**Readme Server**

# **Server**

**Kebutuhan Server:**

1. Flask
2. SQL Alchemy
3. Python 3
4. XAMPP
5. MariaDB versi 10.2.10
6. Pyopenssl
7. Ngrok

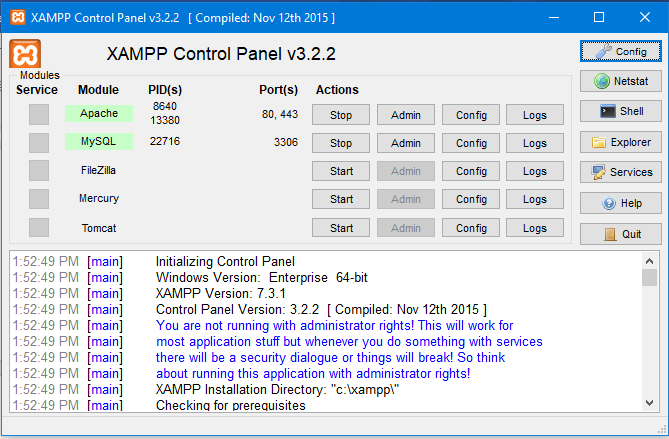
**Konfigurasi Server:**

1. **Lakukan Instalasi XAMPP**
2. **Kemudian upgrade MariaDB ke versi 10.2.10 Pada XAMPP**

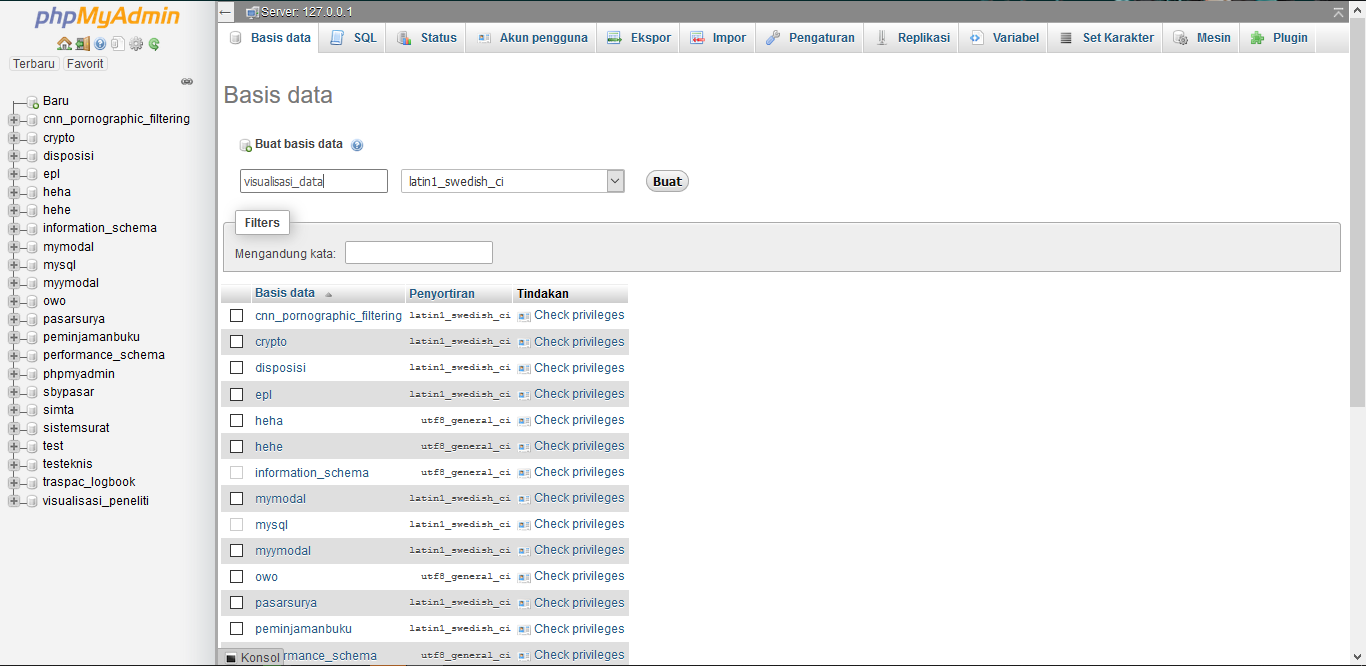
Tujuan upgrade MariaDB adalah untuk menggunakan fungsi Rank() pada MySQL. Berikut adalah langkah-langkah update MariaDB

* 1. Download versi **ZIP** [MariaDB](https://downloads.mariadb.org/mariadb/10.2.10/)
  2. Rename xampp/mysql folder menjadi mysql\_old sebagai backup agar jika terjadi masalah dapat dikembalikan.
  3. **Unzip** atau**Extract** MariaDB.zip ke folder XAMPP.
  4. Rename folder MariaDB menjadi mysql.
  5. Rename xampp/mysql/data menjadi data\_old.
  6. Copy xampp/mysql\_old/data folder ke xampp/mysql/.
  7. Copy xampp/mysql\_old/backup folder ke xampp/mysql/.
  8. Copy xampp/mysql\_old/scripts folder ke xampp/mysql/.
  9. Copy mysql\_uninstallservice.bat and mysql\_installservice.bat dari xampp/mysql\_old/ into xampp/mysql/.
  10. Copy xampp/mysql\_old/bin/my.ini kedalam xampp/mysql/bin.
  11. Edit xampp/mysql/bin/my.ini menggunakan notepad. cari **skip-federated** dan berikan prefix #. Save dan exit editor.
  12. Start-up XAMPP.
  13. Jalankan xampp/mysql/bin/mysql\_upgrade.exe.
  14. Shutdown and restart MariaDB (MySQL)

