Project Discussion + Web Dev Intro

Introduction to Databases
CompSci 316 Fall 2021
Course project

Please refer to the project description—posted on the Schedule page of the website

• Two options: standard vs. open
• Teamwork: 5 members per team
• Three milestones + a final demo
  • First one due in exactly one week
• Six additional weekly updates
  • Individually reported; verified by a teammate
• End-of-semester competition of project videos
  • Extra credits for Audience/Staff Choice Awards!
Standard option: mini-Amazon

Sellers create products and an inventory for sale; Users browse/purchase products, and review products and sellers who fulfill their orders;
Transactions conducted within the website using virtual currency.

• Beginner-friendly; Python-based skeleton codebase provided for you to build on
• Spec’d out for exactly 5 members with clear division of work—but you still must work together
• Team scoring by default
  • Getting all basic features correctly done gives you 90/100
  • Individual scoring possible, but capped at 89/100
Standard option: rough timeline

• **Milestone 1** (in a week): team declaration; nothing else expected
  • We will help you form a complete team if you are unable

• **Milestone 2** (back from fall break): complete DB/UI design; small test DB; get skeleton codebase and development environment up and running

• **Milestone 3** (early Nov.): big test DB; good progress on all components; **5-minute video** update

• **Final demo** (early Dec.): report + **10-minute video**; **15-minute live demo+Q/A** with TA
Open option: overview

• You are expected to do MORE work
• No individual scoring possible
• Depending on what platforms you build on, course staff might not be able to offer you as much help
• Riskier, but more freedom
• Can have a team of 6, if warranted by project scope
Open option: rough timeline

• **Milestone 1** (in a week): team declaration + ~500-word proposal
  - You need a complete team + a concrete proposal of the right scope to be approved; otherwise, you may be asked to switch to the standard option

• **Milestone 2** (back from fall break): progress update + proposed list of basic/bonus features that you will be evaluated on

• **Milestone 3** (early Nov.): progress update with 5-minute video + final list of basic/bonus features

• **Final demo** (early Dec.): self-contained ~1k-word report + 10-minute video; 25-minute Q/A with TA

*If your project is analogous to mini-Amazone, you can use the timeline for mini-Amazone to guide your progress*
Recap

• Standard vs. open
  • For open, must get team + proposal ready in a week!

• As with the real world, you are expected to
  • Work in a team
  • Learn things as you go
  • And struggle

• But know that we are here to help
  • Be proactive in seeking help from course staff
Mapping NC’s Health

Nadia Bey
Super-Short Intro to Full-Stack Web Dev
A static website

```html
<html>
<head><title>DB-Beers Example</title></head>
<body>
  <h1>DB-Beers</h1>
  <h2 id="heading">All Drinkers</h2>
  <ul>
    <li><a href="/drinker/Amy">Amy</a></li>
    <li><a href="/drinker/Be".en-US/docs/Learn/Server-side/First_steps/Client-Server_overview/basic_static_app_server.png
```
Towards a dynamic website

- Imagine a function that dynamically generates the HTTP response
  - Visiting the URL leads to calling the function
  - The function returns the HTML page (as a string)
  - The function can query the database for content!

```python
def all_drinkers():
    html_out = '<html>
    # more html...
    cur = db.cursor()
    cur.execute('SELECT * FROM drinker')
    html_out += '<ul>
    for name, address in cur:
        html_out += '<li><a href="/drinker/{}">{}</a></li>
'.format(name, name)
    html_out += '</ul>
    # more html...
    html_out += '</html>
    return html_out
```
“Refactor” that function!

• Separate data from presentation, e.g.:
  • Data: list of drinker names
  • Presentation: a HTML template, with some processing directives to embed data items

```python
def all_drinkers():
    cur = db.cursor()
    cur.execute('SELECT * FROM drinker')
    drinker_names = [name for name, address in cur]
    return render_template('index.html', drinkers=drinker_names)
```
Calling these functions

- URL in the HTTP request specifies the function to call
  - AKA routes, endpoints
- HTTP request also encodes any input parameter values to call the function with
  - Can be part of the URL (GET) or the request body (POST)

```python
@app.route('/drinker/<name>')
def drinker(name):
    cur = db.cursor()
    cur.execute('SELECT * FROM drinker WHERE name = %s', (name,))
    name, address = cur.fetchone()
    return render_template('drinker.html',
                           drinker={'name': name, 'address': address})
```
But who calls these functions?

- User usually don’t type specific URLs themselves
- Calls are typically embedded in your HTML pages

```
<html>
<head><title>DB-Beers Example</title></head>
<body>
  <h1>DB-Beers</h1>
  <h2 id="heading">All Drinkers</h2>
  <ul>
    {% for drinker in drinkers %}
      <li><a href="{url_for('drinker', name=drinker)}">{{drinker}}</a></li>
    {% endfor %}
  </ul>
</body>
</html>
```
Full-stack (with “thin client”)

Our standard project skeleton uses this architecture

- Flask for Web application/server
- PostgreSQL for database
- Linux for files/server
Smarter-client alternative

• The client is getting more powerful—why not let it do more processing (beyond displaying HTML)?

• Roughly speaking
  • Initial visit to the website loads a page that essentially carries a JavaScript program
  • In response to user interactions, JavaScript
    • Makes API calls to the server to get data
    • Modifies the current HTML in place
      ☞ On the server, your functions only need to return data (typically in JSON), and don’t need to worry about presentation at all

• A website could behave more like a native app, changing dynamically in response to user actions without loading a new page!
  • Many JavaScript frameworks have adopted this approach
  • We are not expecting or supporting this alternative for the standard project option though!
More project resources

• Skeleton mini-Amazon for open option
  • To be released by next week

• Working web dev examples in Flask
  • See “Help” on course website for more details

• Duke Co-Lab offerings
  • Many interesting “Roots” courses
    • Flask Fundamentals, Intro to JavaScript, “Create a Website” series, Node.js, React, etc.
    • Advance registration required
  • Office hours on full-stack web/app development