FLASK NOTES:

What is Flask?

Flask is a web framework that allows developers to build lightweight web applications quickly and easily with Flask Libraries.

Installation:

virtualenv

pip install virtualenv

Create Python virtual environment

virtualenv venv

Activate virtual environment

windows > venv\Scripts\activate

Hello.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello_world():

    return 'Hello World'

if __name__ == '__main__':

    app.run()
```

Hello2.py

```
from flask import Flask

app = Flask(__name__)

@app.route("/")

def hello_world():

    return "<p>Hello, World!</p>"

if __name__ == '__main__':

   app.run()
```

Flask – Routing

Modern web frameworks use the routing technique to help a user remember application URLs. It is useful to access the desired page directly without having to navigate from the home page.

The route() decorator in Flask is used to bind URL to a function. For example −

```
@app.route(‘/hello’)

def hello_world():

   return ‘hello world’
```

Flask – Variable Rules

It is possible to build a URL dynamically, by adding variable parts to the rule parameter. This variable part is marked as <variable-name>. It is passed as a keyword argument to the function with which the rule is associated

demo1.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/hello/<name>')
def hello_name(name):
   return 'Hello %s!' % name
if __name__ == '__main__':
   app.run(debug = True)
```

In the following code, all these constructors are used.
demo2.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/blog/<int:postID>')
def show_blog(postID):
   return 'Blog Number %d' % postID

@app.route('/rev/<float:revNo>')
def revision(revNo):
   return 'Revision Number %f' % revNo

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

Flask – HTTP methods

1	GET Sends data in unencrypted form to the server. Most common method.
2	HEAD Same as GET, but without response body
3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.
5	DELETE Removes all current representations of the target resource given by a URL

login.html

```
<html>
   <body>
      <form action = "http://localhost:5000/login" method = "post">
         <p>Enter Name:</p>
         <p><input type = "text" name = "nm" /></p>
         <p><input type = "submit" value = "submit" /></p>
      </form>
   </body>
</html>
```

Demo3.py

```
from flask import Flask
app = Flask(__name__)
@app.route('/success/<name>')
def success(name):
   return 'welcome %s' % name


@app.route('/login',methods = ['POST', 'GET'])
def login():
   if request.method == 'POST':
      user = request.form['nm']
      return redirect(url_for('success',name = user))
   else:
```

```
    user = request.args.get('nm')
    return redirect(url_for('success',name = user))

if __name__ == '__main__':
   app.run(debug = True)
```

Flask – Templates

demo4.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
   return '<html><body><h1>Hello World</h1></body></html>'

if __name__ == '__main__':
   app.run(debug = True)
```

demo5.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
   return render_template(‘hello.html’)
```

```
if __name__ == '__main__':
   app.run(debug = True)
```

hello.html

```
<!doctype html>
<html>
   <body>

      <h1>Hello {{ name }}!</h1>

   </body>
</html>
```

Flask – SQLAlchemy

SQLAlchemy, a Python toolkit,Flask-SQLAlchemy is the Flask extension that adds support for SQLAlchemy to your Flask application.

new.html

```
<!DOCTYPE html>
<html>
   <body>
      <h3>Students - Flask SQLAlchemy example</h3>
      <hr/>
      {%- for category, message in get_flashed_messages(with_categories = true)
%}
         <div class = "alert alert-danger">
            {{ message }}
```

```
        </div>
      {%- endfor %}
      <form action = "{{ request.path }}" method = "post">
        <label for = "name">Name</label><br>
        <input type = "text" name = "name" placeholder = "Name" /><br>
        <label for = "email">City</label><br>
        <input type = "text" name = "city" placeholder = "city" /><br>
        <label for = "addr">addr</label><br>
        <textarea name = "addr" placeholder = "addr"></textarea><br>
        <label for = "PIN">City</label><br>
        <input type = "text" name = "pin" placeholder = "pin" /><br>
        <input type = "submit" value = "Submit" />
      </form>
   </body>
</html>
```

app.py

```
from flask import Flask, request, flash, url_for, redirect, render_template
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.sqlite3'
app.config['SECRET_KEY'] = "random string"

db = SQLAlchemy(app)
class students(db.Model):
```

```
   id = db.Column('student_id', db.Integer, primary_key = True)
   name = db.Column(db.String(100))
   city = db.Column(db.String(50))
   addr = db.Column(db.String(200))
   pin = db.Column(db.String(10))

def __init__(self, name, city, addr,pin):
   self.name = name
   self.city = city
   self.addr = addr
   self.pin = pin

@app.route('/')
def show_all():
   return render_template('show_all.html', students = students.query.all() )

@app.route('/new', methods = ['GET', 'POST'])
def new():
   if request.method == 'POST':
      if not request.form['name'] or not request.form['city'] or not
request.form['addr']:
         flash('Please enter all the fields', 'error')
      else:
         student = students(request.form['name'], request.form['city'],
            request.form['addr'], request.form['pin'])

         db.session.add(student)
         db.session.commit()
```

```
        flash('Record was successfully added')
        return redirect(url_for('show_all'))
    return render_template('new.html')

if __name__ == '__main__':
    db.create_all()
    app.run(debug = True)
```