```
from fastapi import FastAPI,
HTTPException, status
from pydantic import BaseModel,
Field
from typing import List, Optional
from enum import Enum

app = FastAPI(
    title="Fast Food App API",
    description="A fully featured Fast
Food REST API with menu and order
management",
    version="1.0.0"
)

# ---------------------------------
# Models
# ---------------------------------

class FoodCategory(str, Enum):
    burger = "Burger"
    pizza = "Pizza"
    drink = "Drink"
    dessert = "Dessert"
    other = "Other"

class FoodItemBase(BaseModel):
    name: str = Field(..., min_length=1,
max_length=100)
    description: Optional[str] =
Field(None, max_length=300)
```

```
    price: float = Field(..., gt=0)
    category: FoodCategory
    is_available: bool = True

class
FoodItemCreate(FoodItemBase):
    pass

class FoodItemUpdate(BaseModel):
    name: Optional[str] = Field(None,
min_length=1, max_length=100)
    description: Optional[str] =
Field(None, max_length=300)
    price: Optional[float] = Field(None,
gt=0)
    category: Optional[FoodCategory]
    is_available: Optional[bool]

class FoodItem(FoodItemBase):
    id: int

    class Config:
        orm_mode = True

class OrderStatus(str, Enum):
    pending = "Pending"
    preparing = "Preparing"
    ready = "Ready"
    completed = "Completed"
    cancelled = "Cancelled"

class OrderItem(BaseModel):
    food_id: int
```

```
    quantity: int = Field(..., gt=0)

class OrderCreate(BaseModel):
    customer_name: str = Field(...,
min_length=1, max_length=100)
    items: List[OrderItem] = Field(...,
min_items=1)
    special_requests: Optional[str] =
Field(None, max_length=500)

class
OrderUpdateStatus(BaseModel):
    status: OrderStatus

class OrderItemDetail(BaseModel):
    food: FoodItem
    quantity: int

class Order(BaseModel):
    id: int
    customer_name: str
    items: List[OrderItemDetail]
    special_requests: Optional[str]
    status: OrderStatus

    class Config:
        orm_mode = True

# ---------------------------------
# In-memory "Database"
# ---------------------------------
```

```
orders: List[Order] = []

food_id_seq = 1
order_id_seq = 1

# ---------------------------------
# Food Menu Routes
# ---------------------------------

@app.get("/", tags=["Root"])
def read_root():
    return {"message": "Welcome to
the Fast Food App API! Visit /docs
for API documentation."}

@app.get("/menu",
response_model=List[FoodItem],
tags=["Menu"])
def list_menu():
    return food_menu

@app.get("/menu/{food_id}",
response_model=FoodItem,
tags=["Menu"])
def get_food_item(food_id: int):
    for food in food_menu:
        if food.id == food_id:
            return food
    raise
HTTPException(status_code=404,
detail="Food item not found")

@app.post("/menu",
```

```
response_model=FoodItem, status_
code=status.HTTP_201_CREATED,
tags=["Menu"])
def create_food_item(food:
FoodItemCreate):
    global food_id_seq
    new_food =
FoodItem(id=food_id_seq,
**food.dict())
    food_id_seq += 1
    food_menu.append(new_food)
    return new_food

@app.put("/menu/{food_id}",
response_model=FoodItem,
tags=["Menu"])
def update_food_item(food_id: int,
food_update: FoodItemUpdate):
    for index, food in
enumerate(food_menu):
        if food.id == food_id:
            updated_data = food.dict()
            update_fields = food_update.di
ct(exclude_unset=True)

updated_data.update(update_fields)
            updated_food =
FoodItem(**updated_data)
            food_menu[index] =
updated_food
            return updated_food
    raise
HTTPException(status_code=404,
```

```
detail="Food item not found")

@app.delete("/menu/{food
_id}", status_code=status.
HTTP_204_NO_CONTENT,
tags=["Menu"])
def delete_food_item(food_id: int):
    for index, food in
enumerate(food_menu):
        if food.id == food_id:
            del food_menu[index]
            return
    raise
HTTPException(status_code=404,
detail="Food item not found")

# ---------------------------------
# Orders Routes
# ---------------------------------

@app.get("/orders",
response_model=List[Order],
tags=["Orders"])
def list_orders():
    return orders

@app.get("/orders/{order_id}",
response_model=Order,
tags=["Orders"])
def get_order(order_id: int):
    for order in orders:
        if order.id == order_id:
            return order
```

```
        special_requests=order_create.s
pecial_requests,
        status=OrderStatus.pending
    )
    order_id_seq += 1
    orders.append(new_order)
    return new_order

@app.put("/orders/{order_id}
/status", response_model=Order,
tags=["Orders"])
def update_order_status(order_id: int,
status_update: OrderUpdateStatus):
    for index, order in
enumerate(orders):
        if order.id == order_id:
            orders[index].status =
status_update.status
            return orders[index]
    raise
HTTPException(status_code=404,
detail="Order not found")

@app.delete("/orders/{order
_id}", status_code=status.
HTTP_204_NO_CONTENT,
tags=["Orders"])
def delete_order(order_id: int):
    for index, order in
enumerate(orders):
        if order.id == order_id:
            del orders[index]
            return
```

```
            return
    raise
HTTPException(status_code=404,
detail="Order not found")

# ---------------------------------
# Extra utility routes
# ---------------------------------

@app.get("/menu
/category/{category}",
response_model=List[FoodItem],
tags=["Menu"])
def get_food_by_category(category:
FoodCategory):
    return [food for food in food_menu
if food.category == category and
food.is_available]

@app.get("/orders/status/{status}",
response_model=List[Order],
tags=["Orders"])
def get_orders_by_status(status:
OrderStatus):
    return [order for order in orders if
order.status == status]
```

12:00 PM