QA

Private Driving Accompany

Python App

Hua Hui Wang
09/12/2022

Contents

Background

This document is used to explain the fabrication of driving accompany by use the CICD technologies. The app allows the learner to practice their driving skill by find the private companion in a suitable area and time. At the same time allow the people with more than 3 years of driving experience to gain some income within their spare time.

Project management

User Story

The user story is used to generate the system requirement for design the basic system features and functions.

1. Customer

1) As customer, I want to keep my account safe, so I need a login page to keep all my information.

2) As customer, I want to view the private instructor or deals, so that I can choose which lesson I would like to attend.

3) As customer, I want to create, update, and delete a request for a lesson, so that I can make extra fees to take my time off on lesson searching.

4) As customer, I want to view the review of driver, so that I can choose the reliable and skilful instructor.

5) As customer, I want to write, update and delete the review of driver, so that driver's performance can be develop and hold accountable.

6) As customer, I want to become instructor, so I can offer driving accompany to make revenue.

2. Instructor

1) As instructor, I want to view the request in my area, so that I can provide service to the pupil I want.

2) As instructor, I want to create, update and delete my lesson in the area, so that pupil can follow my schedule.

3. Admin

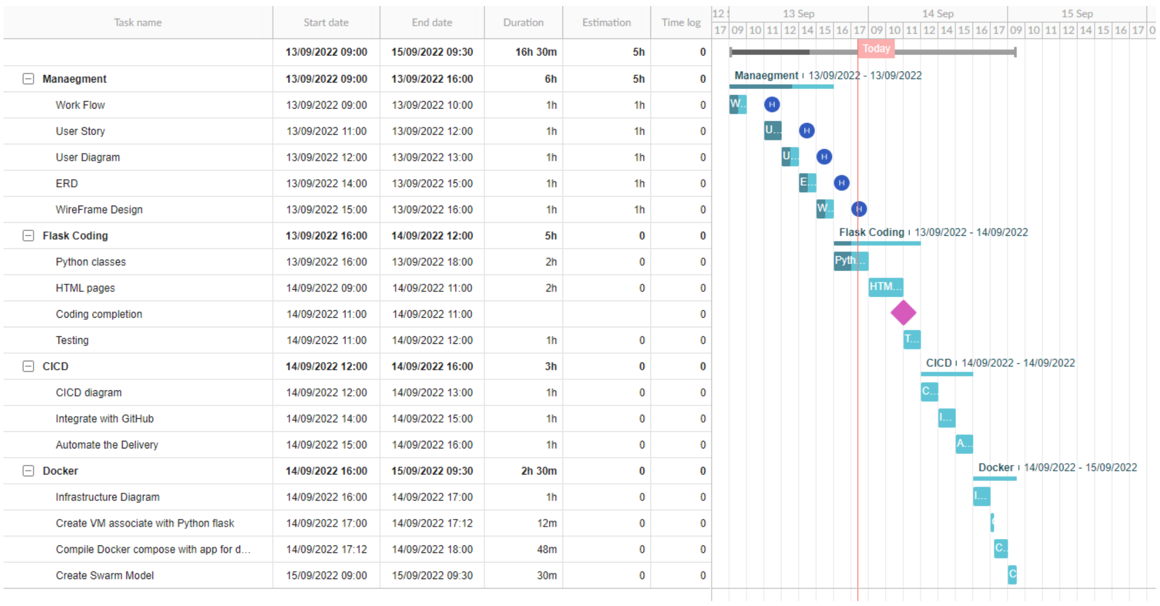
1) As Admin, I want to approve the driver who is qualified for the instructor in different level, so that pupil is in good hand.

2) As Admin, I want to approve the pupil is covered with proper insurance before the lesson started, so that regulation and law is followed.

3) As admin, I want to solve the dispute in the driving lesson, so that fair and justice can be reserved.

Workflow

This section shows the planning of whole project to meet limit time. The URL included.



Task name	Start date	End date	Duration	Estimation	Time log
	13/09/2022 09:00	**15/09/2022 09:30**	**16h 30m**	**5h**	**0**
Manaegment	**13/09/2022 09:00**	**13/09/2022 16:00**	**6h**	**5h**	**0**
Work Flow	13/09/2022 09:00	13/09/2022 10:00	1h	1h	0
User Story	13/09/2022 11:00	13/09/2022 12:00	1h	1h	0
User Diagram	13/09/2022 12:00	13/09/2022 13:00	1h	1h	0
ERD	13/09/2022 14:00	13/09/2022 15:00	1h	1h	0
WireFrame Design	13/09/2022 15:00	13/09/2022 16:00	1h	1h	0
Flask Coding	**13/09/2022 16:00**	**14/09/2022 12:00**	**5h**	**0**	**0**
Python classes	13/09/2022 16:00	13/09/2022 18:00	2h	0	0
HTML pages	14/09/2022 09:00	14/09/2022 11:00	2h	0	0
Coding completion	14/09/2022 11:00	14/09/2022 11:00			0
Testing	14/09/2022 11:00	14/09/2022 12:00	1h	0	0
CICD	**14/09/2022 12:00**	**14/09/2022 16:00**	**3h**	**0**	**0**
CICD diagram	14/09/2022 12:00	14/09/2022 13:00	1h	0	0
Integrate with GitHub	14/09/2022 14:00	14/09/2022 15:00	1h	0	0
Automate the Delivery	14/09/2022 15:00	14/09/2022 16:00	1h	0	0
Docker	**14/09/2022 16:00**	**15/09/2022 09:30**	**2h 30m**	**0**	**0**
Infrastructure Diagram	14/09/2022 16:00	14/09/2022 17:00	1h	0	0
Create VM associate with Python flask	14/09/2022 17:00	14/09/2022 17:12	12m	0	0
Compile Docker compose with app for d…	14/09/2022 17:12	14/09/2022 18:00	48m	0	0
Create Swarm Model	15/09/2022 09:00	15/09/2022 09:30	30m	0	0

13 Sep
14 Sep
15 Sep
Today
Manaegment | 13/09/2022 - 13/09/2022
Flask Coding | 13/09/2022 - 14/09/2022
CICD | 14/09/2022 - 14/09/2022
Docker | 14/09/2022 - 15/09/2022

ERD Diagram

This section shows the entity relationship diagram of project which allows the customer to check the lessons online, and to be approved for a new instructor base the evidence provided. Moreover, be able to write the interview about the driver.

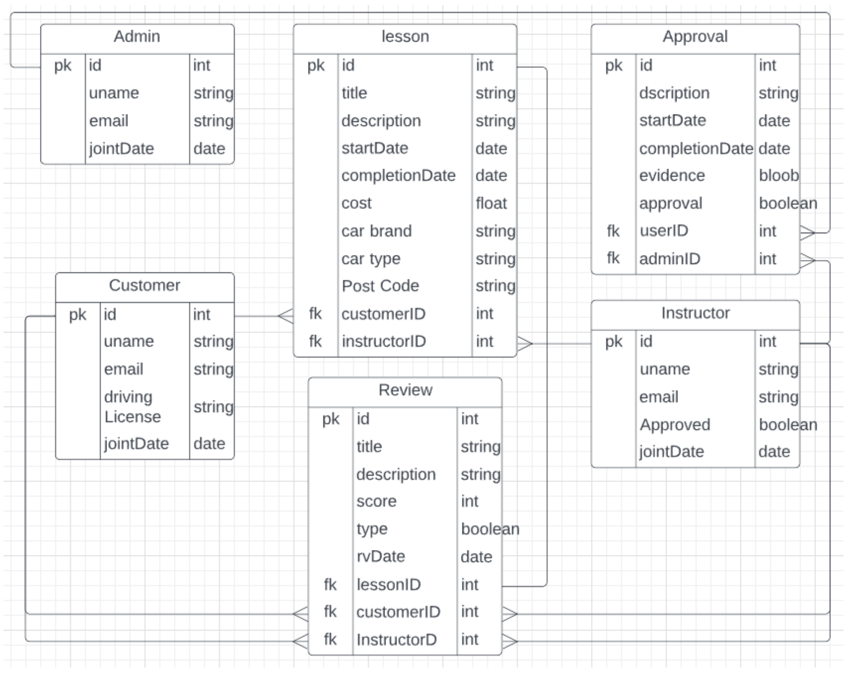
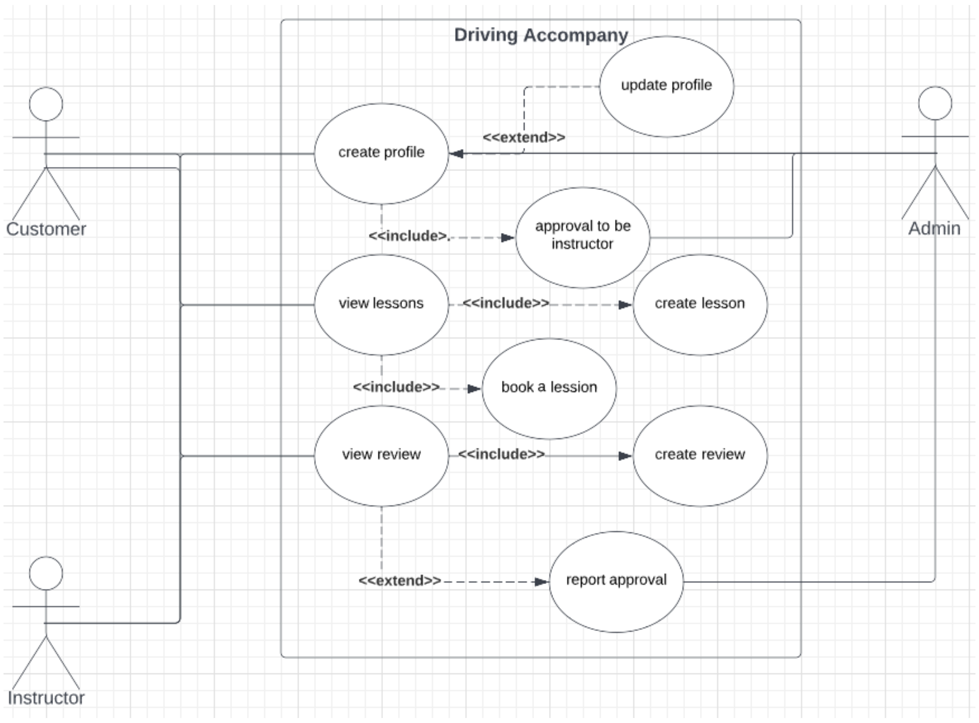


