2021-03-01 by Jason Femino

Why You Should Hire Me...

... maybe

Code

https://github.com/jlfemino/simulmedia

Python + Flask + Swagger + SQLite:

- Choices were about delivering a small breadth of features, without incurring any deep costs in any one. (I mean, I only have an hour.)
- It uses a repository.
- It uses some basic object-oriented principles.
- It uses some basic RESTful-ish design principles.
- It has tiny bit of documentation:
  - A `README.md` to get running
  - API documentation available Swagger

Architechture and Design

Separation of concerns

Classes:

- Used classes to allow simple data validation and utility methods.

Data Access Objects:

- Focused on marshalling/unmarshalling of persisted data (i.e. little to no business logic).

Services:

- Implement business logic by interacting with DAOs and/or other services.
- Process normalized data classes (vice raw input).

Testing

There's never enough

- Build tests early.
- Coverage isn't everything, but it can expose potential problem areas.
- Pursue all the types: unit, integration, functional, regression, stress, performance, smoke.
- Plan testing around common mistakes:
  - Developers too busy to try their code out in the UI?
    (Lean on integration, regression, performance tests to find issues.)
  - QA culture relies heavily on manual UI testing?
    (Automate, even if you have to refactor your UI).

Scalability

Always seems easy, until it's time...

- Build in scalability early:
  - Keep services as stateless as possible.
  - Decide how to divide your project's resources.
- Techniques:
  - Keep individual service interfaces and relationships simple.
  - Take advantage of cloud computing auto-scaling.

Optimizations:

- CDNs for basic data web data (static HTML resources, global/regional config)
- Gateway-to-Service caching
- Service-to-Service caching
- Service-to-DB caching
- Load-balancing for services and databases

Security

Not my strongest suit, but I take it seriously

- Keep as much of the surface area of your applications locked-down as practical.
- Take time to plan auth UX and workflows.
- Favor industry's evolving best-practices over bespoke solutions.
- Stop storing passwords/credentials in source code and config files. Seriously.