BESTE PRACTICES FLASK

GHENT UNIVERSITY

TEMPLATES (HTML) STRUCTURE

Why put reusable HTML in a separate file:

- No duplication
- You can override what changes

templates/partials/navbar.html

templates/partials/footer.html

```html
<!-- navbar.html -->
<nav>
  <a href="/">Home</a>
  <a href="/about">About</a>
</nav>
```

```
<body>
  {% include 'partials/navbar.html' %}

  {% block content %}{% endblock %}

  {% include 'partials/footer.html' %}
</body>
```

GHENT UNIVERSITY

CSS STRUCTURE

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/base.css') }}">
```

```
/project
  /static
    /css
      base.css
      layout.css
      components.css
  /templates
    base.html
    page.html
```

Why CSS is separate file (not in HTML):

- Cleaner code
- reusability, no duplication

GHENT UNIVERSITY

ROUTES STRUCTURE

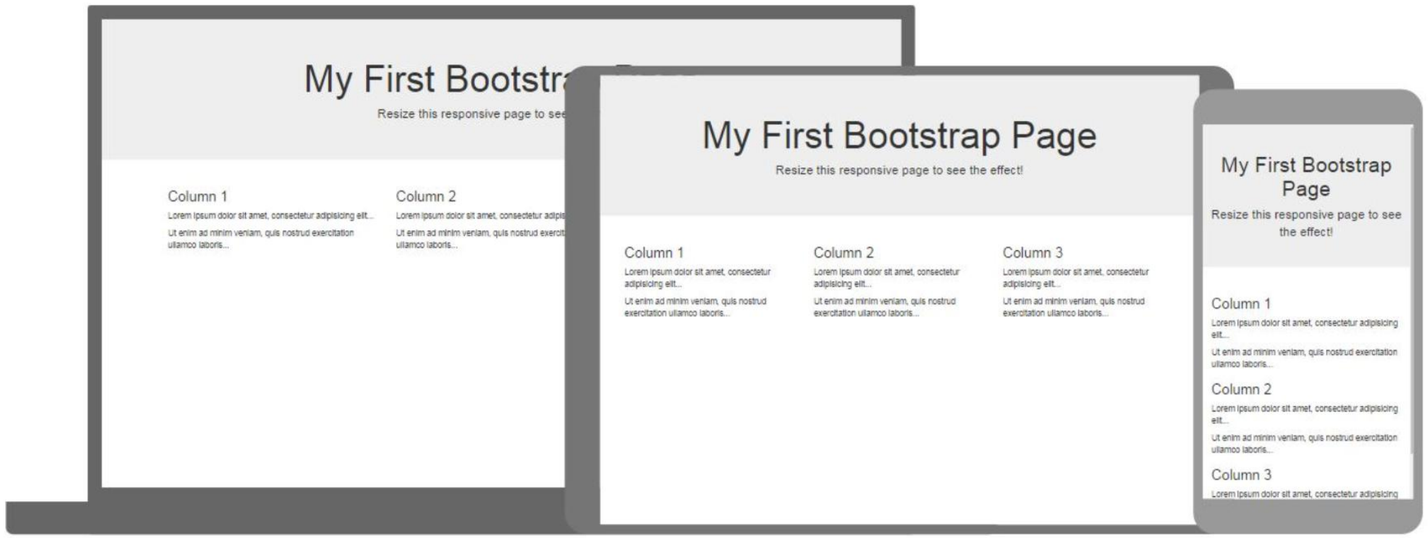
```python
@app.route("/trips")
def trips():
    # big block of logic here
    ...

@app.route("/trips/<int:id>")
def trip_detail(id):
    # copy-pasted logic here
    ...
```

```python
def get_trip_data(id=None):
    # shared logic
    ...

