

UJIAN AKHIR SEMESTER
PEMROGRAMAN WEB FRAMEWORK (PHYTON)
Projek Yang Diambil Organisasi Pencinta Hewan Anjing



OLEH:

MAHASISWA : JET J KRISNADI (32180014)

Kepada :

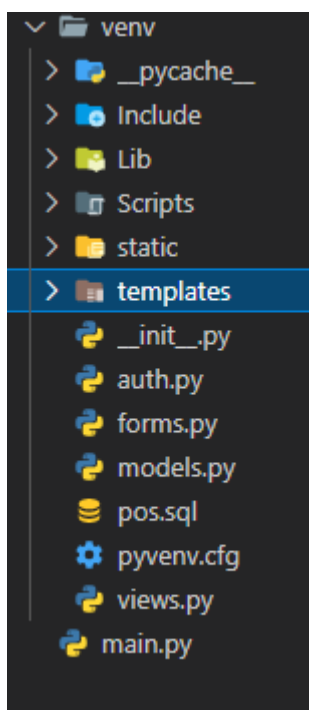
DOSEN :Ester Lumba , S.Si, M.Kom

UNIVERSITAS BUNDA MULIA
2020/2021

How to setup

1. `virtualenv -p python3 env`
2. `pip install -r requirements.txt`
3. Bikin database menggunakan mysql di phpmyadmin dengan nama model
4. Import pos.sql ke dalam database model. Setelah itu run aplikasi

1. Membuat virtual environment



```
PS C:\Users\ACER\Downloads\flask-pos-master> py -3 -m venv venv
PS C:\Users\ACER\Downloads\flask-pos-master> venv\scripts\activate
(venv) PS C:\Users\ACER\Downloads\flask-pos-master> 
```

2. Database relasional atau MySQL yang melibatkan minimal 5 buah tabel.

Table	Action	Row
<input type="checkbox"/> historyproducts	★ Browse Structure Search Insert Empty Drop	
<input type="checkbox"/> products	★ Browse Structure Search Insert Empty Drop	
<input type="checkbox"/> transactions	★ Browse Structure Search Insert Empty Drop	
<input type="checkbox"/> transaction_products	★ Browse Structure Search Insert Empty Drop	
<input type="checkbox"/> user	★ Browse Structure Search Insert Empty Drop	
5 tables	Sum	

Database Master products

```
class Products(db.Model):
    id = db.Column(db.Integer, primary_key=True,
autoincrement=True)
    barang = db.Column(db.String(45), nullable=False)
    harga = db.Column(db.Integer, nullable=False)
    jumlah = db.Column(db.Integer, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
```

Transactions

```
class Transactions(db.Model):
    id = db.Column(db.Integer, primary_key=True,
autoincrement=True)
    create_on = db.Column(db.DateTime, nullable=False)
    def __init__(self):
        self.create_on = datetime.now()
```

Database Transaksi :

```
class TransactionProducts(db.Model):
    id = db.Column(db.Integer, primary_key=True,
autoincrement=True)
    transaction_id = db.Column(db.Integer,
db.ForeignKey("transactions.id"), nullable=False)
    product_id = db.Column(db.Integer,
db.ForeignKey("products.id"), nullable=False)
    product_qty = db.Column(db.Integer, nullable=False)
    transaction = relationship("Transactions",
backref="transaction_products")
    product = relationship("Products",
backref="transaction_products")
```

Database HistoryProducts

```
class HistoryProducts(db.Model):
    __tablename__ = "historyproducts"
    id = db.Column(db.Integer, primary_key=True)
    id_barang = db.Column(db.Integer,
db.ForeignKey('products.id'))
    id_user = db.Column(db.Integer, db.ForeignKey('user.id'))
    action = db.Column(db.String(150))
```

Database User :

```
class User(db.Model, UserMixin):
```

```

        id = db.Column(db.Integer, primary_key=True,
autoincrement=True, default='0')


        username = db.Column(db.String(45), nullable=False)
        email = db.Column(db.Integer, nullable=False)
        role = db.Column(db.String(64), default='customer')

    def __repr__(self) -> str:
        return '<User %s>' % self.username

```

3. Aplikasi web di bangun menggunakan arsitektur model dan view.

Models :

 models.py

```

class Products(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    barang = db.Column(db.String(45), nullable=False)
    harga = db.Column(db.Integer, nullable=False)
    jumlah = db.Column(db.Integer, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

