



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Malaysia-Japan
International
Institute of Technology
(MJIT)

ADVANCED PROGRAMMING

(SMJE 4383)

WEB BASED PROJECT REPORT:

EMPLOYEE ATTENDANCE

GROUP :

MEMBERS NAME	MATRIC NO.	IC NO.
NASUHA BINTI ANUAR	A17MJ0208	960502-01-5316
NUR AZLINA BINTI MAHFUL	A17MJ0108	970901-01-5742
LEE KAR OON	A17MJ0056	970521-07-5241

COURSE/SEC : 4 SMJE/ SEC 01

LECTURER : IR. DR. ZOOL HILMI BIN ISMAIL

TABLE OF CONTENTS

TITLE	PAGE
CHAPTER 1 INTRODUCTION	
1.1 Background Research	1
1.2 Problem Statement	1
1.3 Objective	2
1.4 Distribution of Workload	3
CHAPTER 2 METHODOLOGY	
2.1 Project Development	
2.1.1 Flowchart for Project Web Based Application	4
2.1.2 Web Based Application Interface	5-6
CHAPTER 3 RESULTS AND DISCUSSION	
3.1 Project Analysis	7
3.2 Project Simulation	8-11
CHAPTER 4 CONCLUSION	
4.1 Project Outcome	12
CHAPTER 5 APPENDICES	
5.1 Source code	
5.1.1 App.py	13-14
5.2.1 Config.py	15
5.3.1 Models.py	16
5.4.1 Manage.py	17-18

CHAPTER 1

INTRODUCTION

1.1 Background Research

In this project assignment, the goal of it is to let every students to experience the process of creating web-based applications. Each student is required to develop real-world applications as a group and implementing the use of standard technology such as HTML, Python language, git, Flask and etc. Web applications are shooting up in its everyday use in all kind of sectors. For example, industrial sector. In fact, web-based applications are a great replacement to desktop applications and becoming a significant instrument for small and large businesses around the world. According to Alex Chaffee, 2012, “a web application is a collection of servlets, html pages, classes, and other resources that can be bundled and run-on multiple containers from multiple vendors. A web application is rooted at a specific path within a web server”. The popularity and convenience of web-based apps is because developers don’t have to write multiple versions of the same application for different operating systems, the users won’t have to install additional software. In this project, Linux platform is used as it is a great web development platform as it has tools to make the task easier. Due to lots of advantages offered, web-based applications are commonly used in businesses for its promising future.

1.2 PROBLEM STATEMENT

During the study for this project, we are planning to develop an application that facilitates the need in businesses. We fairly acknowledged the problem faced by most of employees and managers. The traditional way of attendance tracking such as card swiping, manually putting names and working hours on spreadsheets and etc are somewhat a hassle, especially during this Covid-19 pandemic, many employees are encouraged to work from home. So the traditional way of tracking attendance need to be further developed. To keep up with the technology, we have developed a web-based attendance tracker to minimize the hassle faced by both employers and employees as attendance management is essential to every companies.

1.3 OBJECTIVES

Several objectives in this project are stated to achieve solutions for solving the problem statements. Following are the objectives from this project:

1. To develop a web based application using python 3 and virtual environment
2. To solve the industrial-based problem and easy work environment and experience.

1.4 DISTRIBUTION OF WORKLOAD

TASK	NASUHA	AZLINA	KAR OON
Project discussion and online group meeting	/	/	/
Project idea presentation		/	/
Construct web based app using Ubuntu 20.04	/		
Present the roughly idea and design for the app interface			/
Prepared the project report	/	/	/
Code and simulate the program in python	/		
Project report: Introduction		/	
Project report: Methodology	/		
Project report: Result and discussion	/		
Project report: Conclusion			/

