Let me continue with documenting the Review APIs for the FoodShare application based on the code analysis.

FoodShare API Documentation - Review APIs

6. Review APIs

Based on my analysis of the codebase, especially the `ReviewCard` component, `DonationDetailsPage`, and `DonationTrackingPage`, I'll document the Review APIs available in the FoodShare application.

6.1 Submit Review

Endpoint URL: `/api/donations/{donationId}/review`

HTTP Method: `POST`

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

Request Body Schema:
```json
{
  "rating": 5,
  "comment": "The food quality was excellent and the pickup process was smooth.",
  "reviewType": "donor-to-beneficiary"
}
```

Response Body Schema:

```json
{
  "id": "rev123",
  "donationId": "don456",
  "reviewerId": "u789",
  "reviewerName": "John Smith",
  "reviewerRole": "Beneficiary",
  "recipientId": "u012",
  "recipientName": "Fresh Grocers",
  "rating": 5,
  "comment": "The food quality was excellent and the pickup process was smooth.",
  "reviewType": "donor-to-beneficiary",
  "status": "Active",
  "createdAt": "2025-04-14T15:30:00Z"
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:
- `201 Created`: Review submitted successfully
- `400 Bad Request`: Invalid review data
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have permission to review this donation
- `404 Not Found`: Donation not found
- `409 Conflict`: User has already submitted a review for this donation
- `500 Internal Server Error`: Server error

Description: Submits a review for a completed donation. Either the donor can review the beneficiary/NGO, or the beneficiary/NGO can review the donor. The user must have been a participant in the donation transaction to submit a review.

6.2 Get Donation Reviews

Endpoint URL: `/api/donations/{donationId}/reviews`

HTTP Method: `GET`

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

Response Body Schema:
```json
[
  {
    "id": "rev123",
    "donationId": "don456",
    "reviewerId": "u789",
    "reviewerName": "John Smith",
    "reviewerRole": "Beneficiary",
    "rating": 5,
    "comment": "The food quality was excellent and the pickup process was smooth.",
    "reviewType": "beneficiary-to-donor",
    "status": "Active",
    "createdAt": "2025-04-14T15:30:00Z"
  },
  {
    "id": "rev124",
    "donationId": "don456",
    "reviewerId": "u012",
```

  "reviewerName": "Fresh Grocers",
  "reviewerRole": "Donor",
  "rating": 4,
  "comment": "Pickup was on time, very professional interaction.",
  "reviewType": "donor-to-beneficiary",
  "status": "Active",
  "createdAt": "2025-04-15T09:15:00Z"
 }
]
```

**Authentication Requirements**: JWT Token Required

**Status Codes and Messages**:
- `200 OK`: Successful request
- `401 Unauthorized`: Invalid or missing token
- `404 Not Found`: Donation not found
- `500 Internal Server Error`: Server error

**Description**: Retrieves all reviews associated with a specific donation.

### 6.3 Get User Reviews

**Endpoint URL**: `/api/users/{userId}/reviews`

**HTTP Method**: `GET`

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

Request Parameters:

- `role` (optional): Filter by review role ('reviewer' or 'recipient')

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

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

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

Response Body Schema:

```json
{
 "reviews": [
  {
   "id": "rev123",
   "donationId": "don456",
   "donationTitle": "Fresh Vegetables Bundle",
   "reviewerId": "u789",
   "reviewerName": "John Smith",
   "recipientId": "u012",
   "recipientName": "Fresh Grocers",
   "rating": 5,
   "comment": "The food quality was excellent and the pickup process was smooth.",
   "reviewType": "beneficiary-to-donor",
   "status": "Active",
   "createdAt": "2025-04-14T15:30:00Z"
  }
 ],
 "pagination": {
  "total": 12,
  "page": 1,
  "limit": 10,
  "pages": 2
```

 }

}

```

**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 reviews

- `404 Not Found`: User not found

- `500 Internal Server Error`: Server error

**Description**: Retrieves all reviews either written by or about a specific user, depending on the `role` parameter.

### 6.4 Update Review

**Endpoint URL**: `/api/reviews/{reviewId}`

**HTTP Method**: `PUT`

**Request Headers**:

```

Content-Type: application/json

Authorization: Bearer {token}

~~~

Request Body Schema:

```json

{
~~~

  "rating": 4,

  "comment": "Updated comment after further consideration."

}

**Response Body Schema**:

```json

{

  "id": "rev123",

  "donationId": "don456",

  "reviewerId": "u789",

  "rating": 4,

  "comment": "Updated comment after further consideration.",

  "updatedAt": "2025-04-16T10:20:30Z",

  "status": "Active"

}

```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Review updated successfully

- `400 Bad Request`: Invalid review data

- `401 Unauthorized`: Invalid or missing token

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

- `404 Not Found`: Review not found

- `500 Internal Server Error`: Server error

Description: Updates an existing review. Users can only update their own reviews, and typically within a limited timeframe after submission (e.g., 24 hours).

6.5 Delete Review

Endpoint URL: `/api/reviews/{reviewId}`

HTTP Method: `DELETE`

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

Response Body Schema:
```json
{
  "id": "rev123",
  "message": "Review successfully deleted"
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:
- `200 OK`: Review deleted successfully
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have permission to delete this review
- `404 Not Found`: Review not found
- `500 Internal Server Error`: Server error

Description: Deletes a review. Users can only delete their own reviews, and typically within a limited timeframe after submission.

