Demo Day Script

Pitch:

This is DoulaHoop. I came up with the idea for this project before I started Hackbright, when I was training to become a doula, which is a person who provides non-medical care for women during childbirth. This app provides an online space for doulas and expectant parents to find each other.

Location is a key factor in finding a doula, since she will have to be able to reach the mom quickly once labor starts, so I made the doula user database searchable by zipcode.

Security, accuracy of information, and functionality are critical aspects of my application.

I wanted to make this a safe space for parents to share personal information and find care and support during this vulnerable time. Visibility settings allow me to restrict access to personal information based on parent preferences.

As parents needs evolve and doulas’ services change, I want my users to be able to keep their profiles updated.

Testing the application enabled me to demonstrate that I could accurately store data and maintain a usable application. It also gave me time to consider additional security measures (injection attacks, being able to hack into the database and obtain user information) that would enhance user experience.

Why would one user class that the two types inherit from be better?

* ideally each user would have a unique id
* unique user ids would enable me to use a single user database to query users
* in python, I could use a User class that each user type would inherit from to control user id assignment

Why is it important that there are two user types?

* patients want privacy
* they have different needs
  + doulas are here to vendor a product
  + patients are seeking a service
* allows me to assign correct permissions

What was the most challenging part of the project?

* Testing!
  + Did not cover it in course, so had to learn outside of instruction
  + Learning how to refactor code
  + Making smart choices about refactoring code so that I’m testing parts of my application that are necessary for functionality
* If they ask what you mean --
  + learning how to evaluate what a payload includes
  + learning how to set up test databases and test users

What do you want to do? What kind of role?

* web application design
  + programming for business logic (creating user types, understanding relationships among users, optimizing for user experience)
  + leveraging background in statistics (MPH) for data engineering
  + using python, ruby, javascript; some interest in d3, data viz

What do you wish you had known before you started this project? / what would do differently if you were to start over?

* database design
* writing code for testing
* maybe sketch out messaging tooling, calendar functionality, search based on preferences

Additional things I would do:

* increased security
* messaging (secure)
* calendar
* more testing
* postgres database/deploy to Heroku
* increased search capability
* see above

How things work

**x-editable**

<a href="#" class="editable" data-type="text" data-pk="{{ doula.id }}" data-name="firstname" data-original-title title data-url="/doula\_edit" data-title="Enter your first name">

* put an <a> tag around the item that you want to be editable
* when it is clicked, it will send all of the data listed in the tag to the appropriate app route (/doula\_edit)
* based on the NAME (”field”) that gets passed, and the VALUE (what will replace the old value for that attribute), controller will change that information in the database and then commit it.
* Button function is done with Javascript,

$('#enable').click(function() {

$('.editable').editable('toggleDisabled');

})

* Javascript is only active if the user (from the session) owns that profile page (done with an if-statement around the script tag)

**Flask**

* Web framework
* works with python and Jinja for templating
* I also used the werkzeug extension for encrypting and checking passwords
* I also used the Flask extension for logins (protects pages that require log in, manages the session for logged-in users, manages log-out)

DB structure

**Python Unittest**

* did unit-testing of specific methods (required refactoring my code)
* used it as well to set up a test-client to test the flask performance, serve an HTTP request, and make sure everything that is expected to be in it is actually in it.

**PyQuery**

* used for parsing HTML in Python
* Similar to BeautifulSoup, but can be faster if it is parsing multiple documents at a time.
* like jQuery, it searches for identifiers to isolate items in the page

**Werkzeug**

* Web Server Gateway Interface utility library for Python

**scoped session**

* The [Session](http://docs.sqlalchemy.org/en/rel_0_9/orm/session.html#sqlalchemy.orm.session.Session) establishes all conversations with the database and represents a “holding zone” for all the objects which you’ve loaded or associated with it during its lifespan. It provides the entrypoint to acquire a [Query](http://docs.sqlalchemy.org/en/rel_0_9/orm/query.html#sqlalchemy.orm.query.Query) object, which sends queries to the database using the [Session](http://docs.sqlalchemy.org/en/rel_0_9/orm/session.html#sqlalchemy.orm.session.Session) object’s current database connection, populating result rows into objects that are then stored in the [Session](http://docs.sqlalchemy.org/en/rel_0_9/orm/session.html#sqlalchemy.orm.session.Session), inside a structure called the [Identity Map](http://martinfowler.com/eaaCatalog/identityMap.html) - a data structure that maintains unique copies of each object, where “unique” means “only one object with a particular primary key”.

SQLAlchemy

* SQLAlchemy is an [open source](http://en.wikipedia.org/wiki/Open_source) [SQL](http://en.wikipedia.org/wiki/SQL) toolkit and [object-relational mapper](http://en.wikipedia.org/wiki/Object-relational_mapping)(ORM) for the [Python programming language](http://en.wikipedia.org/wiki/Python_(programming_language))
* allows me to create tables as user classes (with Base as the object type),
* Each class attribute is a column in the table
* Each instance of the class is a row in the table
* Allows me to query more easily, translates my queries into SQL when it talks to the database.
* Allows me to change/save data in the database