Table 1: Project Task Allocation of Members

CHAPTER 2

METHODOLOGY

2.1 PROJECT DEVELOPMENT

2.1.1 FLOWCHART FOR WEB BASED APPLICATION

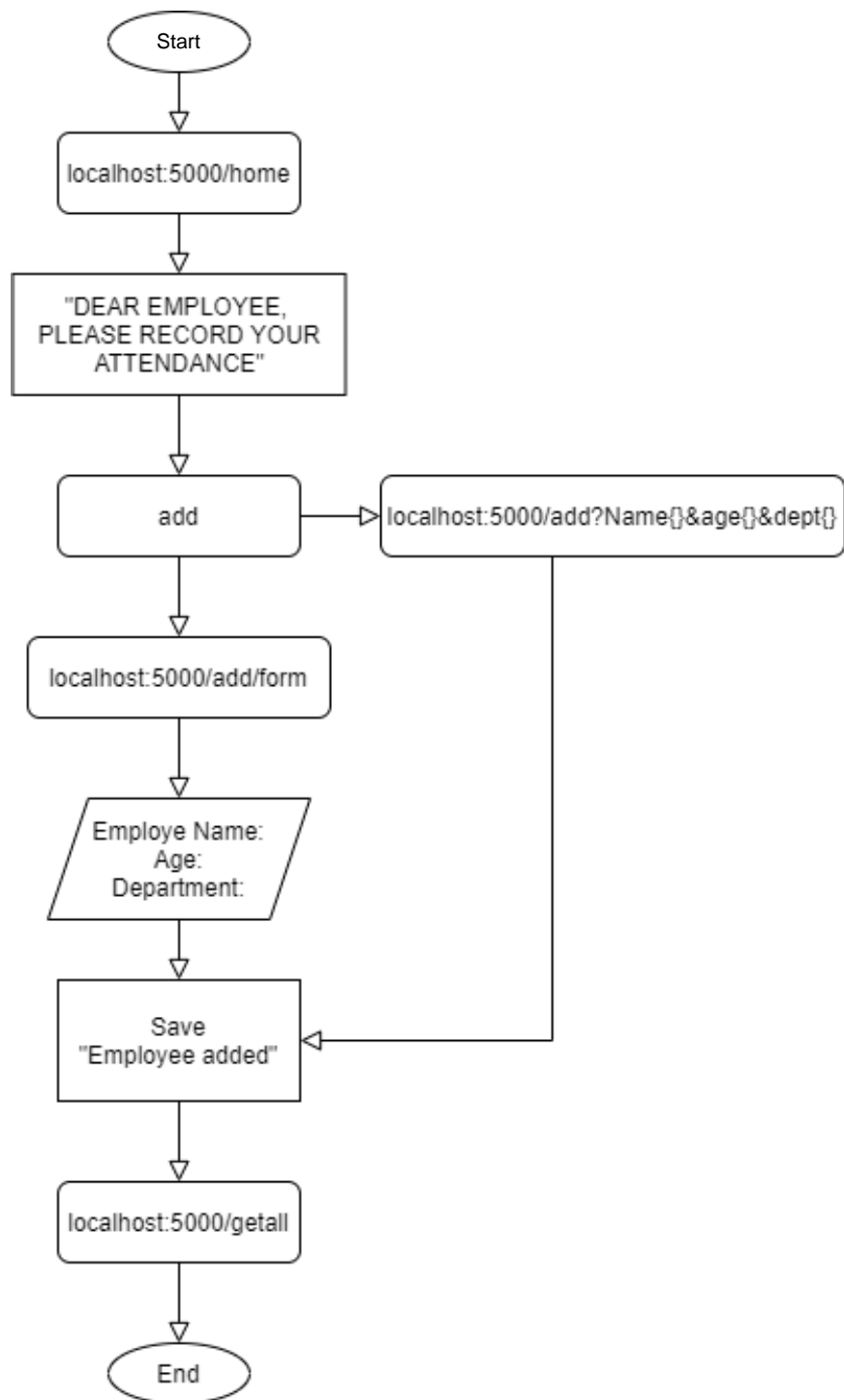


Fig.2 Flowchart for Project Application

2.1.2 WEB BASED APPLICATION INTERFACE

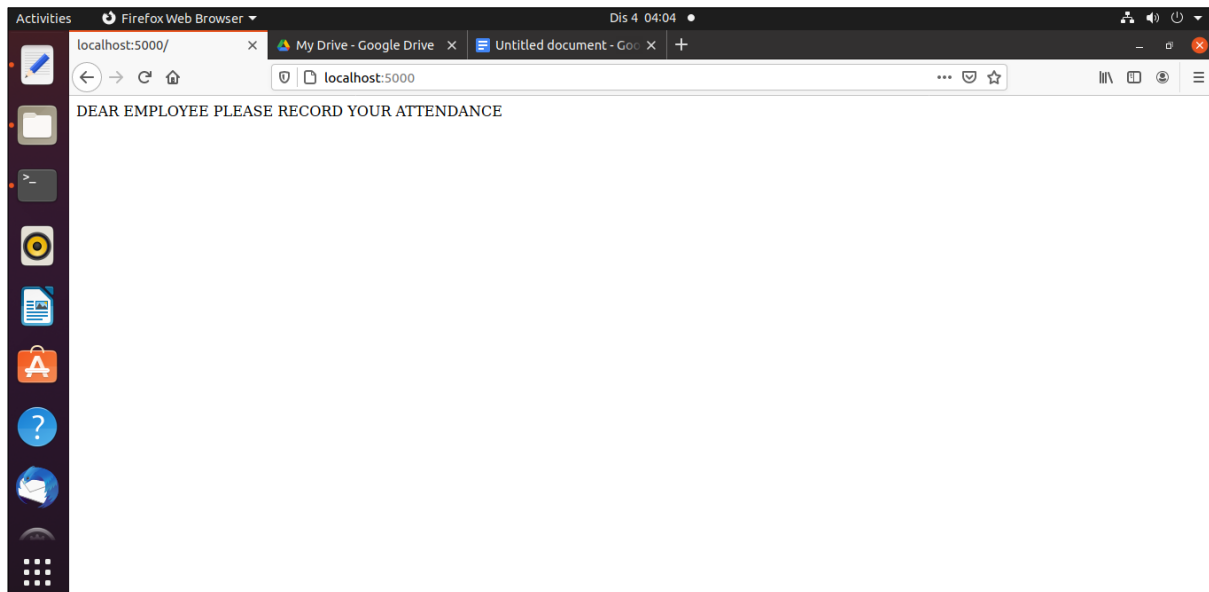


Fig 2.1 localhost:5000/home

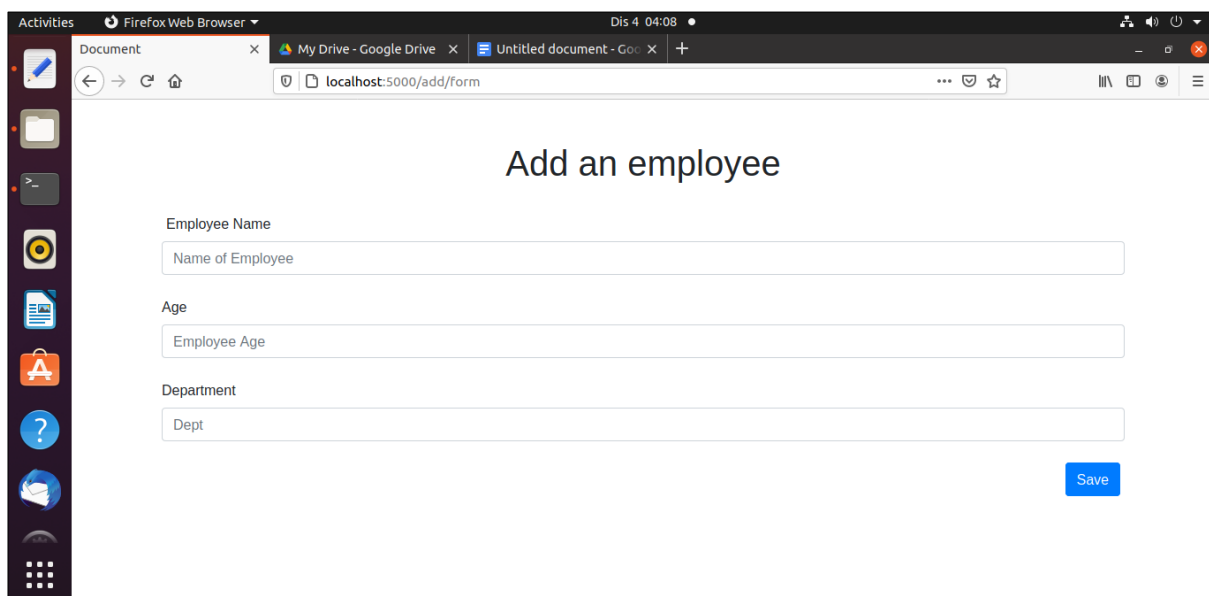


Fig 2.2 localhost:5000/add/form

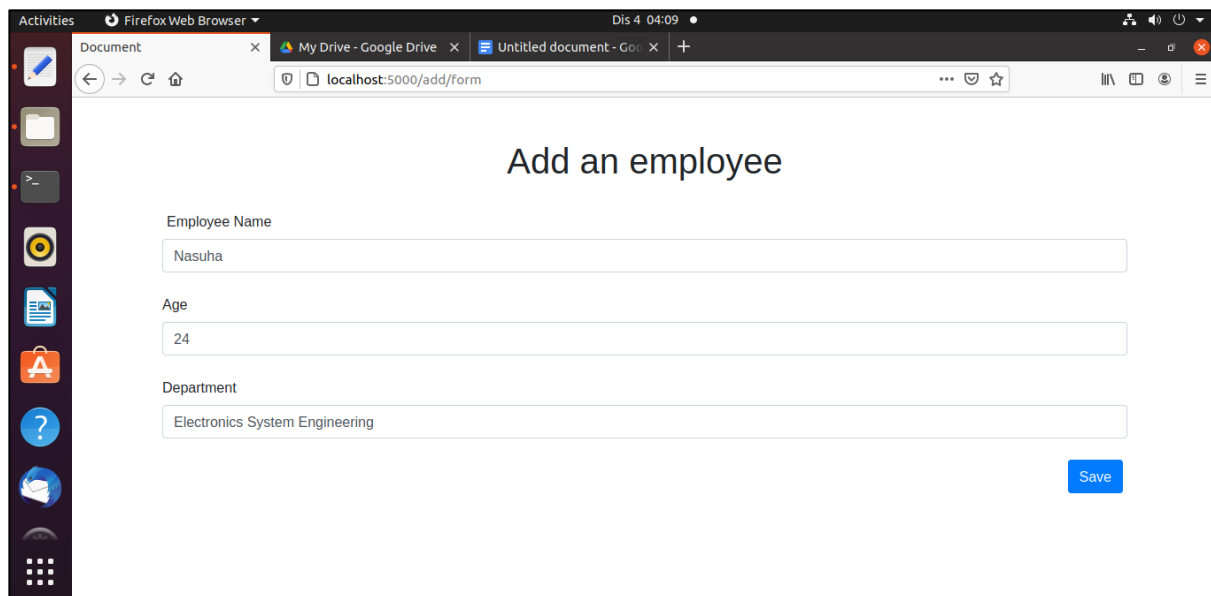


Fig 2.3 key in employee details

