Nami Kim

John Quan

CS 425

December 5, 2019

Kim's personal Information Manager

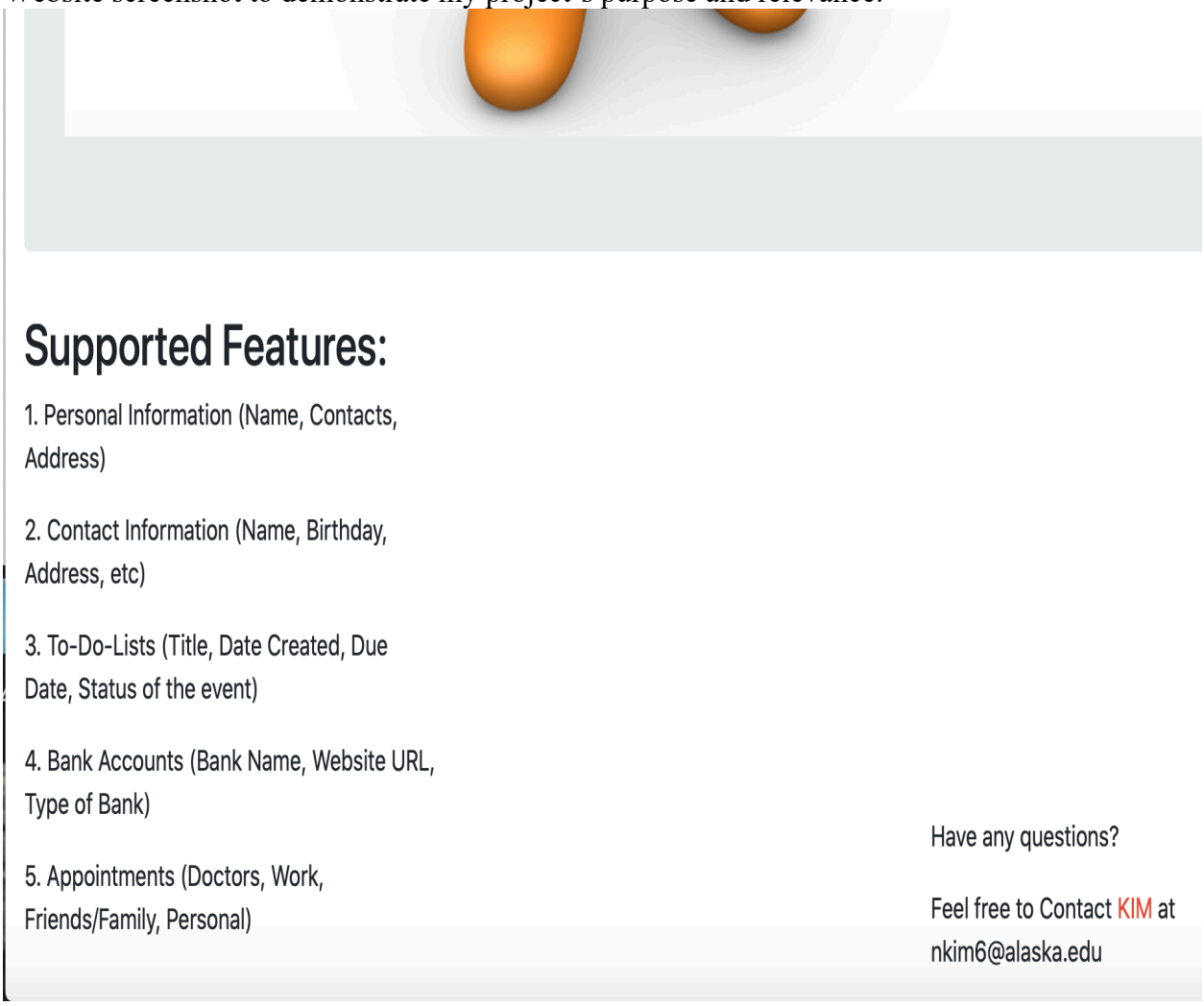
Final Report

Phase 1: Requirements Collection and Analysis

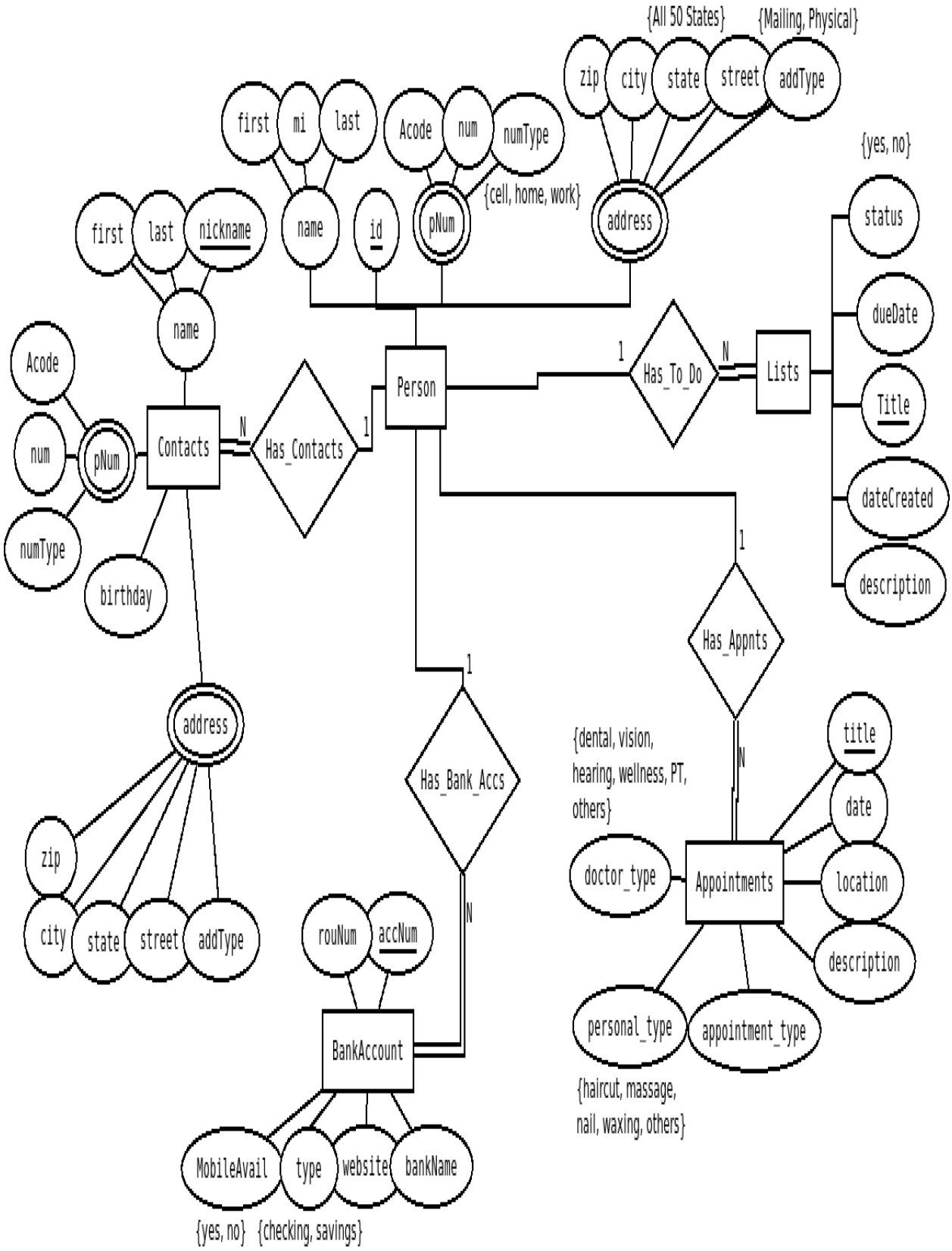
Purpose/Relevance:

- Can be used to store/access personal information

Website screenshot to demonstrate my project's purpose and relevance:



Phase 2: Conceptual Database Design (EER Diagram)



