PROJECT REPORT
CONTACT MANAGER

Made by-Ram Khandelwal,Mohit Ahirwar

Contents

Starting with Scrum Practices

Scrum Master:-This role is ensured by Ram for managing and facilitating the scrum, selecting the sprint backlogs and ensuring that sprint is completed on time.He also ensures that both the team members have clear understanding of the scum activities and all impediments are removed.

Product Owner:-This role is managed by Mohit.He selects the product backlog items along with other items that are to be run during the sprint.He makes the ordered list of PBIs in the product backlog and ensures clear understanding of PBIs.He also checks that the team is leading to the optimization of the product and incrementing each time the sprint runs.

Scrum Team:-As part of scrum team both Mohit and Ram are their and will run the sprint by themselves.Both will ensure the delivering of the product is iterative and incremented each time the sprint runs.This team is self organizing and cross functional in itself.

Product Backlog and Product Backlog Items:-

S.NO	Product Backlog Items
1	Starting to work with django
2	Templates and static files
3	Django Models and admin site
4	Homepage passing data from views to template
5	Accessing individual contact object

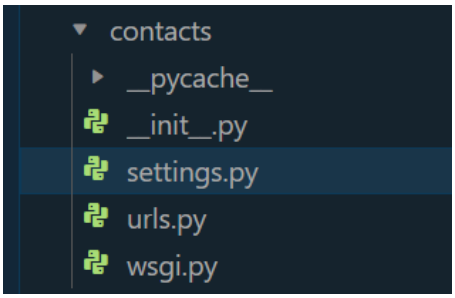
1st Sprint(15-03-2021 to 17-03-2021)

Sprint Backlog Item
So the first sprint backlog item is Starting to work with django.

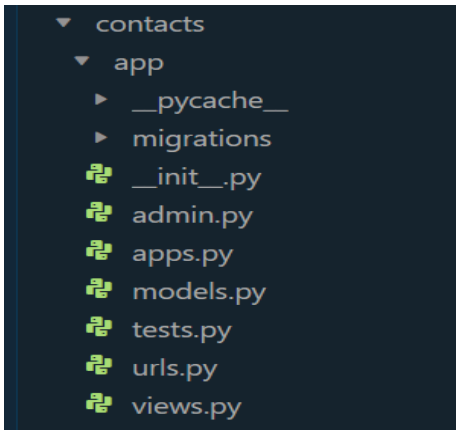
Sprint Planning
The sprint is planned to include installation of python and django.It is assumed that vs code already installed in the developer's system.Also various modules will be installed that are necessary to initiate the django Contact project.

Sprint

- Install django using pip command
- Go to the terminal and write “django-admin start project contacts.” and then run it.It will create the contact project folder and will initiate the following files:



- Install migrations and then enter this command in the terminal “python manage.py startapp app”.In settings.py add ‘app’ in the Installed apps.It will add the following files in the migrations folder



- In urls.py write the following code:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('app.urls')),
```

Sprint Review

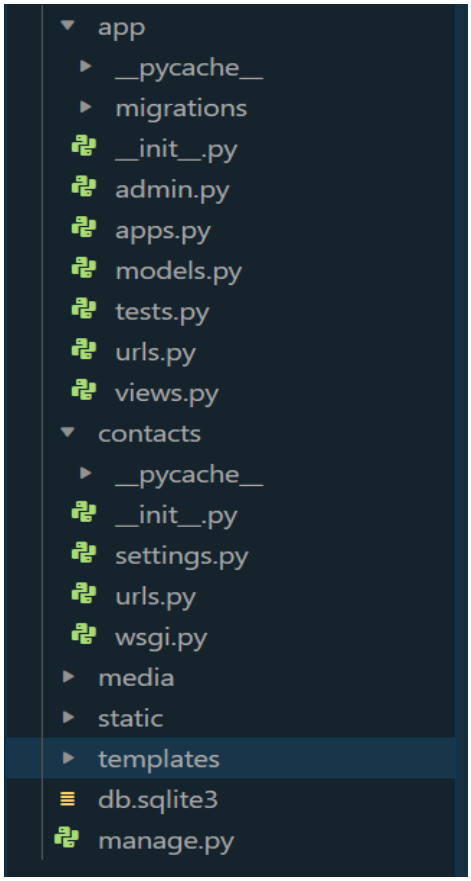
The sprint went well.We have installed all the related folders that are required to start our django project.We have imported various modules that are imported by running various inbuilt commands that are shown in the sprint.

Sprint Retrospective

Overall the sprint went well without any errors while we were installing the important modules that are required to start with our django project.We have also worked on the urls models and wrote a short code to install the app.These modules will be further extended to give us the required functionalities in the future sprints.

Sprint Increment

In this increment we got various modules to start our django project.These are collectively shown below:



2nd Sprint(18-03-2021 to 21-03-2021)

Sprint Backlog Item

The second sprint backlog item is templates and static files.

Sprint planning

In this sprint we will focus on the designing the layout of the home page.We will first add template and static files using the inbuilt commands and then update our html files which will include index,base and search.By the end of the sprint we would be able to show the layout of the homepage along with some contacts also.

Sprint

- To work with templates first go to setting.py and scroll down to Templates and inside it update the DIRS as follows:

```
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [os.path.join(BASE_DIR, 'templates')],
    'APP_DIRS': True,
```

- This will allow us to work with template files.
- For static files, scroll down to last and write the below code

```
STATIC_URL = '/static/'
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)
```

- Now crete the base.html file in the contacts folder and copy the contents of index.html file inside it.Remove the main section.Remove the contents from index. html and write the code for main section inside it.Similar code for search.html

Sprint Review

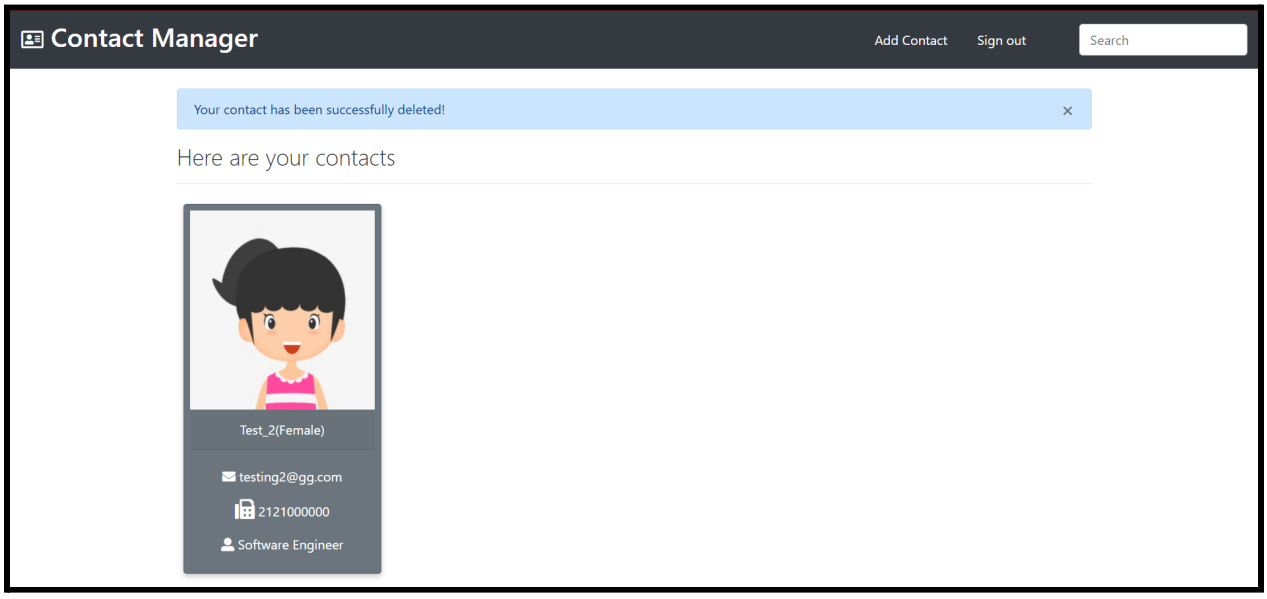
The sprint went well.We had added the required template and static files that were necessary to design the homepage view.The html files are also created which will further help in building the design part of the website.

Sprint Retrospective

Overall the sprint went well without facing much errors.We got some error when we were displaying the layout of webpage but it was resolved afterwards.

Sprint Increment

In this sprint we have prepared the layout for the homepage which include headers, main section and footers.We had successfully added templates and static files which are required for the homepage view.Also we have created html files which include index.html,search.html,base.html.You could see the following increment:



3rd Sprint(22-03-2021 to 25-03-2021)

Sprint backlog item

The third sprint backlog item is Django Models and admin site.

Sprint Planning

In this sprint we will create the database that will store all the information of the contact.Also we will access the admin site by creating a superuser that will have access to all the contacts stored on the webpage and will be able to use all the functionalities from the admin page itself.Also we will customize the admin page and update it in the admin.py file.

Sprint

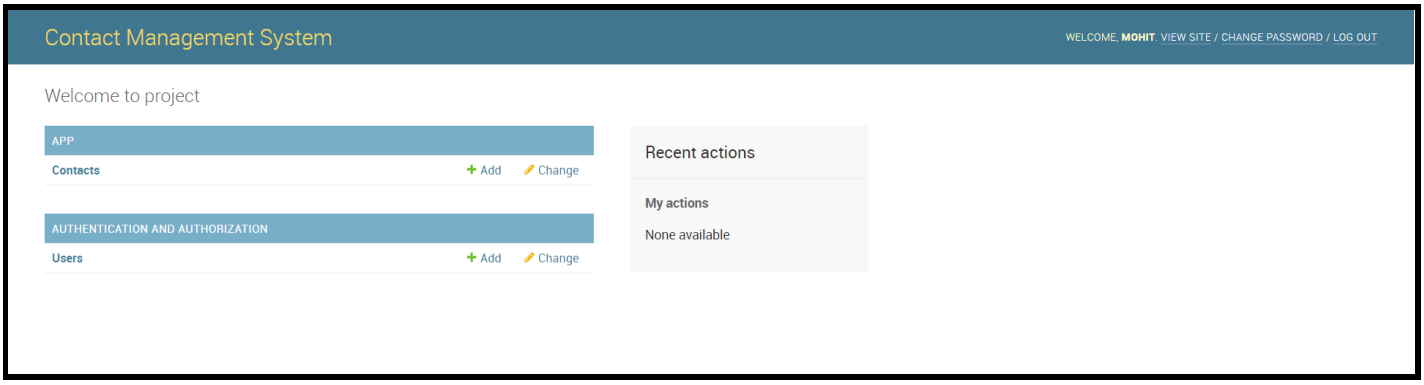
- Create the database for the contact which will include contact's name,email,phone,gender,image,info and data_added.