Figure 1 Database ERD

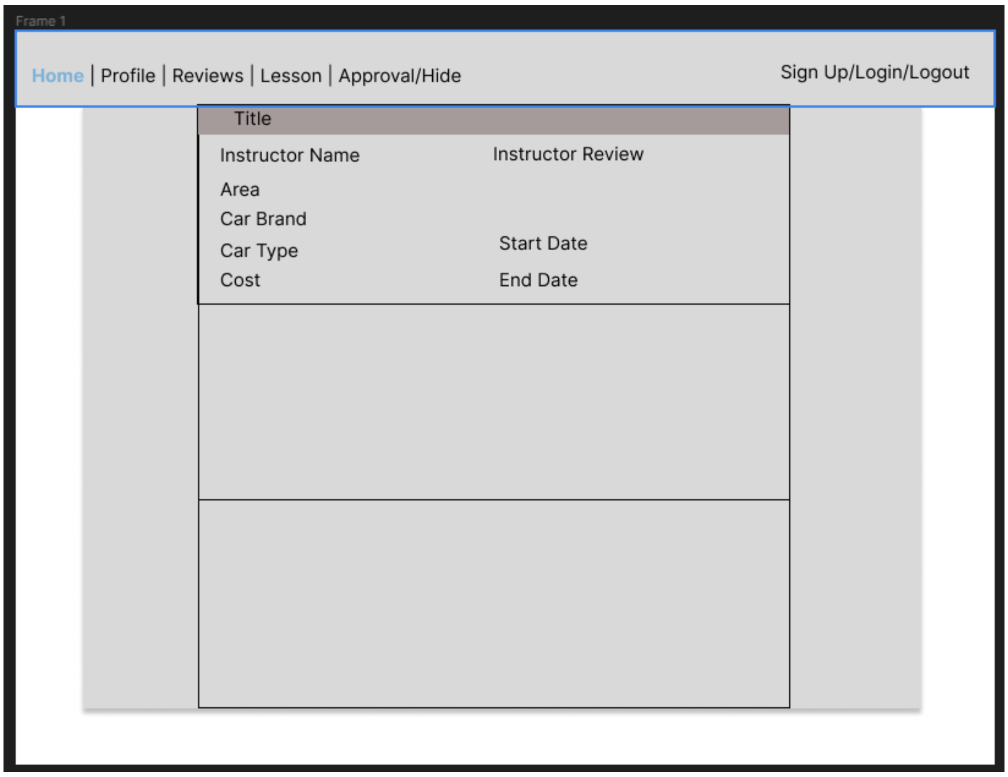
Use Case Diagram

The following image show how users interact within the system in a basic level.