Phase 3: Data Model Mapping

```
# -*- coding utf-8 -*-
from django.db import models

from django.db import models
from django.urls import reverse

# Create your models here.

# Used Arrays for Many-to-Many relationships
PHONE_CHOICES = (
        ('C', 'Cell'),
        ('H', 'Home'),
        ('W', 'Work'),
)

ADDRESS_CHOICES = (
        ('M', 'Mailing'),
        ('P', 'Physical'),
)

STATE_CHOICES = (
        ('AL', 'Alabama'),
        ('AK', 'Alaska'),
        ('AZ', 'Arizona'),
        ('AR', 'Arkansas'),
        ('CA', 'California'),
        ('CO', 'Colorado'),
        ('CT', 'Connecticut'),
        ('DE', 'Delaware'),
        ('FL', 'Florida'),
        ('GA', 'Georgia'),
        ('HI', 'Hawaii'),
        ('ID', 'Idaho'),
        ('IL', 'Illinois'),
        ('IN', 'Indiana'),
        ('IA', 'Iowa'),
        ('KS', 'Kansas'),
        ('KY', 'Kentucky'),
        ('LA', 'Louisiana'),
        ('ME', 'Maine'),
        ('MD', 'Maryland'),
        ('MA', 'Massachusetts'),
        ('MI', 'Michigan'),
        ('MN', 'Minnesota'),
```

```
        ('MS', 'Mississippi'),
        ('MO', 'Missouri'),
        ('MT', 'Montana'),
        ('NE', 'Nebraska'),
        ('NV', 'Nevada'),
        ('NH', 'New Hampshire'),
        ('NJ', 'New Jersey'),
        ('NM', 'New Mexico'),
        ('NY', 'New York'),
        ('NC', 'North Carolina'),
        ('ND', 'North Dakota'),
        ('OH', 'Ohio'),
        ('OK', 'Oklahoma'),
        ('OR', 'Oregon'),
        ('PA', 'Pennsylvania'),
        ('RI', 'Rhode Island'),
        ('SC', 'South Carolina'),
        ('SD', 'South Dakota'),
        ('TN', 'Tennessee'),
        ('TX', 'Texas'),
        ('UT', 'Utah'),
        ('VT', 'Vermont'),
        ('VA', 'Virginia'),
        ('WA', 'Washington'),
        ('WV', 'West Virginia'),
        ('WI', 'Wisconsin'),
        ('WY', 'Wyoming'),
)

STATUS_CHOICES = (
        ('Y', 'Yes'),
        ('N', 'No'),
)

MOBILE_CHOICES = (
        ('Y', 'Yes'),
        ('N', 'No'),
)

BANK_CHOICES = (
        ('C', 'Checking'),
        ('S', 'Savings'),
)

DOCTOR_CHOICES = (
```

```
        ('D', 'Dental'),
        ('V', 'Vision'),
        ('H', 'Hearing'),
        ('W', 'Wellness'),
        ('PT', 'Phyical Therapy'),
        ('O', 'Others'),
)

PERSONAL_CHOICES = (
        ('H', 'Haircut'),
        ('M', 'Massage'),
        ('N', 'Nail'),
        ('W', 'Waxing'),
        ('O', 'Others'),
)

APPOINTMENT_CHOICES = (
        ('D', 'Doctors'),
        ('W', 'Work'),
        ('F', 'Friends/Family'),
        ('P', 'Personal'),
)

class Person(models.Model):
    first_name  = models.CharField(max_length=45)
    middle_initial = models.CharField(max_length=1, blank=True, null=True)
    last_name = models.CharField(max_length=45)

    def __str__(self):
        return "%s %s %s" % (self.first_name, self.middle_initial,
self.last_name)

    def get_absolute_url(self):
        return reverse("person-detail", kwargs={"pk": self.pk})


class PhoneNumber(models.Model):
    area_code = models.CharField(max_length=3)
    number = models.CharField(max_length=7)
    number_type = models.CharField(max_length=1, choices=PHONE_CHOICES)
    person = models.ForeignKey('Person', on_delete=models.PROTECT)

    def __str__(self):
        return "%s %s %s" % (self.area_code, self.number, self.number_type)
```

