## Kirana Store Backend

The Kirana Store Backend facilitates the management of transaction registers for small businesses. It allows tracking of credit and debit transactions across multiple currencies with financial reports, user authentication, API rate limiting, and caching features.

### Transaction API

#### POST /api/transactions

Description:  
This endpoint is used to record financial transactions (credit or debit) for a Kirana store. The transaction is posted in one currency and is automatically converted into another currency based on real-time exchange rates.

* URL: http://localhost:8080/api/transactions
* Method: POST
* Authorization: Bearer Token ({{JWT}})

Request Body:

json

Copy code

{

"amount": 500,

"originalCurrency": "USD",

"targetCurrency": "INR",

"transactionType": "debit"

}

Fields:

* amount (double): The amount of money for the transaction.
* originalCurrency (string): The currency code of the original transaction (e.g., "USD").
* targetCurrency (string): The currency code to convert the transaction to (e.g., "INR").
* transactionType (string): Type of transaction; either "credit" or "debit".

Response Example:

json

Copy code

{

"id": "66f2ba231a72fc177cb21b6f",

"amount": 1000.0,

"originalCurrency": "USD",

"targetCurrency": "INR",

"convertedAmount": 83573.6653812,

"timeStamp": 1727183395111,

"transactionType": "credit"

}

Response Fields:

* id (string): Unique identifier for the transaction.
* amount (double): The original amount of the transaction.
* originalCurrency (string): The original currency of the transaction.
* targetCurrency (string): The target currency to which the original amount was converted.
* convertedAmount (double): The amount after conversion to the target currency.
* timeStamp (long): The timestamp of when the transaction was recorded.
* transactionType (string): Indicates whether the transaction was a credit or debit.

### Report API

#### GET /api/reports/weekly

Description:  
Fetches a report that summarizes all the financial transactions for the week, grouped by the provided currency.

* URL: http://localhost:8080/api/reports/weekly?currency=USD
* Method: GET
* Authorization: Bearer Token ({{JWT}})

Query Parameters:

* currency (string): Currency code for which to generate the report (e.g., "USD").

Response Example:

json

Copy code

{

"id": "66f2ba3e1a72fc177cb21b70",

"reportType": "weekly",

"totalCredits": 3000.0,

"totalDebits": 2200.0,

"netFlow": 800.0,

"currency": "USD",

"timeStamp": 1727183422169

}

Response Fields:

* id (string): Unique identifier for the report.
* reportType (string): Type of the report, in this case, "weekly".
* totalCredits (double): Total credit transactions for the week.
* totalDebits (double): Total debit transactions for the week.
* netFlow (double): The difference between credits and debits.
* currency (string): The currency code for the report.
* timeStamp (long): The timestamp when the report was generated.

#### GET /api/reports/monthly

Description:  
Generates a monthly report, summarizing credits, debits, and net flow for the given currency.

* URL: http://localhost:8080/api/reports/monthly?currency=USD
* Method: GET
* Authorization: Bearer Token ({{JWT}})

Query Parameters:

* currency (string): Currency code for the report (e.g., "USD").

Response Example:

json

Copy code

{

"id": "66f2ba501a72fc177cb21b71",

"reportType": "monthly",

"totalCredits": 3000.0,

"totalDebits": 2200.0,

"netFlow": 800.0,

"currency": "USD",

"timeStamp": 1727183440986

}

Response Fields:

* id (string): Unique identifier for the report.
* reportType (string): Type of the report, in this case, "monthly".
* totalCredits (double): Total credit transactions for the month.
* totalDebits (double): Total debit transactions for the month.
* netFlow (double): The difference between credits and debits.
* currency (string): The currency for the report.
* timeStamp (long): The timestamp when the report was generated.

### Authentication API

#### POST /api/auth/register

Description:  
This endpoint allows new users to register by providing a username, password, and role. The system returns the encrypted password.

* URL: http://localhost:8080/api/auth/register
* Method: POST

Request Body:

json

Copy code

{

"username": "example619",

"password": "testpassword",

"role": "ROLE\_READ\_WRITE"

}

Fields:

* username (string): The username of the user to register.
* password (string): The password for the user.
* role (string): The role assigned to the user. Possible values are "ROLE\_READ\_ONLY" or "ROLE\_READ\_WRITE".

Response Example:

json

Copy code

{

"id": "66f2b8fa1a72fc177cb21b6b",

"username": "exampleuser",

"password": "$2a$10$waRupbb/8LX39yrk2FKcAekF.KPmLBRd5cx7k5dKI/Booi4R/RD/G",

"role": "ROLE\_READ\_WRITE"

}

Response Fields:

* id (string): Unique identifier for the user.
* username (string): The username of the registered user.
* password (string): The encrypted password of the user.
* role (string): The role assigned to the user.

#### POST /api/auth/login

Description:  
Logs a user into the system by verifying the username and password and returns a JWT token if successful.

* URL: http://localhost:8080/api/auth/login
* Method: POST
* Authorization: Bearer Token ({{JWT}})

Request Body:

json

Copy code

{

"username": "example619",

"password": "testpassword"

}

Fields:

* username (string): The username of the user.
* password (string): The password for the user.

Response Example:

json

Copy code

{

"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJleGFtcGxldXNlciIsImlhdCI6MTcyNzE4MzE2NiwiZXhwIjoxNzI3MjY5NTY2LCJhdXRob3JpdGllcyI6IlJPTEVfUkVBRF9XUklURSJ9.3MRJ38W\_tc2N1GpN6bGwnB1BQivhGCJ9btCfQtXutcguUDr1bcUOoQNP6ua5o3OMcFJVnCVMmYBxnkxYQB78YQ"

}

Response Fields:

* token (string): The JWT token for authenticated API access.