6.6 Report Review

Endpoint URL: `/api/reviews/{reviewId}/report`

HTTP Method: `POST`

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

Request Body Schema:
```json
{
  "reason": "inappropriate",
  "details": "This review contains offensive language."
}
```

Response Body Schema:
```json
{
  "id": "rep456",
  "reviewId": "rev123",
  "reporterId": "u789",
  "reason": "inappropriate",
  "details": "This review contains offensive language.",
  "status": "Pending",
  "createdAt": "2025-04-14T15:30:00Z"
}
```

```

**Authentication Requirements**: JWT Token Required

**Status Codes and Messages**:

- `201 Created`: Report submitted successfully

- `400 Bad Request`: Invalid report data

- `401 Unauthorized`: Invalid or missing token

- `404 Not Found`: Review not found

- `500 Internal Server Error`: Server error

**Description**: Reports a review for moderation. This flags the review for administrator review without immediately removing it from view.

### 6.7 Moderate Review (Admin)

**Endpoint URL**: `/api/admin/reviews/{reviewId}/moderate`

**HTTP Method**: `POST`

**Request Headers**:

```
Content-Type: application/json

Authorization: Bearer {token}
~~~

Request Body Schema:

```json
{

  "action": "approve",

  "notes": "Review meets community guidelines."
~~~

}

**Response Body Schema**:
```json
{
  "id": "rev123",
  "status": "Approved",
  "moderatedAt": "2025-04-14T16:00:00Z",
  "moderatorId": "u001",
  "moderatorName": "Admin User",
  "notes": "Review meets community guidelines."
}
```

Authentication Requirements: JWT Token Required (Admin role only)

Status Codes and Messages:
- `200 OK`: Review moderated successfully
- `400 Bad Request`: Invalid moderation action
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have admin role
- `404 Not Found`: Review not found
- `500 Internal Server Error`: Server error

Description: Allows administrators to moderate a review - approving it, flagging it, or removing it from the system.

6.8 Get Reported Reviews (Admin)

Endpoint URL: `/api/admin/reviews/reported`

    },

HTTP Method: `GET`

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

Request Parameters:
- `status` (optional): Filter by report status ('pending', 'resolved')
- `page` (optional): Page number for pagination
- `limit` (optional): Number of reports per page

Response Body Schema:
```json
{
  "reports": [
    {
      "id": "rep456",
      "reviewId": "rev123",
      "review": {
        "id": "rev123",
        "donationId": "don456",
        "reviewerId": "u789",
        "reviewerName": "John Smith",
        "rating": 2,
        "comment": "The food was of poor quality.",
        "status": "Flagged"
      },
      "reporterId": "u012",
      "reporterName": "Fresh Grocers",
      "reason": "disputed",
      "details": "This review is inaccurate. The food was freshly prepared that morning.",
      "status": "Pending",
      "createdAt": "2025-04-14T15:30:00Z"
    }
  ],
  "pagination": {
    "total": 5,
    "page": 1,
    "limit": 10,
    "pages": 1
  }
}
```

Authentication Requirements: JWT Token Required (Admin role only)

Status Codes and Messages:

- `200 OK`: Successful request
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have admin role
- `500 Internal Server Error`: Server error

Description: Retrieves a list of reviews that have been reported, along with the report details, for administrator moderation.

7. Statistics APIs

Based on the codebase analysis, particularly the donation service and admin dashboard components, I'll document the Statistics APIs for the FoodShare application.

7.1 Get User Statistics

}

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

HTTP Method: `GET`

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

Response Body Schema:
```json
{
 "donations": {
  "total": 27,
  "active": 3,
  "completed": 24
 },
 "received": {
  "total": 12,
  "active": 1,
  "completed": 11
 },
 "impact": {
  "foodSaved": 135,
  "unit": "kg",
  "mealsProvided": 450,
  "impactPoints": 2350
 },
 "ratings": {
  "average": 4.8,
    "count": 25
  },
  "activity": {
    "firstActivity": "2025-01-15T10:20:30Z",
    "lastActivity": "2025-04-12T14:30:00Z",
    "daysActive": 88
  }
}
```

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 comprehensive statistics for a specific user's activity on the platform, including donation metrics, impact metrics, and ratings.

7.2 Get Donation Statistics

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

HTTP Method: `GET`

Request Headers:

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