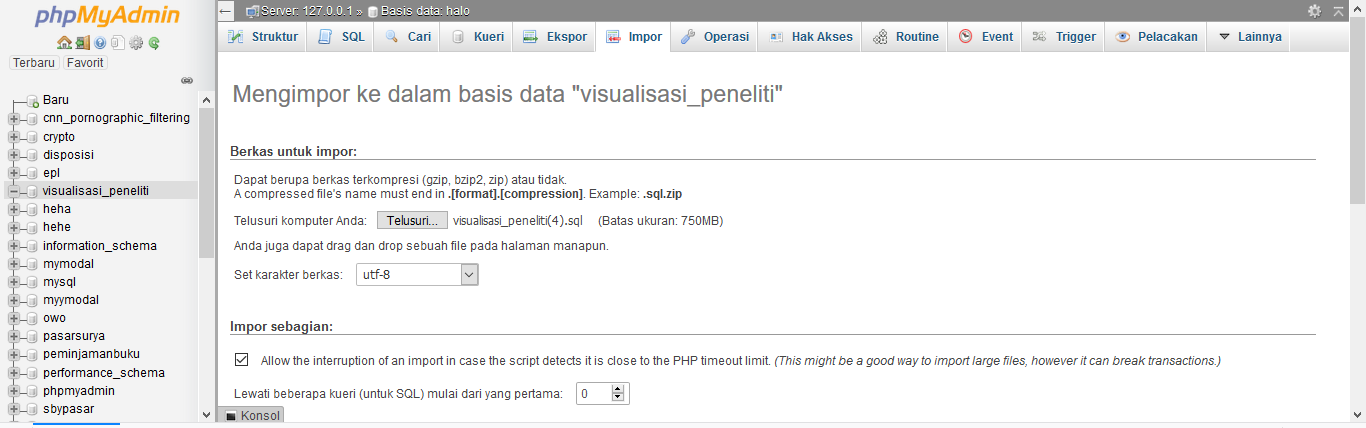
1. **Jalankan Apache dan MySQL**

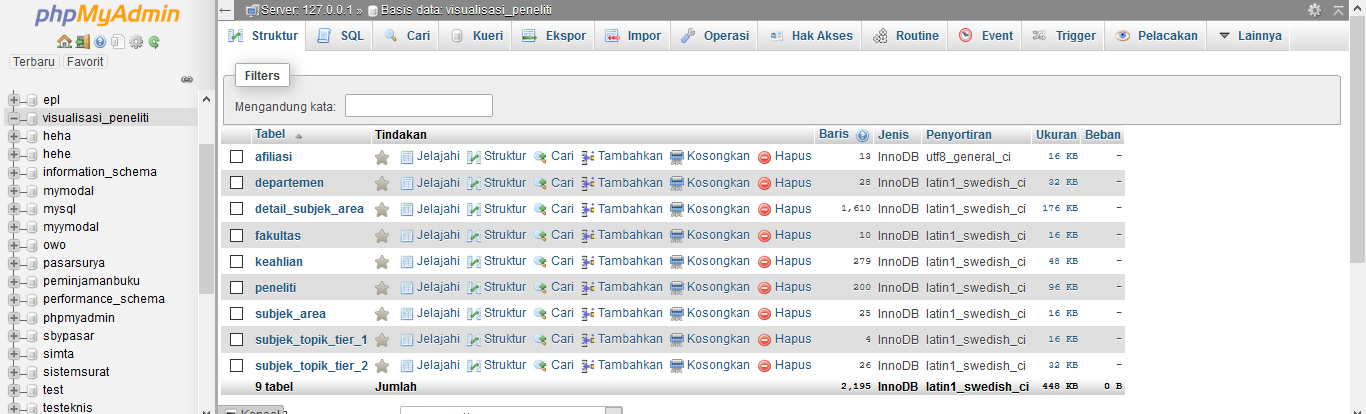


1. **Bukalah *PHP MyAdmin* dengan mengisikan** [**http://localhost/phpmyadmin/**](http://localhost/phpmyadmin/) **pada browser.**
2. **Import Database *PHP MyAdmin .***
   1. Pertama Buatlah Database Pada *PHP MyAdmin*



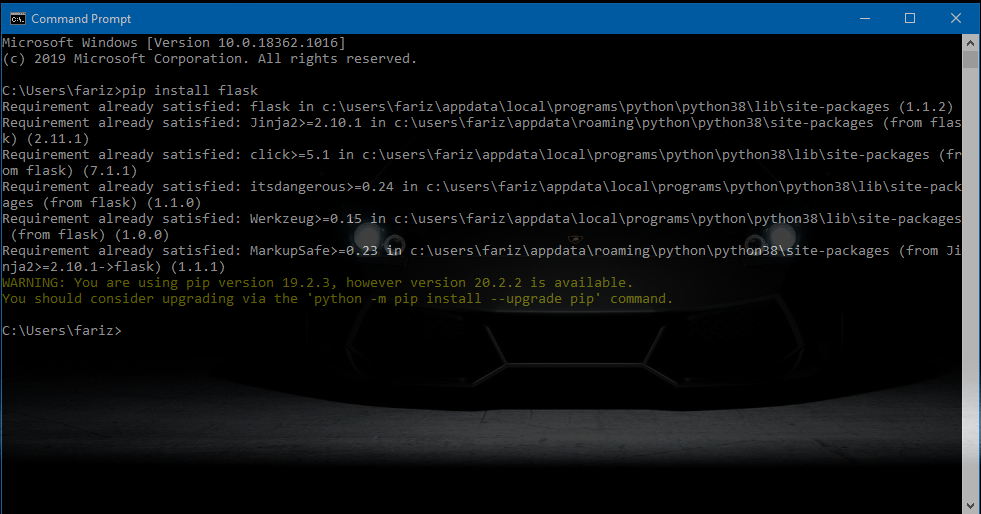
* 1. Kemudian importlah file sql visualisasi\_peneliti.sql yang telah disediakan pada folder database.





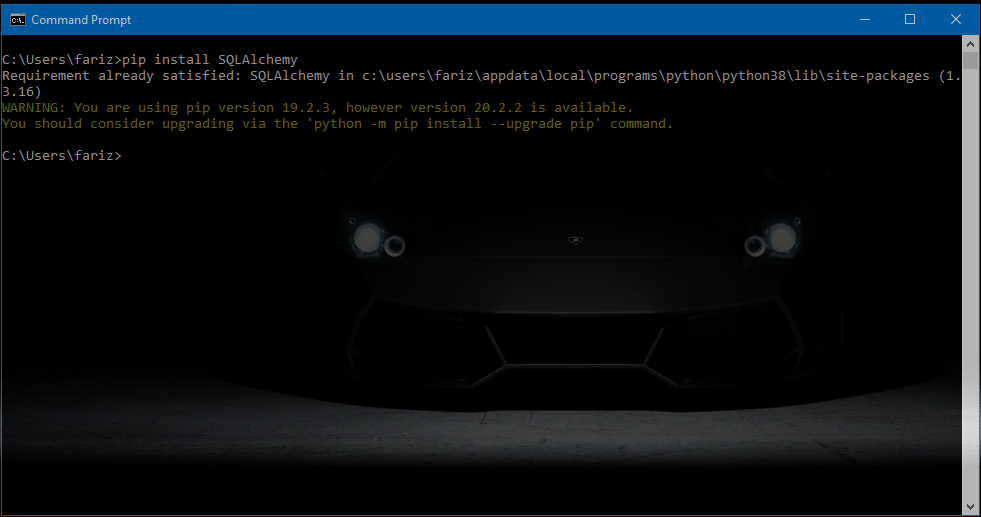
1. **Install Flask Untuk Library Python**