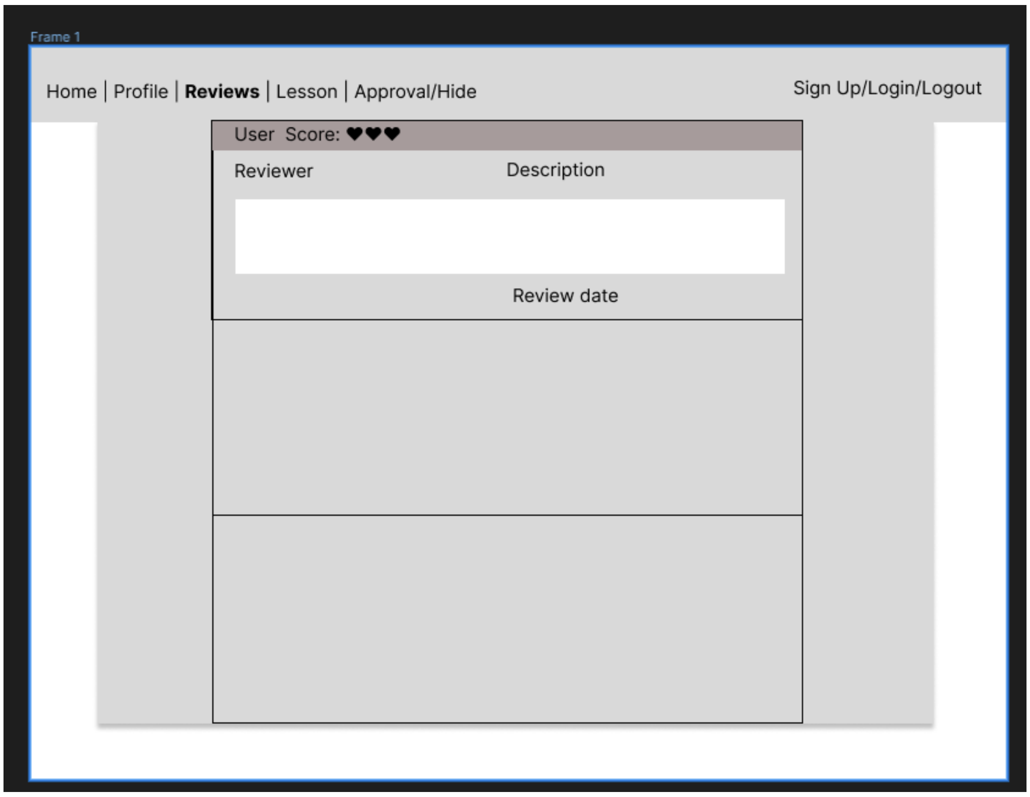


WireFrame Diagram

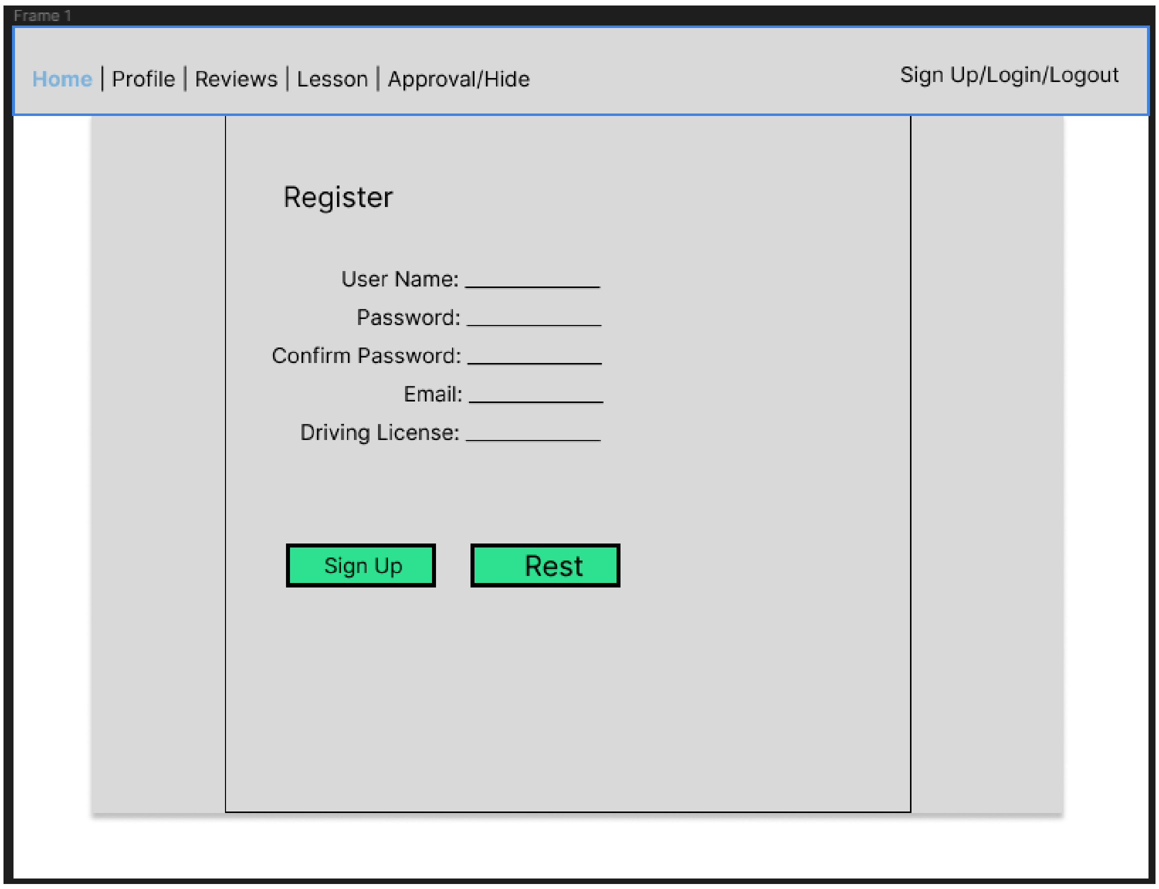
1. Lessons Page



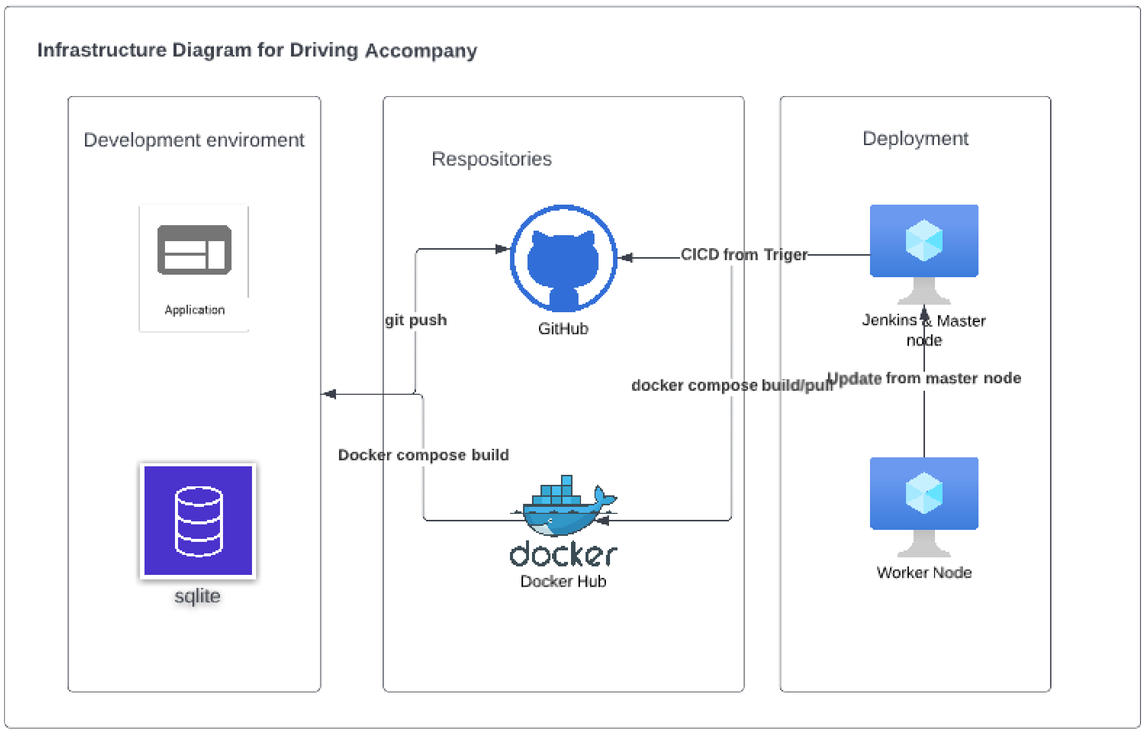
2. Review Page



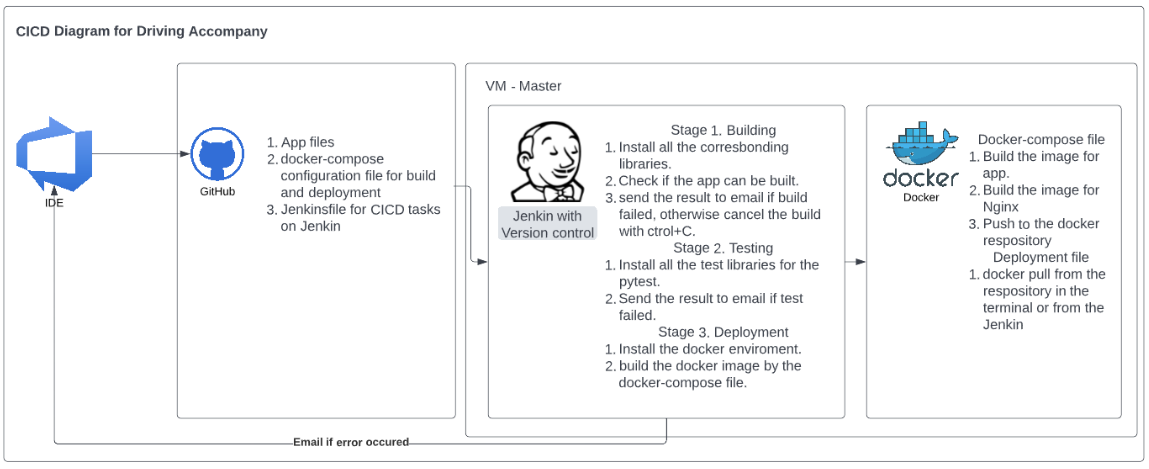
3. Sign Up Page



Infrastructure Diagram



CICD Diagram


CICD Diagram for Driving Accompany
IDE
GitHub
VM - Master
Jenkin with Version control
docker
Docker
Email if error occured

1. App files
2. docker-compose configuration file for build and deployment
3. Jenkinsfile for CICD tasks on Jenkin

Stage 1. Building

1. Install all the corresbonding libraries.
2. Check if the app can be built.
3. send the result to email if build failed, otherwise cancel the build with ctrol+C.