```
class Contact(models.Model):
    manager = models.ForeignKey(User, on_delete=models.CASCADE,default=None,null = True , blank = True)
    name = models.CharField(max_length=20,null = True , blank = True)
    email = models.EmailField(max_length=100,null = True , blank = True)
    phone = models.IntegerField(null = True , blank = True)
    info = models.CharField(max_length=30,null = True , blank = True)
    gender = models.CharField(max_length=50, choices=(
        ('male', 'Male'),
        ('female', 'Female'),
    ))
    image = models.ImageField(upload_to='images/', blank=True,null = True)
    date_added = models.DateTimeField(default=datetime.now)
```

- To use the image field we will install the 'Pillow' python package so that it could work properly.We will install it using the pip command.
- Now create the media folder in which we will inform django that we will store all the user uploaded files and images in this directory.
- Create the path to media directory in the settings.py file.

```
# Setting media files
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

- Now we will create a superuser that will have access to the admin page of the webpage.First create username and password after setting the virtual environment in the terminal.Now go to the admin page and login with this username and password.The admin page will open as:



- For customizing the admin site,in the admin.py file write the following code.This will help admin to use the functionalities to change all the contact info.

```
from django.contrib import admin
from django.contrib.auth.models import Group
from .models import Contact
from import_export.admin import ImportExportModelAdmin

class ContactAdmin(ImportExportModelAdmin):
    list_display = ('id', 'name', 'gender', 'email', 'info', 'phone')
    list_display_links = ('id','name')
    list_editable = ('info',)
    list_per_page = 10
    search_fields = ('name', 'gender', 'email', 'info', 'phone')
    list_filter = ('gender', 'date_added')


admin.site.register(Contact, ContactAdmin)
admin.site.unregister(Group)
```

Sprint Review

The sprint went well.We have added the required database table for the contact and also successfully accessed the admin site and customized it.

Sprint Retrospective

Overall the sprint went well without facing any issues in creating the database for the contact and accessing the admin site.

Sprint Increment

The increment include the customization and working of the admin site and creating the database for the contacts.This database would further help us in adding the functionalities for the webpage.

4th Sprint(26-03-2021 to 29-03-2021)

Sprint backlog item

The 4th sprint backlog item is Homepage passing data from views to template.