Masukkan perintah *pip install flask* pada *Command Prompt*



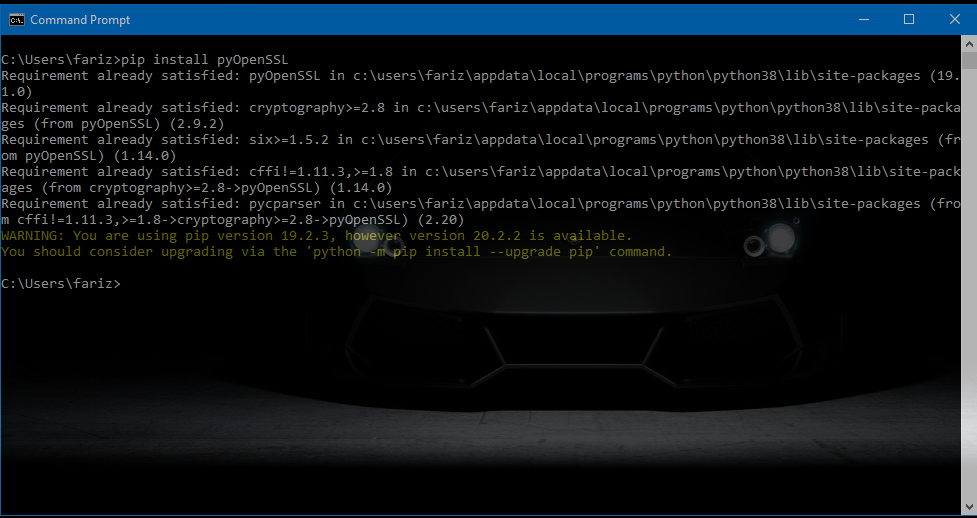
1. **Install Flask Alchemy**

Masukkan perintah *pip install SQLAlchemy* pada *Command Prompt*



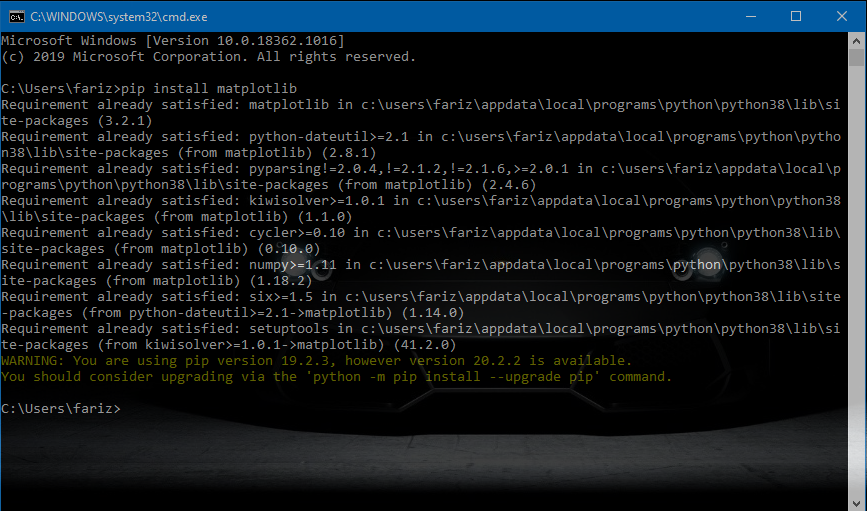
1. **Install PyOpenSSL**

Masukkan perintah *pip install pyOpenSSL* pada *Command Prompt*



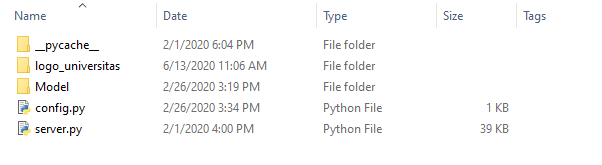
1. **Install MatPlotLib**

Masukkan perintah *pip install matplotlib* pada *Command Prompt*



1. **Aplikasi Web Server Visualisasi Data Peneliti**

Web server visualisasi data peneliti menggunakan Flask yang telah diinstall sebelumnya. Dalam aplikasi web server terdapat 2 buah folder yaitu folder logo\_universitas untuk menyimpan logo, folder Model terdapat file Model.py untuk memudahkan administrasi database dengan merepresentasikan tabel kedalam bentuk class serta file python untuk server.py sebagai backend dan config.py untuk konfigurasi ke database.



* 1. **config.py**

File config.py bertujuan untuk memudahkan konfigurasi agar aplikasi server dapat tersambung ke database dengan menggunakan *SQL Alchemy*. *SQL Alchemy adalah* library Python untuk Flask web server untuk menghandle database. Berikut adalah kode sumber untuk file config.py.

1. **import** sqlalchemy
2. **from** flask\_sqlalchemy **import** SQLAlchemy
3. **from** flask **import** Flask
4. **from** sqlalchemy.ext.declarative **import** declarative\_base
5. **from** sqlalchemy **import** MetaData
7. **class** config:
8. app = Flask(\_\_name\_\_)
9. db\_host = 'localhost' #host name
10. db\_user = 'root' #mysql username
11. db\_pass = '' #mysql password
12. db\_name = 'visualisasi\_peneliti' #database name
13. engine = sqlalchemy.create\_engine('mysql+pymysql://'+db\_user+':'+db\_pass+'@'+db\_host+'/'+db\_name) #koneksi ke db engine
14. uri = 'mysql://' + db\_user + ':' + db\_pass + '@' + db\_host + '/' +db\_name #uri db yg dipakai
15. db = SQLAlchemy(app)
16. Base = declarative\_base()
17. metadata = MetaData(bind=engine)
    1. **Model.py**

File Model.py terletak pada folder Model uang bertujuan untuk memudahkan administrasi database dengan merepresentasikan tabel kedalam bentuk class. Tabel-tabel dalam database dideklarasikan dalam bentuk class lalu atribut dalam tabel juga dideklarasikan sesuain dengan apa yang ada dalam database. Berikut adalah kode sumber programnya.