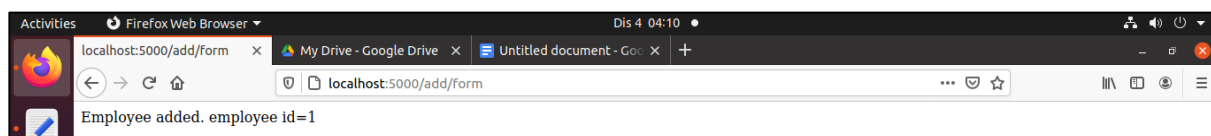


Fig 2.4 Employee added

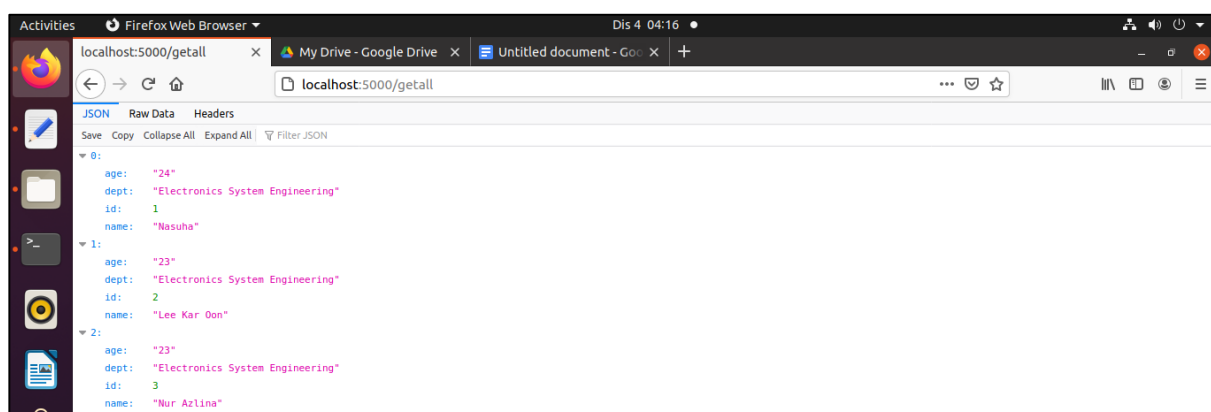


Fig 2.5 localhost:5000/getall

CHAPTER 3

DISCUSSION

3.1 PROJECT ANALYSIS

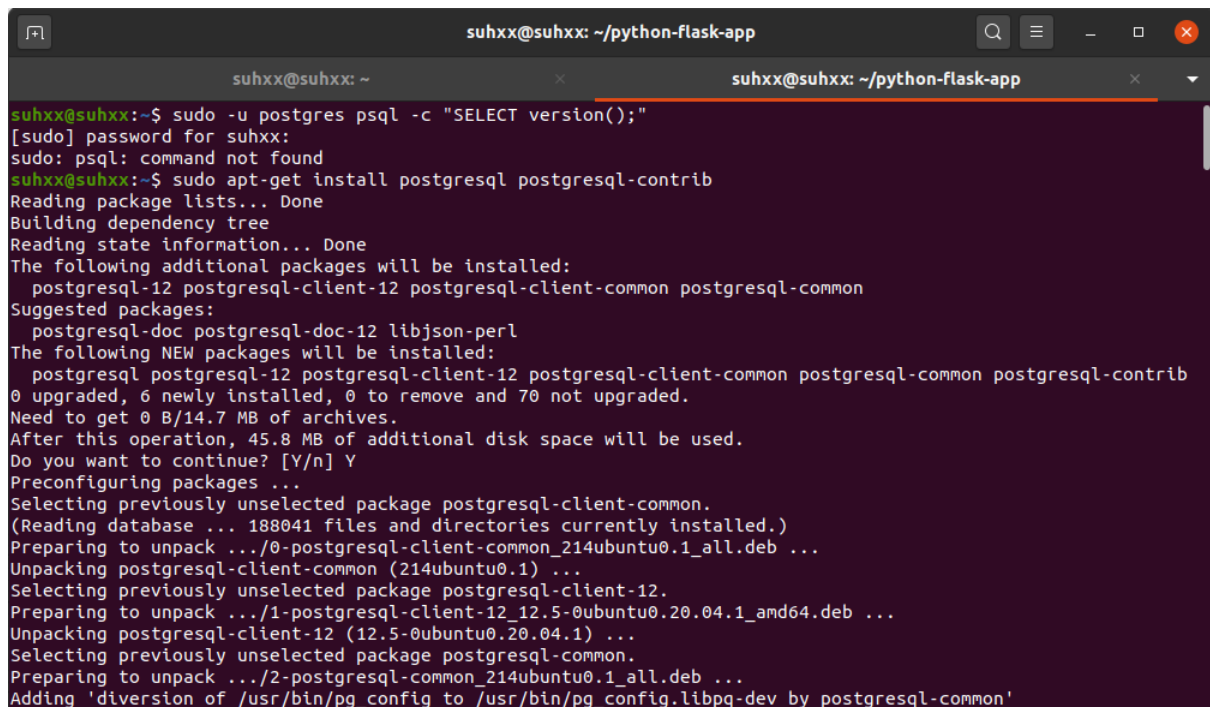
In this project, Ubuntu version 20.04, and python3 were installed along with pip3. Also PostgreSQL version 12.5 were installed to create a database and a user. An installation of a flask and virtual environment in python is run as well to use all the required package needed for developing the web app before lastly run database migration. Database migration is a process of writing code in python file to create a table instead of creating table directly in psql (postgres shell). Hence, by run those code in python file created the database and the table. “devopsdb” and “devopsuser” are the name used to create the database and user in postgres sql. This project application developed running using local host and TCP port 5432, 127.0.0.1:5432.