Stage 2. Testing

1. Install all the test libraries for the pytest.
2. Send the result to email if test failed.

Stage 3. Deployment

1. Install the docker enviroment.
2. build the docker image by the docker-compose file.

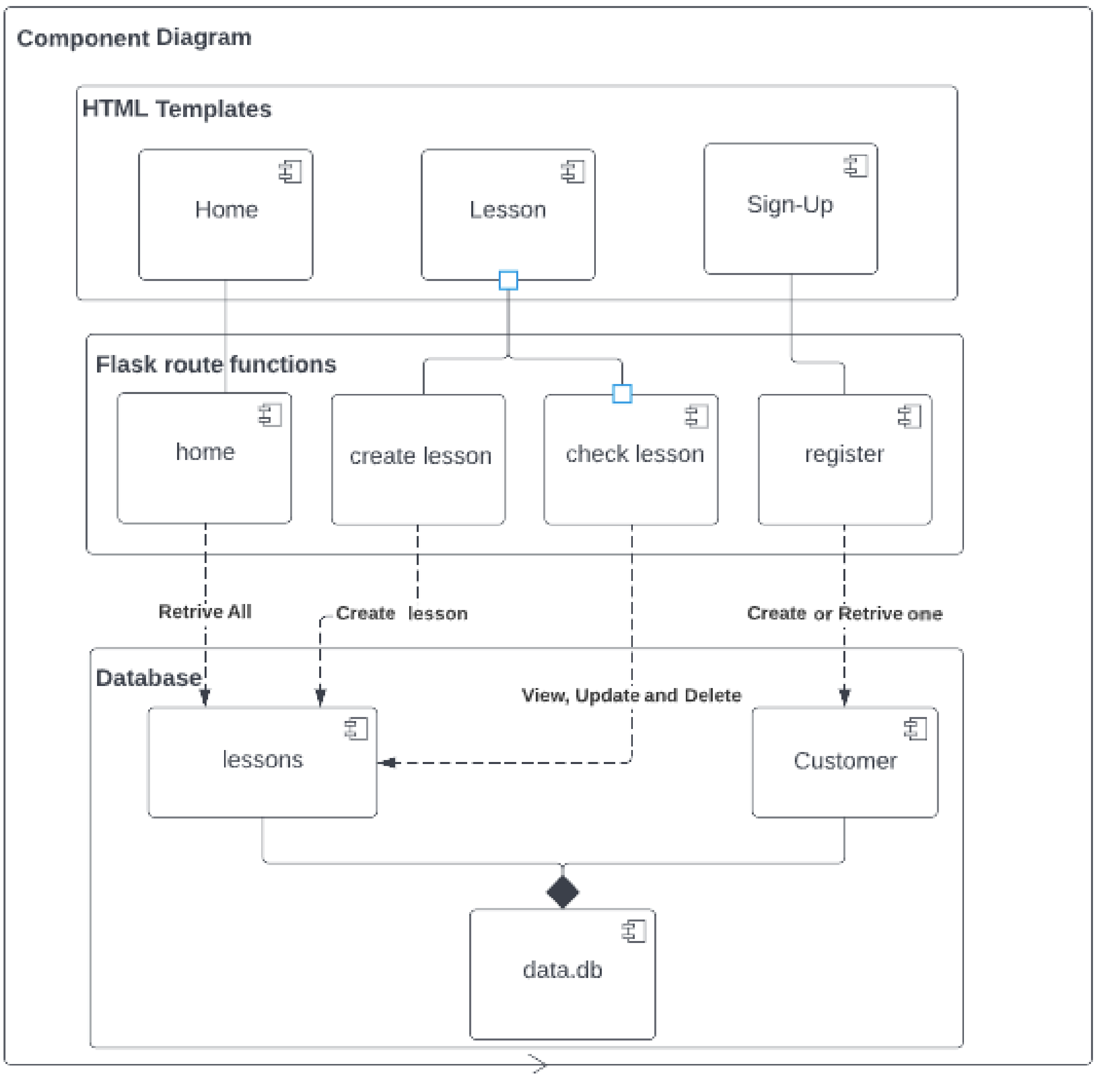
Docker-compose file

1. Build the image for app.
2. Build the image for Nginx
3. Push to the docker respository

Deployment file

1. docker pull from the respository in the terminal or from the Jenkin

Component Diagram



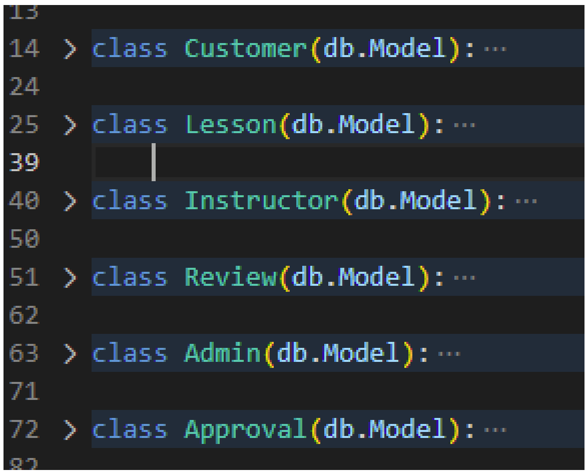
Python Flask

Functions – MVC

I have separated the functions into MVC model which include model, view and controller modules.

1. Model

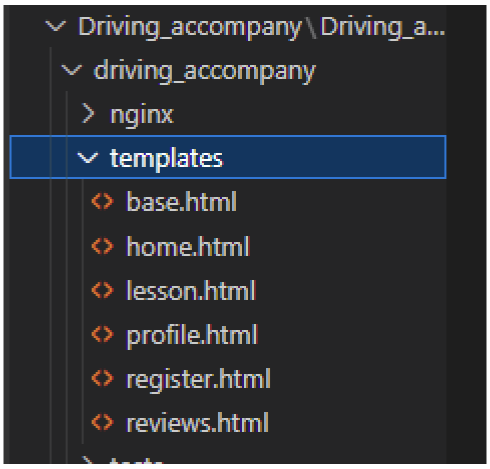
In the model, I have created my database entities with different classes according to Figure 1 Database ERD. In this case, I can manage my different database entries easier and effective. At same time, I can utilise the python libraries such as database or flask to make the application more persistent.



```
class Customer(db.Model): ···

class Lesson(db.Model): ···

class Instructor(db.Model): ···

class Review(db.Model): ···

class Admin(db.Model): ···

class Approval(db.Model): ···
```

2. View

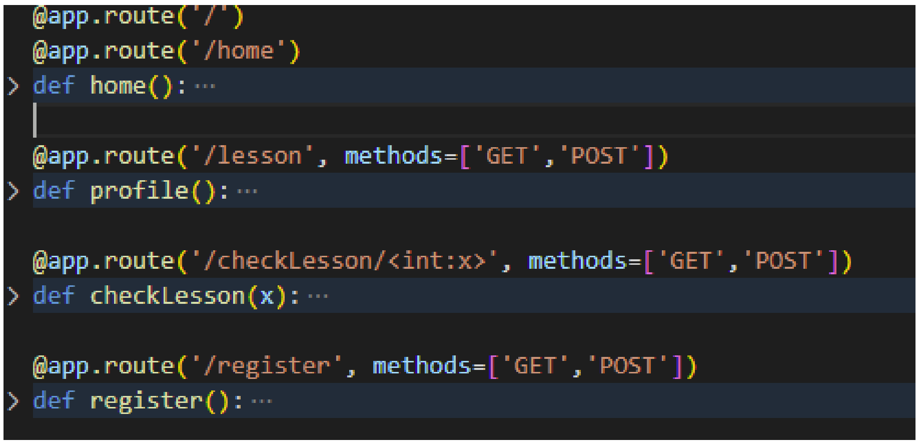
For the view, I have created the GUI from html files to allow the user to view the data from database. These not only give me the idea of data flow, but also allow me to change the code or logics to make it more adaptable and efficient. Meanwhile, make one of the html reusable for some other task.



```
Driving_accompany\Driving_a...
  driving_accompany
    nginx
    templates
      base.html
      home.html
      lesson.html
      profile.html
      register.html
      reviews.html
```

3. Controller

For the controller, the route features have been used to allow the user to view, create, update and delete the data in the application.



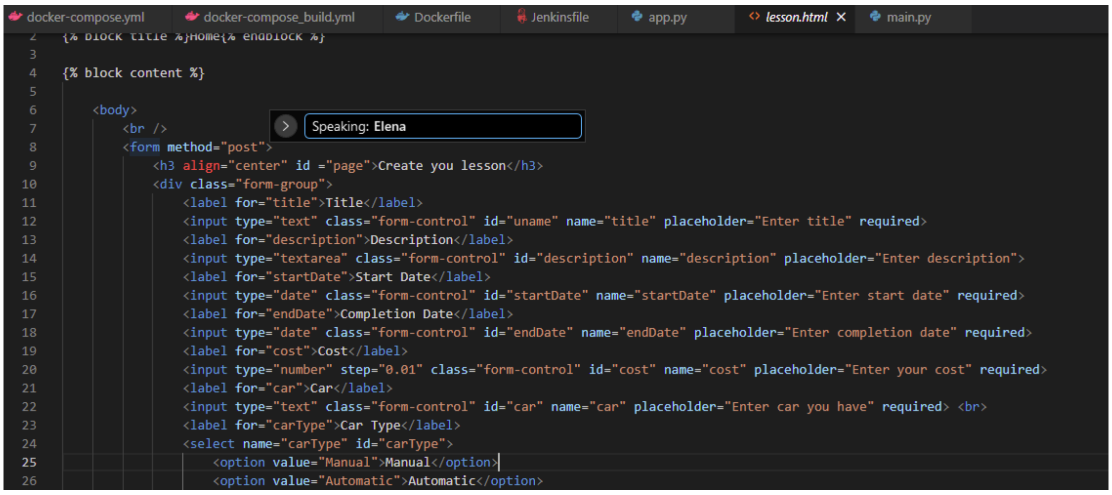
```
@app.route('/')
@app.route('/home')
def home(): ···

@app.route('/lesson', methods=['GET','POST'])
def profile(): ···

@app.route('/checkLesson/<int:x>', methods=['GET','POST'])
def checkLesson(x): ···

@app.route('/register', methods=['GET','POST'])
def register(): ···
```

Input Validation

The validation of input has been used in the app to make sure the necessary information have taken, as well as the correct information has been taken to use the application.