```

```

class HistoryProducts(db.Model):
    __tablename__ = "historyproducts"
    id = db.Column(db.Integer, primary_key=True)
    id_barang = db.Column(db.Integer, db.ForeignKey('products.id'))
    id_user = db.Column(db.Integer, db.ForeignKey('user.id'))
    action = db.Column(db.String(150))

```

```

class TransactionProducts(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    transaction_id = db.Column(db.Integer, db.ForeignKey("transactions.id"), nullable=False)
    product_id = db.Column(db.Integer, db.ForeignKey("products.id"), nullable=False)
    product_qty = db.Column(db.Integer, nullable=False)
    transaction = relationship("Transactions", backref="transaction_products")
    product = relationship("Products", backref="transaction_products")

```

```

class Transactions(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    create_on = db.Column(db.DateTime, nullable=False)
    def __init__(self):
        self.create_on = datetime.now()

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True, default='0')
    username = db.Column(db.String(45), nullable=False)
    email = db.Column(db.Integer, nullable=False)
    role = db.Column(db.String(64), default='customer')

    def __repr__(self) -> str:
        return '<User %s>' % self.username

```

Views Admin :

```

views = Blueprint('views', __name__, static_folder='static')

@views.route('/', methods=['GET', 'POST'])
def home():
    data = Products.query.all()

    for row in data:
        print(row.id, row.barang, row.harga, row.user_id, row.jumlah)

    return render_template("index.html", data=data)

@views.route("/admin")
@login_required
@requires_roles('admin')
def admin():
    return redirect("admin/products")

@views.route('/admin/products', methods=['GET', 'POST'])
@login_required
@requires_roles('admin')
def product_list():
    headings = ("ID", "Nama Barang", "Harga Barang", "Jumlah Barang",
               "User ID", "Gambar Barang")

    data = Products.query.all()

    for row in data:
        print(row.id, row.barang, row.harga, row.user_id, row.jumlah)

```

```

        return render_template("admin/products/list.html",
data=data,user=current_user,headings=headings)

@views.route('/admin/products/add', methods=['GET', 'POST'])
@login_required
@requires_roles('admin')
def products_add():
    if request.method == 'POST':
        nama_barang = request.form.get('barang')
        harga_barang = request.form.get('harga')
        jumlah_barang = request.form.get('jumlah')
        image = request.files['foto']

        if len(nama_barang) == 0:
            flash('Pastikan data terisi dengan benar !',
category='error')
        else:
            new_Products = Products(barang=nama_barang,harga=harga_barang,
jumlah=jumlah_barang,user_id=current_user.id)
            db.session.add(new_Products)
            db.session.commit()
            tempid = new_Products.id
            fileid = str(new_Products.id)
            flash('Data Ditambahkan added!', category='success')
            image.filename = fileid + ".jpg"
            image.save(os.path.join("pos/", "static/", "image/",
secure_filename(image.filename)))
            new_HistoryProducts = HistoryProducts(id_barang=tempid,
id_user=current_user.id, action="Insert")
            db.session.add(new_HistoryProducts)
            db.session.commit()

    return render_template("admin/products/form_add.html")

@views.route('/admin/products/edit', methods=['GET', 'POST'])
@login_required
@requires_roles('admin')
def products_edit():
    id_barang = request.args.get('id')

```

```

        if (request.method == 'POST'):
            nama_barang = request.form.get('name')
            harga_barang = request.form.get('price')
            jumlah_barang = request.form.get('stock')
            image = request.files['foto']
            print("Edit berjalan")
            if len(nama_barang) == 0:
                flash('Pastikan data terisi dengan benar !',
category='error')
            else:
                new_Products =
Products.query.filter_by(id=id_barang).first()
                new_Products.name = nama_barang
                new_Products.price = harga_barang
                new_Products.stock = jumlah_barang
                db.session.commit()
                fileid = str(new_Products.id)
                flash('Inventory added!', category='success')
                image.filename = fileid + ".jpg"
                image.save(os.path.join("pos/", "static/", "image/",
secure_filename(image.filename)))
                new_HistoryProducts = HistoryProducts(id_barang=id_barang,
id_user=current_user.id, action="Insert")
                db.session.add(new_HistoryProducts)
                db.session.commit()

            return redirect("/admin/products")

        return render_template("admin/products/form_edit.html",
id_barang=id_barang)

@views.route('/admin/products/delete', methods=['GET', 'POST'])
@login_required
@requires_roles('admin')
def products_delete():
    id_barang = request.args.get('id')
    deletepro = Products.query.filter_by(id=id_barang).first()
    db.session.delete(deletepro)
    db.session.commit()
    new_HistoryProducts = HistoryProducts(id_barang=id_barang,
id_user=current_user.id, action="Delete")
    db.session.add(new_HistoryProducts)