Sprint Planning

In this sprint we will work on passing contact objects to templates and create a temporary webpage consisting of two contacts with their objects including their email address,phone number,their profile image,and other info.

Sprint

- To pass contacts objects to templates, in the views.py file write the following code:

```
def home(request):
    context = {
        'contacts': Contact.objects.all()
    }
    return render(request, 'index.html', context)
```

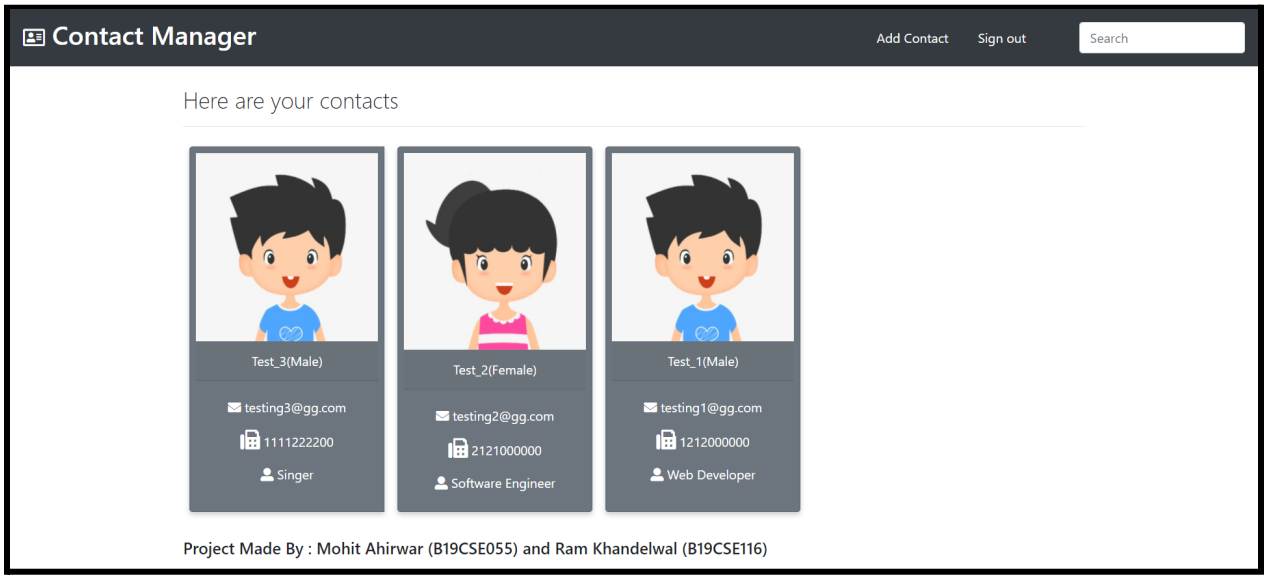
- In the index.html write the contact objects for 3 cases:no contact, when contact is girl where if image is not provided then default image will be used and similarly when contact is a boy.

- For importing images for the contacts, edit in the urls.py file and write the url of the images we want.Write the following code:

```python
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('app.urls')),
    path('',include('django.contrib.auth.urls')),
]+static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

# Customizing admin texts
admin.site.site_header = 'Contacts'
admin.site.index_title = 'Welcome to project'
admin.site.site_title = 'Control Panel'
```

- You will see the contacts getting updated as:



- Modify the date_added function inside the models.py file as

```python
TIME_ZONE = 'UTC'
```

```python
date_added = models.DateTimeField(default=datetime.now)
```

For this to work we have to also import the datetime function from django.utils.timezone

Sprint Review
The sprint went well.We have successfully created our temporary webpage with some of the functionality with all tests and automations performed.

Sprint Retrospective
There was one error which was coming for the date_added function which we got while running the webpage.This was modified by adding the datetime function as shown in the sprint.

