# Health Connect REST API Documentation

## Introduction

The Next app API is a RESTful API built using Django REST Framework. It serves as the backend for Next app providing a scalable and modular system for handling data, authentication, and business logic.

## Technologies Used

* Python
* Django
* Django REST API (DRF)
* Message Central OTP verification
* Razor pay payment gateway
* Postgres DB
* Firebase FCM

## Authentication Flow

The API implements token-based authentication with OTP verification for both users and partners.

### Authentication Process

* Registration or login request (phone number provided)
* OTP sent to user/partner phone
* OTP verification
* Token issuance (access + refresh tokens)
* Token usage for API access

## API Routes

## Auth Routes (Partner and User)

## Registration end points

## auth/register/user/

## This route is used to register a user where it will take phone number, email and full name to give out OTP and a verification id(part of OTP verification) which needs to be used in the route auth/verify/user. There is max retry count of 3 every minute. Full name and email is stored in cache so it can be accessed by auth/verify/user to add to db.

## auth/verify/user/

## This route is used to verify the user’s OTP and verification ID and they are verified by Message central service If verification is ok then email and full name is taken from cache and a user object is created then the cache is cleared. The user will receive their appropriate JWT access and refresh tokens.

## auth/register/partner/

## This route is used to register a partner where it will take phone number, email, education. Experience, medical certificate and full name to give out OTP and a verification id(part of OTP verification) which needs to be used in the route auth/verify/user. There is max retry count of 3 every minute. Full name and email and others are stored in cache so it can be accessed by auth/verify/user to add to db.

## auth/verify/partner/

## This route is used to verify the partner’s OTP and verification ID and they are verified by Message central service If verification is ok then email and full name and others are taken from cache and a user and its partner object is created then the cache is cleared. If a already registered user registers as partner then the user’s partner object is created. The user will receive their appropriate JWT access and refresh tokens for partner views.

## Login end points

## auth/login/user/

## This route is used for sending OTP and verification id so they can login to the app. Phone number is taken here. Max retry of 3 is set and phone number and verification is stored in cache so it can be accessed by auth/verify/login/user to verify and login.

## auth/verify/login/user/

## This route is used to verify the user’s OTP and verification ID and they are verified by Message central service and user will receive their appropriate JWT access and refresh tokens.

## auth/login/partner/

## This route is used for sending OTP and verification id so they can login to the app. Phone number is taken here. Max retry of 3 is set and phone number and verification is stored in cache so it can be accessed by auth/verify/login/user to verify and login.

## auth/save-fcm-token/

## This route is used to save fcm token received from especially mobile frontends to issue notifications on some actions.

## auth/verify/login/partner/

## This route is used to verify the partner’s OTP and verification ID and they are verified by Message central service and user will receive their appropriate JWT access and refresh tokens.

## JWT Token end points

## auth/token/refresh/

## This route is used to create access tokens for auth by providing refresh if expired. This app needs access token in header to access each protected routes.

## auth/token/verify/

## This route is used to check if a access token has expired so the app can use auth/token/refresh to create another from refresh token

## auth/logout/

## This route is used to logout for both user and partner. In context of JWT logging out means abandoning both refresh and access by the API consumer (web-app , mobile app etc) if stored so the user can’t use it anymore to login. Here a blacklisting mechanism is implemented where if logged out the old refresh token is stored in blacklisted table in database so it can’t be used anymore.

## User Routes

## User end points

## user/home/

## Home page route for user to access after logging in. This route returns user details.

## user/services/

## This route gives out the available services details which are Hospital care , check-up companion, adult care, baby sitting. This route also gives hourly rate of the above services.

## user/bookings/history/

## This route gives out booking history of users with status either completed or cancelled.

## user/bookings/<int:booking\_id>/

## This route gives out detailed view of a specific booking.

## user/bookings/create/

## This route is used to create a booking for either book now or book later options.This route has validations for avoiding same day bookings if another booking exists and time overlap for book laters.