@app.route("/trips")
def trips():
    data = get_trip_data()
    return render_template("trips.html", **data)

@app.route("/trips/<int:id>")
def trip_detail(id):
    data = get_trip_data(id=id)
    return render_template("trip_detail.html", **data)
```

GHENT UNIVERSITY

BOOTSTRAP

- Bootstrap is an HTML and CSS framework that provides some pre-built CSS classes, an easy way to style your web application and make it responsive.
- Add via CDN in the head of the HTML template: https://www.techwithtim.net/tutorials/flask/flask-adding-bootstrap
- Tutorial: https://www.w3schools.com/bootstrap/



GHENT UNIVERSITY

FILTERING IN FRONTEND VS BACKEND

Frontend (Javascript)

- Load all data **once** from the backend (e.g., fetch("/api/listings"))
- Use JavaScript to filter, sort, and search **inside the browser**

Why?

- Faster UX
- Best for small-medium datasets

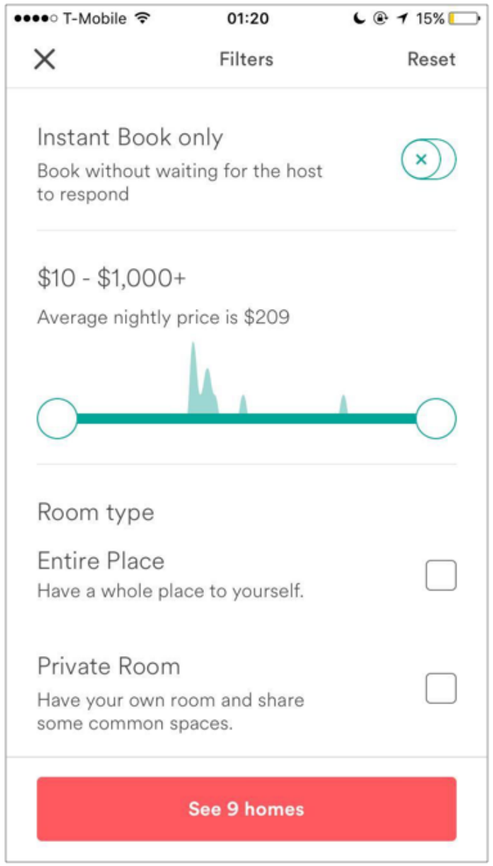
Backend (Flask routes)

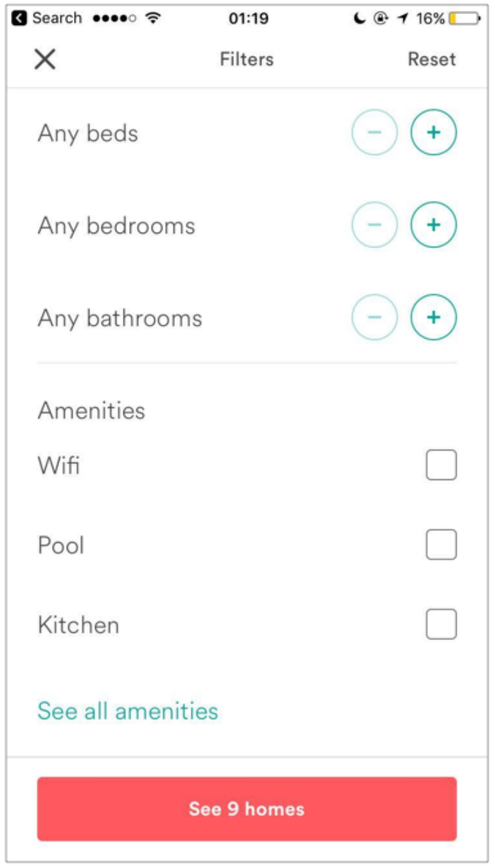
- More secure
- only sends needed records

Why?

For large datasets