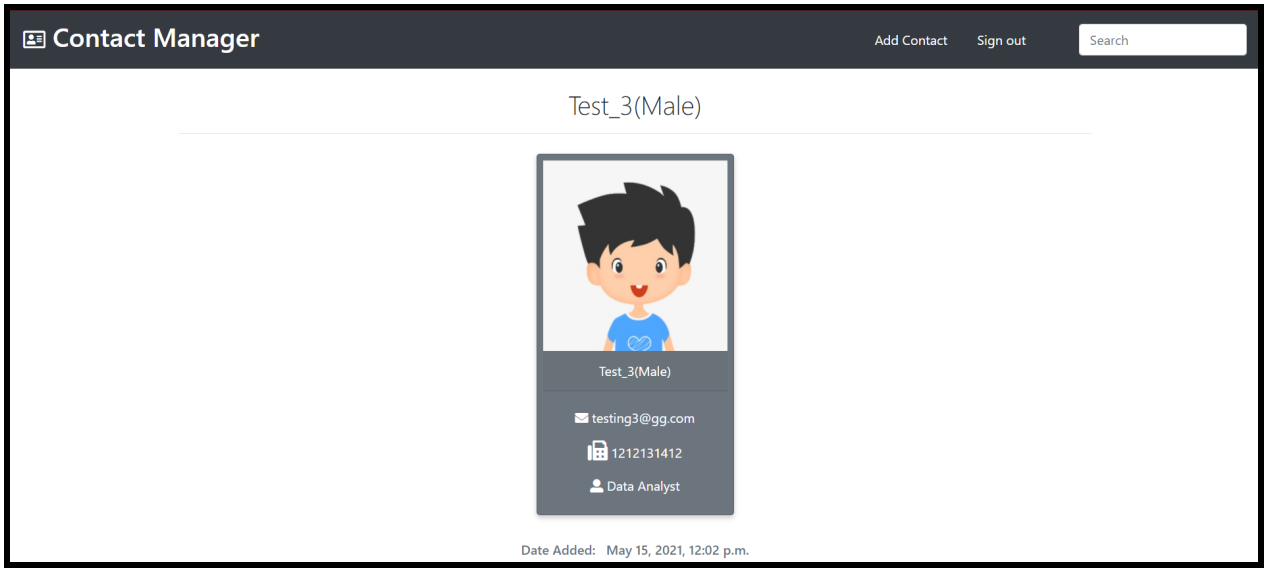
Sprint Increment
In this sprint one could say that we have prepared a prototype kind of thing.We have created contact manager that manages some contacts with its objects which include email address,phone number,profile image and other info.This had given us clear idea about how we have to move in further sprints and what will our final webapp look like.

5th Sprint(30-03-2021 to 2-04-2021)

Sprint Backlog item
The 5th sprint backlog item is Accessing Individual contact object.

Sprint Planning
In this sprint we will add individual page for each contact.When we will click any object on the home page it will open up the new page showing that particular contact.

Sprint
- Edit the title and card heading in the detail.html file.Move the contents from detail.html to _card.html
- Do small modifications on the size of the contact displayed on the new page inside the _card.html file.
- Finally the individual contact page will look like as shown below:



Sprint Review

The sprint went well.The individual contact page design follows the SRS and design document.All the functionalities are added as discussed with the stakeholders.

Sprint Retrospective

While performing the sprint we had not got any error and the individual contact page ran successfully.It passed all the test automations perfectly.

Sprint Increment

In this sprint we have created a new individual contact page that will be displayed when user will click on a particular contact on the home page. And finally we have shown how it will look like in the sprint.

Starting with Use Case Practise: Understanding The Requirements

1.Find Actors and Use cases

Use case Model

Here we will list the use cases, what they will do, and how users will benefit from them.

- **Search functionality-**This is very helpful for the users that use contact manager and have a large list of contacts.The search or the navigation bar on the top helps users to find out the contact quickly.
- **Creating a contact-**As contact manager stores a large amount of contacts, one needs to store his/her preferable list of contacts accordingly.For this user has to create the contact and add all its info by itself.
- **Editing a contact-**Suppose if user wants to edit some info in the contact maybe a person's phone,email,profession or profile he/she must be able to edit it.
- **Deleting a contact-**Sometimes users find that if a particular contact is not useful for them,they want to delete it.So this feature lets them do this task.
- **Login/Logout functionality-**User's privacy is very important and so user's contacts could only be accessible by user only through his username and password that will be set by the user itself.

Use case narrative

Our main aim is to implement these use cases for the ease of users.All these use cases will be implemented using various classes and objects and all will be run independently.

Use case

Here we will write about the scope of the use cases.Contact manager gives them the ease of accessing their contacts from desktop,mobile or any other platform.

- Primary actors named-These use cases are used by user that want their contacts to be managed properly and safely.
- Goal and value clearly described-Our goal is to make the separate classes for each use case and implement them.
- Stakeholders agree upon the goal-Yes, the stakeholders have also agreed upon the goal.
- In or out of scope-All the use cases are in the scope.