## Instant booking is where partners can accept work and start work immediately.

## {

## "service\_type": 1,

## "partner\_type": "trained",

## "is\_instant": true,

## "hours": 4,

## "user\_location": "123 Main Street, Anytown",

## "hospital\_location": "General Hospital, Anytown",

## "notes": "Bring n95 Mask",

## "long": -74.0060,

## "lang": 40.7128

## }

## Booking later is where partners can accept work and start work at given time and date.

## {

## "service\_type": 2,

## "partner\_type": "normal",

## "is\_instant": false,

## "scheduled\_date": "2025-04-30",

## "scheduled\_time": "14:30",

## "hours": 3,

## "user\_location": "456 Elm Street, Anytown",

## "hospital\_location": "City Health Clinic, Anytown",

## "notes": "Patient needs wheelchair assistance",

## "long": -74.0059,

## "lang": 40.7130

## }

## In Service type checkup companion there needs to give a additional field called hospital location.

## {

## "service\_type": 3,

## "partner\_type": "trained",

## "is\_instant": false,

## "scheduled\_date": "2025-05-02",

## "scheduled\_time": "10:00",

## "hours": 2,

## "user\_location": "789 Oak Avenue, Anytown",

## "hospital\_location": "Downtown Medical Center, Anytown",

## "notes": "Elderly patient, please assist with forms",

## "long": -74.0070,

## "lang": 40.7110

## }

## user/bookings/<int:booking\_id>/cancel/

## This route cancels a specific booking and issues a refund to user.

## user/bookings/pending/

## This route gives out bookings which are pending and not accepted by partner.

## user/bookings/<int:booking\_id>/available-partners/

## This route gives available partners for a booking.

## user/bookings/<int:booking\_id>/select-partner/<int:partner\_id>

## This route is used to select a partner using partner id gevin from available partner.

## user/bookings/<int:booking\_id>/create-order/

## This route is used to create the razorpay order with necessary details like amount etc.

## user/bookings/active/

## This route gives out bookings that are active ie where partner has accepted.

## user/bookings/<int:booking\_id>/extention/

## This route is used to requests extentions for bookings that are in progress but not completed.

## user/extentions/<int:extention\_id>/create-order/

## This route is used to create the razorpay extention order with necessary extention amount.

## user/bookings/<int:booking\_id>/review/

## This route is used to create review after booking work is finished.

## user/razorpay/webhook/

## This route is used to verify payments using the razorpay webhook.

## Partner Routes

## Partner end points

## partner/home/

## This route is used to get partner name and other partner related details

## partner/services/

## This route is used to get services data.

## partner/book-slot/

## This route is used to book slot in 2hr format to receive bookings.

## partner/booked-slots/

## This route is used to see booked slots.

## partner/bookings/available/

## This route is used to see available bookings if a slot is active.

## partner/bookings/<int:booking\_id>/accept/

## This route is used to accept a booking and wait for user approval.

## partner/bookings/<int:booking\_id>/release/

## This route is used to to release a booking if cant complete it .

## partner/bookings/active/

## This route is used to see active bookings which are paid and confirmed by user.

## partner/bookings/<int:booking\_id>/toggle-status/

## This route is used to start and stop work , first call will start and second call will stop.

## partner/bookings/<int:booking\_id>/

## This route is used to the specified booking in detail.

## partner/bookings/<int:booking\_id>/extentions/

## This route is used to see extention requests for a given work.

## partner/extentions/<int:extention\_id>/respond/

## This route is used to respond to extentions ie either accept or reject.

## partner/bookings/completed/

## This route is used to see completed bookings.

## partner/reviews/

## This route is used to see reviews for each bookings.

## partner/update-bank-details/

## This route is used to update bank details like ifsc etc

## partner/wallet-details/

## This route is used to see wallet details.

## Admin Routes

## Admin end points

## adminapp/login/

## This route is used to login with ph no and password.

## adminapp/totp-verify/

## This route is used to verify totp that was gevin from totp app after registered using totp-setup route

## adminapp/totp-setup/

## This route is used to setup totp in auth app like google totp using a qr code.

## adminapp/logout/

## This route is used to logout.

## adminapp/users/

## This route is used to see users list and do crud.

## adminapp/partners/

## This route is used to see partners list and verify details and approve.

## adminapp/services/

## This route is used to see all services and do crud , this can be also used to temporarily disable services.

## adminapp/bookings/

## This route is used to see bookings and do crud.

## adminapp/bookings/stuck/

## This route is used to see stuck bookings ie bookings where partner has released and no other partner has accepted for more than 30min.