```python
@app.route("/api/mainactivities", methods=["GET"])
def get_mainactivities():
    country = request.args.get("country")  # always provided
    activities = MainActivityType.query.filter_by(country=country).all()
    return [a.to_dict() for a in activities]
```





TYPES, CATEGORIES & LISTS

Not in HTML or routes

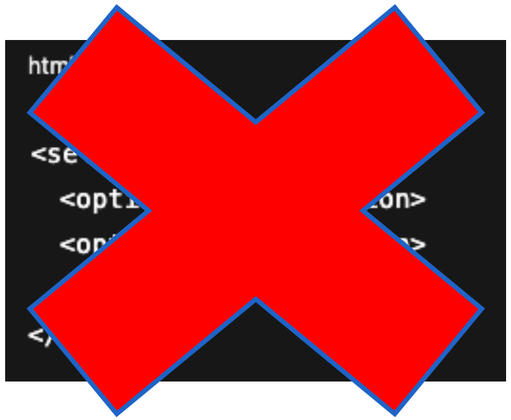
- Fast, but not flexible.
- Only for small, static options (e.g., gender)

In Model (enum)

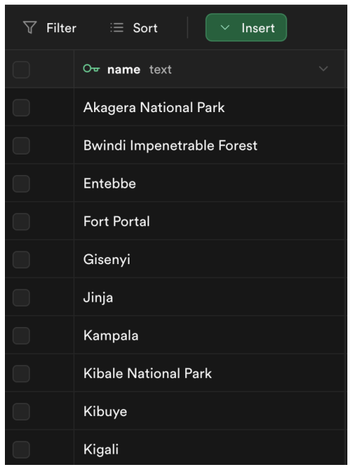
- List is fixed,

Database table

- Fully dynamic
- Perfect for large lists (e.g., countries)
- Perfect for lists that change often (e.g., status dropdown options updatable by admins)



```
class ActivityType(Enum):
    HIKING = "Hiking"
    BIKING = "Biking"
    SAFARI = "Safari"
```



GHENT UNIVERSITY

BACKREF IN MODEL

```python
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)

class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey("user.id"))
    user = db.relationship("User", backref="posts")
```

```
post.user

user.posts
```

```
User (1) ---- (*) Post
```

GHENT UNIVERSITY

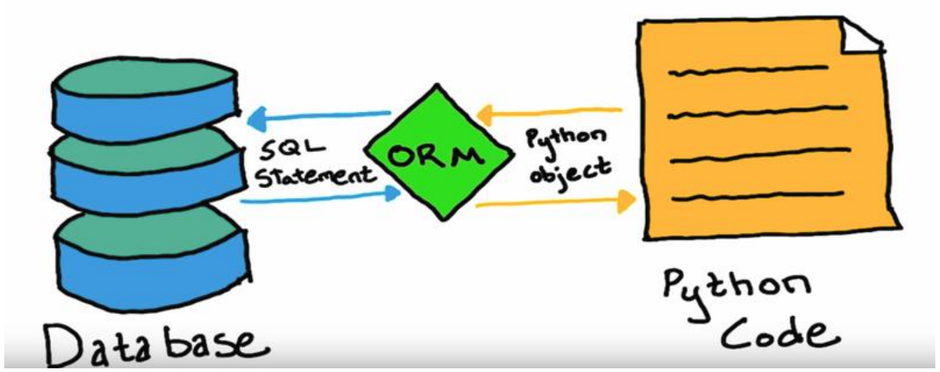
SQALCHEMY ORM VS JSON

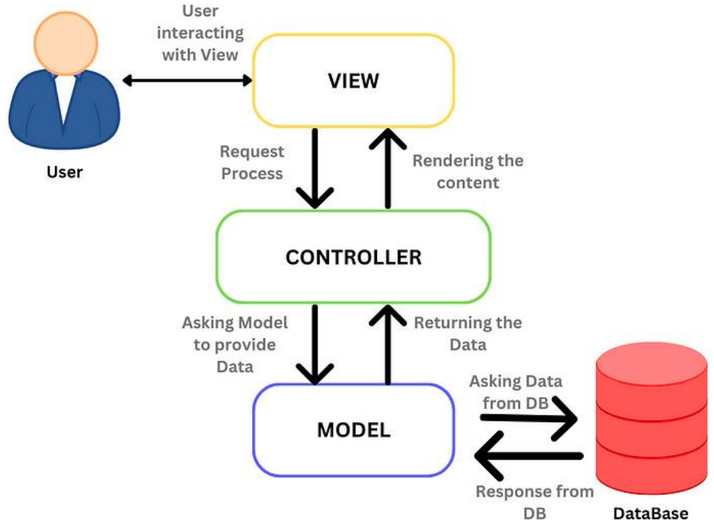
Why via ORM (e.g., SQAlchemy)?

- Direct connection to database
- less code, less bugs
- Lazy loading (data is not loaded until the moment you actually need it)

When JSON?

- When only building a backend API (connected to a frontend in React /Angular)





GHENT UNIVERSITY

SQALCHEMY ORM VS JSON (SUPABASE QUERY)