1. **import** sqlalchemy
2. **from** sqlalchemy **import** MetaData, Table, Column, Integer, ForeignKey, BigInteger
3. **from** flask\_sqlalchemy **import** SQLAlchemy
4. **import** config
5. **from** sqlalchemy.orm **import** relationship, Session
6. **from** sqlalchemy.ext.declarative **import** declarative\_base
7. Base = config.config.Base
8. metadata = config.config.metadata
10. **class** Fakultas(Base):
11. \_\_tablename\_\_ = 'fakultas'
12. \_\_table\_args\_\_ = {'extend\_existing': True}
13. id\_fakultas = Column(primary\_key=True)
14. nama\_fakultas = Column()
15. ambilDepartemen = relationship("Departemen", back\_populates="ambilFakultas")
17. **def** \_\_repr\_\_(self):
18. **return** '<Fakultas %s>' % self.id\_fakultas
20. **class** Departemen(Base):
21. \_\_tablename\_\_ = 'departemen'
22. \_\_table\_args\_\_ = {'extend\_existing': True}
23. id\_departemen = Column(primary\_key=True)
24. nama\_departemen = Column()
25. id\_fakultas\_departemen = Column(ForeignKey(Fakultas.id\_fakultas))
26. ambilFakultas = relationship("Fakultas", back\_populates="ambilDepartemen")
27. ambilPeneliti = relationship("Peneliti", back\_populates="ambilDepartemen")
29. **def** \_\_repr\_\_(self):
30. **return** '<Departemen %s>' % self.id\_departemen
32. **class** Afiliasi(Base):
33. \_\_tablename\_\_ = 'afiliasi'
34. \_\_table\_args\_\_ = {'extend\_existing': True}
35. id\_afiliasi = Column(primary\_key=True)
36. universitas = Column()
37. kota = Column()
38. negara = Column()
39. logo = Column()
40. ambilPeneliti = relationship("Peneliti", back\_populates="ambilAfiliasi")
42. **def** \_\_repr\_\_(self):
43. **return** '<Afiliasi %s>' % self.id\_afiliasi
45. **class** Peneliti(Base):
46. \_\_tablename\_\_ = 'peneliti'
47. id\_peneliti = Column(primary\_key = True)
48. kode\_peneliti = Column()
49. nama\_peneliti = Column()
50. dokumen\_peneliti = Column()
51. dokumen\_dalam\_scopus\_peneliti = Column()
52. h\_index\_peneliti = Column()
53. total\_sitasi\_peneliti = Column()
54. x = Column()
55. y = Column()
56. keyword\_peneliti = Column()
57. show\_centroid\_peneliti = Column()
58. id\_departemen = Column(ForeignKey(Departemen.id\_departemen))
59. id\_afiliasi = Column(ForeignKey(Afiliasi.id\_afiliasi))
60. ambilDetailSubjekArea = relationship('DetailSubjekArea')
61. ambilDepartemen = relationship("Departemen", back\_populates="ambilPeneliti")
62. ambilAfiliasi = relationship("Afiliasi", back\_populates="ambilPeneliti")
63. ambilKeahlian = relationship('Keahlian')
65. **def** \_\_repr\_\_(self):
66. **return** '<Peneliti %s>' % self.id\_peneliti
68. **class** SubjekArea(Base):
69. \_\_tablename\_\_ = 'subjek\_area'
70. id\_subjek\_area = Column(primary\_key=True)
71. nama\_subjek\_area = Column()
73. **def** \_\_repr\_\_(self):
74. **return** '<SubjekArea %s>' % self.id\_subjek\_area
76. **class** DetailSubjekArea(Base):
77. \_\_tablename\_\_ = 'detail\_subjek\_area'
78. id\_detail\_subjek\_area = Column(primary\_key = True)
79. id\_subjek\_area = Column(ForeignKey(SubjekArea.id\_subjek\_area))
80. id\_peneliti = Column(ForeignKey(Peneliti.id\_peneliti))
81. ambilSubjekArea = relationship('SubjekArea')
83. **def** \_\_repr\_\_(self):
84. **return** '<DetailSubjekArea %s>' % self.id\_detail\_subjek\_area
86. **class** SubjekTopikTier1(Base):
87. \_\_tablename\_\_ = 'subjek\_topik\_tier\_1'
88. id\_subjek\_topik\_tier\_1 = Column(primary\_key=True)
89. kelas\_tier\_1  = Column()
91. **def** \_\_repr\_\_(self):
92. **return** '<SubjekTopikTier1 %s>' % self.id\_subjek\_topik\_tier\_1
94. **class** SubjekTopikTier2(Base):
95. \_\_tablename\_\_ = 'subjek\_topik\_tier\_2'
96. id\_subjek\_topik\_tier\_2 = Column(primary\_key=True)
97. kelas\_tier\_2 = Column()
98. id\_subjek\_topik\_tier\_1 = Column(ForeignKey(SubjekTopikTier1.id\_subjek\_topik\_tier\_1))
99. ambilSubjekTopikTier1 = relationship('SubjekTopikTier1')
100. **def** \_\_repr\_\_(self):
101. **return** '<SubjekTopikTier2 %s>' % self.id\_subjek\_topik\_tier\_2
103. **class** Keahlian(Base):
104. \_\_tablename\_\_ = 'keahlian'
105. id\_keahlian = Column(primary\_key = True)
106. id\_subjek\_topik\_tier\_2 = Column(ForeignKey(SubjekTopikTier2.id\_subjek\_topik\_tier\_2))
107. id\_peneliti = Column(ForeignKey(Peneliti.id\_peneliti))
108. jumlah\_dokumen = Column()
109. ambilSubjekTopikTier2 = relationship('SubjekTopikTier2')
111. **def** \_\_repr\_\_(self):
112. **return** '<Keahlian %s>' % self.id\_keahlian
     1. **Melakukan Query menggunakan ORM (Object Relational Mapping) pada file server.py**

