Dec 2019: FullStackPDX

Two-Step Registration *and* Role-Based Access Control

Summary of what will be covered

- Introduction
  - Inspiration for the project
  - Some challenges that I faced
- Review the anatomy of the site
  - Login and Registration
  - Special pages (based on account confirmation status)
  - Membership purchase funnel
  - Member-only pages (access based on role)
  - Notification Emails
- Upcoming Features
- Questions

Main Features

- Login
- Registration
- Confirmation step
  - Routing based on confirmation status
  - Content filtering based on confirmation status
- Membership system (roles)
  - Routing based on role (membership level)
  - Content filtering based on role (membership level)
- Email notifications
- *Upcoming features*
  - *User profile pages for managing account and membership*

Under Research

- Where is best place to implement access control?
- Using optional parameters for custom functions
- Replacing current registration system in Lending app
- Using RBAC to create private “groups” in Lending App
- Enforcing strong passwords
- Better error handling (Try/Except)
- Moving to class-based views
- TESTING! adding unit tests
- Experimenting with Selenium
- Ensuring design & code follows best practices

Project Assets

GitHub

- https://github.com/lar-mo/registration-rbac

App's Readme.md

- https://github.com/lar-mo/registration-rbac/blob/master/README.md

Asana project

- https://app.asana.com/0/1153713004006177/1153713004006251

Questions?

```
import random
import re
def answer(q):
    ynm = ["Yes!", "No!", "Maybe!"]
    qa = {"who": "The identity is unknown.",
          "what": "Cannot predict now.",
          "when": "When the time is right.",
          "where": "The path is unclear.",
          "how": "There are many ways.",
          "why": "Your question is deep.",
          "are": ynm[random.randint(0,2)],
          "will": ynm[random.randint(0,2)],
          "can": ynm[random.randint(0,2)]}
    character_regex = r'^\w+'
    question_lower = q.lower()
    start_word = ''.join(re.findall(character_regex, question_lower))
    return qa.get(start_word,"Concentrate and ask again")
def get_all_answers(questions):
    for question in questions:
        response = answer(question)
        print(f"{question}? {response}")
questions = []
while True:
    q = input("Submit your question, or type (done): ")
    if q == 'done':
        get_all_answers(questions)
        break
    else:
        questions.append(q)
```