name** function if the key is present in the record.
If the mobile and name is validated, then the record is appended to the **DATA** variable which is a **constant** and logged in the log file.

```
def post_new_record(self, record):
        try:
            logging.debug(f"Validating for new user {record}")
            if isinstance(record, dict):
                if 'mobile' in record:
                    logging.debug(f"Validating for mobile...")
                    if is_valid_mobile(record['mobile']):
                        logging.debug(f"Mobile number validated successfully")
                        if 'name' in record:
                            logging.debug(f"Validating for name...")
                            if is_valid_name(record['name']):
                                logging.debug(f"Name validated successfully")
```

```
                                DATA['records'].append(record)
                                response = {"Message":"Record Inserted SUCCESSFULLY",
                                logging.debug(f"{response}")
                                return response
                            else:
                                response = {"Message": "Missing name key in record", "reco
                                logging.debug(f"{response}")
                                return response
                    else:
                        response = {"Message": "Missing mobile key in record", "record":re
                        logging.debug(f"{response}")
                        return response
                else:
                    response = {"Message": f"Incorrect record type {type(record)}"}
                    logging.debug(f"{response}")
                    return response

            except Exception as err:
                logging.error(f"Error while validating user details :- {err}")
            finally:
                logging.info(f"Execution completed")
                print("Program executed succesfully")
```

We create and object to the class name **PostUserData** and call the method **post_user_data** with the dict which consists of user details

```
obj = PostUserData()
obj.post_new_record({'mobile':469036318984, 'name':'Yaswant_Kumar.P'})
```

LOGGER.PY

logging module is imported and the path for log file, level, time, append mode are set to store the log's

```
import logging

logging.basicConfig(
    filename = "c:/Users/w125682/Downloads/practice.log",
    level = logging.DEBUG,
    format = '%(asctime)s - %(levelname)s - %(message)s',
    filemode = 'a'
    )
```