1. Client-side validation

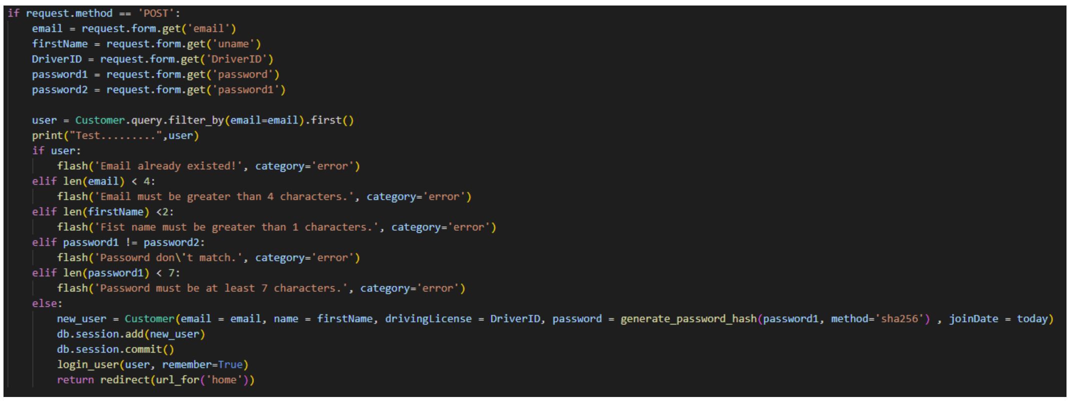
The client side validation has been done by the required attribute in the input field which it checks the user's input in their device.



```
{% block title %}Home{% endblock %}

{% block content %}

    <body>
        <br />
        <form method="post">
            <h3 align="center" id ="page">Create you lesson</h3>
            <div class="form-group">
                <label for="title">Title</label>
                <input type="text" class="form-control" id="uname" name="title" placeholder="Enter title" required>
                <label for="description">Description</label>
                <input type="textarea" class="form-control" id="description" name="description" placeholder="Enter description">
                <label for="startDate">Start Date</label>
                <input type="date" class="form-control" id="startDate" name="startDate" placeholder="Enter start date" required>
                <label for="endDate">Completion Date</label>
                <input type="date" class="form-control" id="endDate" name="endDate" placeholder="Enter completion date" required>
                <label for="cost">Cost</label>
                <input type="number" step="0.01" class="form-control" id="cost" name="cost" placeholder="Enter your cost" required>
                <label for="car">Car</label>
                <input type="text" class="form-control" id="car" name="car" placeholder="Enter car you have" required> <br>
                <label for="carType">Car Type</label>
                <select name="carType" id="carType">
                    <option value="Manual">Manual</option>
                    <option value="Automatic">Automatic</option>
```

2. Server-side validation

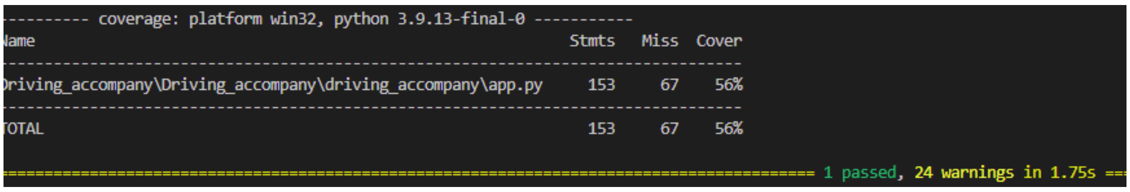
The server-side validation will check the input in the server-side or database before create an entity in the database.



```
if request.method == 'POST':
    email = request.form.get('email')
    firstName = request.form.get('uname')
    DriverID = request.form.get('DriverID')
    password1 = request.form.get('password')
    password2 = request.form.get('password1')

    user = Customer.query.filter_by(email=email).first()
    print("Test.........",user)
    if user:
        flash('Email already existed!', category='error')
    elif len(email) < 4:
        flash('Email must be greater than 4 characters.', category='error')
    elif len(firstName) <2:
        flash('Fist name must be greater than 1 characters.', category='error')
    elif password1 != password2:
        flash('Passowrd don\'t match.', category='error')
    elif len(password1) < 7:
        flash('Password must be at least 7 characters.', category='error')
    else:
        new_user = Customer(email = email, name = firstName, drivingLicense = DriverID, password = generate_password_hash(password1, method='sha256') , joinDate = today)
        db.session.add(new_user)
        db.session.commit()
        login_user(user, remember=True)
        return redirect(url_for('home'))
```

Testing

I only did the view test which the testing application create a new lesson class with title and check it in the home page with its html file. The testing is important that I can use in the CICD pipeline or in other automation task.

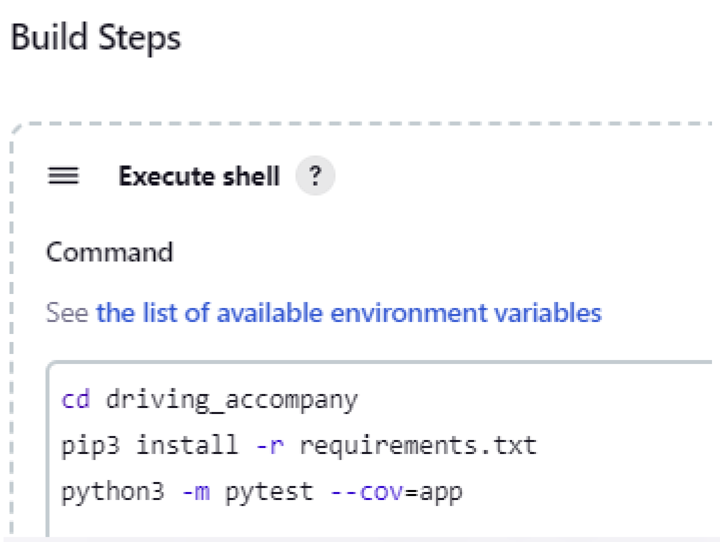


```
---------- coverage: platform win32, python 3.9.13-final-0 -----------
ame                                                     Stmts   Miss  Cover
---------------------------------------------------------------------------
riving_accompany\Driving_accompany\driving_accompany\app.py     153     67    56%
---------------------------------------------------------------------------
OTAL                                                      153     67    56%

============================================================================ 1 passed, 24 warnings in 1.75s ==
```

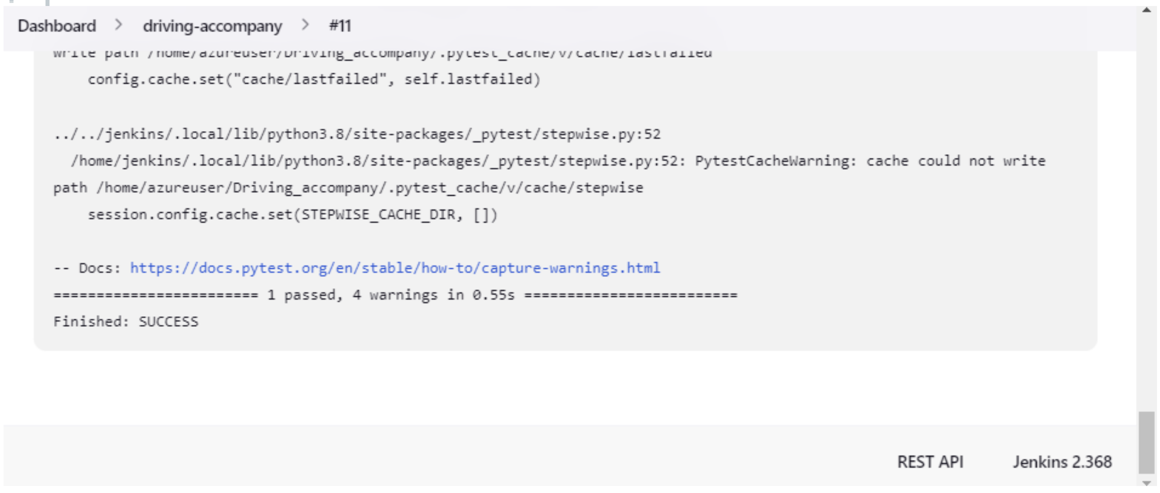
CICD

Jenkins freestyle project

In this section, I have created a testing environment with my GitHub project. Firstly, the Jenkins will need to install the python and pytest libraries, and test the app' view like previous section to make sure the app is successfully built.