- Use case narrative-It is briefly described.

2.Slice the use cases

Use case Model

Here we will define the scope and structure of each use case:

- **Search functionality-**This we will achieve by creating a navigation bar at the top of the web site on the home page.User will go their and type the name of the contact they want to search and press the enter.This will immediately take them to the particular contact they want to access.
- **Creating a contact-**For this we have a create contact option on the home page itself.User will click on it and then a page will appear where user has to fill all the details of the contact.This will add a new contact in the list.
- **Editing a contact-**User could click on a contact that he/she wants to edit.There will be a option to update the contact, user will click on it.This will open all the details of that particular contact and user can then edit the details accordingly.
- **Deleting a contact-**Near to the update contact,there will also be the option of delete contact.User can click on it and it will immediately remove that contact from the contact list.
- **Login/Logout functionality-**Before entering the home page user has to enter his/her correct login details that is username and password and then only the web page will open.

Use case

- Basic flow determined-The basic flow that how user will use the use case is each explained the use case model itself.
- Nature of other flows determined-We do not have other flows for the use cases in this project.
- Start and end are clear-Yes it is also clear.
- Most common stories identified-In this project all the use cases are equally important and independent.There is no common story between them.

Use case slice

Here we have to identify use case slices so that we could divide the larger use cases into smaller parts.So for each use case:

- **Search functionality**
    - One is the simple use that user could search the contact by name of the particular person.
    - Another slice includes the complex search query.It allows users to search using phone,email and other detail of the contact also.
- **Creating a contact**
    - One could be to add contact details which include phone,email,and type of contact.
    - Another is adding image of the person whose contact user is adding.
- **Editing a contact**
    - It has only one slice as it is a small use case.
- **Deleting a contact**
    - It has only one slice as it is a small use case.
- **Login/Logout functionality**
    - First the user will sign up and create its account.Here user will also get assistance for creating a strong password.
    - And then user could sign in by entering username and password on the login page.

3.Prepare the use case slice

Use case narrative(Essential outline)

Now all the use cases are all set.The story structure is also well understood.Here we have not created any alternative flows.We have only basic flows for each use case.Therefore the software also has no exceptions or alternate usages for the use cases.

Use case slice-Test case

Here we will define the inputs and outputs of the use case clearly.We have:

- **Search functionality**
    - Type a name of the contact present in the contact list.Output will display the particular contact.
    - Type a name of the contact that is not present in the contact list.Output will display a message that contact not present in the list.
    - Type the phone number/email/info of the contact present.Output will display the particular contact.
- **Creating a contact**
    - Add a contact by clicking on the add contact button and enter the contact details.Output will display the contact in the contact list.
- **Editing a contact**
    - Update a contact by clicking on a particular contact and then click on update button,then edit the contact details.Output will display the updated contact in the contact list.
- **Deleting a contact**
    - Delete a contact by clicking on a particular contact and then click on delete button.Output will display contact list with that deleted contact removed.
- **Login/Logout functionality**
    - Create a username and password.Try to set a weak password.Output is a message which will display that password set is too weak.
    - Now set a strong password.Output will display the web page where user could manage its contacts.

Essentialized Microservice practise

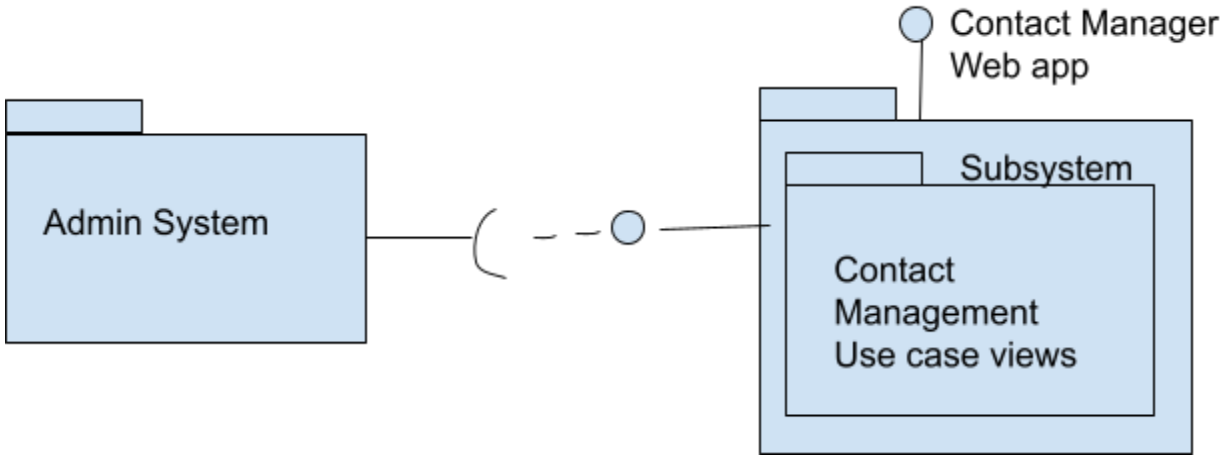
We have defined only one microservice for this software project.We will define it,make it evolvable and then evolve it.Actually all our use cases are defined inside one module in separate classes.Microservice activity space include shaping the system amd implement the system of the solution part.