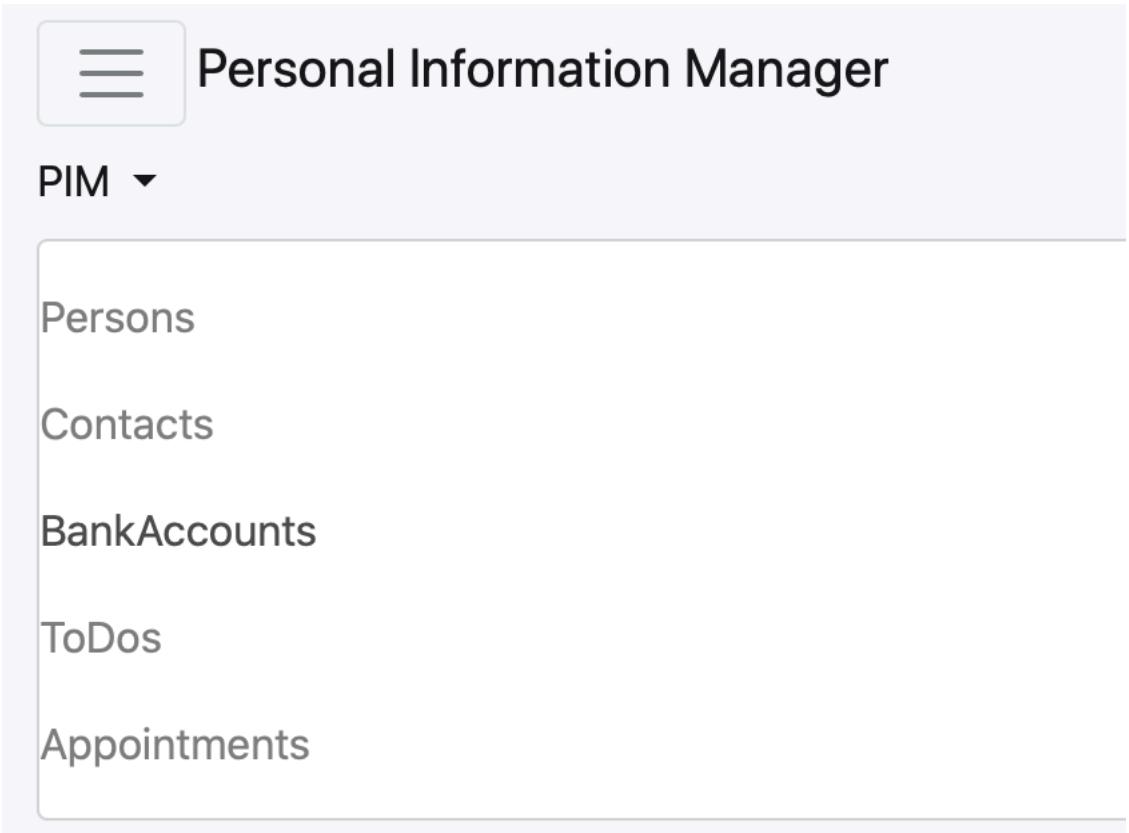
```
    def formatted_number(self):
        return "%s-%s" % (self.area_code, self.number)

    class Meta:
        unique_together = ('number_type', 'person',)


class Address(models.Model):
    zip_code = models.CharField(max_length=5)
    city = models.CharField(max_length=45)
    street = models.CharField(max_length=45)
    state = models.CharField(max_length=2, choices=STATE_CHOICES)
    address_type = models.CharField(max_length=1, choices=ADDRESS_CHOICES)
    person = models.ForeignKey('Person', on_delete=models.PROTECT)

    def __str__(self):
        return "%s %s %s %s %s" % (self.zip_code, self.city, self.street,
self.state, self.address_type)

    def full_address(self):
        return "%s %s %s %s" % (self.street, self.ciy, self.state,
self.zip_code)

    class Meta:
        unique_together = ('address_type', 'person')


class ToDo(models.Model):
    title = models.CharField(max_length=20, unique=True)
    description = models.CharField(max_length=20)
    date_created = models.DateField()
    date_due = models.DateField()
    status = models.CharField(max_length=1, choices=STATUS_CHOICES)
    person = models.ForeignKey('Person', on_delete=models.PROTECT)

    def __str__(self):
        return "%s %s %s %s %s" % (self.title, self.description,
self.date_created, self.date_due, self.status)

    def get_absolute_url(self):
        return reverse("todo-detail", kwargs={"pk": self.pk})


class Contact(models.Model):
    first_name = models.CharField(max_length=45)
```

```
    last_name= models.CharField(max_length=45)
    nickname = models.CharField(max_length=20, unique=True)
    birthday = models.DateField()
    person = models.ForeignKey('Person', on_delete=models.PROTECT)

    def __str__(self):
        return "%s %s %s %s" % (self.first_name, self.last_name,
self.nickname, self.birthday)

    def name(self):
        return "%s %s" % (self.first_name, self.last_name)

    def get_absolute_url(self):
        return reverse("contacts-detail", kwargs={"pk": self.pk})


class ContactPhoneNumber(models.Model):
    area_code = models.CharField(max_length=3)
    number = models.CharField(max_length=7)
    number_type = models.CharField(max_length=1, choices=PHONE_CHOICES)
    contact_person = models.ForeignKey('Contact', on_delete=models.PROTECT)