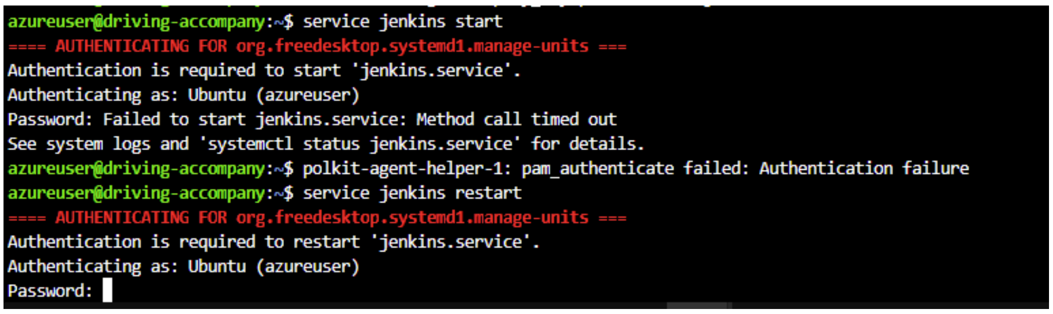
```
cd driving_accompany
pip3 install -r requirements.txt
python3 -m pytest --cov=app
```



```
write path /home/azureuser/Driving_accompany/.pytest_cache/v/cache/lastfailed
    config.cache.set("cache/lastfailed", self.lastfailed)

../../jenkins/.local/lib/python3.8/site-packages/_pytest/stepwise.py:52
  /home/jenkins/.local/lib/python3.8/site-packages/_pytest/stepwise.py:52: PytestCacheWarning: cache could not write
path /home/azureuser/Driving_accompany/.pytest_cache/v/cache/stepwise
    session.config.cache.set(STEPWISE_CACHE_DIR, [])

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================== 1 passed, 4 warnings in 0.55s ========================
Finished: SUCCESS
```

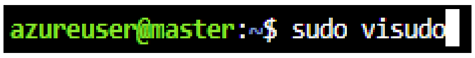
Jenkins pipeline

Jenkins pipeline allow us to create multiple tasks in the same time, but it come with authentication problem with some application which it like image down below.

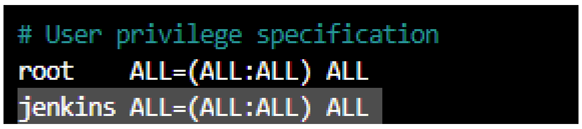


```
azureuser@driving-accompany:~$ service jenkins start
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to start 'jenkins.service'.
Authenticating as: Ubuntu (azureuser)
Password: Failed to start jenkins.service: Method call timed out
See system logs and 'systemctl status jenkins.service' for details.
azureuser@driving-accompany:~$ polkit-agent-helper-1: pam_authenticate failed: Authentication failure
azureuser@driving-accompany:~$ service jenkins restart
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to restart 'jenkins.service'.
Authenticating as: Ubuntu (azureuser)
Password:
```

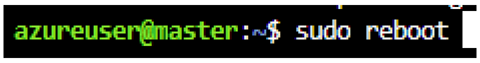
The solution is add the jenkins to the super user group without the need for password for root operation, to reboot the VM to make sure the immediate affect.



```
azureuser@master:~$ sudo visudo
```

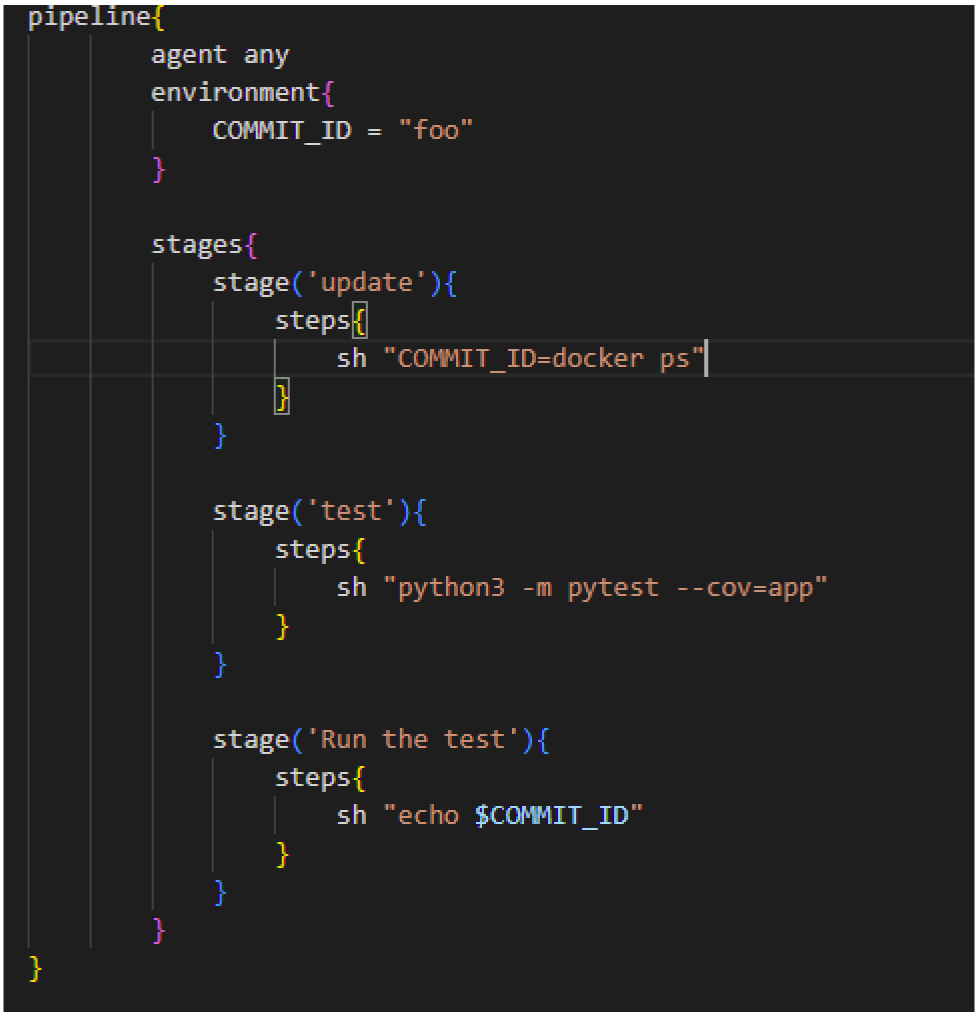


```
# User privilege specification
root    ALL=(ALL:ALL) ALL
jenkins ALL=(ALL:ALL) ALL
```

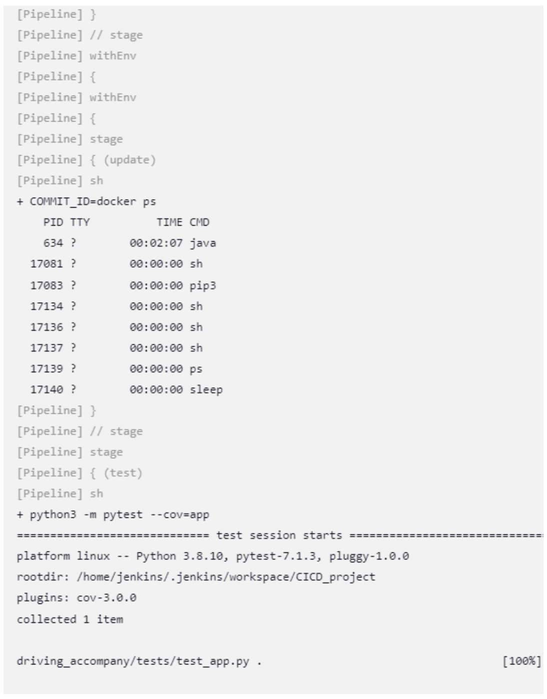


```
azureuser@master:~$ sudo reboot
```

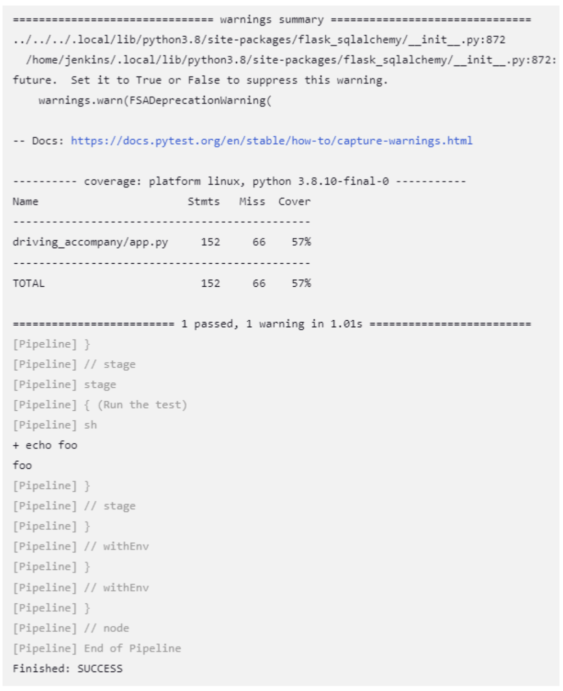
For the image downbelow, I was trying to connect with different stages with variable, but the result shows the stages works like class in programming which the variable will be identicaled to the gloable varialble even the variable has been altered.



```
pipeline{
        agent any
        environment{
            COMMIT_ID = "foo"
        }

        stages{
            stage('update'){
                steps{
                    sh "COMMIT_ID=docker ps"
                }
            }

            stage('test'){
                steps{
                    sh "python3 -m pytest --cov=app"
                }
            }

            stage('Run the test'){
                steps{
                    sh "echo $COMMIT_ID"
                }
            }
        }
}
```



