**ORF401: eCommerce: Spring 2021**

**Lab 3**

**Robert and Levi**

***Databases and deploying Online: Completing*** [***HandyRides, Inc.***](https://web.archive.org/web/20180615232159/http:/orfe.princeton.edu/courses/orf401/labs/lab1/aboutHandyRides.html)

***Background***

In Lab 2 you added some styling and interaction to HandyRides. However, the app is still not self sustaining in the sense that data does not get generated on the website itself. In this we will add the ability for users to register and to create rides for themselves. To do this, we will be altering the database. Finally, we will host our app on the web.

**Setup**

1) Use the same project from Lab 2. If you want to make a copy, do it now!

2) We will be deleting the data we manually insert from Lab 1 by using the command `python manage.py flush` and just type yes.

3) Copy the following settings files needed for deployment: requirements.txt and Procfile go in the same folder as “manage.py”. Replace HandyRides/settings.py and rides/urls.py with the ones provided.

4) Using pip, install the following: `pip install dj-database-url` and `pip install whitenoise`

5) I have created a couple of new files for our app. Replace the files rides/urls.py and rides/forms.py with the ones provided on Blackboard. Copy the following to the bottom of your rides/index\_view.html file (above the endblock) so that people can create new rides:

**<form action="{% url 'rides:create' %}" method="post">**

**{% csrf\_token %}**

**{{ new\_ride\_form }}**

**<input type="submit" value="Submit">**

**</form>**

Inside rides/views.py add the following import at the top:

**from .forms import RideForm, NewRideForm**

And add this line of code right before the render inside the index function:

**context["new\_ride\_form"] = NewRideForm()**

6) Additionally, add the following function to your rides/views.py:

**def create(request):**

**if request.method == "POST":**

**new\_ride = NewRideForm(request.POST)**

**new\_ride.save()**

**return redirect("/rides")**

**Assignment**

1. Update the database. You will need to edit HandyRides/rides/models.py to have the following characteristics (25 pts):

* Update the Person model to have all the characteristics of a ride (you can modify the following fields or add to them, just giving ideas, have fun):
  + Origination state
  + Last name, email?
  + Vehicle type?
  + Is it a regular CartyCity or a Premium CartyCity?
* Once you are satisfied with your changes, you will need to run `python manage.py makemigrations` and `python manage.py migrate`
* What are [migrations](https://docs.djangoproject.com/en/3.1/topics/migrations/)? Include your summary as part of your writeup.

Migrations basically change the format of data in the database every time the form and models are updated. This makes sure accesses to the database can properly retrieve existing data. For our project, we changed the format which phone numbers are stored from an integer to a string validated with a regular expression. The makemigrations command allowed us to replace existing data with a default value or allow us to rename model parameters.

1. Create a couple of rides using your new model by simply adding them manually on the webapp. Once you have everything working, we don’t actually want the form to be on the same page as the search page. Create a new url and template (like we did with the splash page) and put the registration form on a new page. You have all the pieces to make a usable webapp (adding new rides to a public list), make it usable by incorporating some CSS and JS as you did in Lab 2. Additionally, add some links to the pages to allow users to navigate to the search and to the “add ride” page. In your write up, include what your customizations for your site are (can’t be the same as Lab2). Make sure everything works on your live website after deploying it. (25 pts)

Made rows get highlighted when moused over. Submit button also lights up and mouse cursor changes styles. Added more information such as specific address for origin and destination, so we can possible work in code from last semester with lat long lookup to get shortest distances. Also added phone number. We had issues with using a simple integer field since there were overflow problems. Instead kept it a char field and use regex to validate. Changed the format of the add\_ride page to make it look a little more appealing, by increasing size of text and input fields. Search page still the same with only state and city lookup. Still working on what to do for that.

1. Following the instructions, deploy the app to Heroku, it’s free. (25 pts). The general steps are:

* Install git and heroku
* Create a Heroku account
* Once you have git installed, type the following commands in your webapp folder:
  + `git init`
  + `git add .`
  + `git commit -m “My first commit”`
* Run the following commands in your webapp folder to deploy to heroku:
  + `heroku create`
  + `git push heroku master` which will push your webapp to Heroku’s servers.
  + `heroku run python manage.py migrate` to create the database
  + `heroku ps:scale web=1` which actually assigns compute power to your app
  + Run ` heroku ps:scale web=1` which will open your live website.

Set it up so out website address works. Added heroku’s generated dns to godaddy: <http://www.ridedirs.com/>

1. Describe the major shortcomings of using a standard WWW browser as a client for this kind of application. (10pts)

It takes a long time to push changes so it is faster to make changes and verify locally. It requires the user to navigate to either a bookmark if they like the page enough, or they have to find a link to the site or just type the whole address in. A phone app would be a whole lot more convenient to access. A www browser is usually associated with desktop computers, which isn’t totally compatible with an app designed for mobility and trip planning.

1. In your ideal app, what additional models would you introduce that would be helpful to building your version of a ride sharing app. (10 pts)

Models of the events people are going to and from would be helpful. It could offer information about the surrounding area and the types of people/levels of sobriety that are expected to happen there. These factors could have an effect especially on planning trips leaving events.

1. The current webapp doesn’t implement any user authentication. Part of the reason why is because implementing safe login requires countermeasures against Cross-Site Request Forgery (CSRF). You may have noticed the csrf-token in the new ride form. What is CSRF and how would you explain it to my 5 year old niece? (5pts)

CSRF is cross site request forgery, which is where an attacker gets a victim to submit posts and requests that could have ill intent/consequences. To explain to a child, whenever we visit a webpage, it is like receiving a letter and looking at it. When we click on something on the page, we send a letter and wait for the site to return a letter. But a bad person could send a letter pretending to be you and trick the site into doing bad things or making a mistake. That’s why we need a way for the site to make sure you are the one sending and receiving letters with tricks like giving you a special envelope or letterhead everytime they send you a letter.