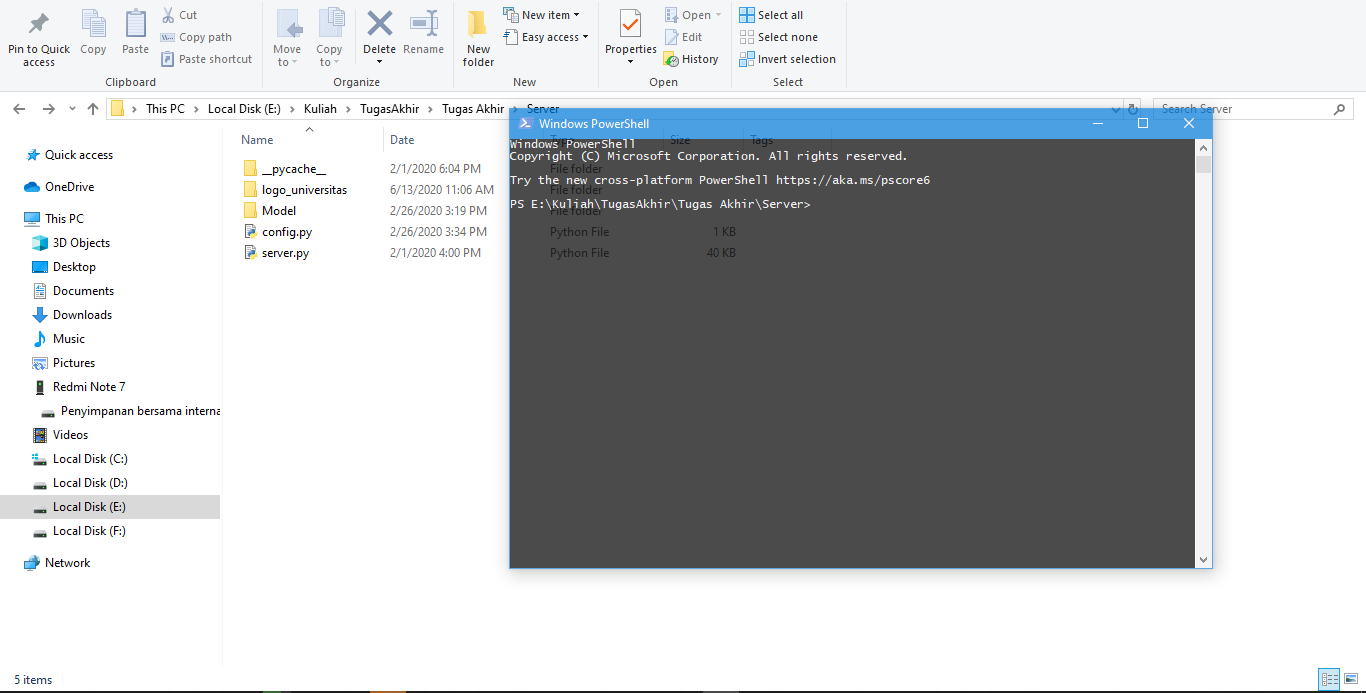
Server.py adalah file python yang digunakan untuk backend atau memproses permintaan yang dikirimkan oleh aplikasi VR. Server.py akan melakukan pencarian data menggunakan ORM sesuai dengan apa yang diminta oleh client. Contoh melakukan query untuk peneliti menggunakan ORM adalah sebagai berikut dan lebih lengkapnya dapat dilihat pada file server.py.

* + 1. Lakukanlah pemanggilan model dan library untuk ORM pada file Server.py seperti berikut.

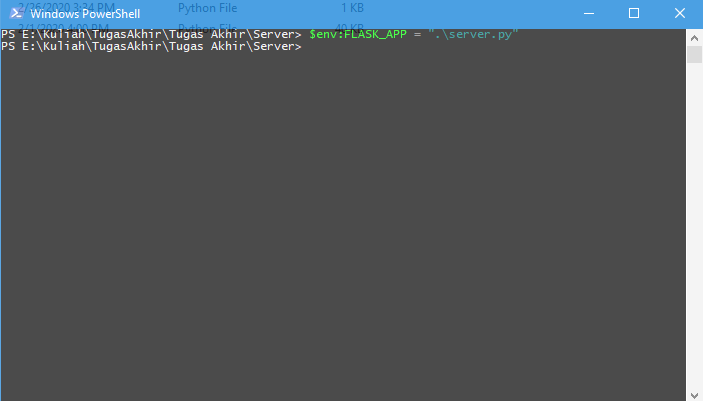
1. **import** json
2. **import** sqlalchemy
3. **from** sqlalchemy.orm **import** Session, sessionmaker, mapper
4. **from** Model.Model **import** Peneliti
5. **from** Model.Model **import** Departemen
6. **from** Model.Model **import** Fakultas
7. **from** Model.Model **import** Afiliasi
8. **from** Model.Model **import** Keahlian, DetailSubjekArea, SubjekArea, SubjekTopikTier2
9. **from** sqlalchemy **import** func,desc,asc, distinct, or\_
10. Lakukanlah query menggunakan ORM. Dalam contoh ORM ini terdapat pada fungsi peneliti() pada server.py.
11. session = Session()
12. data = session.query(Peneliti, func.dense\_rank().over(
13. order\_by=Peneliti.h\_index\_peneliti.desc()
14. ).label('rnk')
15. ).outerjoin(Keahlian, Keahlian.id\_peneliti == Peneliti.id\_peneliti
16. ).outerjoin(Afiliasi, Afiliasi.id\_afiliasi == Peneliti.id\_afiliasi
17. ).outerjoin(DetailSubjekArea, DetailSubjekArea.id\_peneliti == Peneliti.id\_peneliti
18. ).filter(or\_(Keahlian.id\_subjek\_topik\_tier\_2 == keahlian, keahlian == ''), or\_(Peneliti.id\_afiliasi == id\_afiliasi, id\_afiliasi == ''), or\_(DetailSubjekArea.id\_subjek\_area == id\_subjek\_area, id\_subjek\_area == ''), or\_(or\_(Peneliti.nama\_peneliti.like(search), search == ''), or\_(Peneliti.keyword\_peneliti.like(search), search == ''))).order\_by('rnk').all()
19. session.commit()
20. Kemudian lakukan serialisasi atau penyusunan data kedalam json
21. data\_len = len(data)
22. **for** row **in** data:
23. x = Serialisasi(id\_peneliti = row[0].id\_peneliti,
24. kode\_peneliti = row[0].kode\_peneliti,
25. nama\_peneliti = row[0].nama\_peneliti,
26. dokumen\_peneliti = row[0].dokumen\_peneliti,
27. dokumen\_peneliti\_dalam\_scopus = row[0].dokumen\_dalam\_scopus\_peneliti,
28. h\_index\_peneliti = row[0].h\_index\_peneliti,
29. total\_sitasi\_peneliti = row[0].total\_sitasi\_peneliti,
30. x = row[0].x,
31. y = row[0].y,
32. keyword\_peneliti = row[0].keyword\_peneliti,
33. show\_centroid\_peneliti = row[0].show\_centroid\_peneliti,
34. id\_departemen = row[0].id\_departemen,
35. nama\_afiliasi = row[0].ambilAfiliasi.universitas,
36. nama\_departemen = row[0].ambilDepartemen.nama\_departemen,
37. nama\_fakultas = row[0].ambilDepartemen.ambilFakultas.nama\_fakultas,
38. h\_index\_rank = row[1])
40. temp\_row = x.\_\_dict\_\_
41. array\_temp = []
42. **for** detail\_subjek\_area **in** row[0].ambilDetailSubjekArea:
43. y = Serialisasi(id\_detail\_subjek\_area = detail\_subjek\_area.id\_detail\_subjek\_area,
44. id\_subjek\_area = detail\_subjek\_area.id\_subjek\_area,
45. nama\_subjek\_area = detail\_subjek\_area.ambilSubjekArea.nama\_subjek\_area)
46. array\_temp.append(y.\_\_dict\_\_)
47. temp\_row['detail\_subjek\_area'] = array\_temp
49. array\_temp = []
50. **for** keahlian **in** row[0].ambilKeahlian:
51. y = Serialisasi(id\_keahlian = keahlian.id\_keahlian,
52. id\_subjek\_topik\_tier\_2 = keahlian.id\_subjek\_topik\_tier\_2,
53. nama\_keahlian\_tier\_2 = keahlian.ambilSubjekTopikTier2.kelas\_tier\_2,
54. total\_dokumen = keahlian.jumlah\_dokumen,
55. id\_subjek\_topik\_tier\_1 = keahlian.ambilSubjekTopikTier2.ambilSubjekTopikTier1.id\_subjek\_topik\_tier\_1,
56. nama\_keahlian\_tier\_1 = keahlian.ambilSubjekTopikTier2.ambilSubjekTopikTier1.kelas\_tier\_1
57. )
58. array\_temp.append(y.\_\_dict\_\_)
59. temp\_row['keahlian\_peneliti'] = array\_temp
60. arrayQuery.append(temp\_row)
61. **Start Web Server**
    1. **Start area WLAN**

