

Team Richards

UService

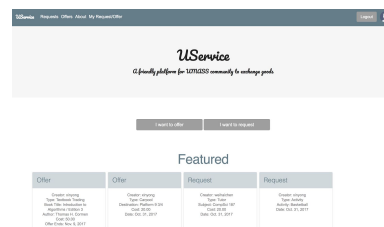
Fall 2017

<https://github.com/kali044/UService-.git>

Team Member Names	Github Usernames
Weihai Chen	weihaichen
Brandon Lam	brandonnlam
Ka Yu Li	kali044
Tuck Soon Liew	timmliew
Xinyong Qiu	mahenryqiu
Jeff Zhao	jzhao1223

- **Overview:** Our website, UService, helps students on the UMass Campus exchange services and goods become easier and more accessible. The services are tailored toward college students, including the services of tutoring, textbook trading, carpooling, and meeting up for an activity. Users can create requests and offers with the essential information and the other users can contact the creator if they are interested in the listing. Our application is a new take on sites giving sites, because sites like Craigslist don't have searches or filters, they only have categories, and since you can only find out what a listing is by reading the title there is no way to sort. By giving our listings quantifiable data, we can sort through them and make them easily accessible to the average user.
- **User Interface:**

Home - The first screen users see when they open the website, navbar allows access to most parts of the site



Request/Offer pages - Lets you see the listings of other users, requests and offers are different pages but look and function the same way, only listing different things

Title	Creator	Description	Cost	Date	Status
To Singapore	Tim Liew	Requesting a flight ticket to Singapore	\$100	May 15, 2017	Request
To London	Jeff Zhao	Requesting a flight ticket to London	\$150	May 15, 2017	Request
To New York	Brandon Lam	Requesting a flight ticket to New York	\$200	May 15, 2017	Request


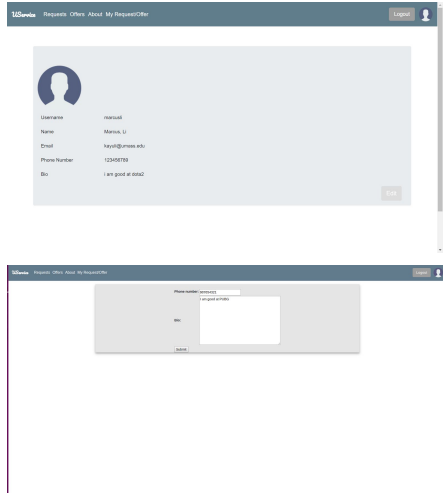
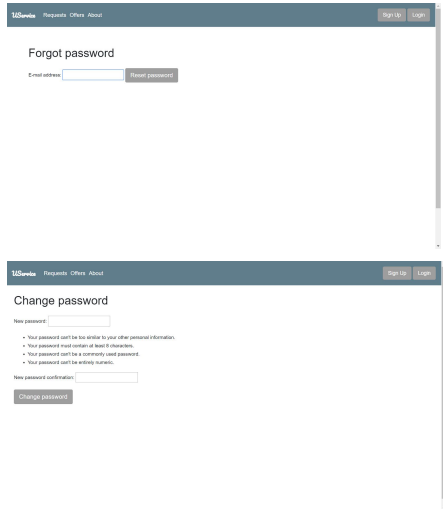
Login/Signup - modals attached to the navbar allowing you to login and signup to the website

Detail - Allows you to view the detail of a request/offer, and user can contact the creator and leave a comment

Create - Allows you to create your request and offer. Form fields are tailored to your listing

