Replecon REST API Documentation

All returns are in JSON format.

Bank

banks/
	GET: Returns a list of all the bank objects
	POST: Creates a new bank object

banks/<int:id>
	GET: Returns the bank object with the specific id
	PUT: Updates the bank object with the specific id
	DELETE: Deletes the bank object with the specific id

bank object

{
	id: 1,
	class_code: ‘123456’,
	intrest_rate: 12,
	payout_rate: 1,
}

transactionintrestrates/
	GET: Returns all the transaction interest rate objects
	POST: Creates a new transaction interest rate object

transactionintrestrates/<int:id>
	GET: Returns the transaction interest rate object with the specific id
	PUT: Updates the transaction interest rate object with the specific id
	DELETE: Deletes the transaction interest rate object with the specific id

transactionintrestrates object

{
	id: 1,
	set_intrest_rate: 12.6,
	transaction_id: 2,
	active: true,
	end_date: 30/4/21,
}

Classroom

classrooms/

GET: Returns a list of all the classroom objects
POST: Creates a new classroom object

classrooms/<int:id>

GET: Returns the classroom object with the specific id
PUT: Updates the classroom object with the specific id
DELETE: Deletes the classroom object with the specific id

classroom object

```
{
    id: 1,
    class_name: ‘my class’,
    teacher_id: 2,
    class_code: ‘123456’,
}
```

Store

shops/

GET: Returns a list of all shop objects
POST: Creates a new shop

shops/<int:id>

GET: Returns a details of shop with the specific id
PUT: Updates the data of the shop with the specific id
DELETE: Deletes the object of shop with the specific id

shops object

```
{
    id: 1,
    shop_name: ‘shop name’,
    class_code: ‘123456’,
}
```

items/

GET: Returns a list of all the item shops
POST: Creates new a item object

items/<int:id>

GET: Returns the item with the specific id
PUT: Updates the data of the item with the specific id
DELETE: Deletes the item with the specific id

item object

```
{
    id: 1,
    item_name: “Water”,
    description: “drink some water”,
    price: 12.99,
    shop_id: 1,
}
```

Tax

taxes/

GET: Returns a list of all the taxes objects
POST: Creates a new tax object

taxes/<int:id>

GET: Returns the tax object with the specific id
PUT: Updates the tax object with the specific id
DELETE: Deletes the tax object with the specified id

tax object

```
{
    id: 1,
    class_code: ‘123456’,
    sales_tax: 12,
    percentage_tax: 15.5,
    flat_tax: 100,
}
```

progressivebrackets/

GET: Returns a list of all the progressive brackets objects
POST: Creates a new progressive bracket object

progressivebrackets/<int:id>

GET: Returns the progressive bracket object with the specific id
PUT: Updates the progressive bracket object with the specific id
DELETE: Deletes the progressive bracket object with the specific id

progressivebracket object

{
    id: 1,
    tax_id: 1,
    lower_bracket: 1,
    higher_bracket:100,
    percentage: 12,
{

regressivebrackets/
    GET: Returns a list of all the regressive brackets objects
    POST: Creates a new regressive bracket object

regressivebrackets/<int:id>
    GET: Returns the regressive bracket object with the specific id
    PUT: Updates the regressive bracket object with the specific id
    DELETE: Deletes the regressive bracket object with the specific id

regressivebracket object

{
    id: 1,
    tax_id: 1,
    lower_bracket: 1,
    higher_bracket:100,
    percentage: 20,
}

Transaction

transactions/
    GET: Returns a list of all transactions in the database.
    POST: Creates a new transaction between two users, requires recipient_id, sender_id, category, and amount as request.

transactions/<int:id>
    GET: Returns transaction with specified id.

transactions/getAllStudentTransactions/
    GET: Returns a list of all transactions that a specified user_id is a part of.

transactions/buyFromStore/

POST: Creates a new transaction between the logged in user and the STORE user, requires an amount as request.

transactions/teacherPayStudents/

POST: Creates a new transaction between the logged in teacher and a specified student, requires a user_id and amount as request.

transactions/banksavings/

POST: Creates a new transaction for a student's savings with a bank, requires an amount, and done as request.

transaction object

```
{
    id: 1,
    recipient_id: 2,
    sender_id: 5,
    category: 'banking',
    amount: 145,
}
```

Users

auth/register/

POST: Creates a new user in auth_user table, requires username, password, and first_name as request.

auth/logout/

GET: Logs out the user by deleting their auth_token.

users/

GET: Returns list of all users.

users/<int:id>

GET: Returns user with id.

user object

{
    id: 1,
    username: “user@gmail.com”,
    password: “password”,
    first_name: “User”,
}

students/
    GET: Returns an alphabetically sorted list of all students in the database.

students/create/
    POST: Creates a new student object from a base user and returns it, requires a class_code as request.

students/class_code/
    GET: Returns a sorted list of all students with the same class_code as the logged in student

students/balance/
    GET: Returns the logged in student’s current balance
    PUT: If the logged in user is a student it updates their balance, requires user_id, amount, and recipient as request. If the logged in user is a teacher it updates a specified student’s balance, requires user_id and amount.

students/current/
    GET: Returns the logged in student.

students/store/
    GET: Returns the STORE account.

students/bank/
    GET: Returns the BANK account.

student object

{
    user: 1,
    balance: “53.50”,
    class_code: “HJKLYU”,
}

teachers/

GET: Returns all teachers in database.

teachers/create/

POST: Creates a new teacher object from a base user and returns it, requires a last_name as request.

teachers/isTeacher/

GET: Checks if logged in user is a teacher, if so it returns true, otherwise it returns false.

teacher object

```
{
    user: 3,
    last_name: “Smith”,
}
```

setup/

POST: Used to setup the app with required BANK and STORE users.