WLAN adalah jaringan nirkabel yang hanya dapat diakses dari dalam dan tidak dapat diakses dari luar jaringan.

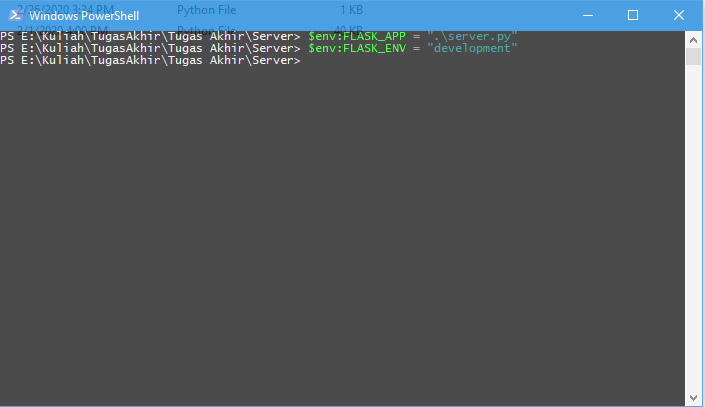
* + 1. Bukalah Powershell pada direktori server.



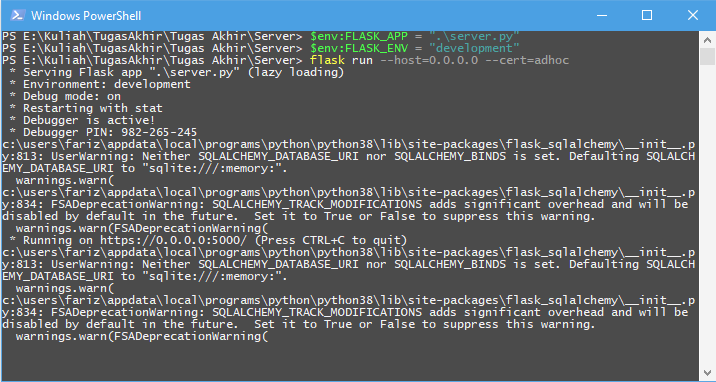
* + 1. Masukkan perintah *$env:FLASK\_APP = ".\server.py"*



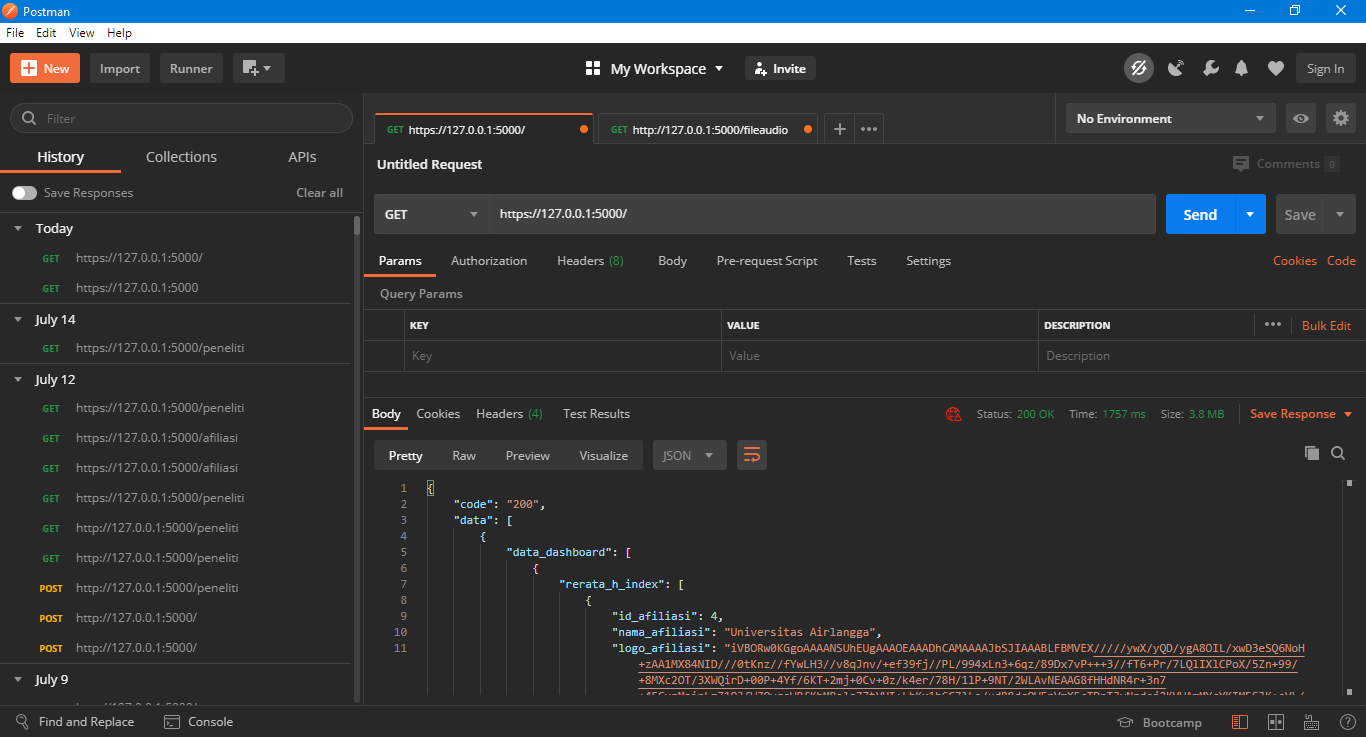
* + 1. Masukkan perintah *$env:FLASK\_ENV = "development"*