## adminapp/bookings/assign-partner/<int:bookings\_id>

## This route is used to update partners in a booking manually.

## adminapp/bookings/edit/<int:bookings\_id>

## This route is used to edit bookings.

## adminapp/bookings/delete/<int:bookings\_id>

## This route is used to delete bookings.

## adminapp/partners/<int:partner\_id>/trigger-payout/

## This route is used to trigger payout to single or multiple partners.

## adminapp/bookings/int:bookings\_id>/refund/

## This route is used to do manual refunds to users due to unforeseen circumstances.

## Data Models

## Models related to auth and users

1. class UserManager(BaseUserManager):
2. def create\_user(self, phone\_number, email, full\_name, password=None):
3. if not phone\_number:
4. raise ValueError("Users must have a phone number")
5. user = self.model(phone\_number=phone\_number, email=email, full\_name=full\_name)
6. if password:  # Only set a password if provided
7. user.set\_password(password)  # Hash the password
8. user.save(using=self.\_db)
9. return user
11. def create\_superuser(self, phone\_number, email, full\_name, password):
12. """
13. Create and return a superuser with the given details.
14. """
15. user = self.create\_user(phone\_number, email, full\_name, password)
16. user.is\_admin = True
17. user.is\_staff = True
18. user.set\_password(password)
19. user.is\_superuser = True
20. user.save(using=self.\_db)
21. return user
22. # User Model
23. class CustomUser(AbstractBaseUser):
24. phone\_number = models.CharField(max\_length=15, unique=True)
25. email = models.EmailField(unique=True)
26. full\_name = models.CharField(max\_length=255)
27. user\_address = models.CharField(max\_length=255, null=True, blank=True)
28. is\_partner = models.BooleanField(default=False)
30. USERNAME\_FIELD = 'phone\_number'
31. REQUIRED\_FIELDS = ['email', 'full\_name']
32. is\_staff = models.BooleanField(default=False)
33. is\_active = models.BooleanField(default=True)
34. is\_superuser = models.BooleanField(default=False)
35. objects = UserManager()
36. def get\_full\_name(self):
37. return self.full\_name
38. # Permission checks for Django Admin
39. def has\_perm(self, perm, obj=None):
40. """
41. Returns True if the user has the given permission.
42. """
43. # Superusers have all permissions
44. if self.is\_superuser:
45. return True
46. # Implement other permission checks as needed, for example:
47. return False
48. def has\_module\_perms(self, app\_label):
49. """
50. Returns True if the user has permission to access the given app's module.
51. """
52. # Superusers have access to all modules
53. if self.is\_superuser:
54. return True
55. # Implement more checks if needed for other roles
56. return False
57. def \_\_str\_\_(self):
58. return f"{self.full\_name} {self.phone\_number} "
59. # Partner Model
60. class Partner(CustomUser):
61. is\_verified = models.BooleanField(default=False)
62. experience = models.CharField(max\_length=255, null=True, blank=True)
63. total\_earnings = models.DecimalField(max\_digits=12, decimal\_places=2, default=0.00)
64. # Personal Information
65. adhar\_card\_front = models.ImageField(upload\_to='documents/adhar\_cards/front/', null=True, blank=True)  # Aadhar front image
66. adhar\_card\_back = models.ImageField(upload\_to='documents/adhar\_cards/back/', null=True, blank=True)   # Aadhar back image
67. driving\_license\_front = models.ImageField(upload\_to='documents/driving\_licenses/front/', null=True, blank=True)  # Driving license front image
68. driving\_license\_back = models.ImageField(upload\_to='documents/driving\_licenses/back/', null=True, blank=True)  # Driving license back image
69. profile\_picture = models.ImageField(upload\_to='documents/profile\_pictures/', null=True, blank=True)  # Profile picture
70. medical\_certificate = models.FileField(upload\_to='documents/certificates/',null=True, blank=True) # will this accept picture ?
71. education = models.CharField(max\_length=255)
72. # Additional Information
73. dob = models.DateField(null=True, blank=True)  # Date of Birth
74. languages\_known = models.CharField(max\_length=255, null=True, blank=True)  # Languages known, comma-separated
75. secondary\_phone\_number = models.CharField(max\_length=20, null=True, blank=True)  # Secondary phone number
76. # bank details
77. bank\_username = models.CharField(max\_length=255, null=True, blank=True)
78. bank\_account\_number = models.CharField(max\_length=50, null=True, blank=True)
79. ifsc\_code = models.CharField(max\_length=20, null=True, blank=True)
80. # Location Details
81. address = models.CharField(max\_length=255, null=True, blank=True)
82. def \_\_str\_\_(self):
83. return f"Partner: {self.id} {self.full\_name} ({self.phone\_number})"
84. # OTP Model
85. class OTP(models.Model):
86. phone\_number = models.CharField(max\_length=15, unique=True)
87. otp = models.CharField(max\_length=6)
88. created\_at = models.DateTimeField(auto\_now\_add=True)
89. def \_\_str\_\_(self):
90. return f"OTP for {self.phone\_number} - {self.otp}
91. class FCMToken(models.Model):
92. user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE, related\_name='fcm\_tokens')
93. token = models.CharField(max\_length=255)  # Removed unique=True
94. created\_at = models.DateTimeField(auto\_now\_add=True)
96. class Meta:
97. unique\_together = ('user', 'token')  # Only prevent duplicate user-token pairs
98. class AdminProfile(models.Model):
99. """Profile for admin users with TOTP settings"""
100. user = models.OneToOneField(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, related\_name='admin\_profile')
101. last\_login = models.DateTimeField(null=True, blank=True)
103. def \_\_str\_\_(self):
104. return f"Admin: {self.user.full\_name}"
106. @property
107. def has\_active\_totp(self):
108. """Check if user has configured TOTP"""
109. return TOTPDevice.objects.filter(user=self.user, confirmed=True).exists()

