Users

POST /api/users/
Description: Registers a new user.
Body: { "name": string, "netid": string }
Response: 201 Created with user JSON object. {“id”: <ID>, “name”:string, “netid”:string, “rating”: float, “goods”: [ ], “transactions”: [ ] }

GET /api/users/{id}/
Description: Retrieves a user's public profile information.
Path Parameters: id - The ID of the user.
Response: 200 OK with user JSON object. {“id”: <ID>, “name”:string, “netid”:string, “rating”: float, “goods”: [<SERIALIZED GOOD WITHOUT USER FIELD>] }

GET /api/users/{id}/transactions/
Description: Retrieves the transaction history for the user (private).
Path Parameters: id - The ID of the user.
Response: 200 OK with a list of transaction JSON objects. {“transactions”: [<SERIALIZED TRANSACTION WITHOUT USER FIELD>] }

DELETE /api/users/{id}/
Description: Deletes a user's account and associated data.
Path Parameters: id - The ID of the user to delete.
Response: 200 OK if the deletion is successful. {“id”: <ID>, “name”:string, “netid”:string, “rating”: float, “goods”: [<SERIALIZED GOOD WITHOUT USER FIELD>], “transactions”: [<SERIALIZED TRANSACTION WITHOUT USER FIELD>] }

PATCH /api/users/{id}/
Description: Updates the specified user's details.
Path Parameters: id - The ID of the user to update.
Body: JSON object containing the fields to update (e.g., { "name": string, "netid": string }).
Response: 200 OK with the updated user JSON object. {“id”: <ID>, “name”:string, “netid”:string, “rating”: float, “goods”: [<SERIALIZED GOOD WITHOUT USER FIELD>], “transactions”: [<SERIALIZED TRANSACTION WITHOUT USER FIELD>] }

Goods

POST /api/goods/
Description: Creates a new good listing.
Body: { "good_name": string, "images": array of strings (URLs), "price": int }
Response: 201 Created with good JSON object. {“id”: <ID>, “good_name”:string, “images”: array of strings (URLs), "price": int}

GET /api/goods/{id}/
Description: Retrieves information about a specific good.
Path Parameters: id - The ID of the good.
Response: 200 OK with good JSON object. {“id”: <ID>, “good_name”:string, “images”: array of strings (URLs), "price": int, “seller”: [<PUBLIC SERIALIZE OF USER>]}

GET /api/goods/
Description: Retrieves a list of all goods available in the marketplace.
Response: 200 OK with an array of good JSON objects. {“goods”: [{“id”: <ID>, “good_name”:string, “images”: array of strings (URLs), "price": int}] }

DELETE /api/goods/{id}/
Description: Deletes a specific good from the marketplace.
Path Parameters: id - The ID of the good to delete.
Response: 200 OK if the deletion is successful. {“id”: <ID>, “good_name”:string, “images”: array of strings (URLs), "price": int, “seller”: [<PUBLIC SERIALIZE OF USER>]}

PATCH /api/goods/{id}/
Description: Updates the specified good's details.
Path Parameters: id - The ID of the good to update.
Body: JSON object containing the fields to update (e.g., { "good_name": string, "price": int, “sold”: boolean}).
Response: 200 OK with the updated good JSON object. {“id”: <ID>, “good_name”:string, “images”: array of strings (URLs), "price": int, “seller”: [<PUBLIC SERIALIZE OF USER>]}

Transactions

POST /api/transactions/
Description: Creates a new transaction record.
Body: { "buyer_id": int, "seller_id": int, "amount": int, "timestamp": db.DateTime }
Response: 201 Created with transaction JSON object.

Ratings

POST /api/rating/{transaction_id}/
Description: Submits a rating for a seller for a specific transaction.
Body: { "value": int (1 to 5), "transaction_id": int }
Response: 201 Created with rating JSON object.

User model:

- Represents a user in the system with a unique ID, name, netid, and a rating
- Has a one-to-many relationship with Good, indicating the goods associated with the user
- Participates in two many-to-many relationships with Transaction through buyers_table and sellers_table, representing the transactions where the user is either a buyer or a seller

Good model:

- Represents an item or good listed in the system with an ID, name, images (urls stored as JSON text), and a price (in cents)
- References the User model (one-to-many) to identify the seller of the good
- Has a one-to-one relationship with Transaction, tracking the corresponding transaction (once completed)

Transaction model:

- Represents a transaction in the system with a unique ID, the amount of the transaction, and a timestamp
- Has two many-to-many relationships with the User model to track the buyer and seller involved in the transaction
- Has a one-to-one relationship with the Good model to track the item sold in the transaction
- Has a one-to-one relationship with Rating identifying the buyer's rating

Rating model:

- Represents a rating given to a transaction with a unique ID and a value (out of 5)
- Has a one-to-one relationship with the Transaction model, linking a rating to its respective transaction