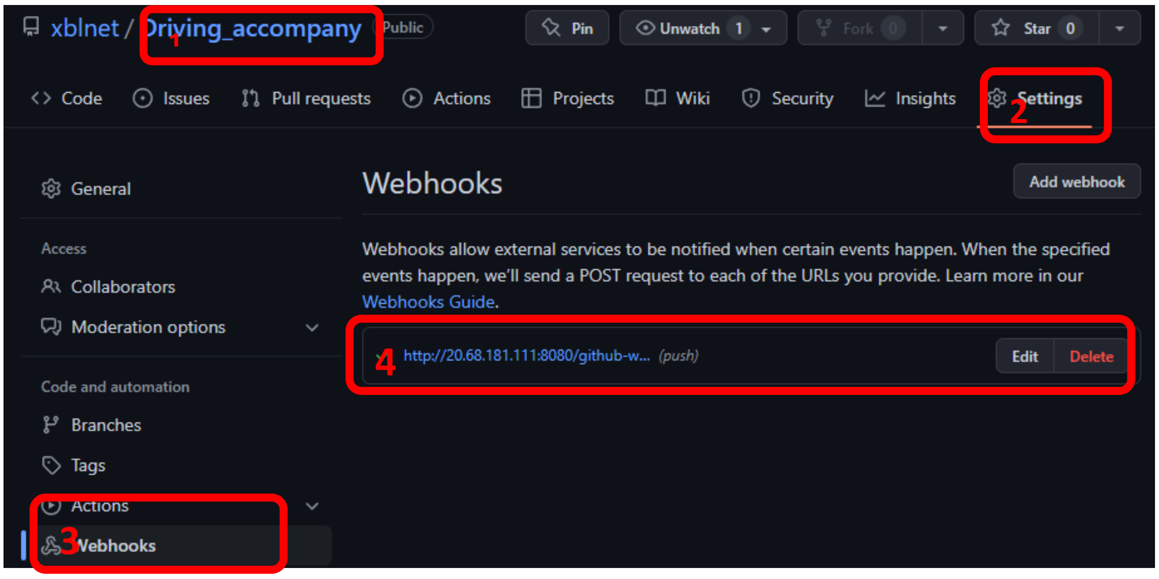
```
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (update)
[Pipeline] sh
+ COMMIT_ID=docker ps
    PID TTY          TIME CMD
    634 ?        00:02:07 java
  17081 ?        00:00:00 sh
  17083 ?        00:00:00 pip3
  17134 ?        00:00:00 sh
  17136 ?        00:00:00 sh
  17137 ?        00:00:00 sh
  17139 ?        00:00:00 ps
  17140 ?        00:00:00 sleep
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (test)
[Pipeline] sh
+ python3 -m pytest --cov=app
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-7.1.3, pluggy-1.0.0
rootdir: /home/jenkins/.jenkins/workspace/CICD_project
plugins: cov-3.0.0
collected 1 item

driving_accompany/tests/test_app.py .                                    [100%]
```



```
=============================== warnings summary ===============================
../../../.local/lib/python3.8/site-packages/flask_sqlalchemy/__init__.py:872
  /home/jenkins/.local/lib/python3.8/site-packages/flask_sqlalchemy/__init__.py:872:
future.  Set it to True or False to suppress this warning.
    warnings.warn(FSADeprecationWarning(

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html

---------- coverage: platform linux, python 3.8.10-final-0 -----------
Name                          Stmts   Miss  Cover
-------------------------------------------------
driving_accompany/app.py        152     66    57%
-------------------------------------------------
TOTAL                           152     66    57%

========================= 1 passed, 1 warning in 1.01s =========================
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run the test)
[Pipeline] sh
+ echo foo
foo
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Automatic Trigger

There are two automatic triggers in Jenkins, one trigger periodically which it is inefficient for the GitHub. Another one is only triggered once it sensed the update on the GitHub. For the automatic trigger from GitHub update I have to set the Webbook in GitHub project and configure my Jenkins project setting to achieve this function.

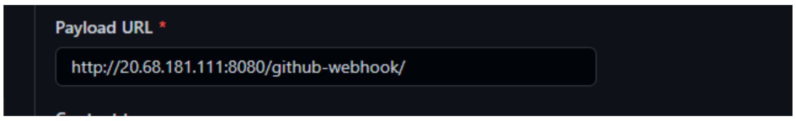


Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

http://20.68.181.111:8080/github-w... (push)

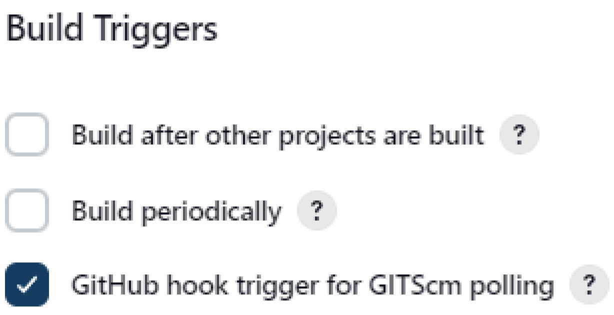
Setup in the GitHub project setting page.



Payload URL *

http://20.68.181.111:8080/github-webhook/

Add the your Jenkins URL to the webhook. Otherwise, set default and save.


Build Triggers

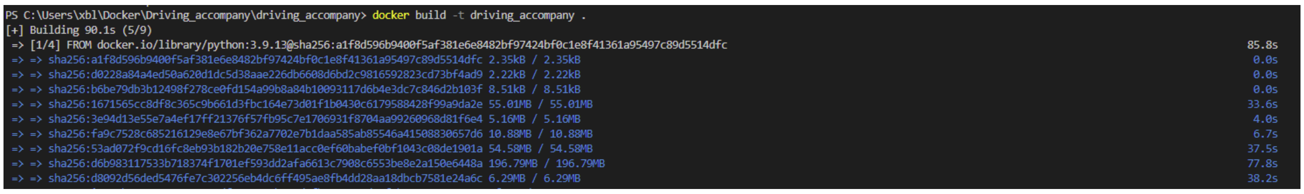
- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling

In your Jenkins project configuration section, checked the option from image on top to achieve automatic triggers.

Docker

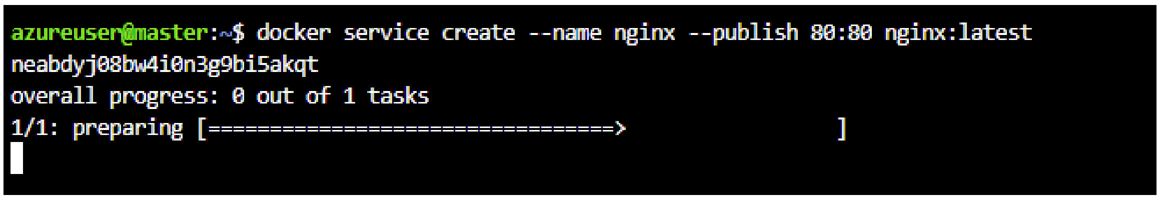
Build and push to the Docker Hub

In this section, I have to build the docker image with Nginx by docker-compose and deploy the images by docker stack deploy function with the yaml file. Unfortunately, It cannot display the app in the swarm manager VM which it might need the time to tweak some of setting. Meanwhile, I have built the app without the Nginx, but it still failed. For this instance, I will just demonstrate the default Nginx in the swarm for the short deadline.



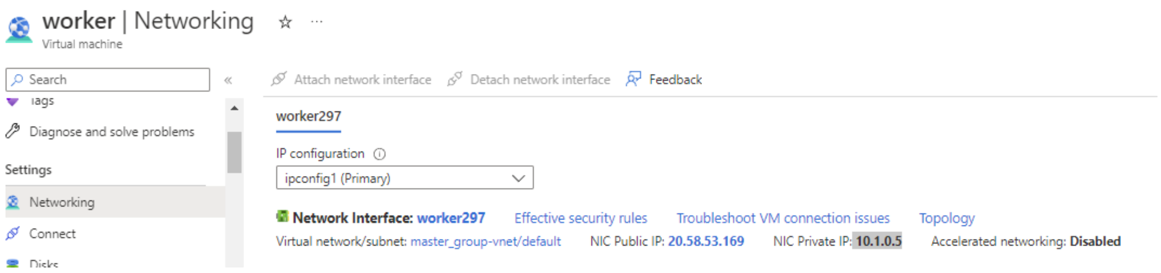
```
PS C:\Users\xbl\Docker\Driving_accompany\driving_accompany> docker build -t driving_accompany .
[+] Building 90.1s (5/9)
 => [1/4] FROM docker.io/library/python:3.9.13@sha256:a1f8d596b9400f5af381e6e8482bf97424bf0c1e8f41361a95497c89d5514dfc                    85.8s
 => => sha256:a1f8d596b9400f5af381e6e8482bf97424bf0c1e8f41361a95497c89d5514dfc 2.35kB / 2.35kB                                              0.0s
 => => sha256:d0228a84a4ed50a620d1dc5d38aae226db6608d6bd2c9816592823cd73bf4ad9 2.22kB / 2.22kB                                              0.0s
 => => sha256:b6be79db3b12498f278ce0fd154a99b8a84b10093117d6b4e3dc7c846d2b103f 8.51kB / 8.51kB                                              0.0s
 => => sha256:1671565cc8df8c365c9b661d3fbc164e73d01f1b0430c6179588428f99a9da2e 55.01MB / 55.01MB                                           33.6s
 => => sha256:3e94d13e55e7a4ef17ff21376f57fb95c7e1706931f8704aa99260968d81f6e4 5.16MB / 5.16MB                                              4.0s
 => => sha256:fa9c7528c685216129e8e67bf362a7702e7b1daa585ab85546a41508830657d6 10.88MB / 10.88MB                                            6.7s
 => => sha256:53ad072f9cd16fc8eb93b182b20e758e11acc0ef60babef0bf1043c08de1901a 54.58MB / 54.58MB                                           37.5s
 => => sha256:d6b983117533b718374f1701ef593dd2afa6613c7908c6553be8e2a150e6448a 196.79MB / 196.79MB                                         77.8s
 => => sha256:d8092d56ded5476fe7c302256eb4dc6ff495ae8fb4dd28aa18dbcb7581e24a6c 6.29MB / 6.29MB                                              38.2s
```

Build image from the docker-compose.yml file



