MoneyPortal API Documentation

This document is to give a brief overview about the internal working of the application APIs.

About Application

MoneyPortal is a **financial web application** complete with **bank accounts**, **budgets** & **graphing**. It utilities a **role-based** security system to create permission-based roles. The role of **Owner**, **Member**, and **Personal** restricts what the user can see and do.

MoneyPortal: Service API

Bank accounts

POST - BankAccounts/AddAccount

```
.post('BankAccounts/AddAccount', (req, res) => {
  /*
  * Reads upcoming data from the request body and adds a new bank account to
   a user with their Email in the database.
  */
}
```

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
name	required	**The account name.**	string
account type	required	**The bank account type.**	String
starting Balance	required	**The starting balance of the account. Will also be set as the current balance.**	number
low balance	required	**The low balance alert threshold.**	number
email	required	**The email of the user the account is being added to.**	string

`GET` - BankAccounts/AccountDetails

```
.get('BankAccounts/AccountDetails', (req, res) => {
  /*
  * Extracts the current account Id with (req.params.Id), saves it in a
    variable. Based on this Id finds the details of the bank account
    along with the total transactions and account statistics.
  */
}
```

Response Class (Status 200)

Example Values - *Account Details*

```
{
  _id: ObjectId,
  email: 'String',
  name: 'String',
  accountType: 'String',
  CurrentBalance: 0,
  LowBalance: 0,
  HouseholdId: null,
  FirstName: 'String',
  LastName: 'String',
}
```

Example Values - *Transaction Details*

```
[
  {
    _id: ObjectId,
    BankAccountId: String,
    Amount: 0,
    TransactionType: String,
    Memo: String,
    CategoryId: null,
    CategoryItemId: null,
    Created: { createMoment: [Object] },
  }
]
```

Example Values - *Account Statistics*

```
{
  Deposits: 0,
  Spending: 0,
  Transactions: 0
}
```

DELETE - BankAccounts/Delete/:accountId

```
.get('BankAccounts/Delete/:accountId', (req, res) => {
  /*
  * This takes a accountId as parameter and based on that Id deletes a
    specific user bank account along with all the transactions belongings
    from the database permanently.
  */
}
```

Implementation Notes

This will delete a specific bank account.

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
accountId	required	**Id of the account.**	ObjectId

POST - BankAccounts/Edit/:accountId

```
.get('BankAccounts/Edit/:accountId', (req, res) => {
  /*
  * This takes a accountId as parameter and request data by the user which
    has to updated. Based on the Id saves the new bank account data inside
    the database.
  */
}
```

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
name	required	**The account name.**	string
account type	required	**The bank account type.**	String
low balance	required	**The low balance alert threshold.**	number
accountId	required	**Id of the account.**	ObjectId

Budgets

`POST` - Budgets/CreateBudget

```
.get('Budgets/CreateBudget', (req, res) => {
  /*
  * Reads upcoming data from the request body  and adds a new budget category
    to a household and saves it in the database.
  */
}
```

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
name	required	**The Budget category name.**	string
description	required	**A description of the Budget category.**	string
target amount	required	**The target amount of your Budget.**	number
household	required	**The ID number of the household receiving the new Budget.**	ObjectId

`POST` - Budgets/CreateBudgetItem

```
.get('Budgets/CreateBudgetItem', (req, res) => {
  /*
  * Extracts the BudgetId from the request body so, we could know that for
    which specific budget category user is trying to make this budget item.
    Further we are save this budget item, inside our Budget Model we have an
    field of type Array with the "budgetItems".
  */
}
   /*
    Note: Before we were storing the budget items into a different collection
    but because the application requirement was to constantly fetch the
    budget item it became difficult to find each budget item for specific
    categories. So, to decrease the complexity of the program we are now
    storing the budget item inside an array field which is a part of Budget
    category for easy retrival.
    */

  //Inside Budget Category Model
  budgetItems: {
    type: mongoose.Schema.Types.Array,
  }
```

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
name	required	**The Budget Item name.**	string
item Id	required	**Id of the Item.**	ObjectId
budget Id	required	**Id of the Budget Category.**	ObjectId

`Post` - Budgets/RemoveItem

```
.get('Budgets/RemoveItem', (req, res) => {
  /*
  * Reads BudgetId & BudgetItemId from request body and performs searching
    on the basis of BudgetId in the database, once the budget category is
    founded then looks for the specific Item inside the budgetItem array and
    removes the item from the database.
  */
}
```

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
item Id	required	**Id of the Item**	ObjectId
budget Id	required	**Id of the Budget Category**	ObjectId

Households

`POST` - Households/Create

```
.post('Households/Create', (req, res) => {
  /*
  * Reads upcoming data from the request body and creates a new household
    to a user with their Email in the database.
  */
}
```

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
name		**The name for the household.**	string
greeting		**The greeting of the household.**	string
email		**The email to set the owner of the household.**	string

`GET` - Households/Transactions

```
.get('Households/Transactions', (req, res) => {
  /*
  * Returns all the spending type transaction of the Joined bank account to
    the household.
  */
}
```

Response Class (Status 200)

Example Values - Household Transactions

```
[
  {
    _id: ObjectId,
    BankAccountId: String,
    Amount: 0,
    TransactionType: String,
    Memo: String,
    CategoryId: null,
    CategoryItemId: null,
    Created: { createMoment: [Object] },
    OwnerName: String,
    AccountName: String
  }
]
```

`GET` - Households/Members

```
.get('Households/Members', (req, res) => {
  /*
  * Returns all the members of the household
  */
}
```

Response Class (Status 200)

Example Values - *Household Members*

```
{
    Id: ObjectId,
    FullName: String,
    TransactionsCount: 0,
    JoinedAccountsCount: 0
}
```

`GET` - Households/HouseholdStatistics

```
.get('Households/Members', (req, res) => {
  /*
  * Returns the household statistics
  */
}
```

Response Class (Status 200)

Example Values - *Household Statistics*

```
 {
   TotalBalance: 0,
   TotalBudget: 0,
   MonthlySpending: 0,
   MonthlyDeposit: 0
 }
```

`GET` - Households/BankAccounts

```
.get('Households/Members', (req, res) => {
  /*
  * Returns all the joined bank accounts within the household
  */
}
```

Response Class (Status 200)

Example Values - *Household Bank Accounts*

```
[
  {
   _id: ObjectId,
   BankAccountId: String,
   Amount: 0,
   TransactionType: String,
   Memo: String,
   CategoryId: null,
   CategoryItemId: null,
   Created: { createMoment: [Object] },
  }
]
```

Transactions

`POST` - Transactions/AddTransaction/:accountId

```
.post('Transactions/AddTransaction/:accountId', (req, res) => {
  /*
  * This takes a accountId as parameter and request data by the user and adds
    the transacton into the database for the specific bank account.
  */
}
```

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
amount	required	**Amount of the transaction.**	number
memo	required	**Details about the transaction.**	string
transactionType	required	**The transaction type.**	String
bankAccountId	required	**The Id of the bank account receiving transaction.**	number

`POST` - Transactions/DeleteTransaction

```
.post('Transactions/DeleteTransaction', (req, res) => {
  /*
  * Reads accountId & transactionId from the request body and deletes the
    specific transaction from the database, also updates the value of
    current balance for the bank account in the database
}
```

Implementation Notes

This will delete a specific transaction for a bank account.

Response Class (Status 200)

Request Body

Parameter	Value	Description	Data Type
transactionId	required	**Id of the transaction to be deleted.**	ObjectId
accountId	required	**Id of the associated bank account.**	ObjectId