```
listings = Listing.query.all()
```

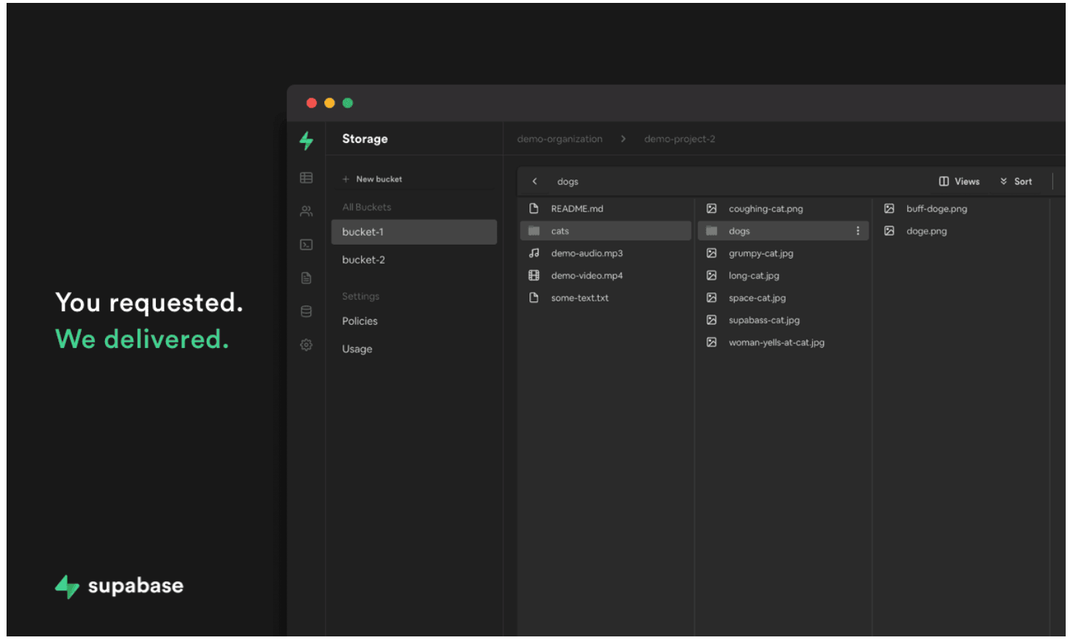
Why?

- Dependence on Supabase
- Less code
- Less complexity (perfect for MVP)

GHENT UNIVERSITY

IMAGES VIA BUCKETS

1. Users upload an image via a form or file picker
2. Flask receives the file
3. Flask uploads the file to a **Supabase Storage bucket**
4. Your Postgres database (table) stores only the **file path**, not the file
5. Flask builds a **an URL** to display the image in templates



GHENT UNIVERSITY

ICAL INTEGRATION

What is iCal (iCalendar)?

- Standard for calendar events (.ics files & feeds)
- Supported by Google Calendar, Apple Calendar, Outlook,
- Format for events, bookings, reminders, time ranges, and availability

Advantages

- Easy integration into calendars
- Automatic reminders on user devices (phone, smartwatch, laptop)
- Easy sync — users import once, calendar updates itself (if using feeds)
- Timezone-safe handling (e.g., TZID=Africa/Kigali)