    def __str__(self):
        return "%s %s %s" % (self.area_code, self.number, self.number_type)

    def PhoneNumber(self):
        return "%s %s" % (self.area_code, self.number)


class ContactAddress(models.Model):
    zip_code = models.CharField(max_length=5)
    city = models.CharField(max_length=45)
    street = models.CharField(max_length=45)
    state = models.CharField(max_length=2, choices=STATE_CHOICES)
    address_type = models.CharField(max_length=1, choices=ADDRESS_CHOICES)
    address_person = models.ForeignKey('Contact', on_delete=models.PROTECT)

    def __str__(self):
        return "%s %s %s %s %s" % (self.zip_code, self.city, self.street,
self.state, self.address_type)

    def full_address(self):
        return "%s %s %s %s" % (self.street, self.city, self.state,
self.zip_code)
```

```
class BankAccount(models.Model):
    account_number = models.CharField(max_length=10, unique=True)
    routing_number = models.CharField(max_length=9)
    bank_name = models.CharField(max_length=45)
    website = models.CharField(max_length=45)
    mobile_availability = models.CharField(max_length=1,
choices=MOBILE_CHOICES)
    bank_type = models.CharField(max_length=1, choices=BANK_CHOICES)
    person = models.ForeignKey('Person', on_delete=models.PROTECT)

    def __str__(self):
        return "%s %s %s %s %s %s" % (self.account_number,
self.routing_number, self.bank_name, self.website, self.mobile_availability,
self.bank_type)

    def get_absolute_url(self):
        return reverse("bankaccount-detail", kwargs={"pk": self.pk})


class Appointment(models.Model):
    title = models.CharField(max_length=45, unique=True)
    date = models.DateField()
    location = models.CharField(max_length=45)
    person = models.ForeignKey('Person', on_delete=models.PROTECT)
    appointment_type = models.CharField(max_length=1,
choices=APPOINTMENT_CHOICES)
    doctor_type = models.CharField(max_length=2, choices=DOCTOR_CHOICES,
blank=True, null=True)
    personal_type = models.CharField(max_length=1, choices=PERSONAL_CHOICES,
blank=True, null=True)
    description = models.CharField(max_length=50, blank=True, null=True)

    def __str__(self):
        return "%s %s %s" % (self.title, self.date, self.location)

    def get_absolute_url(self):
        return reverse("appointment-detail", kwargs={"pk": self.pk})
```