## Models related to booking

1. # Service types
2. class ServiceType(models.Model):
3. SERVICE\_CHOICES = (
4. ('hospital\_care', 'Hospital Care'),
5. ('checkup\_companion', 'Checkup Companion'),
6. ('adult\_care', 'Adult Care'),
7. ('baby\_sitting', 'Baby Sitting'),
8. )
10. name = models.CharField(max\_length=50, choices=SERVICE\_CHOICES, unique=True)
11. description = models.TextField()
12. base\_hourly\_rate = models.DecimalField(max\_digits=10, decimal\_places=2)
13. is\_active = models.BooleanField(default=True)  # <-- New field
15. def \_\_str\_\_(self):
16. return self.get\_name\_display()
18. from decimal import Decimal
19. # Booking Model
20. class Booking(models.Model):
21. STATUS\_CHOICES = (
22. ('pending', 'Pending'),
23. ('confirmed', 'Confirmed'),
24. ('in\_progress', 'In Progress'),
25. ('completed', 'Completed'),
26. ('cancelled', 'Cancelled'),
27. )
29. PARTNER\_TYPE\_CHOICES = (
30. ('trained', 'Trained (2+ years)'),
31. ('regular', 'Regular (Less than 2 years)'),
32. )
34. user = models.ForeignKey(CustomUser, on\_delete=models.CASCADE, related\_name='bookings')
35. service\_type = models.ForeignKey(ServiceType, on\_delete=models.CASCADE)
36. partner\_type = models.CharField(max\_length=10, choices=PARTNER\_TYPE\_CHOICES)
37. partner = models.ForeignKey(Partner, on\_delete=models.SET\_NULL, null=True, blank=True, related\_name='assignments')
39. # Booking details
40. is\_instant = models.BooleanField(default=True)  # True for "Book Now", False for "Book Later"
41. hours = models.PositiveIntegerField()
42. scheduled\_date = models.DateField(null=True, blank=True)
43. scheduled\_time = models.TimeField(null=True, blank=True)
44. notes = models.TextField(blank=True, null=True)  # Optional field for storing additional notes
46. # Locations
47. user\_location = models.CharField(max\_length=255)
48. lang = models.FloatField(null=True, blank=True)  # Latitude
49. long = models.FloatField(null=True, blank=True)  # Longitude
50. hospital\_location = models.CharField(max\_length=255, null=True, blank=True)  # Only for "Checkup Companion"
52. # Status tracking
53. status = models.CharField(max\_length=15, choices=STATUS\_CHOICES, default='pending')
54. created\_at = models.DateTimeField(auto\_now\_add=True)
55. partner\_accepted\_at = models.DateTimeField(null=True, blank=True)
56. # partner release tracking
58. released\_by = models.ForeignKey(Partner, on\_delete=models.SET\_NULL, null=True, blank=True, related\_name='released\_bookings')
59. released\_at = models.DateTimeField(null=True, blank=True)
60. cancellation\_reason = models.TextField(null=True, blank=True)
61. work\_started\_at = models.DateTimeField(null=True, blank=True)
62. work\_ended\_at = models.DateTimeField(null=True, blank=True)
64. # Payment
65. total\_amount = models.DecimalField(max\_digits=10, decimal\_places=2, null=True, blank=True)
66. payment\_status = models.CharField(max\_length=20, default='pending')
68. def \_\_str\_\_(self):
69. return f"{self.user.full\_name} - {self.service\_type} - {self.status} - {self.id}"
71. def calculate\_total\_amount(self):
72. base\_rate = self.service\_type.base\_hourly\_rate
73. # Add premium for trained partners
74. rate\_multiplier = Decimal(1.5) if self.partner\_type == 'trained' else Decimal(1.0)
75. self.total\_amount = base\_rate \* rate\_multiplier \* Decimal(self.hours)
76. return self.total\_amount
77. # Booking requests from users to partners
78. class BookingRequest(models.Model):
79. STATUS\_CHOICES = (
80. ('pending', 'Pending'),
81. ('accepted', 'Accepted'),
82. ('rejected', 'Rejected'),
83. ('released', 'Released'),
84. )
86. booking = models.ForeignKey(Booking, on\_delete=models.CASCADE, related\_name='requests')
87. partner = models.ForeignKey(Partner, on\_delete=models.CASCADE, related\_name='booking\_requests')
88. status = models.CharField(max\_length=10, choices=STATUS\_CHOICES, default='pending')