```python
from flask import Response
from icalendar import Calendar, Event
from datetime import datetime

@app.route("/booking/<int:id>/export")
def export_booking(id):
    cal = Calendar()

    event = Event()
    event.add("summary", "Stay at Lakeview Apartment")
    event.add("dtstart", datetime(2025, 12, 1, 15, 0))   # Check-in
    event.add("dtend", datetime(2025, 12, 5, 11, 0))     # Check-out
    event.add("location", "123 Main Street, Kigali, Rwanda")
    event.add("description", "Your Airbnb-style booking")

    cal.add_component(event)

    return Response(
        cal.to_ical(),
        mimetype="text/calendar",
        headers={"Content-Disposition": "attachment; filename=booking.ics"}
    )
```

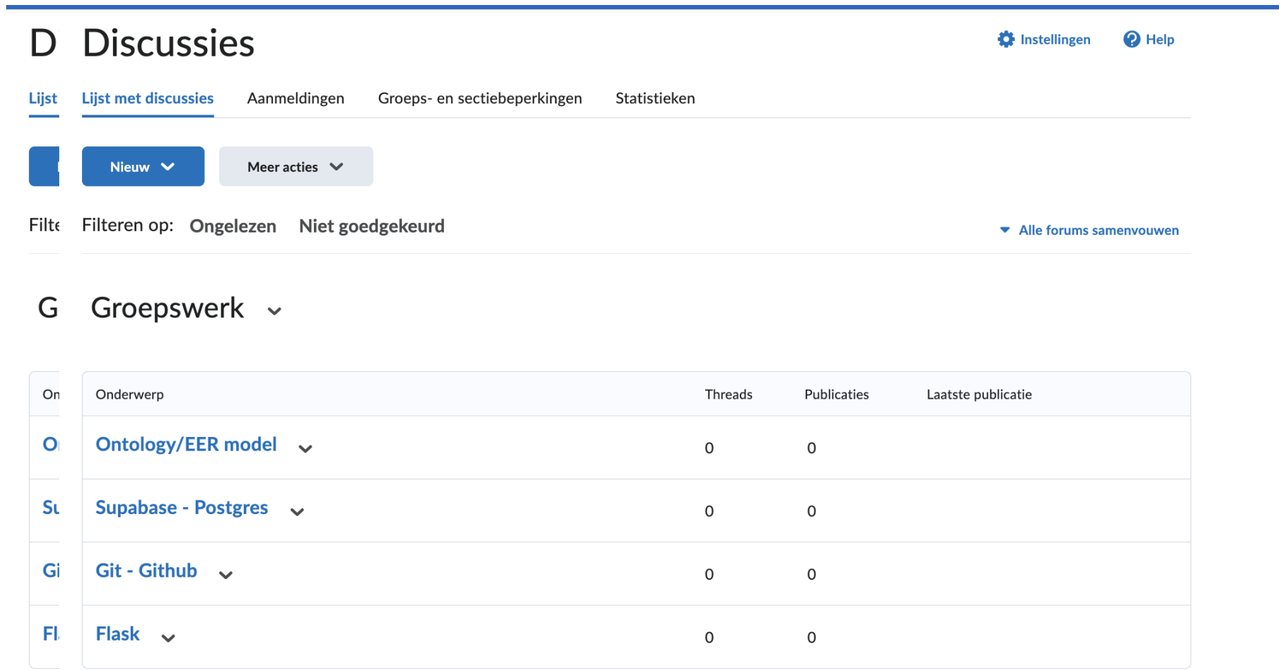
GHENT UNIVERSITY

HOW TO USE CHATGPT/COPILOT

- First give all relevant code you already have
- Mention your stack (Flask, bootstrap, SQAlchemy, …)
- Paste full error
- Understand—ask to explain code and other options