An installation of virtual environment is done by running command `pip3 install virtual env` in a directory file name ‘python-flask-app’ where a virtual environment named `venv` is created inside. The virtual environment is activated during the whole process of developing a web app. Then run the `pip3` command to install all the dependencies such as flask, flask sqlalchemy, flask script, flask migrate and psycopg2 binary.

Next, a file `.env` is created in which to set up the environment variables and source `.env` file to pick up the development config from config module. Furthermore, the database is connected using the url for connectivity. Means, connect to postgresql with the user and database in local host.

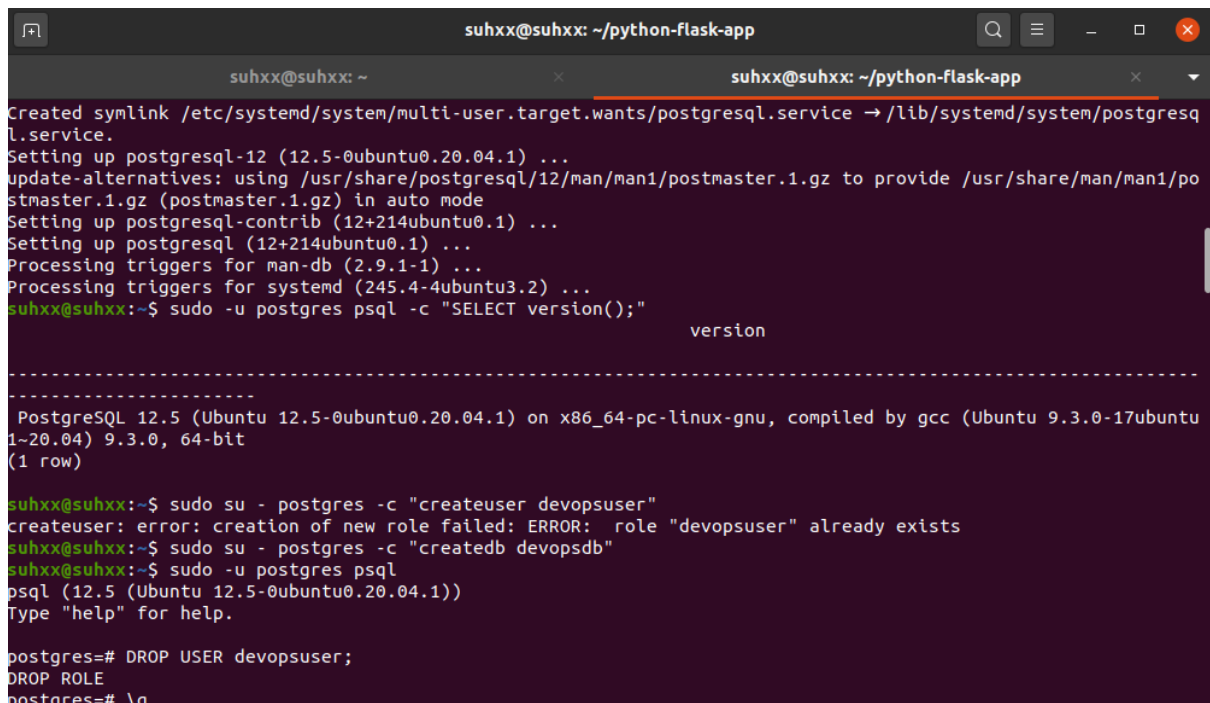
Lastly, the migration were run after created a database scheme by initially run a command `python3 manage.py db init` where it is going to create a migration static and database scheme. Created a migrations file by running migrate command `python3 manage.py db migrate`. A code is provided after migration which is a code to create the table. Run the command `python3 manage.py db upgrade` to create required table. Finally, run the web application using command `python3 manage.py runserver`. Thus, our application running on local host 5000 port.

