👨‍💻 Typito Assignment I

- **Pre coding questions** :
  - For the API, which language should I use ? Using python-django would use less development time or I can use Node.js-Express also.
  - The image post-processing after the upload would be easier using python as there are libraries available for image processing.
  - For the UI, using vanilla JavaScript with jQuery sprinkled in there would be fine as this is not that big project. In fact, using React or Angular would take more time to build the prototype.
  - Have to think about Deployent options. Using AWS or Digital Ocean would be overkill. Have to go for Heroku.
  - Which plugin to use for infinite scroll and how to integrate with api pagination?
  - which javascript plugin to use for drag and drop feature
  - How to show progress of upload. Use a custom loader or user some fancy library?
  - Which solution to use for full screen image view and carousol with smooth transition?
  - Should I include user authentication in the app and the api. Won't it go out of scope ?
- **Development** :
  - For UI, I am going for plain bootstrap. I am not too focused on fancy designing now. Just want to get the layout down and make sure it is responsive and fast. So keep dependencies to minimum.
  - Since the app should be extensible, I am going for the django template extending system : So 5 template files :
    - base.html (skeleton template)
    - login.html (for authentication)
    - navbar.html (for separting navbar and make in inheratible as and when necessary, since I don't want navbar in login page.
    - gallery.html (main UI page which will extend base and navbar
    - upload.html (upload page with drag and drop feature, which will also extend base.html navbar.html, containing hyperlinks to ages and logout button)
    - First, I have to work on the back end and create the models

- Since the image will have 3 different rendation, the model will have three different field. Here image field will be better over file field as it will be easier to later on implement Pillow for image processing.
- The view will have to send paginated images and I have to find a way to convert the pagination into infinite scroll in the template. Also, the pagination has to be combined with grouping for showing image based on upload date.
- For the infinite scroll, went for waypoints.js. In django template, paginated the queryset along with django regrouping tag to combine infinite scroll along with grouping with upload date.
- For full screen lightbox, went for lightgallery.js.
- Challenge with lightbox and infinite scroll was. The lightbox only showed initial paginated results. So, I had to modify the lightgallery javascript to destroy the lightgallery instane and reinstantiate, everytime new batch of images were loaded using infinite scrolling.
- Now, for showing caption and upload date on fullscreen, I had to use lightgalleries caption settings. But that left me with the caption and upload date showing with the image in the gallery all the time, breaking the view. So, quick solution was to dispay:none the caption and upload date and then show them when the images were launged in the slideshow.
- Since the app has been targated towards user with slow internet connection, I focused on getting as much performance out of it as I can. So, my major concern was image processing. I was basically gonna upload the original image then use Pillow to resize image keeping the aspect ratio. And then in the ui, show the 240 rendation inthe thumbnail and when the image is launched, show the 1200 rendation. But I still wasn't satisfied with the process. The issue was, I am deploying to heroku and heroku doesn't support django media files. So, I had to store the static files and media (uploaded photos) to AWS S3 bucket. And then do collectstatic to transfer all the assets over to S3 and server from S3 to heroku. So, if I upload an image and THEN process it, then it is gonna end up eating a lot of bandwith as I have to upload to S3 and then load the image back into the memory and do processing and then reupload. So, the solution was to use django resizeimage field and with with some custom methods to resize the image in-memory, after drag and drop, before uploadng to S3 and then save all the rendation with original filename as caption. This drastically increased performance. I noticed the image were faster than instagram

upload. Even when I was doing bulk upload it was blazing fast. So was quite happy with the result.
    - Now the challenge is to get the multiple files from drag and drop and upload them in bulk. For that I had to access individual files. So, no django form class. I had to manually loop through each form file and do validation and preprocessing.
    - Another challenge is to show error message to the dropzone area. For that I have to server json response with modified header with appropreate HTTP status code. And then fire the on success and on failure event in dropzone.js javascript and show the error mesage.
    - For deloyment, I am going with the heroku and AWS S3.
    - After a few hours of tinkering, got the heroku and aws setup.
    - Now, I have to implement authentication to lock the app out.
    - Initially, I was thiking about reusing django admin's login. It was possible with a little bit of inheritance and custom routing. But wasn't happy with the result. I had to run my own validation and have a custom view.
    - So, I went with custom template with separet authenication system of my own.
- Tradeoffs made:
    - Heroku uses sqlite for database storage, which is fast, but for production, I probably have to migrate to postgresql
    - The fullscreen slideshow could be more dynamic.