# LAMPIRAN

## LISTING PROGRAM

Admin Controller

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from app.core.database import get_db
from . import crud, schemas

router = APIRouter()

@router.get("", response_model=list[schemas.AdminResponse])
async def read_users(db: Session = Depends(get_db)):
    return crud.get_users(db)

@router.post("", response_model=schemas.AdminResponse)
async def create_new_user(user: schemas.AdminCreate, db: Session =
Depends(get_db)):
    existing_user = db.query(crud.Admin).filter(crud.Admin.email ==
user.email).first()
    if existing_user:
        raise HTTPException(status_code=400, detail="email already
registered")
    return crud.create_user(db, user)

@router.get("/{user_id}", response_model=schemas.AdminResponse)
async def read_user(user_id: int, db: Session = Depends(get_db)):
    db_user = crud.get_user(db, user_id)

    if db_user is None:
        raise HTTPException(status_code = 404, detail="User Not
Found")

    return db_user

@router.put("/{user_id}", response_model=schemas.AdminResponse)
def update_user(user_id: int, user: schemas.AdminCreate, db:
Session = Depends(get_db)):
    db_user = crud.update_user(db, user_id, user.nama, user.email,
user.password)
    if db_user is None:
        raise HTTPException(status_code=404, details="User not
found")
    return db_user

@router.delete("/{user_id}", response_model=schemas.AdminResponse)
def delete_user(user_id: int, db: Session = Depends(get_db)):
    db_user = crud.delete_user(db, user_id)
    if db_user is None:
        raise HTTPException(status_code=404, detail="User not
found")
    return db_user
```

Kendaraan Controller

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from app.core.database import get_db
from . import crud, schemas

router = APIRouter()

@router.get("", response_model=list[schemas.KendaraanResponse])
async def read_kendaraans(db: Session = Depends(get_db)):
    return crud.get_kendaraans(db)

@router.post("", response_model=schemas.KendaraanCreate)
async def create_kendaraan(kendaraan: schemas.KendaraanCreate, db:
Session = Depends(get_db)):
                                  existing_kendaraan               =
db.query(crud.Kendaraan).filter(crud.Kendaraan.number           ==
kendaraan.number).first()
    if existing_kendaraan:
            raise  HTTPException(status_code=400,  detail="Number
Kendaraan already registered")
    return crud.create_kendaraan(db, kendaraan)

@router.get("/{kendaraan_id}",
response_model=schemas.KendaraanResponse)
async  def  read_kendaraan(kendaraan_id:  int,  db:  Session  =
Depends(get_db)):
    db_kendaraan = crud.get_kendaraan(db, kendaraan_id)

    if db_kendaraan is None:
         raise HTTPException(status_code = 404, detail="Kendaraan
Not Found")

    return db_kendaraan

@router.put("/{kendaraan_id}",
response_model=schemas.KendaraanResponse)
def      update_kendaraan(kendaraan_id:      int,      kendaraan:
schemas.KendaraanCreate, db: Session = Depends(get_db)):
       db_kendaraan  =  crud.update_kendaraan(db,  kendaraan_id,
kendaraan.nama)
    if db_kendaraan is None:
        raise HTTPException(status_code=404, detail="Kendaraan not
found")
    return db_kendaraan

@router.delete("/{kendaraan_id}",
response_model=schemas.KendaraanResponse)
def    delete_wisata(kendaraan_id:    int,    db:    Session    =
Depends(get_db)):
    db_kendaraan = crud.delete_kendaraan(db, kendaraan_id)
    if db_kendaraan is None:
        raise HTTPException(status_code=404, detail="Kendaraan not
found")
    return db_kendaraan
```

Pelanggan Controller

```
   from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from app.core.database import get_db
from . import crud, schemas

router = APIRouter()

@router.get("", response_model=list[schemas.UserResponse])
async def read_pelanggan(db: Session = Depends(get_db)):
    return crud.get_users(db)

@router.post("", response_model=schemas.UserResponse)
async def create_new_pelanggan(user: schemas.UserCreate, db:
Session = Depends(get_db)):
    existing_user = db.query(crud.User).filter(crud.User.email ==
user.email).first()
    if existing_user:
        raise HTTPException(status_code=400, detail="Email already
registered")
    return crud.create_user(db, user)

@router.get("/{user_id}", response_model=schemas.UserResponse)
async def read_pelanggan(user_id: int, db: Session =
Depends(get_db)):
    db_user = crud.get_user(db, user_id)

    if db_user is None:
        raise HTTPException(status_code = 404, detail="User Not
Found")

    return db_user

@router.put("/{user_id}", response_model=schemas.UserResponse)
def update_pelanggan(user_id: int, user: schemas.UserCreate, db:
Session = Depends(get_db)):
    db_user = crud.update_user(db, user_id, user.name, user.email)
    if db_user is None:
        raise HTTPException(status_code=404, detail="User not
found")
    return db_user

@router.delete("/{user_id}", response_model=schemas.UserResponse)
def delete_pelanggan(user_id: int, db: Session = Depends(get_db)):
    db_user = crud.delete_user(db, user_id)
    if db_user is None:
        raise HTTPException(status_code=404, detail="User not
found")
    return db_user
```