3.2 PROJECT SIMULATION



```
suhxx@suhxx: ~/python-flask-app
suhxx@suhxx: ~
suhxx@suhxx:~$ sudo -u postgres psql -c "SELECT version();"
[sudo] password for suhxx:
sudo: psql: command not found
suhxx@suhxx:~$ sudo apt-get install postgresql postgresql-contrib
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  postgresql-12 postgresql-client-12 postgresql-client-common postgresql-common
Suggested packages:
  postgresql-doc postgresql-doc-12 libjson-perl
The following NEW packages will be installed:
  postgresql postgresql-12 postgresql-client-12 postgresql-client-common postgresql-common postgresql-contrib
0 upgraded, 6 newly installed, 0 to remove and 70 not upgraded.
Need to get 0 B/14.7 MB of archives.
After this operation, 45.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Preconfiguring packages ...
Selecting previously unselected package postgresql-client-common.
(Reading database ... 188041 files and directories currently installed.)
Preparing to unpack .../0-postgresql-client-common_214ubuntu0.1_all.deb ...
Unpacking postgresql-client-common (214ubuntu0.1) ...
Selecting previously unselected package postgresql-client-12.
Preparing to unpack .../1-postgresql-client-12_12.5-0ubuntu0.20.04.1_amd64.deb ...
Unpacking postgresql-client-12 (12.5-0ubuntu0.20.04.1) ...
Selecting previously unselected package postgresql-common.
Preparing to unpack .../2-postgresql-common_214ubuntu0.1_all.deb ...
Adding 'diversion of /usr/bin/pg_config to /usr/bin/pg_config.libpq-dev by postgresql-common'
```

Fig 3.1 Installing postgres sql



```
suhxx@suhxx: ~/python-flask-app
suhxx@suhxx: ~
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service → /lib/systemd/system/postgresql.service.
Setting up postgresql-12 (12.5-0ubuntu0.20.04.1) ...
update-alternatives: using /usr/share/postgresql/12/man/man1/postmaster.1.gz to provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
Setting up postgresql-contrib (12+214ubuntu0.1) ...
Setting up postgresql (12+214ubuntu0.1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.2) ...
suhxx@suhxx:~$ sudo -u postgres psql -c "SELECT version();"
          version
-----
 PostgreSQL 12.5 (Ubuntu 12.5-0ubuntu0.20.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0, 64-bit
(1 row)

suhxx@suhxx:~$ sudo su - postgres -c "createuser devopsuser"
createuser: error: creation of new role failed: ERROR: role "devopsuser" already exists
suhxx@suhxx:~$ sudo su - postgres -c "createdb devopsdb"
suhxx@suhxx:~$ sudo -u postgres psql
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# DROP USER devopsuser;
DROP ROLE
postgres=# \q
```

Fig 3.2 Creating database and user

```
suhxx@suhxx: ~/python-flask-app
postgres=# \q
suhxx@suhxx:~$ sudo su - postgres -c "createuser devopsuser"
suhxx@suhxx:~$ sudo -u postgres psql
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# grant all privileges on database devopsdb to devopsuser;
GRANT
postgres=# alter user devopsuser with password 'devops';
ALTER ROLE
postgres=# \q
suhxx@suhxx:~$ sudo vi /etc/postgresql/10/main/postgresql.conf
suhxx@suhxx:~$ netstat -tlupn | grep 5432
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:5432        0.0.0.0:*          LISTEN      -
suhxx@suhxx:~$ mkdir python-flask-app
suhxx@suhxx:~$ cd python-flask-app
suhxx@suhxx:~/python-flask-app$ pip3 install virtualenv
Requirement already satisfied: virtualenv in /home/suhxx/.local/lib/python3.8/site-packages (20.2.1)
Requirement already satisfied: distlib<1,>=0.3.1 in /home/suhxx/.local/lib/python3.8/site-packages (from virtualenv) (0.3.1)
Requirement already satisfied: six<2,>=1.9.0 in /usr/lib/python3/dist-packages (from virtualenv) (1.14.0)
Requirement already satisfied: filelock<4,>=3.0.0 in /home/suhxx/.local/lib/python3.8/site-packages (from virtualenv) (3.0.12)
Requirement already satisfied: appdirs<2,>=1.4.3 in /home/suhxx/.local/lib/python3.8/site-packages (from virtualenv) (1.4.4)
suhxx@suhxx:~/python-flask-app$ sudo apt install python3-venv
```

Fig 3.3 Enable Local host and setup TCP Port 5432

```
suhxx@suhxx: ~/python-flask-app
suhxx@suhxx:~/python-flask-app$ sudo apt install python3-venv
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-venv is already the newest version (3.8.2-0ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 70 not upgraded.
suhxx@suhxx:~/python-flask-app$ python3 -m venv venv
suhxx@suhxx:~/python-flask-app$ source venv/bin/activate
(venv) suhxx@suhxx:~/python-flask-app$ vi requirement.txt
(venv) suhxx@suhxx:~/python-flask-app$ pip3 install -r requirement.txt
Collecting Flask
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting flask_sqlalchemy
  Using cached Flask_SQLAlchemy-2.4.4-py2.py3-none-any.whl (17 kB)
Collecting flask_script
  Using cached Flask-Script-2.0.6.tar.gz (43 kB)
Collecting flask_migrate
  Using cached Flask_Migrate-2.5.3-py2.py3-none-any.whl (13 kB)
Collecting psycopg2-binary
  Using cached psycopg2_binary-2.8.6-cp38-cp38-manylinux1_x86_64.whl (3.0 MB)
Collecting itsdangerous>=0.24
  Using cached itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Collecting click>=5.1
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting Jinja2>=2.10.1
  Using cached Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting Werkzeug>=0.15
  Using cached Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
```

