Project Development Phase

Sprint 1

app.py

```
from dotenv import dotenv_values
from flask import Flask
from flask_cors import CORS
import smtplib
from routes.register import Register
from routes.checkEmail import CheckEmail
from routes.verify import Verify
from routes.getnews import *
from routes.login import Login
from flask_restful import Api

app = Flask(__name__)
api=Api(app)

data=dotenv_values(".env")
app.config['SECRET_KEY']=data["secrect_key"]
app.config['SECURITY_PASSWORD_SALT']=data["security_password_salt"]

s = smtplib.SMTP('smtp.gmail.com',587)
s.starttls()
s.login(data["email"],data["email_password"])

CORS(app, supports_credentials=True)

api.add_resource(Login,'/login')
api.add_resource(CheckEmail,'/register/check')
api.add_resource(Register,'/register')
api.add_resource(Verify,'/register/verify')
```

register.py

```python
from datetime import datetime
from flask import request,after_this_request
from flask_restful import Resource
from utils.dbQuery import insertQuery
from utils.emailSender import newEmailSender
from utils.password import genHash

class Register(Resource):
    def post(self):
        req=request.json
        name=req['name']
        email=req['email']
        password=req['password']
        fav=req['favourite']
        if(name=='' or email=='' or password=='' or fav==''):
            return {"status":"Missing data"},404
        password=genHash(password)
        t=(name,email,password,fav,datetime.now())
        res=insertQuery('INSERT INTO user 
(name,email,password,favourites,resend_time) values (?,?,?,?,?)',t)
        if(not res):
            return {"status":"Error while registering"},400

        @after_this_request
        def emailer(response):
            newEmailSender(email)
            return response

        return {"status":"Successfully registered"},200
```

checkemail.py

```python
from flask import request
from flask_restful import Resource
from utils.dbQuery import selectQuery

class CheckEmail(Resource):
    def post(self):
        data=request.json
        email=str(data["email"])
        res=selectQuery("SELECT email FROM USER WHERE EMAIL=?",(email,))
        print(res)
        if(not res):
            return {"status":True},200
        return {"status":False},400
```

verify.py

```
from flask import request
from flask_restful import Resource
from utils.tokener import confirm_token
from utils.dbQuery import insertQuery

class Verify(Resource):
    def post(self):
        data=request.json
        token=data["token"]
        email=confirm_token(token)
        if(not email):
            return {"status":"Couldn't verify"},400
        res=insertQuery("UPDATE user set verified=True where email=?",(email,))
        if(not res):
            return {"status":"Couldn't Update"},400
        return {"status":"verified successfully"},200
```

dbQuery.py

```
import ibm_db
from utils.dbConfig import getDbCred

conn=ibm_db.connect(getDbCred(),"","")

def selectQuery(query,params=None):
    try:
        stmt=ibm_db.prepare(conn,query)
        if(params==None):
            ibm_db.execute(stmt)
            data=ibm_db.fetch_assoc(stmt)
            return data
        ibm_db.execute(stmt,params)
        data=ibm_db.fetch_assoc(stmt)
        return data
    except:
        return False

def insertQuery(query,params):
    try:
        stmt=ibm_db.prepare(conn,query)
        ibm_db.execute(stmt,params)
        return True
    except:
        return False
```

login.py

```
from datetime import datetime
from flask_restful import Resource
from flask import request,after_this_request
from utils.password import checkPassword
from utils.dbQuery import *
from dateparser import parse
from utils.tokener import generate_confirmation_token
from utils.emailSender import emailSender
import app
import jwt

class Login(Resource):
    def post(self):
        data=request.json
        email=data["email"]
        password=data["password"]
        queryRes=selectQuery("SELECT * from user where email=?",(email,))
        res=checkPassword(password,queryRes["PASSWORD"])
        if(not res):
            return {"status":"Wrong credentials"},400
        if(not queryRes["VERIFIED"]):
            lastTime=parse(queryRes["RESEND_TIME"])
            currTime=datetime.now()
            diff=currTime-lastTime
            diff=diff.total_seconds()
            if(diff>3600):
                token=generate_confirmation_token(email)
                emailSender(email,token)
                insertQuery("UPDATE user set RESEND_TIME=? where
email=?",(datetime.now(),email))
            return {"status":"Not verified"},400
        @after_this_request
        def cookieSender(response):
            access_token=jwt.encode(
                {"email":email},app.app.config['SECRET_KEY']
            )
            response.set_cookie("access_token",str(access_token),httponly=True)
            response.set_cookie("email",str(email),httponly=True)
            return response

        return{"status":"Successfully Logged in"},200
```

password.py

```
import bcrypt

def genHash(password):
    salt=bcrypt.gensalt()
    bytes=password.encode('utf-8')
    hash=bcrypt.hashpw(bytes,salt)
    print(hash)
    return hash

def checkPassword(password,hash):
    hash=hash.encode('utf-8')
    bytes=password.encode('utf-8')
    res=bcrypt.checkpw(bytes,hash)
    return res
```