

CS50 Section 9

Slides and Exercises at github.com/cjleggett/2022-section

Think.

Pair.

Share.

Think. Pair. Share.

- How do we organize files in a Flask app?
- How do we get started with Flask?
- What order should I write my code in?

Outline

- Upcoming Schedule (3:05)
- Overall Web Diagram (3:10)
- Flask (3:15)
- Jinja (3:35)
- Incorporating SQL, CSS, JS, Sessions (3:45)
- Example (3:50)
- Life After CS50 (4:05)

Upcoming Schedule

11/7 - 11/13: Finance + Proposal

- Normal pset, lab, section
- Maybe the most time-consuming pset
- But also one of the most fun!
- **And Extra Tutorials!!!**
- Also time to get into groups and start thinking about final project!
- **Test Review Session 4:30-5:30 pm (recorded)**

Financial Advice

- The walkthrough is 20 minutes long, but worth it
- Don't forget the personal touch!
- Save versions that are working
- Work through piece-by-piece:
 - Implement
 - Test
 - Check50
 - Repeat

11/14 - 11/20: Test

- Test released 11/14
- Test Due 11/20
- No Section / Lecture / Lab
- No problem set
- Harvard-Yale is 11/19!
- **Seminars!!!**



Saturday, November 19 • Harvard Stadium

11/21 - 11/27: Break

- Monday Lecture
- Thanksgiving Break!
- No problem set / section / lab
- You should start working on your final project!



11/28 - 12/4: Final Project Sprint

- No Lecture / Section / Pset / Lab
- 12/1 - 12/2: CS50 Hackathon!



12/5 - 12/7: Finishing Up

- No Lecture / Section / Pset / Lab
- 12/6: Final Projects Due
- 12/7: CS50 Fair



Unfortunate Announcement

- I'm getting surgery tomorrow and will be out of commission for a little while
- So if you have logistical questions, email [heads@cs50.harvard.edu!](mailto:heads@cs50.harvard.edu)

Overall Diagram

User
(View)