```
azureuser@master:~$ docker service create --name nginx --publish 80:80 nginx:latest
neabdyj08bw4i0n3g9bi5akqt
overall progress: 0 out of 1 tasks
1/1: preparing [=================================>                 ]
```

Download the default Nginx docker file

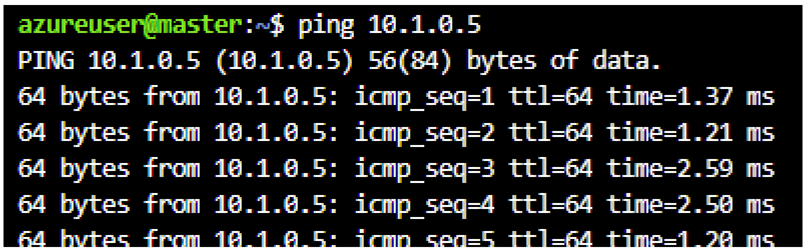


IP configuration: ipconfig1 (Primary)

Network Interface: worker297

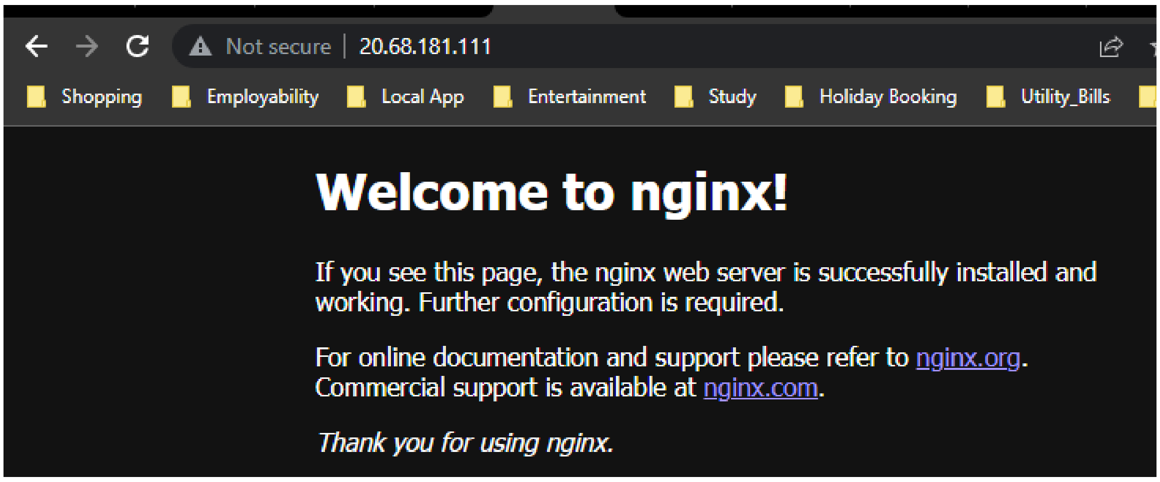
Virtual network/subnet: master_group-vnet/default   NIC Public IP: 20.58.53.169   NIC Private IP: 10.1.0.5   Accelerated networking: Disabled

Check the work's private IP



```
azureuser@master:~$ ping 10.1.0.5
PING 10.1.0.5 (10.1.0.5) 56(84) bytes of data.
64 bytes from 10.1.0.5: icmp_seq=1 ttl=64 time=1.37 ms
64 bytes from 10.1.0.5: icmp_seq=2 ttl=64 time=1.21 ms
64 bytes from 10.1.0.5: icmp_seq=3 ttl=64 time=2.59 ms
64 bytes from 10.1.0.5: icmp_seq=4 ttl=64 time=2.50 ms
64 bytes from 10.1.0.5: icmp_seq=5 ttl=64 time=1.20 ms
```

Pin the work’s IP to make sure it’s in the same private network



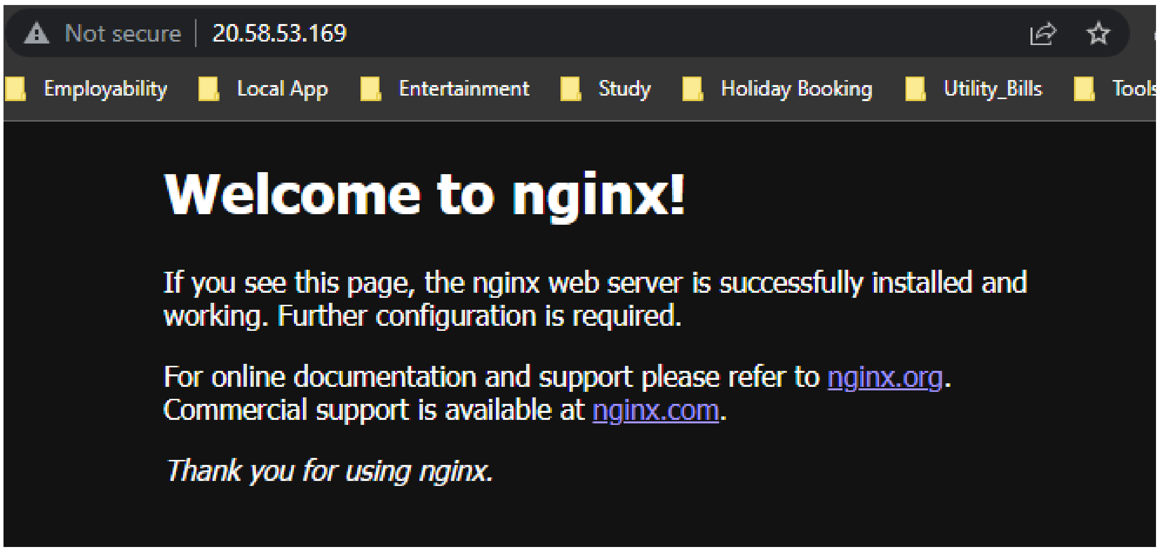
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Access the Nginx in master node.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Access the Nginx in the work node.

Bibliography

shubham, n.d. *How to give Jenkins super user permission.* [Online]
Available at: https://www.edureka.co/community/39390/how-to-give-jenkins-super-user-

permission
[Accessed 18 09 2022].