Phase 4: Database System Implementation and Tuning

Available Views for the Website:



Example View Code: person_views.py

```
from django.views.generic.edit import CreateView, UpdateView, DeleteView
from django.views.generic.detail import DetailView
from django.views.generic.list import ListView

from django.contrib.auth.mixins import (PermissionRequiredMixin,
LoginRequiredMixin)

from django.forms import SelectDateWidget
import datetime

from django.urls import reverse_lazy

from pim.models import Person, ToDo, Contact, PhoneNumber, Address,
ContactPhoneNumber, ContactAddress, Appointment, BankAccount

# Create your views here.

class PersonCreateView(PermissionRequiredMixin, CreateView):
    model = Person
```

```
    fields = '__all__'
    template_name = "pim/add_update.html"
    success_url = "pim/persons"
    login_url = 'accounts/login'
    permission_required = (
            ('pim.add_person'))

    def get_form(self):
        form = super(PersonCreateView, self).get_form()

        return form

    def get_context_data(self, **kwargs):
        context = super(PersonCreateView, self).get_context_data(**kwargs)
        context['object_name'] = self.object.__class__.__name__

        return context

class PersonUpdateView(PermissionRequiredMixin, UpdateView):
    model = Person
    fields = '__all__'
    template_name = "pim/add_update.html"
    login_url = 'accounts/login'
    permission_required = (
            ('pim.change_person'))

    def get_form(self):
        form = super(PersonUpdateView, self).get_form()

        return form

    def get_context_data(self, **kwargs):
        context = super(PersonUpdateView, self).get_context_data(**kwargs)
        context['object_name'] = self.object.__class__.__name__

        return context

class PersonDeleteView(PermissionRequiredMixin, DeleteView):
    model = Person
    template_name = "pim/confirm_delete.html"
    success_url = 'pim/persons'
    login_url = 'accounts/login'
    permission_required = (
            ('pim.delete_person'))
```

```
    def get_context_data(self, **kwargs):
        context = super(PersonDeleteView, self).get_context_data(**kwargs)
        context['object_name'] = self.object.__class__.__name__

        return context

class PersonDetailView(PermissionRequiredMixin, DetailView):
    model = Person
    template_name = 'pim/detail.html'
    login_url = 'accounts/login'
    permission_required = (
            ('pim.change_person'))

    def get_context_data(self, **kwargs):
        """Override this function"""

        # person instance
        person = Person.objects.get(id=self.kwargs['pk'])

        # person_detail
        qs = (Person.objects.filter(id=self.kwargs['pk']).values('id',
            'first_name', 'middle_initial', 'last_name'))
        pim_dict = {}
        pim_dict.setdefault('person_detail', qs[0])

        # phonenumber
        qs =
(PhoneNumber.objects.filter(person=self.kwargs['pk']).values('area_code',
'number', 'number_type'))
        pim_dict.setdefault('phonenumber', qs)

        # address
        qs2 =
(Address.objects.filter(person=self.kwargs['pk']).values('zip_code', 'city',
'street', 'state', 'address_type'))
        pim_dict.setdefault('address', qs2)

        # to-do-list
        qs3 = (ToDo.objects.filter(person=self.kwargs['pk']).values('title',
'description', 'date_created', 'date_due', 'status'))
        pim_dict.setdefault('to do list', qs3)

        # contact
        qs4 = (Contact.objects.filter(person=self.kwargs['pk']).values('id',
'first_name', 'last_name', 'nickname', 'birthday'))
```

```
        pim_dict.setdefault('contact', qs4)

        # contact phone number
        qs5 =
(ContactPhoneNumber.objects.filter(contact_person=self.kwargs['pk']).values('
contact_person_id', 'area_code', 'number', 'number_type'))
        pim_dict.setdefault('contact phone number', qs5)

        # contact address
        qs6 =
(ContactAddress.objects.filter(address_person=self.kwargs['pk']).values('addr
ess_person_id', 'zip_code', 'city', 'street', 'state', 'address_type'))
        pim_dict.setdefault('contact address', qs6)

        # bank account
        qs7 =
(BankAccount.objects.filter(person=self.kwargs['pk']).values('account_number'
, 'routing_number', 'bank_name', 'website', 'mobile_availability',
'bank_type'))
        pim_dict.setdefault('bank account', qs7)

        # appointment
        qs8 =
(Appointment.objects.filter(person=self.kwargs['pk']).values('title', 'date',
'location', 'doctor_type', 'personal_type', 'description'))
        pim_dict.setdefault('appointment', qs8)


        # get the current context so we can add our lists to it
        context = super(PersonDetailView, self).get_context_data(**kwargs)
        # querysets are lists
        context['detail_list'] = pim_dict

        context['object_name'] = self.object.__class__.__name__

        return context

class PersonListView(LoginRequiredMixin, ListView):
    model = Person
    template_name = 'pim/list.html'
    context_object_name = 'my_objects'
    paginate_by = 25 # includes/pagination.html

    queryset = (Person.objects.values(
        'id', 'last_name', 'first_name', 'middle_initial'))
```

```
    def get_context_data(self, **kwargs):
        #person_detail with id for urls
        context = super(PersonListView, self).get_context_data(**kwargs)
        context['object_name'] = self.model.__name__

        return context
```

References

https://docs.google.com/document/u/1/d/1uN02XxMdw4j6bntJyes4eJoUI0LDEIeWHpe3XN_BVMk/edit?usp=drive_web&ouid=113326339872818623350 (Lab 09 Django CMS 2019)

https://docs.djangoproject.com/en/2.2/intro/tutorial01/

Appendices

MySql Workbench Diagram: