I'll continue with documenting the remaining APIs for the FoodShare application. Let me now document the Donation APIs.# FoodShare API Documentation - Donation APIs

Based on my analysis of the codebase, particularly the donation service and related components, I'll now document the Donation APIs available in the FoodShare application.

8. Donation APIs

8.1 Create Donation

Endpoint URL: `/api/donations/create`

HTTP Method: `POST`

Request Headers:
```
Content-Type: application/json
Authorization: Bearer {token}
```

Request Body Schema:
```json
{
  "title": "Fresh Vegetables Bundle",
  "foodType": "Vegetables",
  "quantity": 15,
  "unit": "kg",
  "category": "Vegetables",
  "expiresAt": "2025-04-20T00:00:00Z",
  "location": "123 Main St, Cityville",
  "description": "Assorted fresh vegetables including carrots, broccoli, and bell peppers",
  "pickupTimeStart": "2025-04-15T09:00:00Z",
  "pickupTimeEnd": "2025-04-15T17:00:00Z"
}
```

Response Body Schema:

```json
{
  "id": "don001",
  "title": "Fresh Vegetables Bundle",
  "foodType": "Vegetables",
  "quantity": 15,
  "unit": "kg",
  "category": "Vegetables",
  "expiresAt": "2025-04-20T00:00:00Z",
  "status": "Pending Approval",
  "location": "123 Main St, Cityville",
  "description": "Assorted fresh vegetables including carrots, broccoli, and bell peppers",
  "pickupTimeStart": "2025-04-15T09:00:00Z",
  "pickupTimeEnd": "2025-04-15T17:00:00Z",
  "donorId": "usr112",
  "createdAt": "2025-04-14T15:30:00Z"
}
```

Authentication Requirements: JWT Token Required (Donor role)

Status Codes and Messages:

- `201 Created`: Donation listing created successfully
- `400 Bad Request`: Invalid or missing required fields
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have donor role

- `500 Internal Server Error`: Server error

Description: Creates a new food donation listing. All new donations are initially set to "Pending Approval" status and must be approved by an administrator before they become visible to beneficiaries.

8.2 Get All Donations

Endpoint URL: `/api/donations/list`

HTTP Method: `GET`

Request Headers:
```
Authorization: Bearer {token}
```

Request Parameters:
- `status` (optional): Filter by donation status
- `category` (optional): Filter by food category
- `page` (optional): Page number for pagination
- `limit` (optional): Number of items per page
- `sortBy` (optional): Field to sort by (e.g., 'createdAt', 'expiresAt')
- `sortOrder` (optional): Sort direction ('asc' or 'desc')

Response Body Schema:
```json
{
  "donations": [
    {
      "id": "don001",
      "title": "Fresh Vegetables Bundle",
   "foodType": "Vegetables",
   "quantity": 15,
   "unit": "kg",
   "category": "Vegetables",
   "expiresAt": "2025-04-20T00:00:00Z",
   "status": "Approved",
   "location": "123 Main St, Cityville",
   "donorName": "Green Grocers",
   "createdAt": "2025-04-10T15:30:00Z"
  },
  {
   "id": "don002",
   "title": "Bakery Items",
   "foodType": "Baked Goods",
   "quantity": 10,
   "unit": "items",
   "category": "Baked Goods",
   "expiresAt": "2025-04-16T00:00:00Z",
   "status": "In Transit",
   "location": "456 Flour Ave, Local City",
   "donorName": "Daily Bread",
   "createdAt": "2025-04-12T10:15:00Z"
  }
 ],
 "pagination": {
  "total": 42,
  "page": 1,
  "limit": 10,
  "pages": 5
 }
}
```

```

**Authentication Requirements**: JWT Token Required

**Status Codes and Messages**:

- `200 OK`: Successful request

- `401 Unauthorized`: Invalid or missing token

- `500 Internal Server Error`: Server error

**Description**: Retrieves a list of donation listings. Users with different roles can access different sets of donations: donors see their own donations, beneficiaries see available donations, and admins see all donations.

### 8.3 Get Donation by ID

**Endpoint URL**: `/api/donations/{id}`

**HTTP Method**: `GET`

**Request Headers**:

```
Authorization: Bearer {token}
~~~

Response Body Schema:

```json
{
  "id": "don001",
  "title": "Fresh Vegetables Bundle",
  "foodType": "Vegetables",
  "quantity": 15,
  "unit": "kg",
~~~

```
"category": "Vegetables",
"expiresAt": "2025-04-20T00:00:00Z",
"status": "Approved",
"createdAt": "2025-04-10T15:30:00Z",
"description": "Assorted fresh vegetables including carrots, broccoli, and bell peppers",
"donor": {
  "id": "usr112",
  "name": "Green Grocers",
  "phone": "(555) 123-4567",
  "address": "123 Main St, Cityville"
},
"recipient": {
  "id": "usr456",
  "name": "Community Food Bank",
  "phone": "(555) 987-6543",
  "address": "456 Oak Ave, Townsville"
},
"ngo": {
  "id": "usr789",
  "name": "Food Rescue Initiative",
  "phone": "(555) 456-7890"
},
"logistics": {
  "pickupAddress": "123 Main St, Cityville",
  "dropoffAddress": "456 Oak Ave, Townsville",
  "deliveryType": "NGO Logistics",
  "eta": "2025-04-15T15:30:00Z"
},
"timestamps": {
  "created": "2025-04-10T15:30:00Z",
  "accepted": "2025-04-11T10:30:00Z",
```

    "readyForPickup": "2025-04-14T10:00:00Z",
    "inTransit": "2025-04-14T13:45:00Z",
    "delivered": null,
    "cancelled": null
  },
  "flags": [
    {
      "issue": "Incorrect quantity reported",
      "date": "2025-04-14T08:30:00Z"
    }
  ]
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Successful request
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have permission to view this donation
- `404 Not Found`: Donation not found
- `500 Internal Server Error`: Server error

Description: Retrieves detailed information about a specific donation. The amount of information shown may vary based on the user's role and relationship to the donation.

8.4 Update Donation Status

Endpoint URL: `/api/donations/{id}/status`

HTTP Method: `PUT`
```

**Request Headers**:

```
Content-Type: application/json
Authorization: Bearer {token}
```

Request Body Schema:

```json
{
  "status": "In Transit",
  "notes": "Picked up by delivery driver at 2:30 PM"
}
```

Response Body Schema:

```json
{
  "id": "don001",
  "status": "In Transit",
  "previousStatus": "Ready for Pickup",
  "updatedAt": "2025-04-14T14:30:00Z",
  "updatedBy": "usr789",
  "timestamps": {
    "inTransit": "2025-04-14T14:30:00Z"
  },
  "notes": "Picked up by delivery driver at 2:30 PM"
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Status updated successfully

- `400 Bad Request`: Invalid status transition

- `401 Unauthorized`: Invalid or missing token

- `403 Forbidden`: User doesn't have permission to update this donation

- `404 Not Found`: Donation not found

- `500 Internal Server Error`: Server error

Description: Updates the status of a donation. Different roles can make different status transitions: donors can mark donations as "Ready for Pickup", NGOs can mark donations as "In Transit", beneficiaries can mark donations as "Delivered", and admins can make any status change.

8.5 Get Donor's Donations

Endpoint URL: `/api/donations/donor/{userId}`

HTTP Method: `GET`

Request Headers:

```
Authorization: Bearer {token}
```

Request Parameters:

- `status` (optional): Filter by donation status

- `page` (optional): Page number for pagination

- `limit` (optional): Number of items per page

- `sortBy` (optional): Field to sort by

- `sortOrder` (optional): Sort direction ('asc' or 'desc')

Response Body Schema:

```json
{
 "donations": [
  {
   "id": "don001",
   "title": "Fresh Vegetables Bundle",
   "quantity": "15 kg",
   "category": "Vegetables",
   "expiresAt": "2025-04-20T00:00:00Z",
   "status": "Approved",
   "recipientName": "Community Food Bank",
   "createdAt": "2025-04-10T15:30:00Z"
  }
 ],
 "pagination": {
  "total": 12,
  "page": 1,
  "limit": 10,
  "pages": 2
 },
 "stats": {
  "pending": 2,
  "approved": 5,
  "inTransit": 1,
  "completed": 4
 }
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Successful request

- `401 Unauthorized`: Invalid or missing token

- `403 Forbidden`: User doesn't have permission to view these donations

- `404 Not Found`: User not found

- `500 Internal Server Error`: Server error

Description: Retrieves all donations created by a specific donor. Donors can only access their own donations, while admins can access any donor's donations.

8.6 Claim Donation

Endpoint URL: `/api/donations/claim`

HTTP Method: `POST`

Request Headers:

```
Content-Type: application/json
Authorization: Bearer {token}
```

Request Body Schema:

```json
{
  "donationId": "don001",
  "beneficiaryId": "usr456",
  "notes": "Will pick up ourselves on Wednesday morning."
}
```

Response Body Schema:

```json
{
  "id": "don001",
  "status": "Accepted",
  "previousStatus": "Approved",
  "claimedAt": "2025-04-14T15:30:00Z",
  "beneficiaryId": "usr456",
  "beneficiaryName": "Community Food Bank",
  "message": "Donation successfully claimed"
}
```

Authentication Requirements: JWT Token Required (Beneficiary or NGO role)

Status Codes and Messages:

- `200 OK`: Donation claimed successfully
- `400 Bad Request`: Invalid donation for claiming
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have permission to claim donations
- `404 Not Found`: Donation not found
- `409 Conflict`: Donation already claimed
- `500 Internal Server Error`: Server error

Description: Claims an approved donation, transitioning its status to "Accepted" and associating it with the claiming beneficiary or NGO.

8.7 Get Claimed Donations

Endpoint URL: `/api/donations/claimed/{beneficiaryId}`

HTTP Method: `GET`

Request Headers:
```
Authorization: Bearer {token}
```

Request Parameters:
- `status` (optional): Filter by donation status
- `page` (optional): Page number for pagination
- `limit` (optional): Number of items per page
- `sortBy` (optional): Field to sort by
- `sortOrder` (optional): Sort direction ('asc' or 'desc')

Response Body Schema:
```json
{
  "donations": [
    {
      "id": "don001",
      "title": "Fresh Vegetables Bundle",
      "donorName": "Green Grocers",
      "quantity": "15 kg",
      "category": "Vegetables",
      "expiresAt": "2025-04-20T00:00:00Z",
      "status": "In Transit",
      "location": "123 Main St, Cityville",
      "claimedAt": "2025-04-11T10:30:00Z"
    }
  ],
  "pagination": {
    "total": 8,
    "page": 1,
    "limit": 10,
    "pages": 1
  }
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Successful request
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have permission to view these donations
- `404 Not Found`: User not found
- `500 Internal Server Error`: Server error

Description: Retrieves all donations claimed by a specific beneficiary or NGO. Beneficiaries and NGOs can only access their own claimed donations, while admins can access any user's claimed donations.

8.8 Get Available Donations

Endpoint URL: `/api/donations/available`

HTTP Method: `GET`

Request Headers:

```
Authorization: Bearer {token}
```

Request Parameters:

- `category` (optional): Filter by food category

- `distance` (optional): Maximum distance in kilometers from user's location

- `lat` (optional): User's latitude for distance calculation

- `lng` (optional): User's longitude for distance calculation

- `expiryRange` (optional): Filter by expiration timeframe

- `page` (optional): Page number for pagination

- `limit` (optional): Number of items per page

- `sortBy` (optional): Field to sort by

- `sortOrder` (optional): Sort direction ('asc' or 'desc')

Response Body Schema:

```json
{
  "donations": [
    {
      "id": "don101",
      "title": "Fresh Organic Vegetables",
      "type": "Vegetables",
      "quantity": 8,
      "unit": "kg",
      "expiresAt": "2025-04-18T00:00:00Z",
      "status": "Available",
      "category": "Vegetables",
      "location": "123 Green St, Local City",
      "donorName": "Green Fields Farm",
      "distance": "1.2 km"
    }
  ],
  "pagination": {
```

  "total": 15,

  "page": 1,

  "limit": 10,

  "pages": 2

 }

}

```

**Authentication Requirements**: JWT Token Required (Beneficiary or NGO role)

**Status Codes and Messages**:

- `200 OK`: Successful request

- `401 Unauthorized`: Invalid or missing token

- `403 Forbidden`: User doesn't have appropriate role

- `500 Internal Server Error`: Server error

**Description**: Retrieves all currently available (approved but unclaimed) donations, with options for filtering by various criteria.

### 8.9 Request Donation

**Endpoint URL**: `/api/donations/request`

**HTTP Method**: `POST`

**Request Headers**:

```

Content-Type: application/json

Authorization: Bearer {token}

```
```

Request Body Schema:

```json
{
  "donationId": "don001",
  "beneficiaryId": "usr456",
  "message": "We would like to distribute this food to our community members.",
  "requestType": "pickup"
}
```

Response Body Schema:

```json
{
  "id": "req789",
  "donationId": "don001",
  "beneficiaryId": "usr456",
  "donorId": "usr112",
  "status": "Pending",
  "message": "We would like to distribute this food to our community members.",
  "requestType": "pickup",
  "createdAt": "2025-04-14T15:30:00Z"
}
```

Authentication Requirements: JWT Token Required (Beneficiary or NGO role)

Status Codes and Messages:

- `201 Created`: Request submitted successfully
- `400 Bad Request`: Invalid request data
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have permission to request donations

- `404 Not Found`: Donation not found

- `409 Conflict`: Donation already requested or claimed

- `500 Internal Server Error`: Server error

Description: Submits a request for a donation before actually claiming it, allowing donors to review and approve requests from multiple beneficiaries.

8.10 Search Donations

Endpoint URL: `/api/donations/search`

HTTP Method: `GET`

Request Headers:

```
Authorization: Bearer {token}
```

Request Parameters:

- `query` (required): Search term for donation title, description, or donor name
- `category` (optional): Filter by food category
- `status` (optional): Filter by donation status
- `locationName` (optional): Location name for proximity search
- `lat` (optional): Latitude for proximity search
- `lng` (optional): Longitude for proximity search
- `radius` (optional): Search radius in kilometers
- `fromDate` (optional): Start date for creation date filter
- `toDate` (optional): End date for creation date filter
- `page` (optional): Page number for pagination
- `limit` (optional): Number of items per page

Response Body Schema:

```json
{
 "donations": [
  {
   "id": "don101",
   "title": "Fresh Organic Vegetables",
   "type": "Vegetables",
   "quantity": 8,
   "unit": "kg",
   "expiresAt": "2025-04-18T00:00:00Z",
   "status": "Available",
   "category": "Vegetables",
   "location": "123 Green St, Local City",
   "donorName": "Green Fields Farm",
   "distance": "1.2 km",
   "match": 0.85
  }
 ],
 "pagination": {
  "total": 3,
  "page": 1,
  "limit": 10,
  "pages": 1
 }
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Successful request

- `400 Bad Request`: Invalid search parameters

- `401 Unauthorized`: Invalid or missing token

- `500 Internal Server Error`: Server error

Description: Performs a comprehensive search across donations with multiple filtering criteria. The search results are ranked by relevance to the query.

8.11 Delete Donation

Endpoint URL: `/api/donations/{id}`

HTTP Method: `DELETE`

Request Headers:

```
Authorization: Bearer {token}
```

Response Body Schema:

```json
{
  "id": "don001",
  "message": "Donation deleted successfully"
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Donation deleted successfully

},

- `401 Unauthorized`: Invalid or missing token

- `403 Forbidden`: User doesn't have permission to delete this donation

- `404 Not Found`: Donation not found

- `500 Internal Server Error`: Server error

Description: Deletes a donation listing. Donors can only delete their own donations that haven't been claimed. Admins can delete any donation.

8.12 Get Donation Statistics

Endpoint URL: `/api/donations/stats/{userId}`

HTTP Method: `GET`

Request Headers:
```
Authorization: Bearer {token}
```

Response Body Schema:
```json
{
 "totalDonations": 27,
 "byStatus": {
  "pending": 3,
  "approved": 2,
  "inTransit": 1,
  "completed": 21
 },
 "byCategory": {
  "Vegetables": 8,
    "Fruits": 5,
    "Baked Goods": 7,
    "Canned Goods": 4,
    "Dairy": 3
  },
  "quantityDonated": 135.5,
  "estimatedServings": 450,
  "carbonFootprintSaved": 215.2,
  "timeline": [
    { "month": "Jan", "count": 3, "quantity": 15.2 },
    { "month": "Feb", "count": 4, "quantity": 20.1 },
    { "month": "Mar", "count": 5, "quantity": 25.8 },
    { "month": "Apr", "count": 6, "quantity": 30.3 }
  ],
  "activeDonations": 6,
  "pendingApprovals": 3
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Successful request
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have permission to view these statistics
- `404 Not Found`: User not found
- `500 Internal Server Error`: Server error

Description: Retrieves donation-related statistics for a specific user. The statistics returned vary based on user role: donors see statistics about their donations, beneficiaries see statistics about received donations.

This completes the documentation for the Donation APIs in the FoodShare application. These APIs provide comprehensive functionality for creating, managing, searching, and tracking food donations throughout their lifecycle.