Fig 3.4 installing virtual environment flask

```
suhxx@suhxx: ~/python-flask-app

Running setup.py install for flask-script ... done
Successfully installed Flask-1.1.2 Jinja2-2.11.2 Mako-1.1.3 MarkupSafe-1.1.1 SQLAlchemy-1.3.20 Werkzeug-1.0.1 alembic-
1.4.3 click-7.1.2 flask-migrate-2.5.3 flask-script-2.0.6 flask-sqlalchemy-2.4.4 itsdangerous-1.1.0 pycpg2-binary-2.8
.6 python-dateutil-2.8.1 python-editor-1.0.4 six-1.15.0
(venv) suhxx@suhxx:~/python-flask-app$
(venv) suhxx@suhxx:~/python-flask-app$
(venv) suhxx@suhxx:~/python-flask-app$ vi .env
(venv) suhxx@suhxx:~/python-flask-app$ source .env
(venv) suhxx@suhxx:~/python-flask-app$ echo $APP_SETTINGS
config.DevelopmentConfig
(venv) suhxx@suhxx:~/python-flask-app$ vi app.py
(venv) suhxx@suhxx:~/python-flask-app$ vi config.py
(venv) suhxx@suhxx:~/python-flask-app$ vi models.py
(venv) suhxx@suhxx:~/python-flask-app$ vi manage.py
(venv) suhxx@suhxx:~/python-flask-app$ mkdir -p templates
(venv) suhxx@suhxx:~/python-flask-app$ vi templates/getdata.html
(venv) suhxx@suhxx:~/python-flask-app$ python3 manage.py db init
Creating directory /home/suhxx/python-flask-app/migrations ... done
Creating directory /home/suhxx/python-flask-app/migrations/versions ... done
Generating /home/suhxx/python-flask-app/migrations/script.py.mako ... done
Generating /home/suhxx/python-flask-app/migrations/env.py ... done
Generating /home/suhxx/python-flask-app/migrations/alembic.ini ... done
Generating /home/suhxx/python-flask-app/migrations/README ... done
Please edit configuration/connection/logging settings in '/home/suhxx/python-flask-app/migrations/alembic.ini' before
proceeding.
(venv) suhxx@suhxx:~/python-flask-app$ ls
app.py  config.py  manage.py  migrations  models.py  __pycache__  requirement.txt  templates  venv
(venv) suhxx@suhxx:~/python-flask-app$ ls migrations/versions
```

Fig 3.5 Creating .py files for writing source code in python

```
suhxx@suhxx: ~/python-flask-app

e proceeding.
(venv) suhxx@suhxx:~/python-flask-app$ ls
app.py  config.py  manage.py  migrations  models.py  __pycache__  requirement.txt  templates  venv
(venv) suhxx@suhxx:~/python-flask-app$ ls migrations/versions
(venv) suhxx@suhxx:~/python-flask-app$ python3 manage.py db migrate
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.autogenerate.compare] Detected added table 'employee'
Generating /home/suhxx/python-flask-app/migrations/versions/7fd09944f6f6.py ... done
(venv) suhxx@suhxx:~/python-flask-app$ ls migrations/versions/7fd09944f6f6.py
migrations/versions/7fd09944f6f6.py
(venv) suhxx@suhxx:~/python-flask-app$ python3 manage.py db upgrade
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> 7fd09944f6f6, empty message
(venv) suhxx@suhxx:~/python-flask-app$
(venv) suhxx@suhxx:~/python-flask-app$ python3 manage.py runserver
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
* Debugger is active!
* Debugger PIN: 283-334-010
```

Fig 3.6 Run migration

```

* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
* Debugger is active!
* Debugger PIN: 283-334-010
127.0.0.1 - - [04/Dec/2020 03:50:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 03:50:56] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [04/Dec/2020 04:08:06] "GET /add/form HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:08:11] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [04/Dec/2020 04:10:31] "POST /add/form HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:13:37] "GET /getall HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:14:31] "GET /add/form HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:15:05] "POST /add/form HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:15:20] "GET /add/form HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:15:55] "POST /add/form HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:16:05] "GET /getall HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:17:15] "GET /details HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:20:10] "GET /Nasuha HTTP/1.1" 404 -
127.0.0.1 - - [04/Dec/2020 04:21:26] "GET /name/Nasuha HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:21:45] "GET /details HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:21:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Dec/2020 04:22:10] "GET /add HTTP/1.1" 200 -

```

Fig 3.7 Web app running History/ track

CHAPTER 4

CONCLUSION

In a nutshell, a web-based attendance tracking application is successfully developed by using python 3 and virtual environment. This can be used in industrial to take the attendance remotely especially during this Covid-19 pandemic as employee are encouraged to work from home. The employer can keep track the attendance of the employees through this application, and this enable the employer to make evaluation of their employee and also calculate the pay for their employee. Besides, this application can replace the traditional method of taking attendance of card swiping and bring so much of benefit compare to the old days.