* + 1. Masukkan perintah *flask run --host=0.0.0.0 --cert=adhoc* untuk meng-*online­-*kan pada localhost pada ip address 127.0.0.1:5000



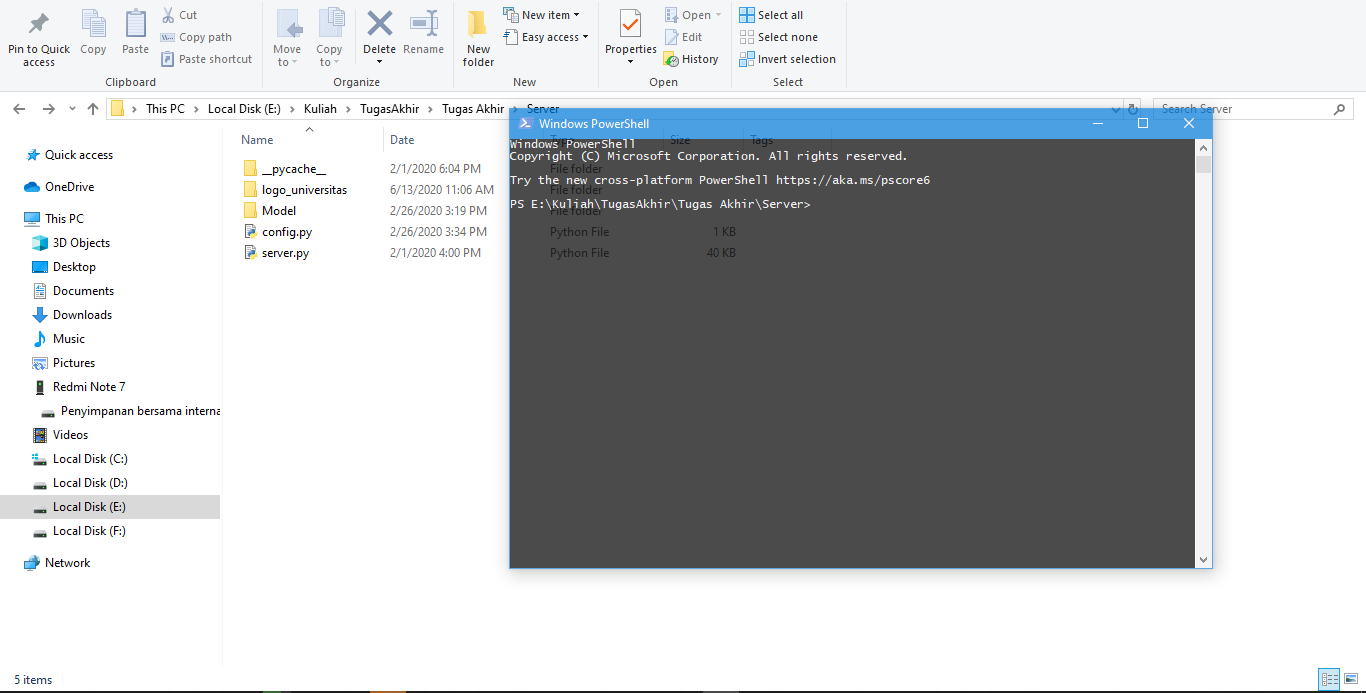
Anda dapat melakukan testing menggunakan Postman untuk menguji apakah server sudah aktif atau belum dengan mengetikkan url <https://127.0.0.5:5000> pada postman dan menghasilkan return JSON.



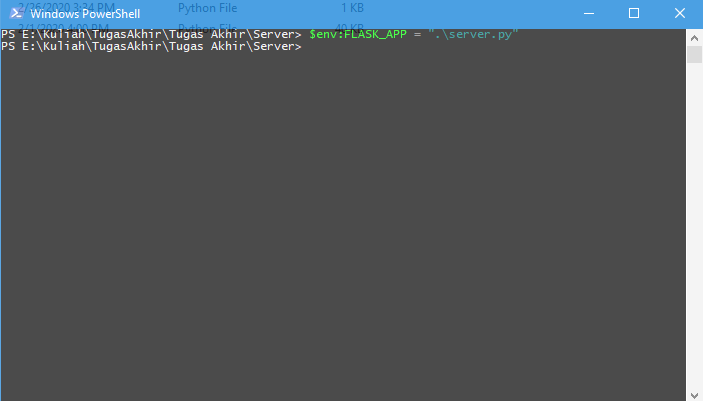
* 1. **Start online server menggunakan *ngrok***

Ngrok adalah software untuk mengonlinekan server pada personal computer tanpa menyewa hosting.

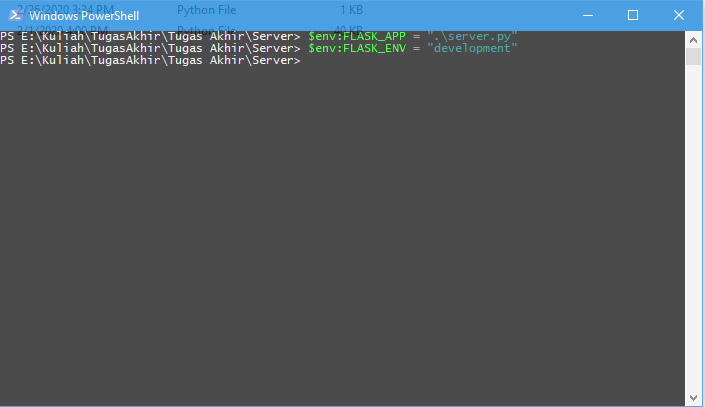
* + 1. Download [ngrok](https://ngrok.com/download)
    2. Lalu extract ngrok
    3. Bukalah Powershell pada direktori server.