```

```

        db.session.commit()
        return redirect("/admin/products")

#transaksi
@views.route("/admin/transactions")
@login_required
@requires_roles('admin')
def transactions_list():

    transactions = Transactions.query.all()

    return render_template("admin/transactions/list.html", data=transactions)

@views.route("/admin/transactions/add", methods=["GET", "POST"])
@login_required
@requires_roles('admin')
def transactions_add():
    if request.method == "POST":
        # ambil data dari form html
        products = request.form.getlist("products")
        products_qty = request.form.getlist("products_qty")

        # buat transacsi utamanya
        transactions = Transactions()
        db.session.add(transactions)

        # flush terlebih dahulu untuk mencegah transaksi apabila gagal
        db.session.flush()


        # loop product
        for i, product in enumerate(products):
            transactions_products = TransactionProducts()
            transactions_products.transaction_id = transactions.id
            transactions_products.product_id = int(product)
            transactions_products.product_qty = int(products_qty[i])
            db.session.add(transactions_products)
            db.session.flush()

        # commit semua transaksi
        db.session.commit()
        return redirect("/admin/transactions")
        return render_template("admin/transactions/form_add.html")

```


Menerapkan operasi CRUD pada masing-masing tabel Products

List Productadd productlist transactionadd transactionLog Out

ID	Nama Barang	Harga Barang	Jumlah Barang	User ID	Gambar Barang	
1	test1	100000	100	1		edit delete

Export To PDF

Transaksi

id	products	create on
----	----------	-----------



4. Aplikasi terdiri dari dua sisi, yaitu sisi pengguna (front end) dan sisi pengelola/admin (back end) dengan menerapkan sistem otentikasi menggunakan session.

Login :

Tampilan Admin :

List Product

add product

list transaction

add transaction

Log Out

ID	Nama Barang	Harga Barang	Jumlah Barang	User ID	Gambar Barang	
1	test1	100000	100	1		edit delete

Export To PDF



Tambah Product :

[List Product](#)
[add product](#)
[list transaction](#)
[add transaction](#)
[Log Out](#)

Nama Barang

Harga

Jumlah Stock

image Pilih file:
No file chosen

Tampilan List Transaksi :

id	products	create on
----	----------	-----------




Tampilan Tambah Transaksi

[List Product](#)
[add product](#)
[list transaction](#)
[add transaction](#)
[Log Out](#)


id	products	create on
----	----------	-----------

Tampilan Front End:



[Register](#)
[Login](#)
[Admin Page](#)

Dogs
Cats
Hamsters
Birds
Other



Medical



test1
100000

Login :

Login

Username

☐ Remember Me

Login

[Create a New Account!](#)

Register:

Register

Username

Email

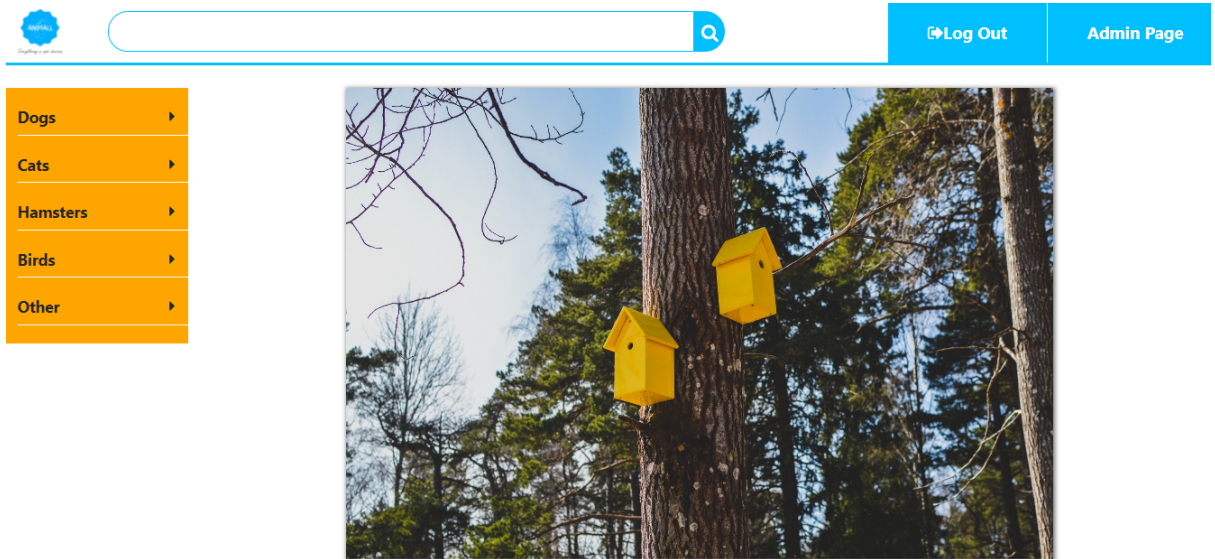
Role

customer ▼

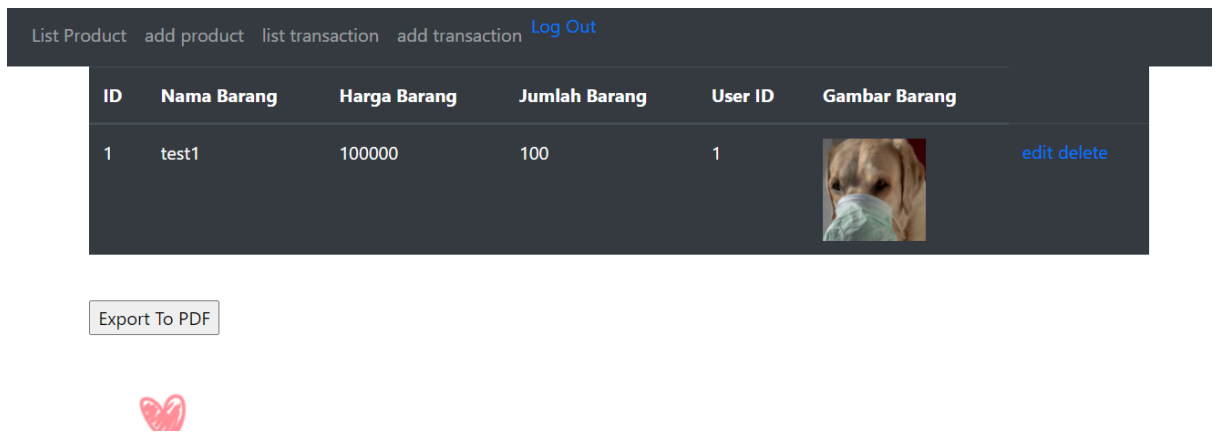
Register

[Login into existing account!](#)

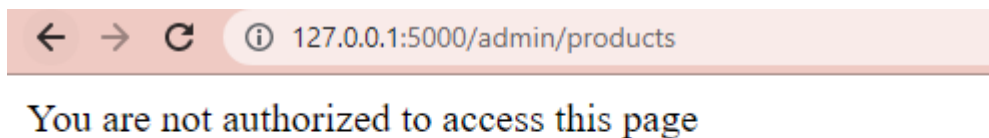
Setelah Login



Jika Admin bisa menekan tombol admin page maka akan masuk ke pages khusus admin



Jika Bukan Admin maka user tidak bisa masuk ke page admin



Code dimana bisa mendeteksi apakah dia user atau admin :

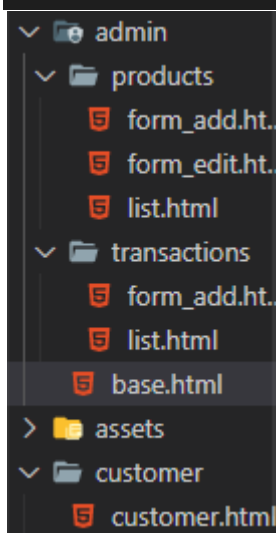
```
def requires_roles(*roles):
    def wrapper(f):
        @wraps(f)
        def wrapped(*args, **kwargs):
            if current_user.role not in roles:
                # Redirect the user to an unauthorized notice!
                return "You are not authorized to access this
page"
            return f(*args, **kwargs)
        return wrapped
    return wrapper
```

Cara manggil code yang mendeteksi user atau admin :

```
@views.route('/admin/products/delete', methods=['GET', 'POST'])
@login_required
@requires_roles('admin')
```