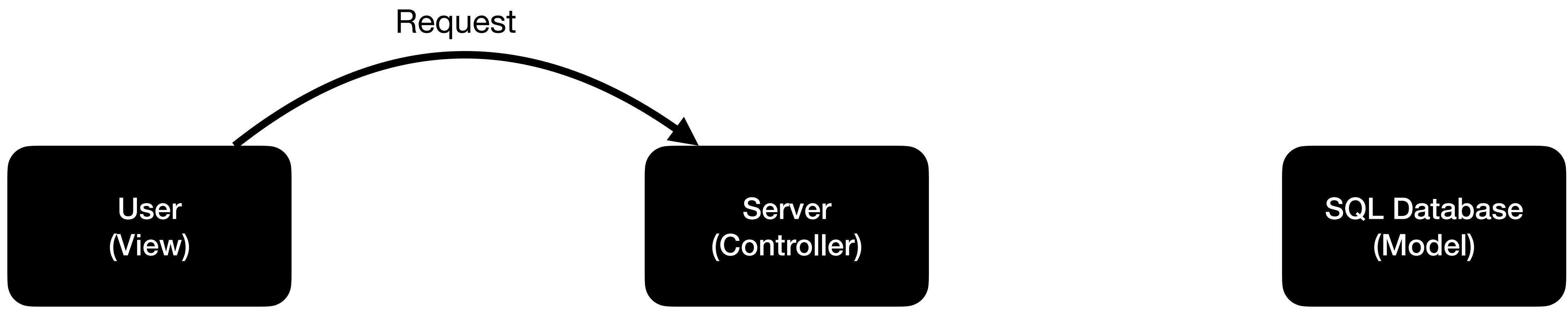
**User
(View)**

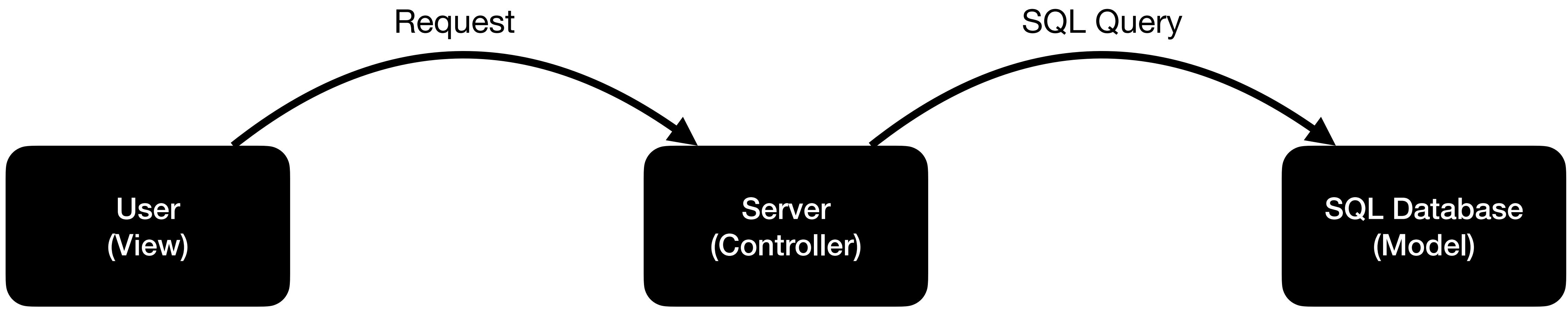
**SQL Database
(Model)**

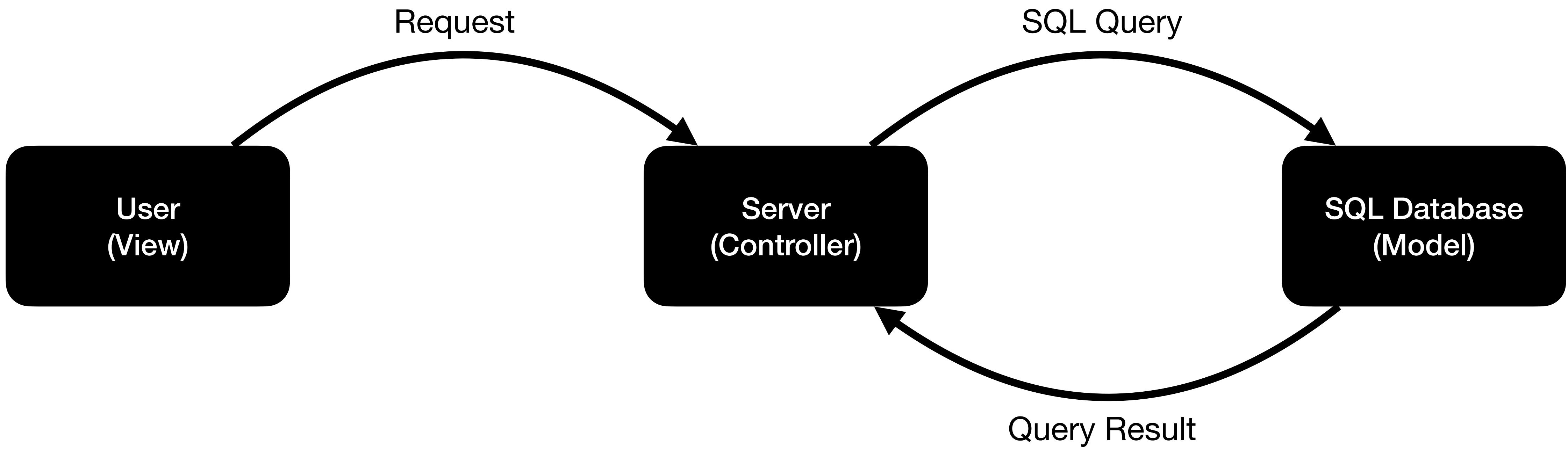
**User
(View)**

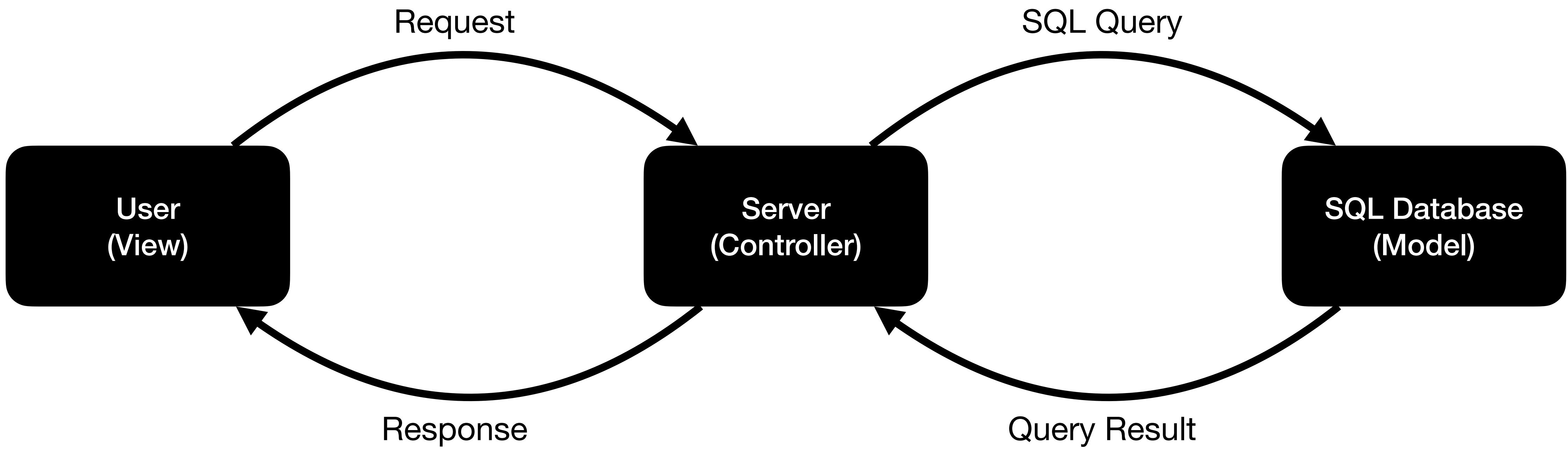
**Server
(Controller)**

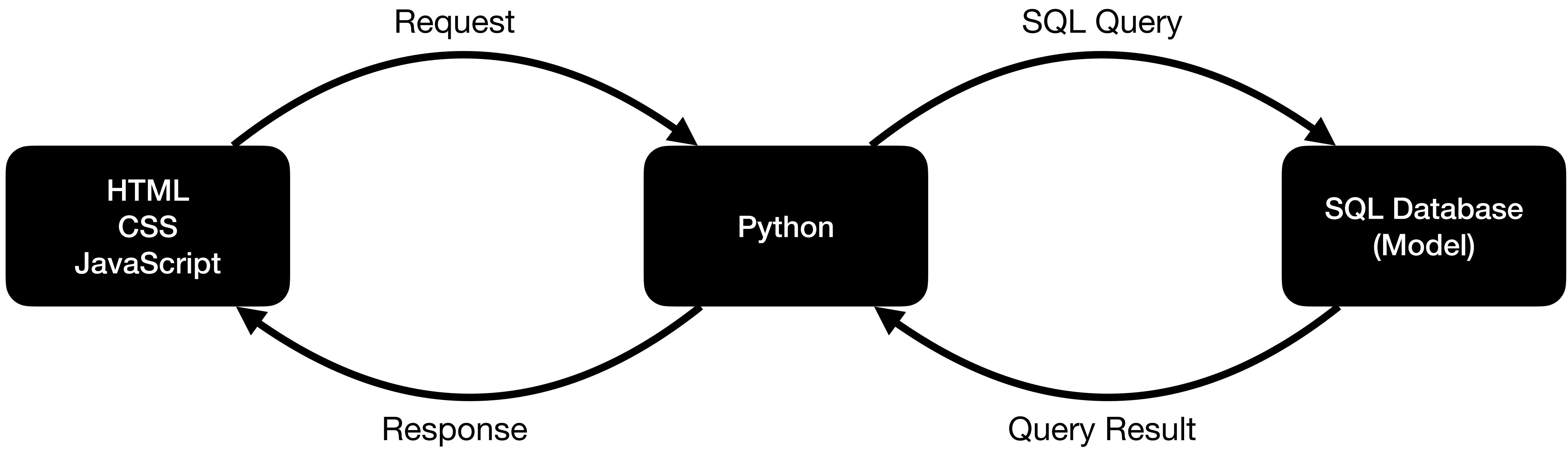
**SQL Database
(Model)**











Flask

Why Flask?

- Allows us to dynamically generate HTML pages
- Allows us to store data over time and over different devices
- It's a simple way to get started compared to some alternatives:
 - Django
 - Express.js
 - Ruby on Rails

Getting Started

```
# Import Flask Library  
from flask import Flask  
  
# Initialize Flask Application  
app = Flask(__name__)
```

Using Functions to Define Behavior

```
# This next line is a “decorator”
# It modifies the function below
@app.route("/")
def index():
    return "You are at the index page"

@app.route("/cat")
def cat():
    return "You are at the cat page"
```

Returning HTML

```
# Import Flask Library
from flask import Flask, render_template

# Initialize Flask Application
app = Flask(__name__)

@app.route("/")
def index(name):
    return render_template("index.html")
```

Passing in Data via URL

```
# We use <> to surround the argument
@app.route("/welcome/<name>")
def welcome(name):
    return f"Welcome, {name}!"
```

Passing in Data via URL query

```
# This handles URL params like ?q=cats for google
@app.route("/welcome")
def welcome():
    name = request.args.get("name")
    return f"Welcome, {name}!"
```

Passing in Data via Form

```
# We use <> to surround the argument
@app.route("/form", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        # Handle the form response
    else:
        # Display the form for the user
    return render_template("form_page.html")
```

