Peter Kralovec-Kirchherr

Meagan Combs

Web Application Write Up

19 November 2014

Our web application is going to be a website in which students at DePauw University can post requests for rides home from school, or post open seats in their car home. Many DePauw students do not know many people from where they live, or they don't know who has a car and is willing to drive them home. This website will give all students a centralized place where they can filter rides based on their nearest hometown city, and will give drivers the opportunity to maximize capacity in the car, therefore spreading the cost of the ride across multiple people. Users will be able to create, edit, and delete their profile, as well as log in and out. While logged in, a user can do things based on whether they are driving or needing a driver. If the user is a driver, they will be able to post a 'drive' that corresponds to their nearest city, which will include the departure date and estimated time, where they are leaving from, how many seats are available in their car, and an extra, optional comments section to write in if they so choose. If the user is a passenger, then they will be able to filter rides down to certain cities in the vicinity of DePauw, and look at all 'drives' that have been posted for that city. The drives will be listed in chronological order of departure time and date, making it easier to locate the ride that will best suit the passenger. Once the user has found the 'drive' that suits them, they can add themselves to that drive as a passenger. The passenger can only be added if there is an open seat in the car, and once they are added, one seat (or the only one) will be now taken, and no other user can occupy that spot in that car. The passenger user will simply click a button to sign up for that drive, which will take them to a confirmation page, letting them know that they are now part of this drive, and have reserved a seat.

What we have completed so far is all functionality regarding the User model and controller, and RESTful actions. We created a user controller with all actions corresponding to the user, which includes new, create, edit, update, destroy, and show. In our users controller, before the edit and update actions can be completed, we check to make sure that we are looking at the correct user, so that only that user can edit and update their profiles. We also created the views corresponding to edit, new, and show. The edit view corresponds to a user updating their profile. They can change their Name, Email address, or password in this view, along with the ability to delete your account. There are also links on the upper right hand side of the page that link to the home page, and a link that will give you pulldown options related to your account. The 'new' view is a sign up page, where a user enters all their information that they can change in the edit view. The show view is currently just a placeholder, and users are redirected there when a user signs in, creates a profile, or edits their profile. In the User model, we have the attributes name, email, a password, and a password confirmation. The name must be within 1 and 50 characters, the password must be between 6 and 255 characters, and match the password confirmation attribute. The email must be in the correct format of an email address. We also validate both the password and name for presence, and uniqueness for the email attribute, which will give an error if there is already a user corresponding to that email.

We use sessions to handle every time a user logs in and out. When a user logs in, we create a session, and store all that information pertaining to that user in the session. When a session is created, it finds a user by the email they enter on the login page, and calls the authenticate method on that user, to ensure that the password entered on the login page is the same one that corresponds to the user that was found, based on the email given. Sessions are destroyed when the user logs out, and it redirects the user to the sign in page. We have used the boostrap gem for styling purposes in our project.

To complete our project, we will have to create the “drive” resource (model, controller, views) which will represent posts on the main page about drives students plan on making. This model will have to include the corresponding attributes that characterize each ride. The drive model will have a belongs\_to relationship with the user, and the user will have a has\_many relationship with the drive model. We will have to create an index page where the drives are all posted, and where users can interact with each drive. The controller will need functionality to update the drive instances in the database when users create, update, or delete drives. We will also need a feature that updates the database when users add themselves to drives. We still need to add the filtering ability on the index page that filters the drives based off of city, which will be a column in the database.

One of the biggest challenges so far is figuring out the how the session and session controller fits into the application. We also struggled to figure out the cybersecurity portion of user passwords and logging in and out. It was difficult to comprehend how the password authentication was functioning, it seemed like what we call rails “magic”. Since password security is so essential, we really had to understand each step to make sure everything would work properly.