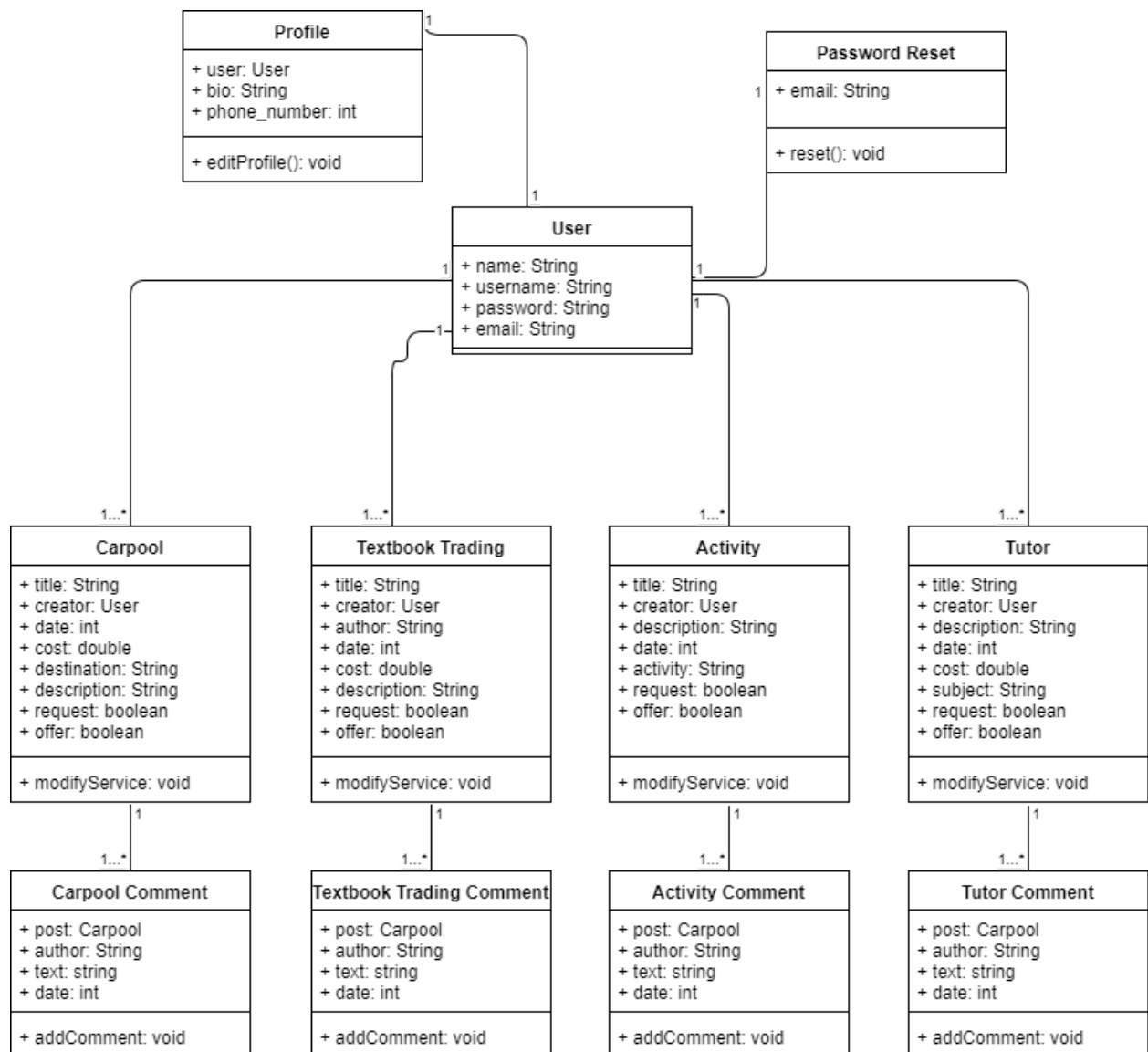
Edit - Allows you to edit your listing if you are the creator

About - A short description of the site and the names of the creators

<p>Mypublish - a list of all the listings you have created, sorted by type.</p>	
<p>Profile - A view of your profile, includes your name, email, phone number and a short biography. And User can edit their profile phone number and bio.</p>	
<p>Password reset - Users can enter their account's valid email to reset their password</p>	

- Data Diagram:** Our data model starts with a user which contains their name, username, password, and email. This has a one-to-one relationship to their profile, which contains their user and a short bio. If a user were to ever forget their password, they can reset their password. This is a one-to-one relationship to each user as every the link sent to each user is unique. After their profile is created, there is a one-to-many relationship between a profile and each of the goods or services offered, which are Carpool, Activities, Textbook Trading, and Tutor. Each of these things contain relevant information tailored to their type, such as title, cost, description and whether it is a request or an offer. Once a service is created, there is a one to many relationship

between the service and comments on the service. The comments will contain information about the post, the user who commented, the comment, and the date the comment was created.



