Product specifications

This application is meant to facilitate inventory observations being performed in warehouses by auditors by using the help of a mobile responsive web application. In the future, I will also be expanding this application to include features that retailers and inventory managers in warehouses can use to perform their inventory counts.

Problem statement:

Performing inventory observations is a hassle for juniors and sometimes things are just not accounted for correctly or documented correctly. This app will help enable users to fully complete their inventory observations on their phone including all of the documentation from start to finish. Basically, the counter can actually focus on performing the count vs focusing on performing the count AND documenting the outcomes.

Technical competencies to accomplish out of this project:

1. Continue to work with form custom validation and making sure I understand it fully, especially that super(mymodelform, self).\_\_init\_\_() thingy.
2. Learn how to work with files being uploaded by users
3. Learn how we can create a method which will allow us to put a default in one field depending on the value of another field
4. Learn how to provide users with the functionality to download the files which contains the final outcome of the inventory observation
5. Learn how to provide users with the functionality to email themselves the files which contains the final outcome of the inventory observation
6. Integrate all file upload and download related features with AWS S3 Buckets
7. Integrate security considerations within the application when files are being uploaded and being downloaded
8. Learn how to effectively work with the Users group and group permissions mechanism in Django to be able to give permissions of the app to different people based on their employee level
9. Take into account when developing this app considerations of how to build out a Saas application in Django
10. Learn how to use Django to create views for different clients using this Saas service
11. Learn how to work with users phone and give them the ability to capture images and then save the image in the app file system
12. Incorporate various sampling methodologies into the application such that a user can either select the samples themselves or have the application determine the samples to select
13. Work on the UX of the app so that it is at least worth looking into by someone
    1. <https://itnext.io/frontend-for-backend-developers-6433adb5a10d>
14. Integrate various sampling methodologies into the mix so that the app actually selects the samples for the users based on the user selected sampling methodology
15. Learn how to integrate speech recognition into the application
16. Learn how to provide functionality to users to be able to work offline OR if that is too complicated for now, figure out a way for me to be able to cache the results of the inventory list to be counted in the cache and then store the users inputted responses somewhere in the browser.
    1. <https://www.youtube.com/watch?v=ksXwaWHCW6k>
    2. <https://hashedin.com/blog/building-progressive-web-applications/>
    3. <https://www.google.com/search?q=django+service+workers&rlz=1C1CHBF_enCA910CA910&oq=django+service+workers&aqs=chrome..69i57j0i457j69i61.4810j0j7&sourceid=chrome&ie=UTF-8>
    4. <https://www.youtube.com/watch?v=cmGr0RszHc8>
    5. <https://www.youtube.com/watch?v=70L8saIq3uo>

<https://www.google.com/search?q=django+service+workers&rlz=1C1CHBF_enCA910CA910&oq=django+service+workers&aqs=chrome..69i57j0i457j69i61.4810j0j7&sourceid=chrome&ie=UTF-8>

1. Configure the various user group permissions correctly such that I can use these User attributes to show custom views to various users based on their user group and permission without the need to add additional fields such as “is\_teacher” or “is\_parent”.

General structure of the application:

* Models:
  1. Enterprise model – This is going to represent the organization using this Saas application. So, like RSM, EY, KPMG etc.
     1. ID – autogenerated
     2. Name – CharField
     3. Plan type – CharField with options such as (Basic, Intermediate and Pro)
     4. Employees – NumberField – This number needs to be an integer
     5. ContactPersonName – CharField
     6. ContactPersonEmail – EmailField
     7. ContactPersonPhone – Charfield – Use regex validator on this
     8. Address – Line 1 – Charfield
     9. Address – Line 2 – Charfield
     10. City – Charfield
     11. State/Province – Charfield
     12. Postal Code – Charfield
  2. Client Model – This is going to represent the organization that my clients are servicing, so like the entities whose
     1. ID – autogenerated
     2. ClientID – Charfield – This is going to represent the ID that
     3. Name – CharField
     4. Plan type – CharField with options such as (Basic, Intermediate and Pro)
     5. Employees – NumberField – This number needs to be an integer
     6. ContactPersonName – CharField
     7. ContactPersonEmail – EmailField
     8. ContactPersonPhone – Charfield – Use regex validator on this
     9. Address – Line 1 – Charfield
     10. Address – Line 2 – Charfield
     11. City – Charfield
     12. State/Province – Charfield
     13. Postal Code – Charfield
     14. RelationshipLeadNameOne – Charfield
     15. RelationshipLeadEmailOne – Charfield – use regex validator on this
     16. RelationshipLeadPhoneOne – Charfield
     17. RelationshipLeadNameTwo – Charfield
     18. RelationshipLeadEmailTwo – Charfield – use regex validator on this
     19. RelationshipLeadPhoneTwo – Charfield
     20. PeriodEnd – DateField – use the date widget
     21. AdditionalFIeld – depending on the requirements of my Saas clients (Enterprise)
  3. Engagement Model – This is going to represent a specific engagement for the client selected above. An example of this could include things such as FY 2020 year end audit engagement.
     1. ID – autogenerated
     2. Name – Charfield – Use as a name to identify this
     3. Client – Either use a Foreign Key or a OnetoOneLink – research which one is more appropriate for this use case
     4. EngagementType – Charfield – with choice of either audit, review, NTR, special engagement and others.
     5. EngagementPeriodEnd – DateField – this is going to represent the period end date for this specific engagement
     6. Engagement partner – Foreign Key or OnetoOneLink pointing to the User object or the Partner object. The main consideration here would be to make sure that when I have a dropdown, that I only get access to users who are Partners and not managers, staff etc.
     7. Engagement manager – ForeignKey or OnetoOneLink pointing to the User object.
  4. Stock Count Model – This is going to represent an instance of a stock count. If there are multiple stock counts being performed at various locations, then each of these counts is going to be encapsulated by a Stock count model class instance.
     1. ID – autogenerated
     2. Name – Charfield – random name provided to this count
     3. CountManager –
     4. ClientContact – User object representing the client contact person
     5. CounterOne – this one can be blank or null
     6. CounterTwo – these ones can be null and blank
     7. CounterThree
     8. CounterFour
     9. CounterFive
  5. User model
     1. Extend the AbstractUser class
     2. Gender
     3. User level
     4. Enterprise user or client user
  6. InventoryList model
     1. ID
     2. File – FileField
  7. SKU model
     1. ID – autogenerated
     2. SKU – Charfield
     3. ProductName
     4. ProductDescription – this one can be blank and null
     5. ClientSKUQuantityOnHand – Floatfield – can be whole number as well
     6. ClientSKUValue – Floatfield – can be whole number as well
     7. EnterpriseQuantityOnHand - Floatfield – can be whole number as well
     8. QuantityOnHandDifference - Floatfield – can be whole number as well – check if there is a way in Django for this to be calculated upon the submission of the form. I can also consider using JavaScript for this automatic calculation. This could be like a hidden field and the value of which is going to be calculated on the onsubmit event handler of the form.
     9. CounterComments – TextField – The text field should append to the submitted text comment the name of the individual making the comment
     10. ImageOne – ImageField
     11. ImageTwo – ImageField
     12. ImageThree – ImageField
     13. ImageFour – ImageField
     14. ImageFive – ImageField
     15. TimeSKUCounted – DateTimeField – Default to timezone.now() – should not be passed into the modelform
* Workflow

1. SuperUser:
   1. Can manually create and Edit any of the above model class instances
   2. Can also upload an macro-enabled excel document which will create all of the instances.
      1. In order to do this, I need some validation to be performed within the excel document before the client can actually upload the file since I want the user to be able to only upload files that are legitimate, don’t have any error and are in the format that I want them to be.
2. Admin:
   1. Can also upload an macro-enabled excel document which will create all of the instances.
      1. In order to do this, I need some validation to be performed within the excel document before the client can actually upload the file since I want the user to be able to only upload files that are legitimate, don’t have any error and are in the format that I want them to be.
3. Partner:
   1. In the current iteration, we only want the manager to be sent an email with the outcomes of the stock count.
   2. In the future iteration, I want the partner to be able to review the inventory observation in real time and provide comments if required.
4. Manager:
   1. In the current iteration, we only want the manager to be sent an email with the outcomes of the stock count.
   2. In the future iteration, I want the manager to be able to review the inventory observation in real time and provide comments if required.
5. Staff:
   1. Can upload the excel file which contains the inventory listing
   2. Can also upload the excel file which contains the samples only
   3. Can edit and save the SKU model class instances
   4. Can submit the Stock count model class instance
6. Client:
   1. Can only upload the excel file which contains the inventory listing.