About money lending app

The lending app will be used to track user budgets and loans.

A user can be described as the user of an application that uses this app for managing loans in a simplified way. The user also stores all data in the cloud.

There are three main types of data stored: Users, budgets, loans and transactions.

Transactions are accounting transactions that debit one account and credit another. For example, the transaction moves funds from budget to loan on loan creation.

There are automatically generated transactions such as interest that are calculated and added periodically per loan configuration.

Contents:

User

The user data type is necessary to join user info, settings, budgets and loans.

User’s data can be accessed and modified by the authorized user.

User data includes the user’s budgets and loans in a non-relational way because all of this data will usually be requested at once.
*Transactions for budgets and loans are stored separately.

TypeScript interface definitions (subject to change):

```
interface User {
    id: string,
    name: string,
    email: string,
    firebaseId: string,
    budgets: [Budget]
    loans: [Loans]
    subscription: {
        revenuecatId: string,
        type: “free” | “standard” | “premium”
    }
    currency: string,
    language: string,
    timezone: string,
}
* Password is stored elsewhere
```

User stories:

As a lender, I want to create a user account, so that I can persist changes.
As a lender, I want to log in, so that I can access my data.
As a lender, I want to log out, so that I can use the application on other people's devices.
As a lender, I want to view my user account information, so that I can make appropriate changes.
As a lender, I want to reset my password, so that I can recover my account in case I forget my password.
As a lender, I want to change my password, so that I can secure my user account.
As a lender, I want to change my user account name, so that I can fix errors in spelling.
As a lender, I want to change my subscription, so that I can pay for exactly what I need.
As a lender, I want to change the UI language, so that I can more easily use the application.
As a lender, I want to change my timezone, so that interest can be calculated at the right time (local midnight).

Budgets

Budgets are used for the separation of funds into user-defined categories (by risk, purpose, someone else's funds etc.).

Budgets persistently store descriptive data such as name, description and default interest rate.

Budgets have calculated financial data such as the total amount and lent amount.
* Future versions of the application may also include analytical data such as ROI and predictions on available funds.

Such calculated financial data is computed on the server from relevant transactions per client request of budget data.
* Optimizations on the server will be required at scale: Caching and storage of pre-calculated data (example: storage of budget total at X date so that only new transactions must be calculated and added to stored value).

A practical use case for budgets 1:

The user wants to budget 20.000€ of his portfolio into low-risk mid-term loans that are expected to yield 7% yearly interest.

The user also wants to budget another 5.000€ of his portfolio in high-risk short-term loans that are expected to yield 10% per month.

A practical use case for budgets 2:

The user has a great opportunity to give out a 100.000€ safe loan that is guaranteed to get paid back in 3 years, but the user does not have enough funds, so he wants to get other investors on board.

The user creates a new budget for each investor

A new loan is created that combines funds from x budgets(investors) with proportional representation.

When loan principal and interests are paid, they are transacted(moved) back to each budget(investor) proportionally.

A practical use case for budgets 3:

The user has an opportunity to give out a good high-risk loan of 500€.

He checks into stats to see if a budget is available for such a high-risk loan.

A practical use case for budgets 4:

The user wants to adjust budget interest rates due to a higher amount of defaults.

The user checks budget statistics to get data on actual money generated by a certain budget and the exact number of defaults.

TypeScript interface definitions (subject to change):

```
interface Budget {
    id: string,
    name: string,
    description: string,
    defaultInterestRate: InterestRate
    calculatedTotalAmount: number,
    calculatedLendedAmount: number,
}
```

User stories:

As a lender, I want to create a budget, so that I can categorize my investments.
As a lender, I want to view a list of budgets with basic information, so that I can have a general overview of my investments.
As a lender, I want to add funds to the budget, so that I can later assign them to loans.
As a lender, I want to withdraw funds from the budget, so that I can make use of interest or move it to another budget.
As a lender, I want to view details about the budget, so that I can make good decisions and appropriate changes.
As a lender, I want to view transactions related to budget, so that I can make decisions.
As a lender, I want to export budget data and transactions, so that I can archive them or import them to other software.
As a lender, I want to change the existing budget name and description, so that I can set a more fitting name and description.
As a lender, I want to change the default interest rate, so that I can adapt my budget to current market conditions.

Loans

Loans are used to group data about a loan and events/transactions related to that specific loan.

The loan holds data such as name, description, list of notes, starting date, predicted closing date, and interest rate.

Definition of interest rate: *The interest rate is defined as a* ***percentage per duration*** *or a* ***fixed amount per duration****. Available durations are predefined as Day, Week, Month, Year (these are self-explanatory) and full duration which simplifies fixed fee loans (Full duration loan example: I loan you 300€ and you will give me back 350€ next month).*

Just like budgets they also have calculated data such as initial principal, remaining principal, the total charged interest, and paid interest.

Loans also have “status” property that can hold the following values:
- Active: Loan is unpaid, payments are expected, interest in being calculated.
- Paused: Loan is unpaid, payments are put on hold, interest is being calculated.
- Paid: Loan is paid, but not finalized.
- Closed: Paid, finalized, saved for archive
- Defaulted: Unpaid, finalized, saved for an archive.