89. created\_at = models.DateTimeField(auto\_now\_add=True)
91. class Meta:
92. unique\_together = ('booking', 'partner')
93. def \_\_str\_\_(self):
94. return f"BookingRequest {self.id}: {self.booking.user.full\_name} - {self.partner.full\_name} - {self.status}"
95. # Extension requests for additional hours
96. class BookingExtension(models.Model):
97. STATUS\_CHOICES = (
98. ('pending', 'Pending'),
99. ('approved', 'Approved'),
100. ('rejected', 'Rejected'),
101. )
103. booking = models.ForeignKey(Booking, on\_delete=models.CASCADE, related\_name='extensions')
104. additional\_hours = models.PositiveIntegerField()
105. status = models.CharField(max\_length=10, choices=STATUS\_CHOICES, default='pending')
106. requested\_at = models.DateTimeField(auto\_now\_add=True)
107. partner\_accepted\_at = models.DateTimeField(null=True, blank=True)
108. cancellation\_reason = models.TextField(null=True, blank=True)
109. # Payment for extension
110. extension\_amount = models.DecimalField(max\_digits=10, decimal\_places=2)
111. payment\_status = models.CharField(max\_length=20, default='pending')
112. def \_\_str\_\_(self):
113. return f"BookingExtension {self.id}: {self.booking.id} - {self.status}"
115. # Reviews and Ratings
116. class Review(models.Model):
117. booking = models.OneToOneField(Booking, on\_delete=models.CASCADE, related\_name='review')
118. rating = models.PositiveIntegerField()  # 1-5 star rating
119. comment = models.TextField()
120. created\_at = models.DateTimeField(auto\_now\_add=True)
121. def \_\_str\_\_(self):
122. return f"Review {self.id} for Booking {self.booking.id} - Rating: {self.rating}"
123. class PartnerSlot(models.Model):
124. partner = models.ForeignKey(Partner, on\_delete=models.CASCADE, related\_name='slots')
125. date = models.DateField()
126. start\_time = models.TimeField()
127. end\_time = models.TimeField()
128. is\_active = models.BooleanField(default=True)
129. class Meta:
130. unique\_together = ('partner', 'date', 'start\_time', 'end\_time')
131. def \_\_str\_\_(self):
132. return f"PartnerSlot {self.id}: {self.partner.full\_name} - {self.date} {self.start\_time}-{self.end\_time}"
134. class PartnerWallet(models.Model):
135. partner = models.OneToOneField(Partner, on\_delete=models.CASCADE, related\_name='wallet')
136. balance = models.DecimalField(max\_digits=10, decimal\_places=2, default=0)
137. last\_payout\_date = models.DateTimeField(null=True, blank=True)
139. def \_\_str\_\_(self):
140. return f"PartnerWallet {self.id}: {self.partner.full\_name} - ₹{self.balance}"
142. class Transaction(models.Model):
143. TRANSACTION\_TYPES = (
144. ('booking\_payment', 'Booking Payment'),
145. ('extension\_payment', 'Extension Payment'),
146. ('partner\_payout', 'Partner Payout'),
147. )
149. booking = models.ForeignKey(Booking, on\_delete=models.SET\_NULL, null=True, blank=True, related\_name='transactions')
150. extension = models.ForeignKey(BookingExtension, on\_delete=models.SET\_NULL, null=True, blank=True, related\_name='transactions')
151. partner\_wallet = models.ForeignKey(PartnerWallet, on\_delete=models.SET\_NULL, null=True, blank=True, related\_name='transactions')
153. amount = models.DecimalField(max\_digits=10, decimal\_places=2)
154. transaction\_type = models.CharField(max\_length=20, choices=TRANSACTION\_TYPES)
155. razorpay\_payment\_id = models.CharField(max\_length=100, null=True, blank=True)
156. razorpay\_order\_id = models.CharField(max\_length=100, null=True, blank=True)
157. status = models.CharField(max\_length=20, default='pending')
158. created\_at = models.DateTimeField(auto\_now\_add=True)
159. refund\_id = models.CharField(max\_length=100, null=True, blank=True)
160. refund\_status = models.CharField(max\_length=20, null=True, blank=True)
161. def \_\_str\_\_(self):
162. return f"Transaction {self.id}: {self.transaction\_type} - ₹{self.amount} - {self.status}"