Identify Microservice

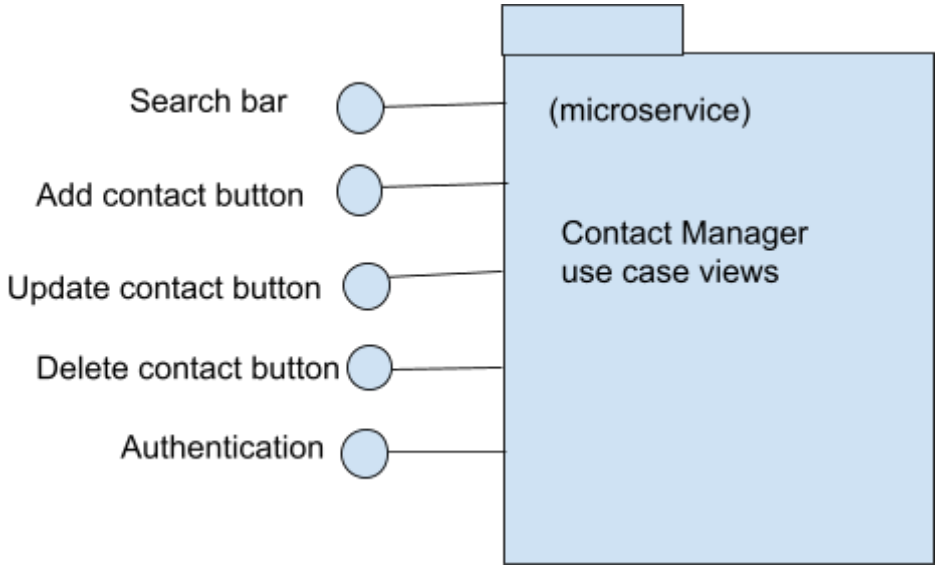
Microservice alpha

As discussed above we have one microservice containing all the use cases which are basically views which include search,Create contact view,Update contact view,Delete contact view and authentication view.

Design Model



Microservice design



Make microservice evolvable

Microservice alpha

Here we make the microservices rapidly deployable.So as we have only one microservice we will write the code for different use cases in separate classes and functions.

Build and deployment scripts

- **Search functionality**

```
@login_required
def search(request):
    if request.GET:
        search_term = request.GET['search_term']
        search_results = Contact.objects.filter(
            Q(name__icontains=search_term) |
            Q(email__icontains=search_term) |
            Q(info__icontains=search_term) |
            Q(phone__iexact=search_term)
        )
        context = {
            'search_term': search_term,
            'contacts': search_results.filter(manager=request.user)
        }
        return render(request, 'search.html', context)
    else:
        return redirect('home')
```

- **Contact Create view**

```
class ContactCreateView(LoginRequiredMixin, CreateView):
    model = Contact
    template_name = 'create.html'
    fields = ['name', 'email', 'phone', 'info', 'gender', 'image']

    def form_valid(self, form):
        instance = form.save(commit=False)
        instance.manager = self.request.user
        instance.save()
        messages.success(
            self.request, 'Your contact has been successfully created!')
        return redirect('home')
```

- **Contact Update view**

```
class ContactUpdateView(LoginRequiredMixin, UpdateView):
    model = Contact
    template_name = 'update.html'
    fields = ['name', 'email', 'phone', 'info', 'gender', 'image']

    def form_valid(self, form):
        instance = form.save()
        messages.success(
            self.request, 'Your contact has been successfully updated!')
        return redirect('detail', instance.pk)
```