Transaksi Controller

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from app.core.database import get_db
from . import crud, schemas

router = APIRouter()

@router.get("", response_model=list[schemas.TransaksiOut])
async def read_transaksis(db: Session = Depends(get_db)):
    return crud.get_transaksis(db)

@router.post("", response_model=schemas.TransaksiCreate)
async          def          create_new_transaksi(transaksi:
schemas.TransaksiCreate, db: Session = Depends(get_db)):
    return crud.create_transaksi(db, transaksi)

@router.get("/{transaksi_id}",
response_model=schemas.TransaksiOut)
async def read_transaksi(transaksi_id: int, db: Session =
Depends(get_db)):
    db_transaksi = crud.get_transaksi(db, transaksi_id)

    if db_transaksi is None:
                raise   HTTPException(status_code   =   404,
detail="Transaksi Not Found")

    return db_transaksi
```

Wisata Controller

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from app.core.database import get_db
from . import crud, schemas

router = APIRouter()

@router.get("", response_model=list[schemas.WisataResponse])
async def read_wisatas(db: Session = Depends(get_db)):
    return crud.get_wisatas(db)

@router.post("", response_model=schemas.WisataCreate)
async def create_new_user(wisata: schemas.WisataCreate, db: Session
= Depends(get_db)):
    existing_wisata                                                =
db.query(crud.Wisata).filter(crud.Wisata.nama                     ==
wisata.nama).first()
    if existing_wisata:
        raise HTTPException(status_code=400, detail="Wisata already
registered")
    return crud.create_wisata(db, wisata)

@router.get("/{wisata_id}", response_model=schemas.WisataResponse)
async def read_user(wisata_id: int, db: Session = Depends(get_db)):
    db_wisata = crud.get_wisata(db, wisata_id)

    if db_wisata is None:
        raise  HTTPException(status_code  =  404,  detail="User  Not
Found")

    return db_wisata

@router.put("/{wisata_id}", response_model=schemas.WisataResponse)
def update_wisata(wisata_id: int, wisata: schemas.WisataCreate, db:
Session = Depends(get_db)):
    db_wisata = crud.update_wisata(db, wisata_id, wisata.nama)
    if db_wisata is None:
        raise  HTTPException(status_code=404,  detail="Wisata  not
found")
    return db_wisata

@router.delete("/{wisata_id}",
response_model=schemas.WisataResponse)
def delete_wisata(wisata_id: int, db: Session = Depends(get_db)):
    db_wisata = crud.delete_wisata(db, wisata_id)
    if db_wisata is None:
        raise  HTTPException(status_code=404,  detail="Wisata  not
found")
    return db_wisata
```

# KRITERIA KELULUSAN

PEMBERITAHUAN SEBELUM UJIAN :
Jika pengumpulan dokumen Tugas Akhir di perpustakaan melewati batas akhir semester berjalan, maka mahasiswa harus menyelesaikan registrasi dan KRS semester berikutnya.

KRITERIA KELULUSAN UJIAN TUGAS AKHIR

1. Lulus dengan memperhatikan catatan ujian tugas akhir, dan atau melakukan perbaikan atau penyempurnaan naskah dan atau produk dalam waktu maksimum dua bulan dari tanggal ujian tugas akhir, yaitu tanggal 4 April 2025

Jika dalam waktu yang ditentukan mahasiswa tersebut tidak dapat menyelesaikan, maka mahasiswa yang bersangkutan dianggap tidak lulus ujian.

2. Tidak lulus, disarankan oleh Ketua Tim Penguji untuk mempelajari ulang materi, merombak produk/naskah, atau mengganti judul.

Ketentuan bagi peserta yang tidak lulus ujian tugas akhir.

