Mahdi Aleboyeh

Medical Appointment System

Medical System Documentation

Underlined fields means not required.  
Bold fields means primary key.  
  
User:

u/user : method = [GET(list only)]

GET: list -> permissions: [Admin], query set=User.objects.all()

u/user/send\_otp : method = [POST]

POST: create User:

permissions: [AnyOne], data request body :[**phone\_number:** char]

u/user/verify\_otp : method = [POST]

POST: create User:

permissions: [AnyOne], data request body :[**phone\_number:** char, otp\_code: char]

u/user/me: methods = [GET, PATCH, PUT], permissions: [Authenticate]

GET: retrieve user information:

query set=User.objects.get(user=user) and medic or patient if exists

PUT, PATCH: update user :

If medic:

data request body : [first\_name(user): char, last\_name(user): char, age(user): int,

image: image, specialization(unique): char,

medical\_system\_number: char]

if patient:

data request body : [first\_name(user): char, last\_name(user): char, age(user):

int, image: image, address: text, medical\_history: text, insurance\_info:

char, blood\_group: char, drug\_allergy: text, special\_medicine: text,

system\_diseases: text]

if only user:

data request body : [first\_name: char, last\_name: char, age: int]

u/user/medic\_entry: method = [POST], permission: [Authenticate]

POST: enter as medic:

Changing user.is\_medic to True and user.is\_patient = False

u/user/patient\_entry: method = [POST], permission: [Authenticate]

POST: enter as patient:

Changing user.is\_medic to False and user.is\_patient = True

u/user/medic\_or\_patient: method = [GET], permission: [Authenticate]

GET: get medic and patient information if exists:

query set=medic.objects.get(user=user) and patient.objects.get(user=user)

Medic:

u/medic: methods=[POST,GET,DELETE]

POST: create medic, permissions = [Authenticate]

If first\_name and last\_name and age not set before:

data request body : [first\_name(user): char, last\_name(user): char, age(user): int,

image: image, specialization: char,

medical\_system\_number(unique): char]

if set:

data request body : [image: image, specialization: char

, medical\_system\_number(unique): char]

\* medic must wait till acceptation from admin:

Medic can change profile but can’t set appointment times and introduce clinics and creating medical records.

GET: list and retreive, permission = [AnyOne]:

query set = Medic.objects.all()

DELETE: delete, permission = [Owenr and Admin]:

Deleting medic object

When delete user.is\_medic turn to False

u/medic/pk/appointment\_times: method = [GET], permission = [AnyOne]

GET - detailed: list of appointment times not filled yet from a specific medic.

Query set = TimeSlot.objects.filter(medic=medic).filter(not fully reserved)   
 \* for next seven days and if time was active.

Patient:

u/patient: methods=[POST,GET(list only),DELETE]

POST: create patient, permissions = [Authenticate]

If first\_name and last\_name and age not set before:

data request body : [first\_name(user): char, last\_name(user): char, age(user): int,

image: image, address: text, medical\_history: text, insurance\_info: char,

blood\_group: char, drug\_allergy: text, special\_medicine: text,

system\_diseases: text]

if set:

data request body : [image: image, address: text, medical\_history: text,

insurance\_info: char, blood\_group: char, drug\_allergy: text,

special\_medicine: text, system\_diseases: text]

GET: list ,permission = [Admin]:

query set = Patient.objects.all()

DELETE: delete, permission = [Owenr and Admin]:

Deleting patient object

When delete user.is\_patient turn to False

Time Slot(medic available time, can be set for repeated in week):

u/available\_times: methods=[GET, POST, PUT, PATCH, DELETE], permissions = [Medic, Admin]

query set:

if admin:

TimeSlot.objects.all()

If medic:

TimeSlot.objects.filter(medic=user.medic)

GET: list and retrieve:

Query set based on user requesting.

POST: create:

data request body : [**medic**(set automatically): prime\_key, **clinic**(can be introduce if not

exists): prime\_key, day\_of\_week: int(choice field), start\_time: time, end\_time: time, avg\_visit\_time: int(minute), avg\_patient\_visit(automatically set by dividing end – start time / avg\_visit time): int, is\_active(default True): Bool]

\* Medic and day\_of\_week : unique together

PATCH, PUT: update:

data request body : [**clinic**: prime\_key, day\_of\_week: int(choice field),

start\_time: time, end\_time: time, avg\_visit\_time: int(minute), avg\_patient\_visit: int, is\_active: Bool]

DELETE: deleting object.

Clinic:

c/clinic : methods = [GET, PUT, PATCH, POST, DELETE]

GET: list and retrieve, permission = [AnyOne]:

Query set = Clinic.objects.all()

POST: create, permission = [Medic and Admin]:

data request body : [name: char, address: text, clinic\_serial(unique): char,

image: image)

\* clinic must be accepted by a super user. Until then clinic can’t be use.

DELETE and PUT and PATCH: deleting and updating, permission = [Admin]:

data request body : [name: char, address: text, clinic\_serial(unique): char,

image: image)

\* after deleting a clinic, all time slots related to clinic will be delete too.

Appointments:

a/appointment : methods = [GET, PUT, PATCH, POST, DELETE]

GET, DELETE: list and retrieve, permission = [Appointment related(patient and medic of appointment) and Admin:

Query set:

If medic:

Appointment.objects.filter(medic=medic)

If patient:

Appointment.objects.filter(patient=patient)

Beside this query set, also shows prescription and medical record of this appointment.

PUT, PATCH: update, permission = [Medic and Admin]:

data request body :[**patient**: prime\_key, **time**: prime\_key,

short\_description: char, appointment\_datetime: datetime,

appointment\_number: int]

POST: create, permission = [Patient and Admin]:

data request body :[**patient**(automatically from request.user.patient): prime\_ket,

**time**: prime\_key, short\_description: char]

**\*** this method also create a prescription for this appointment and create or get the medical record for this medic and patient.

\* appointment\_datetime: datetime and appointment\_number: int will be set automatically base of how many appointment set before this on that date.

a/appointment/my\_appointment: methods = [GET], permission=[Authenticate]

GET: list of appointment reserved for next week.

Query set:

Appointment.objects.filter(user=user).filter(for next week)

Medical Records:

m/medical\_record: methods = [GET, POST, PATCH, PUT, DELETE]

GET: list and retrieve, permission = [Authenticate]

Query set:

If admin:

MedicalRecord.objects.all()

If medic:

MedicalRecord.objects.filter(medic=user.medic)

If patient:

MedicalRecord.objects.filter(patient=user.patient)

POST, PATCH, PUT: create, update, permission = [Medic and Admin]

data request body: [**medic**(automatically set): prime\_key, **patient**: prime\_key,

prescription\_number(Prescription): char, drugs(Prescription): text,

illness\_subject: char, illness: text, hospitalized(default False): bool]

DELETE: permission = [Admin]

Prescription

a/prescription: methods=[GET, POST, PUT, PATCH, DELETE]

GET: list and retrieve, permission = [Authenticate]:

Query set = Prescription.objects.filter(user=user)

PUT, PATCH, CREATE: create and update, permission = [Admin or Medic]

DELETE: permission = [Admin]

\* With celery I send tomorrow appointment to each medic