CSCI-5448 DOCTOR FINDER

Kaiyue An

Sean Tranchetti

Shubham Mudgal

Tahani Almanie

PROJECT SUMMARY

Doctor Finder is a web application that provides a service to help patients to find the right doctors based on some criteria

Functionalities:

- **Patient** can create an account, login, search for doctors, sort the searching results, view doctor's info, write reviews for doctors, locate a doctor's location on map, save/remove favorite doctors and edit account info.

- **Doctor** can login and update information

- **Admin** can add/delete/update users and doctors information.

Doctor Finder Website: http://drfinder.pythonanywhere.com/

Use Case-07: Search for Doctor

Use Case ID:	UC-07
Use Case Name:	Search for Doctor
Description:	Patient can search for a doctor based on doctor's specialty, location, and insurance accepted.

Actors:	Patient		
Pre-conditions:	Patient wants to search for a doctor based on specific search criteria either or not he has an account. Patient's current location is chosen as the default location.		
Post-conditions:	A list of doctors has been displayed based on the provided search criteria.		
Frequency of Use:	Frequently throughout the day by patients		
Flow of Events:		**Actor Action**	**System Response**
	1	Patient selects the required specialty.	
	2	Patient chooses his current location.	DoctorFinder fills in location field (state, city, zip code) automatically.
	3	Patient selects his type of insurance.	
	4	Patient submits his selections.	DoctorFinder displays a list of doctors based on the provided search criteria.
Variations:	2. The Patient enters a specific location (state, city or zip code).		

Doctor Finder

Speciality: Primary Care

City: Denver

State: Colorado

Zip: 80246

Insurance: Aetna

Search

Use Case-13: Review and Rate Doctor

Use Case ID:	UC-13
Use Case Name:	Review and Rate Doctor
Description:	Allow patient to rate and post a review for the doctor on the doctor's profile page

Actors:	Patient	
Pre-conditions:	Patient has previously created an account, is currently logged in, and has navigated to the appropriate doctor profile page.	
Post-conditions:	Rate and review have been added to the doctor's profile page.	
Frequency of Use:	Frequently by the patient.	

Flow of Events:		**Actor Action**	**System Response**
	1	Patient selects "Write a Review" function.	DoctorFinder opens a dialog box for the patient to type in his review and provides options from 1 to 5 to select the doctor's rating.
	2	Patient types in the review.	
	3	Patient gives his rating to the doctor.	
	4	Patient selects "Submit" function.	DoctorFinder saves the review and rating and posts them to the Doctor's profile page.
Variations:			

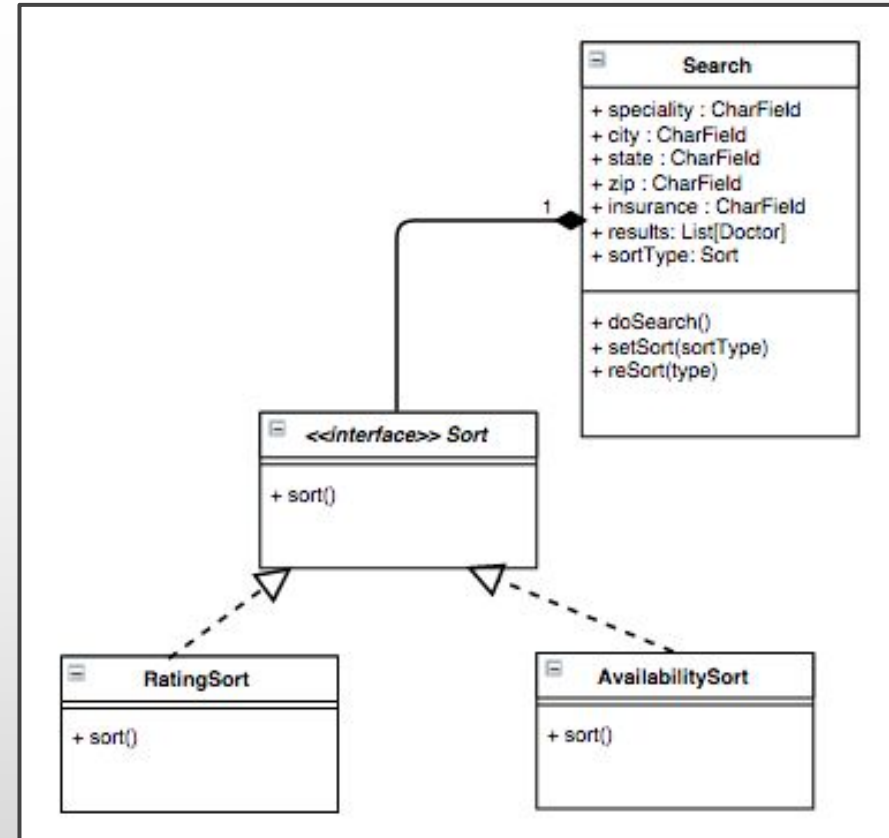**Reviews :** [Add Review!]

Rating: [4]

Good job

Comment:

[Submit]

Design Pattern - Strategy

- Sort doctor results based on sort type:
  - Sort results by doctor's **rating**
  - Sort results by doctor's **availability**

Design Pattern - Strategy

• sort.py

```python
class Sort():
    def sort(self, results):
        raise NotImplementedError("You have not implemented the sort method")

class RatingSort(Sort):
    #sort results in descending order by rating
    def sort(self, results):
        return sorted(results, key=attrgetter('rating'), reverse=True)

class AvailabilitySort(Sort):
    #sort results in ascending order by availability
    def sort(self, results):
        return sorted(results, key=attrgetter('availability'))
```

Design Pattern - Strategy

ABOUT PROJECT

Interesting things that we learned from our project:

★ Use of a Web Framework (**Django**)

★ Applying **Object Oriented** approach using **Python**

★ Learning and implementing **MVC** using **MySQL** for the database

★ Learning how to adopt new approaches by evaluating trade-offs

Project Demo

- Doctor Finder Website: http://drfinder.pythonanywhere.com/

- Full Demo Video: https://youtu.be/bp1lg9qfWsU (5:18)

- Short Demo Video: https://youtu.be/O2OMeN0jRY4 (2:33)