1) Mahasiswa wajib menempuh ujian tugas akhir ulang

2) Kesempatan ujian tugas akhir ulang hanya diberikan dalam rentang waktu maksimum 6 bulan, setelah ujian sidang/pendadaran

3) Jika sampai batas waktu maksimum 6 bulan tersebut belum dapat diajukan/diselesaikan, maka calon peserta ujian dinyatakan sebagai mahasiswa peserta Tugas Akhir baru, dengan segala ketentuan yang berlaku bagi peserta baru

4) Mahasiswa yang akan menempuh ujian tugas akhir ulang ini diwajibkan membayar biaya ujian sesuai tarif yang ditetapkan.

Yogyakarta, _____
Memahami dan bersedia
Mematuhi peraturan di atas,

# CATATAN PENDADARAN

YAYASAN PENDIDIKAN WIDYA BAKTI YOGYAKARTA
UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA
Jl. Raya Janti (Majapahit) No.143, Yogyakarta, 55198, Telp (0274) 486664,
Website: www.utdi.ac.id , E-mail: info@utdi.ac.id

| | | |
|---|---|---|
| Hari, tanggal | : | Selasa, 4 Februari 2025 |
| Waktu | : | 13.00 |
| Nama | : | HERU DWI PANGESTU |
| No. Mahasiswa / Prodi | : | 185410038 / Informatika |

| No | Hal yang harus diperbaiki | Pemberi Catatan |
|---|---|---|
| 1. | Ubah naskah mengikuti format terbaru (cari bedanya dan update di naskah ini).<br>Use case dipecah per user. Activity diagram gunakan bahasa Indonesia.<br>Bab 4 tambahkan uji black-box.<br>kesimpulan diganti.<br>Source code gunakan bagian yang penting saja.<br>transaksi wajib mencatat tanggal waktu kapan transaksi tersebut dilakukan, urutkan data menurun berdasarkan paling baru | Danny |
| 2. | untuk yang pesan wajib login untuk verifikasi data, di transaksi tampilkan smua pesanan baik yang sudah dibayar maupun belum (berikan status pembayaran lunas/belum) tampilkan sesuatu dengan data terbaru yang baru masuk | |
| 3. | Lihat catatan penguji | |
| 4. | | |

# HASIL KEPUTUSAN SIDANG

YAYASAN PENDIDIKAN WIDYA BAKTI YOGYAKARTA

**UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA**

Jl. Raya Janti (Majapahit) No.143, Yogyakarta, 55198, Telp (0274) 486664,
Website: www.utdi.ac.id , E-mail: info@utdi.ac.id

KEPUTUSAN HASIL UJIAN PENDADARAN

| | | | | |
|---|---|---|---|---|
| Sesuai dengan hasil sidang pendadaran pada tanggal | | | 4 Februari 2025 | maka |
| Nama Mahasiswa | HERU DWI PANGESTU | | | |
| NIM / Program Studi | 185410038 / Informatika | | | |
| Jenjang | | | | |
| | dinyatakan | LULUS | | |
| Ketua Penguji | Danny Kriestanto, S.Kom, M.Eng. | | | |

# BUKTI ACC NASKAH

**Heru Dwi Pangestu** <herudwipangestu20@gmail.com> 27 Mar 2025, 01.09
kepada danny

**Satu lampiran** • Dipindai dengan Gmail

185410038 heru d...

**Danny Kriestanto** <danny@utdi.ac.id> 27 Mar 2025, 06.40
kepada saya

Naskah ini telah saya baca dan saya acc untuk dicetak sebagai naskah akhir skripsi.

Email ini silahkan dicetak sebagai bukti acc naskah.

> On 27 Mar 2025, at 01.09, Heru Dwi Pangestu <herudwipangestu20@gmail.com> wrote:
>
>

**SURAT KETERANGAN**

**PERSETUJUAN PUBLIKASI**

Bahwa yang bertanda tangan dibawah ini:

| | | |
|---|---|---|
| Nama | : | Heru Dwi Pangestu |
| No. Mahasiswa | : | 185410038 |
| Jurusan | : | Informatika |
| Jenjang | : | Sarjana |
| Judul | : | Membangun Sistem Pemesanan Paket Wisata Berbasis Website Menggunakan Payment Gateway Midtrans |

Menyerahkan karya ilmiah kepada pihak perpustakaan Universitas Teknologi Digital Indonesia (UTDI) dan menyetujui untuk diunggah ke Digital Library UTDI sesuai dengan ketentuan yang berlaku untuk kepentingan riset dan pendidikan.

Yogyakarta, 20 Agustus 2024

Penulis,

Heru Dwi Pangestu

185410038