~~~

Response Body Schema:

```json
{
  "views": 42,
  "requests": 3,
  "timeToMatch": {
    "hours": 4,
    "minutes": 15
  },
  "ratings": {
    "donorRating": 5,
    "beneficiaryRating": 4,
    "averageRating": 4.5
  },
  "impact": {
    "foodSaved": 15,
    "unit": "kg",
    "mealsProvided": 50
  },
  "timeline": [
    {
      "status": "Created",
      "timestamp": "2025-04-10T09:00:00Z"
    },
    {
      "status": "Approved",
      "timestamp": "2025-04-10T10:30:00Z"
    },
    {
~~~

      "status": "Matched",
      "timestamp": "2025-04-10T13:15:00Z"
    },
    {
      "status": "Completed",
      "timestamp": "2025-04-10T16:45:00Z"
    }
  ]
}
```

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 these statistics
- `404 Not Found`: Donation not found
- `500 Internal Server Error`: Server error

Description: Retrieves detailed statistics about a specific donation, including metrics on how quickly it was matched and its impact.

7.3 Get Platform Statistics (Admin)

Endpoint URL: `/api/admin/stats`

HTTP Method: `GET`

Request Headers:
```

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

Request Parameters:
- `period` (optional): Time period for metrics ('day', 'week', 'month', 'year', 'all')
- `startDate` (optional): Start date for custom period
- `endDate` (optional): End date for custom period

Response Body Schema:
```json
{
 "users": {
  "total": 248,
  "donors": 112,
  "beneficiaries": 96,
  "ngos": 40,
  "growth": {
   "rate": 8.3,
   "trend": [65, 59, 80, 81, 56, 55, 40]
  }
 },
 "donations": {
  "total": 437,
  "active": 53,
  "completed": 384,
  "growth": {
   "rate": 12.5,
   "trend": [45, 52, 49, 60, 55, 58, 62]
  }
 },
 "impact": {
  "foodSaved": 2750,
  "unit": "kg",
  "mealsProvided": 9167,
  "growth": {
   "rate": 15.2,
   "trend": [200, 250, 280, 310, 330, 350, 400]
  }
 },
 "system": {
  "pendingApprovals": 8,
  "pendingReports": 3,
  "averageResponseTime": "4.2 hours"
 }
}
```

Authentication Requirements: JWT Token Required (Admin role only)

Status Codes and Messages:

- `200 OK`: Successful request
- `400 Bad Request`: Invalid date range
- `401 Unauthorized`: Invalid or missing token
- `403 Forbidden`: User doesn't have admin role
- `500 Internal Server Error`: Server error

Description: Retrieves comprehensive platform-wide statistics for administrative purposes, including user metrics, donation metrics, and overall impact.

7.4 Get Impact Report

Endpoint URL: `/api/impact-report`

HTTP Method: `GET`

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

Request Parameters:
- `userId` (optional): Filter report by specific user
- `timeframe` (optional): Time period for report ('month', 'quarter', 'year')
- `format` (optional): Response format ('json', 'pdf', 'csv')

Response Body Schema (JSON format):
```json
{
 "user": {
  "id": "u789",
  "name": "John Smith",
  "role": "Donor",
  "joinedDate": "2025-01-15T10:20:30Z"
 },
 "timeframe": {
  "period": "quarter",
  "start": "2025-01-01T00:00:00Z",
  "end": "2025-03-31T23:59:59Z"
 },
 "summary": {
  "donations": 12,
  "foodSaved": 60,
  "unit": "kg",
  "mealsProvided": 200,
  "carbonOffset": 250,
  "unit": "kg CO2e"
},
"comparison": {
  "percentile": 85,
  "communityAverage": {
    "donations": 8,
    "foodSaved": 40
  }
},
"badges": [
  {
    "name": "Food Rescue Hero",
    "description": "Saved over 50kg of food in a quarter",
    "earnedAt": "2025-03-15T14:20:00Z"
  }
],
"monthlyBreakdown": [
  {
    "month": "January",
    "donations": 3,
    "foodSaved": 15
  },
  {
    "month": "February",
    "donations": 4,
    "foodSaved": 20
  },
  {
    "month": "March",
    "donations": 5,
    "foodSaved": 25
  }
 ]
}
```

Authentication Requirements: JWT Token Required

Status Codes and Messages:

- `200 OK`: Successful request
- `400 Bad Request`: Invalid parameters
- `401 Unauthorized`: Invalid or missing token
- `404 Not Found`: User not found
- `500 Internal Server Error`: Server error

Description: Generates a detailed impact report showing the environmental and social impact of a user's activities on the platform. The report can be filtered by timeframe and delivered in various formats.

This completes the documentation for the Review and Statistics APIs in the FoodShare application.