Passing in Data via Form

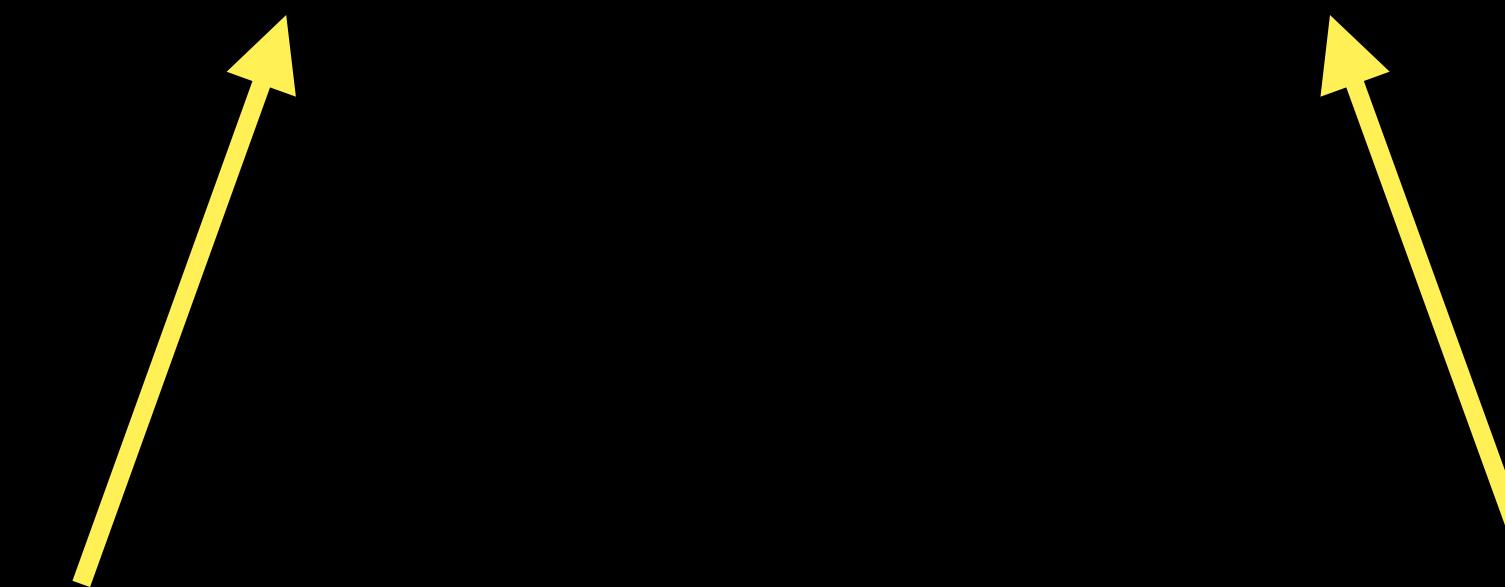
```
# We use <> to surround the argument
@app.route("/form", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        # Handle the form response
        age = request.form.get("age")
        # Do something with age...
    else:
        # Display the form for the user
        return render_template("form_page.html")
```

Passing in Data via Form

```
<form action="/something" method="POST">  
  </form>
```

Route to Request

Request Method



Passing in Data via Form

```
<form action="/something" method="POST">
  <input name="name" type="text">
  <input name="age" type="number">
  <button type="submit">Submit</button>
</form>
```

Passing in Data via Form

```
<form action="/something" method="POST">
  <input name="name" type="text">
  <input name="age" type="number">
  <button type="submit">Submit</button>
</form>
```

Passing in Data via Form

```
<form action="/something" method="POST">
  <input name="name" type="text">
  <input name="age" type="number">
  <button type="submit">Submit</button>
</form>
```

Jinja

Jinja

- A templating language
- Allows us to dynamically generate HTML files
- Allows us to use layouts to avoid repeating code
- Remember to place all HTML files in the “templates” directory

index.html

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <title>Hello!</title>
  <head>
  <body>
    Hello, World!
  </body>
<html>
```

layout.html

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <title>Hello!</title>
  <head>
  <body>
    {%- block body %} 
    {%- endblock %} 
  </body>
<html>
```

index.html

```
{% extends "layout.html" %}

{% block body %}
    Hello, world!
{% endblock %}
```

Response to User

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>Hello!</title>
    </head>
    <body>
        Hello, World!
    </body>
</html>
```

welcome.html

```
{% extends "layout.html" %}

{% block body %}
    Welcome, {{ name }}!
{% endblock %}
```

Response to User

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>Hello!</title>
    <head>
    <body>
        Welcome, Connor!
    </body>
<html>
```

welcome.html

```
{% extends "layout.html" %}

{% block body %}
    Welcome, {{ name }}!
{% endblock %}
```

Response to User

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>Hello!</title>
    <head>
    <body>
        Welcome, Connor!
    </body>
<html>
```

Passing Arguments for Template

```
@app.route("/welcome")
def welcome(name):
    return render_template(
        "welcome.html",
        name=name
    )
```

welcome.html

```
{% extends "layout.html" %}

{% block body %}
    Welcome, {{ name }}!
{% endblock %}
```

Response to User

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <title>Hello!</title>
    <head>
    <body>
        Welcome, Connor!
    </body>
<html>
```

Jinja

- Use {{ }} to input variables
- Use {%-%} to input logic
 - Layouts
 - For Loops
 - If - Else Logic

Incorporating SQL

We already know how to do this!

