Name: Bhatt Priyam

Team; Python

Step 1 — Install Libraries

```
fastapi
uvicorn
sqlalchemy
passlib[bcrypt]
python-jose
```

Step 2 — Database

database.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base

DATABASE_URL = "sqlite:///./food.db"

engine = create_engine(
    DATABASE_URL,
    connect_args={"check_same_thread": False}
)

SessionLocal = sessionmaker(bind=engine, autoflush=False)
Base = declarative_base()
```

Step 3 — Models (Data Storage)

models.py

```
from sqlalchemy import Column, Integer, String, Date, ForeignKey

from database import Base


class User(Base):

    __tablename__ = "users"


    id = Column(Integer, primary_key=True)

    name = Column(String)

    email = Column(String, unique=True)

    password = Column(String)



class FoodItem(Base):

    __tablename__ = "foods"


    id = Column(Integer, primary_key=True)

    user_id = Column(Integer, ForeignKey("users.id"))

    food_name = Column(String)

    quantity = Column(Integer)

    purchase_date = Column(Date)

    expiry_date = Column(Date)
```

Step 4 — Schemas (API Input/Output)

schemas.py

```
from pydantic import BaseModel

from datetime import date


class UserCreate(BaseModel):

   name: str

   email: str

   password: str



class Login(BaseModel):

   email: str

   password: str



class FoodCreate(BaseModel):

   food_name: str

   quantity: int

   purchase_date: date

   expiry_date: date
```

Step 5 — Authentication Logic

auth.py

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from passlib.context import CryptContext

import models, schemas
from database import SessionLocal

router = APIRouter()
pwd_context = CryptContext(schemes=["bcrypt"])

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# Register User
@router.post("/register")
def register(user: schemas.UserCreate, db: Session = Depends(get_db)):

    existing = db.query(models.User).filter(
        models.User.email == user.email
    ).first()
    if existing:
        raise HTTPException(status_code=400, detail="Email already exists")

    hashed = pwd_context.hash(user.password)

    new_user = models.User(
        name=user.name,
        email=user.email,
        password=hashed
    )

    db.add(new_user)
    db.commit()

    return {"message": "User registered successfully"}

# Login User
@router.post("/login")
def login(data: schemas.Login, db: Session = Depends(get_db)):

    user = db.query(models.User).filter(
        models.User.email == data.email
    ).first()

    if not user or not pwd_context.verify(data.password, user.password):
        raise HTTPException(status_code=401, detail="Invalid credentials")

    return {"message": "Login successful"}
```

Step 6 — Food Tracking APIs (Core Task)

food.py

```
from fastapi import APIRouter, Depends

from sqlalchemy.orm import Session

from datetime import date


import models, schemas

from database import SessionLocal


router = APIRouter()


def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()


# Add Food

@router.post("/foods")

def add_food(food: schemas.FoodCreate, db: Session = Depends(get_db)):


    new_food = models.FoodItem(
        user_id=1,
        food_name=food.food_name,
        quantity=food.quantity,
        purchase_date=food.purchase_date,
        expiry_date=food.expiry_date
    )

    db.add(new_food)
    db.commit()

    return {"message": "Food added"}

# Get All Foods
@router.get("/foods")
def get_foods(db: Session = Depends(get_db)):
    return db.query(models.FoodItem).all()

# Expiry Alert Logic (Data Processing)
@router.get("/expiry-alerts")
def expiry_alert(db: Session = Depends(get_db)):

    today = date.today()
    foods = db.query(models.FoodItem).all()

    alerts = []

    for food in foods:
        days_left = (food.expiry_date - today).days

        if days_left <= 3:
            alerts.append({
                "food": food.food_name,
                "days_left": days_left
            })

    return alerts
```

Step 7 — Utility (Smart Logic)

utils.py

```
def waste_percentage(total, expired):
  if total == 0:
    return 0
  return (expired / total) * 100
```

Step 8 — Main File (Run App)

main.py

```
from fastapi import FastAPI

import models

from database import engine

from auth import router as auth_router

from food import router as food_router


models.Base.metadata.create_all(bind=engine)


app = FastAPI(title="Smart Food Waste Reduction System")


app.include_router(auth_router)

app.include_router(food router)



@app.get("/")

def home():

    return {"message": "API Running Successfully"}
```

▶️ Run Project

```
uvicorn main:app --reload
```