- **Contact delete view**

```
class ContactDeleteView(LoginRequiredMixin, DeleteView):
    model = Contact
    template_name = 'delete.html'
    success_url = '/'

    def delete(self, request, *args, **kwargs):
        messages.success(
            self.request, 'Your contact has been successfully deleted!')
        return super().delete(self, request, *args, **kwargs)
```

- **Sign up view**

```
class SignUpView(CreateView):
    form_class = UserCreationForm
    template_name = 'registration/signup.html'
    success_url = reverse_lazy('home')
```

KERNEL ALPHA STATES ACHIEVED

Note that highlighted states are achieved

- **Opportunity**
  - **Identified** - Opportunity are identified which includes the need to manage the contacts on the web app.

    - **Solution needed**- Solution is to make a web app Contact Manager that could manage your contacts and could keep them safe.This could also work as a backup for those who have lost their contact due to their change of sim or phone.
    - **Value established**-Value is established that this app will support editing,creating and deleting contacts.It also has login functionality for security.
    - **Viable**-The app is viable.It requires a django frame work, some html and python coding to built this app.
    - **Addressed**-It is addressed.The requirements are gathered and completed.The app is finally deployed.
    - **Benefit accrued**-Talked to our test users and benefits are accrued.
- **Stakeholder**
    - **Recognized**-Stakeholders are recognized.These are the people who will use our contact manager app.
    - **Represented**-We have talked to the stakeholders about the use cases and functionalities of the app.
    - **Involved**-Stakeholders are involved when we have to talk to them about the requirements and use case practise.
    - **In Agreement**-Stakeholders are agreed to our work and coding.
    - **Satisfied for deployment**-They are satisfied for moving ahead with the deployment.
    - **Satisfied in use**- They are satisfied with the app as whatever they have talked about is used in the app.
- **Requirements**
    - **Conceived**-Requirements are conceived as we have done find actors and use case activity in the use case practise.
    - **Bounded**-Requirements are bounded.The scope is also discussed in SRS document.Constraints are also mentioned their.
    - **Coherent**-It achieves the coherent state in slice the use case activity in the use case practise.

    - **Acceptable**-All these requirements are acceptable to the stakeholders.Value is also clear to them.
    - **Addressed**-All use cases are tested that are given by test cases and automation scripts in the testing document.
    - **Fulfilled-**All requirements are fulfilled except the user profile management which are left uncovered.
- **Software System**
    - **Architecture selected**-Architecture is selected as it is mentioned in the design document.Microservices also mention design model.
    - **Demonstrable**-The software system is demonstrated.Performance measured.Interfaces are also defined in microservice practise.The modules are integrated and tested.
    - **Usable**-After the unit and integration testing the app is usable.
    - **Ready-**The app is ready to use as we also done the system and acceptance testing.
    - **Operational-**System is operational in the user environment as it is deployed successfully and a link is created so that users can go to this managing app through this link.
    - **Retired-**The version is not retired yet.
- **Work**
    - **Initiated**-Work is initiated as we have decided to start with the sprint practise.
    - **Prepared**-All the initial tasks to be done are divided into 5 sprints.
    - **Started**-We achieve the start state at the start of each sprint as we do sprint planning.
    - **Under Control**-We achieve this state as we are performing the sprints under the timeline mentioned in the sprints.
    - **Concluded**-The initial tasks are completed through sprints.The use cases are identified in the use case practise.The use cases are then implemented in the microservice practise.
    - **Closed-**Work is not closed yet as there are some requirements left on which work has to be done.

- **Team**
  - **Seeded**-Team is seeded as we are 2 members working on this software project.
  - **Formed**-At the same time it is formed and work is divided among both the members.
  - **Collaborating**-We have meetings so that 2 members collaborate and get to know the work of each other.
  - **Performing**-The team performing the coding and documentation part is completed.
  - **Adjourned-**The team is not adjourned yet.
- **Way of Working**
  - **Principles established**-Principles and constraints identified.Practices and tools are agreed upon as mentioned in the starting sprint.
  - **Foundation established**-Key practices and tools are integrated.Capability gaps analyzed and understood.
  - **In use**-Practices and tools are understood and accepted by team members.
  - **In Place**-Both the team members are using the tools and key practices and sticking to the guidelines that are discussed during the start of the project.
  - **Working well-**Team members are working well as planned.We have performed scrum,use case and microservice essentialized practices while working on our software project.
  - **Retired-**Work is not retired yet.