TypeScript interface definitions(subject to change):

```
interface Loan {
    id: string,
    name: string,
    description: string,
    notes: [Note],
    startedOn: number,
    closesAt: number
    interestRate: InterestRate,
    initialPrincipal: number,
    calculatedRemainingPrincipal: number,
    calculatedTotalPaidPrincipal: number,
    calculatedChargedInterest: number,
    calculatedPaidInterest: number,
}

interface Note {
    id: string,
    content: string,
    createdAtTimestamp: number
    revisions: Note (recursive)
}
```

```
interface InterestRate {
    type: “percentagePerDuration” | “fixedPerDuration”,
    duration: “day” | “week” | “month” | “year” | “fullDuration”,
    amount: number,
    entryTimestamp: number,
    revisions: InterestRate (recursive)
}
```

User stories:

As a lender, I want to view and search for loans, so that I can find the specific loan.
As a lender, I want to create new loans, so that I can later track specific loan transactions and info.
As a lender, I want to view information and transactions of the specific loan, so that I can make informed decisions.
As a lender, I want to search for loan transactions, so that I can find the specific transaction.
As a lender, I want to add new transactions to the loan, so that I can track payments, interest and fees.
As a lender, I want to edit and delete loan transactions, so that I can make corrections.
As a lender, I want to update descriptive data about the loan, so that it stays current.
As a lender, I want to update the loan interest rate, so that it reflects current market conditions and future interests will be calculated based on the new interest rate.
As a lender, I want to add notes to the loan, so that I can track agreements.
As a lender, I want to change and delete notes, so that I can make them accurate.
As a lender, I want to change the status of the loan, so that status reflects the real world.
As a lender, I want to export budget data and transactions, so that I can archive them or import them to other software.

Notes:

When payment is added one transaction is to balance the interest, and for the leftover amount, one transaction is made to reduce the principal.

When payment is made, principal and interest funds are forwarded back to budgets.

!!!! When a transaction has changed all transactions that were made after the change must be recalculated.

Transactions

Transactions are used to move and track the movement of value.

As such, they can be described as moving x amount from one account to another account.

Traditional accounting/bookkeeping transactions are entered into T Accounts (debit, credit). In my experience this way of accounting is error-prone when past transactions can be edited (as on some accounting software) - mismatches on separate tables can occur on large datasets.

Because of the reasons above transactions will be represented in the following way:

TypeScript interface definitions(subject to change):

```
interface Transaction {
    userId: string,
    id: string,
    transactionTimestamp: number,
    description: string,
    from: TransactionAddress,
    to: TransactionAddress,
    amount: number,
    entryTimestamp: number,
    revisions: Transaction (recursive)
}

interface TransactionAddress {
    datatype: “budget” | “loan” | “Interest” | “outside”,
    id: string
}
```

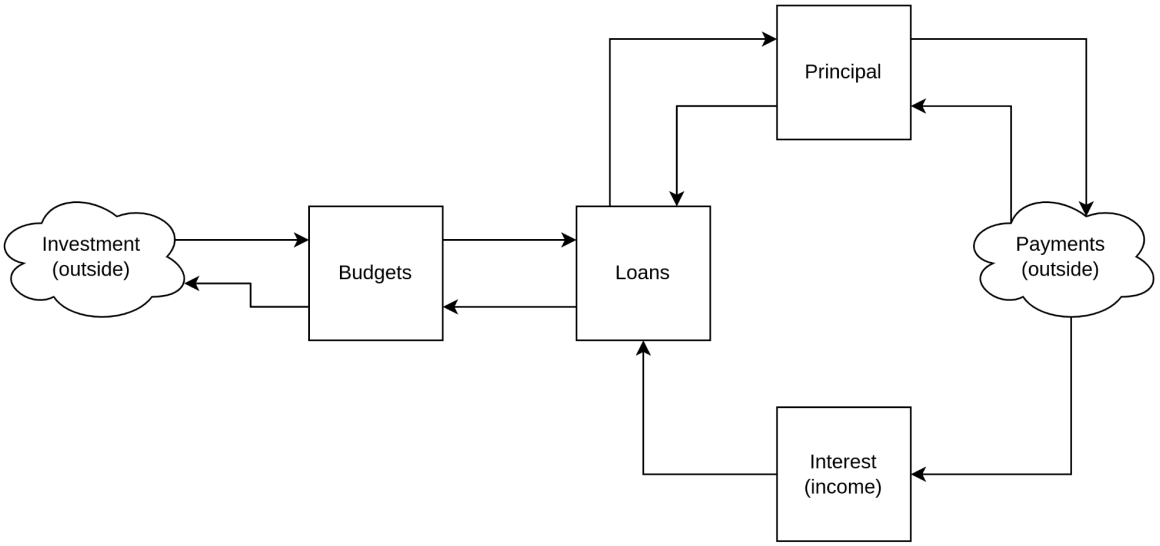
With this approach, any change to the transaction is reflected in both accounts.

A word on accounting and money flow

This application could be explained as a specialized accounting application that deals only with part of accounting.

In accounting, there is an accounts system that ensures balancing and complete tracking of money flow.

Following is a flow chart of transactions in the app:



* Outside means outside an app aka. other parts of the accounting system.

As such this app can be used in tandem with more complex accounting software to provide convenience and enhanced analytics on this specific loan-related part of accounting.

Other concerns:

How percentage per duration is interest calculated:

On the user's timezone day start + 1 hour (01:00) function is run that checks every active loan.

The function creates interest dates (based on interest definition) until today | if the last payday matches today then interest is calculated based on the current principal

!!! Make something like this scalable

How fixed amount per duration is interest calculated:

The fixed amount is added as the transaction to loan.principal ledger when the loan is created

Currencies support

In the first version, the currency is selected per user.

Possible improvements on currencies: Currency is not bound to the user but is selected upon budget creation and can not be changed later. To change currency assets must be transferred withdrawn to a bank account and transacted to the new budget in a new currency. In this way, the currency is a concern of a user but there are some edge cases to handle (loans can only be made up of budgets with the same currency).