APPENDICES

APPENDIX A - CODE FOR PROJECT

APP.PY

```
from flask import Flask, request, jsonify, render_template
from flask_sqlalchemy import SQLAlchemy
import os

app = Flask(__name__)

app.config.from_object(os.environ['APP_SETTINGS'])
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

from models import Employee

@app.route("/")
def hello():
    return "DEAR EMPLOYEE PLEASE RECORD YOUR ATTENDANCE"

@app.route("/name/<name>")
def get_employee_name(name):
    return "name : {}".format(name)

@app.route("/details")
def get_employee_details():
    age=request.args.get('age')
    dept=request.args.get('dept')
    return "Age : {}, Dept: {}".format(age,dept)

@app.route("/add")
def add_employee():
    name=request.args.get('name')
    age=request.args.get('age')
    dept=request.args.get('dept')
    try:
        employee=Employee(
            name=name,
            age=age,
            dept=dept
        )
        db.session.add(employee)
        db.session.commit()
        return "Employee added. employee id={}".format(employee.id)
    except Exception as e:
        return(str(e))

@app.route("/add/form",methods=['GET', 'POST'])
def add_employee_form():
    if request.method == 'POST':
        name=request.form.get('name')
```

```

    age=request.form.get('age')
    dept=request.form.get('dept')
    try:
        employee=Employee(
            name=name,
            age=age,
            dept=dept
        )
        db.session.add(employee)
        db.session.commit()
        return "Employee added. employee id={ }".format(employee.id)
    except Exception as e:
        return(str(e))
    return render_template("getdata.html")

@app.route("/getall")
def get_all():
    try:
        employees=Employee.query.all()
        return jsonify([e.serialize() for e in employees])
    except Exception as e:
        return(str(e))

@app.route("/get/<id_>")
def get_by_id(id_):
    try:
        employee=Employee.query.filter_by(id=id_).first()
        return jsonify(book.serialize())
    except Exception as e:
        return(str(e))

if __name__ == '__main__':
    app.run()

```


CONFIG .PY

```
import os
basedir = os.path.abspath(os.path.dirname(__file__))

class Config(object):
    DEBUG = False
    TESTING = False
    CSRF_ENABLED = True
    SECRET_KEY = 'this-really-needs-to-be-changed'
    SQLALCHEMY_DATABASE_URI = os.environ['DATABASE_URL']

class ProductionConfig(Config):
    DEBUG = False

class StagingConfig(Config):
    DEVELOPMENT = True
    DEBUG = True

class DevelopmentConfig(Config):
    DEVELOPMENT = True
    DEBUG = True

class TestingConfig(Config):
    TESTING = True
```

MODELS .PY

```
from flask_sqlalchemy import SQLAlchemy
from app import db

class Employee(db.Model):
    __tablename__ = 'employee'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String())
    age = db.Column(db.String())
    dept = db.Column(db.String())

    def __init__(self, name, age, dept):
        self.name = name
        self.age = age
        self.dept = dept

    def __repr__(self):
        return '<id {}>'.format(self.id)

    def serialize(self):
        return {
            'id': self.id,
            'name': self.name,
            'age': self.age,
            'dept': self.dept
        }
```

MANAGE .PY

```
from flask_migrate import Migrate, MigrateCommand

from app import app, db

migrate = Migrate(app, db)
manager = Manager(app)

manager.add_command('db', MigrateCommand)

if __name__ == '__main__':
    manager.run()
```

Getdata.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
</head>

<body>
    <div class="container">

        <div class="container">
            <br>
            <br>

            <div class="row align-items-center justify-content-center">
                <h1>Add an employee</h1>
            </div>
            <br>

            <form method="POST">

                <label for="name">Employee Name</label>
                <div class="form-row">
                    <input class="form-control" type="text" placeholder="Name of Employee" id="name" name="name">
                </div>
                <br>
                <div class="form-row">
                    <label for="author">Age</label>
                    <input class="form-control" type="text" placeholder="Employee Age" id="age" name="age">
                </div>
                <br>
                <div class="form-row ">
                    <label for="published">Department</label>
                    <input class="form-control " type="text" placeholder="Dept" id="dept" name="dept">
                </div>
            </form>
        </div>
    </div>
```

```
<br>
<button type="submit " class="btn btn-primary " style="float:right ">Save</button>

</form>
<br><br>
</div>
</div>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js " integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo "
crossorigin="anonymous "></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js " integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIpM49 "
crossorigin="anonymous "></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js " integrity="sha384-
ChfqquxZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy "
crossorigin="anonymous "></script>
</body>
</html>
```