- Use GPT5 with your group (most recent info)?

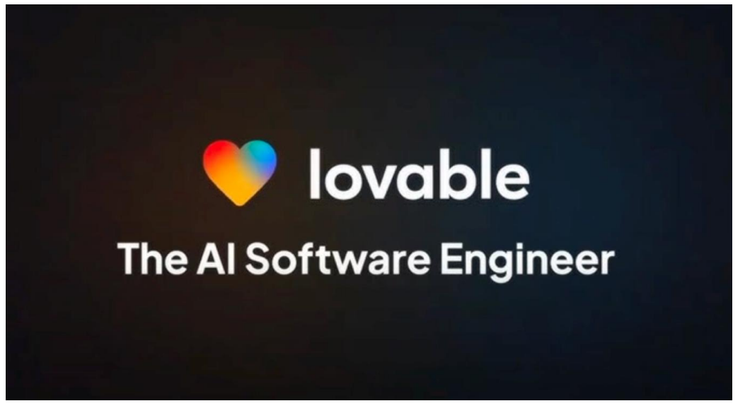
TECHNICAL QUESTIONS? DISCUSSIES PAGE ON A&D

- Choose a good title (name of the error)
- Use Screenshots!
- Help each other (extra point)



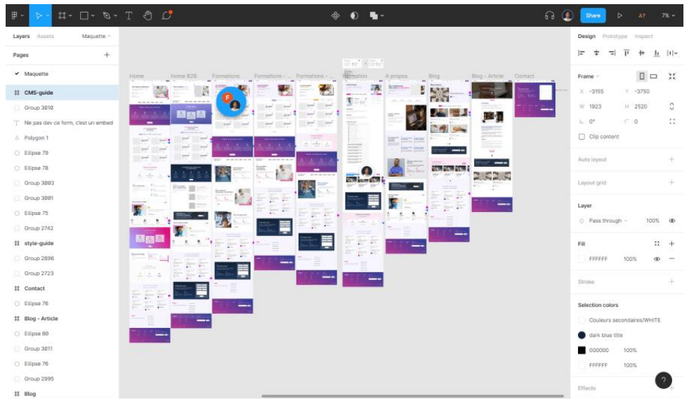
GHENT UNIVERSITY

UI PROTOTYPE



React

Figma



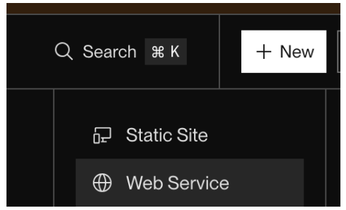
Marvel

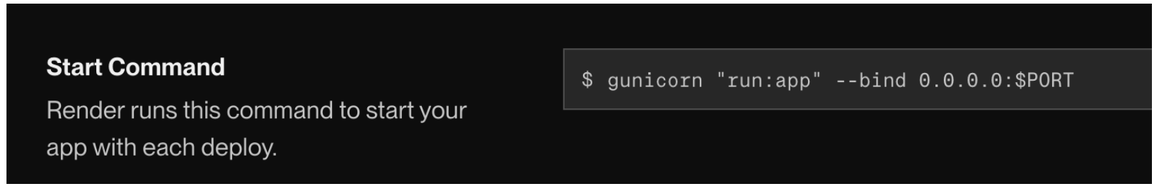
User Testing

Validate designs at speed

DEPLOY WITH RENDER

Render





- New web service (Free)
- Connect go Github repo
- start command: via gunicorn
- Example: https://marketplace-flask-j22l.onrender.com/

GHENT UNIVERSITY

DEFENCE OF PREVIOUS YEAR (LAST WEEK OF EXAMS)

- More info tomorrow

GHENT UNIVERSITY

DEFENCE OF PREVIOUS YEAR

0. Originaliteit	2
1. Layout & Style	3
2. Gebruiksvriendelijkheid	3
3. Complexer algoritme	3
4. Simpelheid & Efficiëntie van Routes	2
5. Responsiveness & css files	2
6. Uitbreiding buiten MVP	2.5
8. Database Model	4
9. Presentatie	1
Total	22.5

GHENT UNIVERSITY