- Using CS50's Python library, we can import SQL!

Incorporating CSS + JS

CSS + JavaScript

- Include .css and .js files in the “static” directory
- Link to those files using “/static/filename”
- Other than that, it’s all the same!

Sessions

Sessions

- Keep track of data for a single user
- Update session with `session["key"] = value`
- Get value from session with `value = session.get("key")`
- You'll use these to keep track of the current user in Finance
- Lots of login/register details are in walkthrough or already written

Example: Registrar

Life After CS50

Ways to Continue with CS

- Classes
- Clubs
- Jobs/Internships/Research
- Projects
- Joining Staff

Classes

“Typical” Next CS Classes for Concentrators

- CS51: Abstraction and Design in Computer Science
- CS61: Systems
- CS20: Discrete Math for Computer Science
- CS121: Introduction to Theoretical Computer Science
- CS120: Introduction to Theory and Algorithms
- CS124: Data Structures and Algorithms
- Stat 110: Introduction to Probability
- Math 1a/1b/21a/21b

“Typical” Next CS Classes for Concentrators

- **CS51: Abstraction and Design in Computer Science**
- **CS61: Systems**
- **CS20: Discrete Math for Computer Science**
- **CS121: Introduction to Theoretical Computer Science**
- **CS120: Introduction to Theory and Algorithms**
- **CS124: Data Structures and Algorithms**
- **Stat 110: Introduction to Probability**
- **Math 1a/1b/21a/21b**

CS Classes for Everyone

- CS Classes Related to Other Fields
 - CS109a: Data Science
 - CS136: Economics and Computation
 - CS171: Data Visualization
- Classes in Other Fields Related to CS
 - Gov50: Data
 - Econ50: Big Data in Economics
 - Stat 111: Introduction to Statistical Inference
 - Econ 1123: Econometrics

CS Classes for Everyone

- CS Classes Related to Other Fields
 - **CS109a: Data Science**
 - **CS136: Economics and Computation**
 - CS171: Data Visualization
- Classes in Other Fields Related to CS
 - Gov50: Data
 - **Econ50: Big Data in Economics**
 - **Stat 111: Introduction to Statistical Inference**
 - **Econ 1123: Econometrics**

CS Classes for Everyone

- CS Classes Related to Other Fields
 - CS109a: Data Science
 - CS136: Economics and Computation
 - CS171: Data Visualization
- Classes in Other Fields Related to CS
 - Gov50: Data
 - Econ50: Big Data in Economics
 - Stat 111: Introduction to Statistical Inference
 - **Econ 1123: Econometrics**

Clubs

- Harvard Computer Society
- Women in Computer Science
- Datamatch
- Engineers Without Borders
- Harvard College Open Data Project
- Harvard Sports Analytics Collective
- Tech position for any club!

Jobs/Internships/Research

- Check out Crimson Careers for Job Postings
- Go to Career Fairs (especially startup fairs)
- Check out research on Harvard CS Website
- Think about applying CS concepts to non-CS jobs/research
 - My roommate who's pre-med uses Python all the time in her lab!
 - My roommate who worked at LucasFilm wrote a program to automate a repetitive task

Personal Projects

- Making a Website
- Making an App
- Analyzing Data
- Automating Some Repetitive Task

Personal Projects

- Making a Website
- Making an App
- Analyzing Data
- **Automating Some Repetitive Task**

Moving Away From Codespace

- Install a Text Editor (VSCode, Sublime, are good and free!)
- Start using GitHub to save work and collaborate
- Look up tutorials on how to use new technologies
- For Python, look into downloading Anaconda to use Python on your computer

Join CS50 Staff

- Emails will come out sometime next semester
- TFs: Lab, OHs, Tutorials
- CAs: OHs, Tutorials
- Helps solidify knowledge and a great way to meet new people!

cjleggett@college.harvard.edu

If you have questions!