5. Elemen form (UI) yang di gunakan harus sesuai dengan fungsi dari tipe input masingmasing field

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <ul class="nav navbar-nav">
      <li><a href="/admin/products">list product</a></li>
      <li><a href="/admin/products/add">add
product</a></li>
      <li><a href="/admin/transactions">list
transaction</a></li>
      <li><a href="/admin/transactions/add">add
transaction</a></li>
      <li><a href="/logout/" >Log Out</a></li>
      <li><a href="/users"> List Akun</a></li>
    </ul>
  </div>
</nav>
```



6. Terdapat fitur pengiriman email, upload file/gambar, fungsi cetak, download dan file reader (sesuai dengan karakteristik aplikasi web yang di buat)

Untuk website saya menggunakan upload file/gambar dan fungsi cetak berupa download dan diubah menjadi file reader(pdf)

Ini saya ingin mengetes apakah beneran masuk atau tidak gambarnya

Nama Barang

tes2

Harga

100000

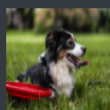
Jumlah Stock

100

image Pilih file: Choose File dogtoyfris.jpg

Submit

Di crud barang masuk maka sekarang kita check apakah di tampilan index atau tampilan umum masuk tidak ?

ID	Nama Barang	Harga Barang	Jumlah Barang	User ID	Gambar Barang	
1	test1	100000	100	1		edit delete
2	tes2	100000	100	1		edit delete

Export To PDF

Ternyata di tampilan index masuk juga

Medical



test1
100000



tes2
100000

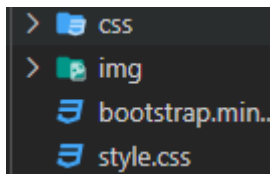
Berarti sukses untuk input gambar
Sekarang saya akan mengetest download pdf

Screenshot of a web browser showing a PDF document titled "Products (4).pdf". The document displays a table with product information:

ID	Nama Barang	Harga Barang	Jumlah Barang	User ID	Gambar Barang
6	test1	100000	1	1	

Berhasil

7. Menerapkan style untuk mengatur tampilan (bootstrap).
Boleh menggunakan template atau layout dan menu-menu untuk navigasi dari class-class pada bootstrap



Contoh :

Navigasi List.html

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <ul class="nav navbar-nav">
  </ul>
  </div>
</nav>
```

8. SDLC

Planning

Website ini akan mempunyai 5 tabel

1. Tabel user
2. Tabel products
3. Tabel Transaksi Products
4. Tabel Transaksi
5. Tabel HistoryProducts

Tabel Products dan Tabel Transaksi menerapkan CRUD sementara tabel transaksi produksi hanya untuk anakan atau menyatukan products dengan tabel transaksi. Sedangkan tables users tidak ada tampilan sama sekali hanya untuk login dan register

Tabel User memiliki kolom id ,username,email,role

Tabel History Products memiliki kolom id ,id_barang ,id_user,action

Tabel Transaksi memiliki id, create_on(dibuat kapan)

Table transaksi products memiliki id,transaction_id,product_id,product_qty

Tabel products memiliki kolom id,barang,harga,jumlah,user_id

- Bagian admin hanya dapat dimasuki oleh admin , user/customer tidak bisa memasuki website tersebut

Implementation

Bahasa yang digunakan adalah Python menggunakan micro framework Flask untuk database menggunakan relational database mysql dan framework bootstrap

Daftar Library yang digunakan :

- Flask Login
- SQLALCHEMY
- OS
- Werkzeug.security
- Re
- Datetime
- Functools untuk penggunaan wraps
- WTFORMS, WTFORMS-VALIDATOR
- FLASK_wtf
- Blueprint

Pengujian :

Kasus pertama yang terjadi adalah menemukan bug dimana image masuk ke dalam database tapi tidak muncul ke tampilan index/tampilan list

Kasus Kedua yang terjadi adalah merubah bug dimana image masuk ke dalam folder internal tidak menggunakan database dan masalahnya sama seperti kejadian pertama

Akhirnya terpaksa untuk membuat permisalan proses dimana database dan penyimpanan internal dipaksa menjadi satu penyimpanan agar dapat image bisa muncul

Link Gdrive:

https://drive.google.com/drive/folders/1IWwciJ6xhRTndUC_15h9A3NWEBGGItTb?usp=sharing