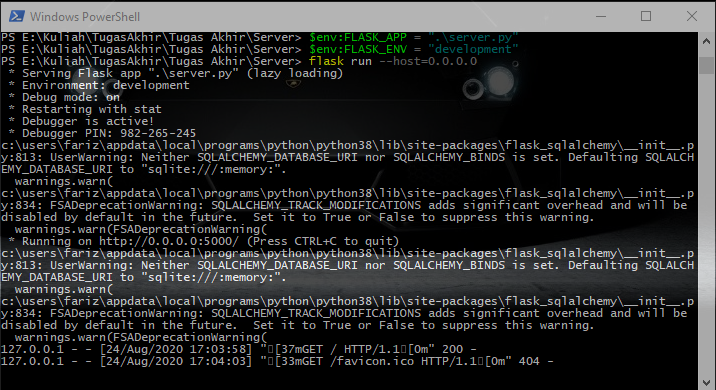
* + 1. Masukkan perintah *$env:FLASK\_APP = ".\server.py"*



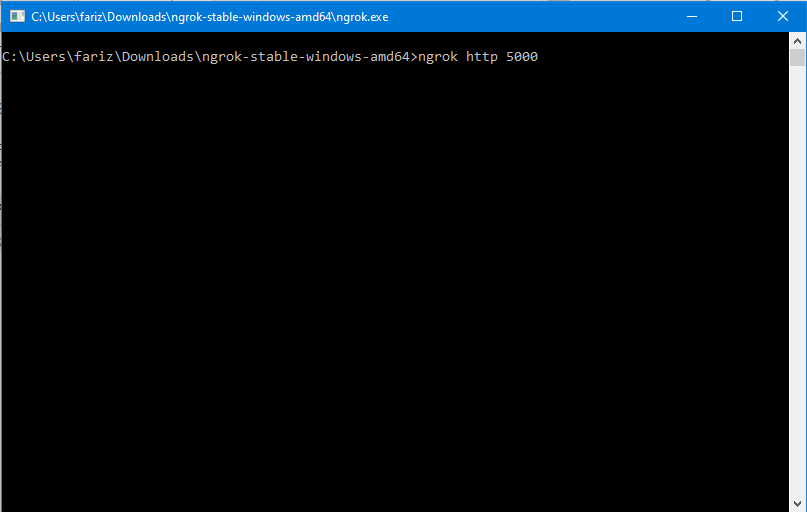
* + 1. Masukkan perintah *$env:FLASK\_ENV = "development"*



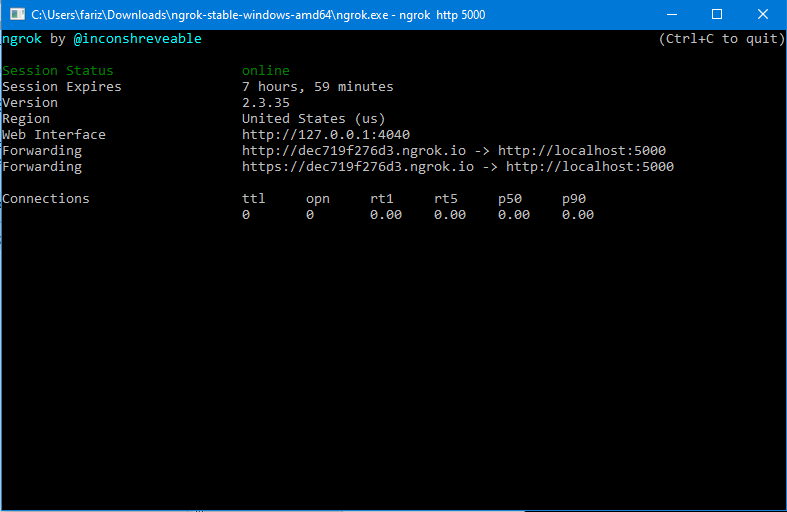
* + 1. Masukkan perintah *flask run --host=0.0.0.0* untuk meng-*online­-*kan pada localhost pada ip address 127.0.0.1:5000 tanpa SSL



* + 1. Bukalah folder ngrok yang telah diextract sebelumnya lalu jalankan ngrok.exe dan masukkan perintah *ngrok http 5000*



* + 1. Tekan *enter* dan ngrok akan melakukan port forwarding kedalam IP Personal Computer kita dengan dynamic URL.

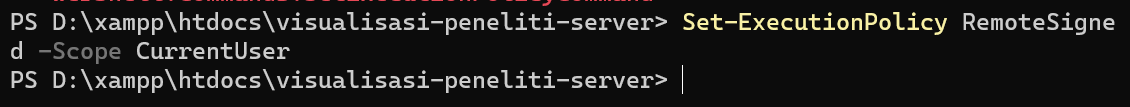


* 1. Jalankan powershell pada directory
  2. Ketikkan command virtualenv venv
  3. Aktifkan virtualenv dengan cara mengetikkan command .\venv\Scripts\activate

Text

Description automatically generated

* 1. Jika muncul error diatas, ketikkan command Set-ExecutionPolicy RemoteSigned -Scope CurrentUser



* 1. Install flask, pip install flask
  2. Install pyopen ssl Masukkan perintah pip install pyOpenSSL
  3. Install flask\_sqlalchemy pip install flask\_sqlalchemy
  4. Install pydobc. pip install pyodbc
  5. Masukkan perintah *$env:FLASK\_APP = ".\server.py"*
  6. Masukkan perintah *$env:FLASK\_ENV = "development"*
  7. Masukkan perintah *flask run --host=0.0.0.0 --cert=adhoc* untuk meng-*online­-*kan pada localhost pada ip address 127.0.0.1:5000
  8. Bukalah folder ngrok yang telah diextract sebelumnya lalu jalankan ngrok.exe dan masukkan perintah *ngrok http 5000*

Export from phpMyAdmin to mssql

Rerun Step

* 1. Aktifkan virtualenv dengan cara mengetikkan command .\venv\Scripts\activate
  2. Masukkan perintah *$env:FLASK\_APP = ".\server.py"*
  3. Masukkan perintah *$env:FLASK\_ENV = "development"*
  4. Masukkan perintah *flask run --host=0.0.0.0 --cert=adhoc* untuk meng-*online­-*kan pada localhost pada ip address 127.0.0.1:5000