- **URL Routes/Mappings:**

url(r'^\$', url(r'^home'	Links to the home page
url(r'^about'	Links to the about page
url(r'^addoffer/')	Links to the creation page for either an offer

<code>url(r'^addrequest/')</code>	or a request. Available when logged in
<code>url(r'^carpoolDetail/(?P<pk>\d+)\$')</code> <code>url(r'^tutorDetail/(?P<pk>\d+)\$')</code> <code>url(r'^textbookDetail/(?P<pk>\d+)\$')</code> <code>url(r'^activityDetail/(?P<pk>\d+)\$')</code>	Links to the page that shows the detail for the service listing. Each url route links to that specific type of service
<code>url(r'^profile/(?P<pk>\d+)\$',</code> <code>views.profileDetailView.as_view(),</code> <code>name='profile'),</code> <code>url(r'^profile/(?P<pk>\d+)/edit/\$',</code> <code>views.ProfileUpdate.as_view(),</code> <code>name='profile_Edit'),</code>	Links to the user's profile, available when logged in, and edit profile page
<code>url(r'^searchoffer/\$')</code> <code>url(r'^searchrequest/\$')</code>	Links to the pages that shows all the offers and requests, respectively
<code>url(r'^mypublish/\$')</code>	Links to the page that shows you all your listings. Available when logged in, and with the permissions as the user to see the info
<code>url(r'^activity/create_request/\$')</code> <code>url(r'^activity/create_offer/\$')</code> <code>url(r'^carpool/create_request/\$')</code> <code>url(r'^carpool/create_offer/\$')</code> <code>url(r'^tutor/create_request/\$')</code> <code>url(r'^tutor/create_offer/\$')</code> <code>url(r'^textbook_trading/create_request/\$')</code> <code>url(r'^textbook_trading/create_offer/\$')</code>	Links to the page that allows you to create requests and offers. Each url route is unique because of the custom forms, so they will be linked whenever the user wants a certain type of listing. Available when logged in.
<code>url(r'^activity/(?P<pk>\d+)/update/\$')</code> <code>url(r'^carpool/(?P<pk>\d+)/update/\$')</code> <code>url(r'^tutor/(?P<pk>\d+)/update/\$')</code> <code>url(r'^textbook_trading/(?P<pk>\d+)/update/\$')</code>	Links to the edit page, so you can edit your offers and requests. Available when logged in and when you have permission from being the creator. Routes also exist for activities, tutor, and textbook trading.
<code>url(r'^password_reset/\$')</code> <code>url(r'^password_reset/done/\$')</code> <code>url(r'^reset/(?P<uidb64>[0-9A-Za-z_-]+)/(?P<token>[0-9A-Za-z]{1,13}-[0-9A-Za-z]{1,20})/\$')</code> <code>url(r'^reset/done/\$')</code>	Links to password reset form, so you can enter your email to reset your password. After that you'll be redirected to password reset done, where it tells you to check your inbox. The link you get in the email will redirect you to password reset confirm for you to change your password. Finally, password reset complete will show up, which means you've successfully changed your password.

- **Authentication/Authorization:** Users are authenticated by logging with their credentials. The login page is attached to the navbar in the form of a modal, and once logged in the user has access to other functions. Once logged in they can create listings, edit their listings, view their profile, edit their profile and view their listings page, which shows all the listings they have created. Every other page is available to everyone.
- **Team Choice:** For our team choice we decided to add a password reset with email confirmation. The lost password button attached in the login page form under the login button will redirect the user to a page to reset his/her password via `url(r'^password_reset/$')`. User has to input the email that he/she set upon registration (user's email in database), otherwise it won't work. If the email is valid, a confirmation page will show up via `url(r'^password_reset/done/$')` and the user will receive a link for password reset. After that, user will be able to change his/her password via `url(r'^reset/(?P<uidb64>[0-9A-Za-z_-]+)/(?P<token>[0-9A-Za-z]{1,13}-[0-9A-Za-z]{1,20})/$')` and a password change success page will show up via `url(r'^reset/done/$')`. The user will be able to login using the new password.
- **Conclusion:** Overall we really enjoyed working on this project. None of us had done web programming before, so it was a new experience for all of us. We learned the data model is always changing as we had many minor changes throughout project 3 and project 4. The data diagram looks significantly different compared to the initial one. One of the most difficult parts for us was dealing with the pushing, the pulling, and merge conflicts from github, since we were all working at the same time. It is also difficult to update pull from GitHub after someone updated the models or the databases, and everytime we would need to clone the repo again. We learned that a lot of things that seemed simple to implement took a lot more work than thought. One thing that we would have liked to know before starting project 2 is that Django includes an user model of their own, and we made the mistake of making a user model initially and didn't find out until we worked on authentication.