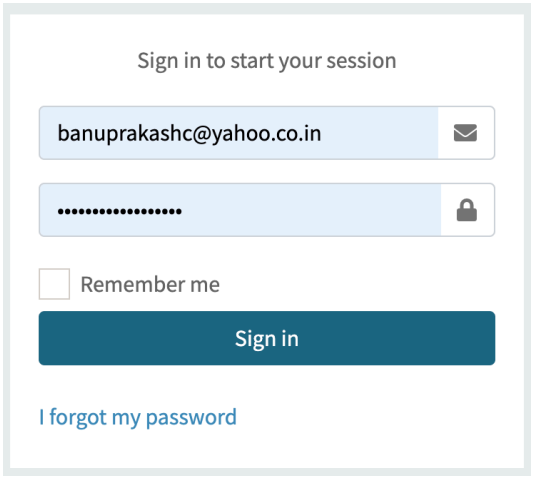
Java and Web technologies Mini-Project

Personal Finance Manager

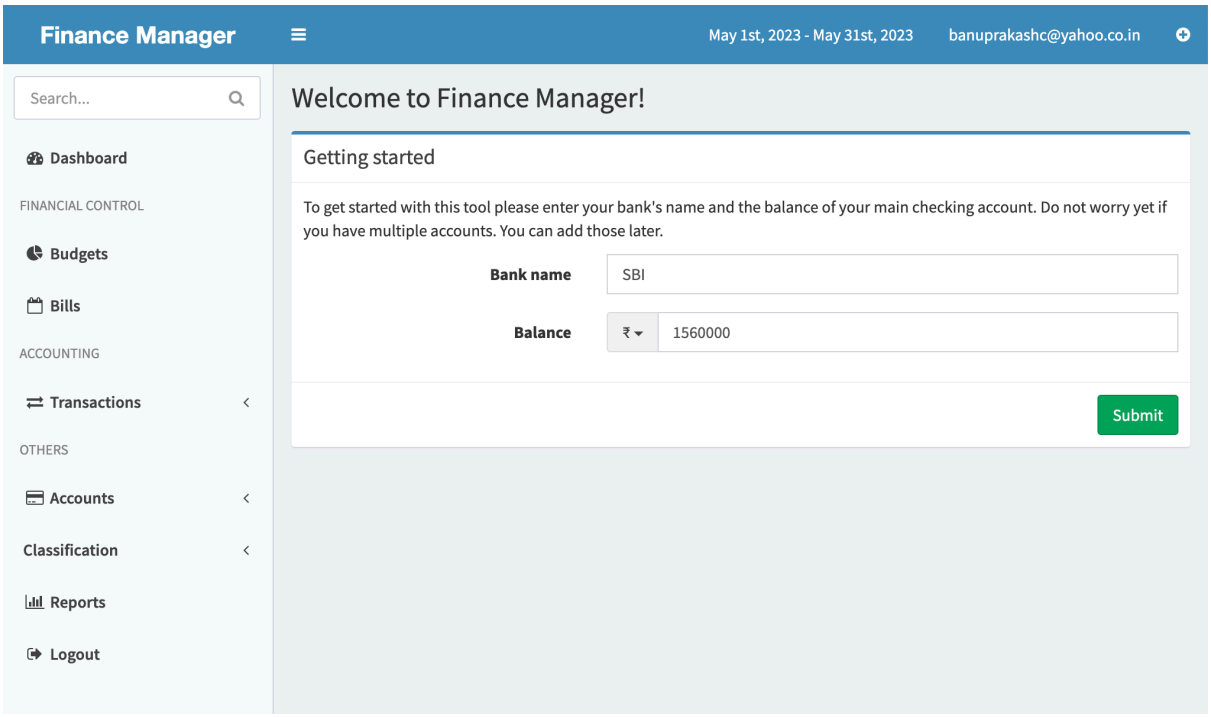
App to help you keep track of your expenses and income, so you can spend less and save more. By keeping track of your expenses and your income you can budget accordingly and save money.

Who is it for?
This application is for people who want to track their finances, keep an eye on their money without having to upload their financial records to the cloud.

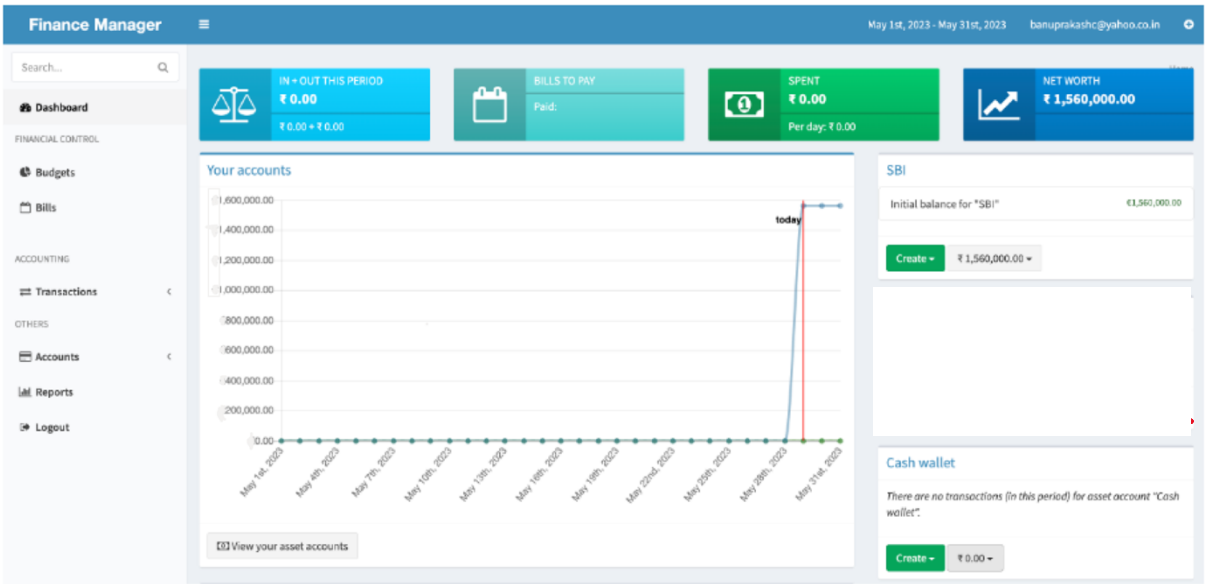
1. Registration: User needs to register before using the application.
2. Login



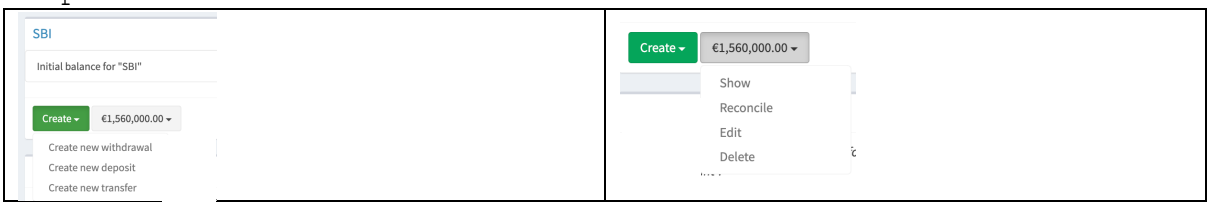
3. Once on successful login for first time user is redirected to "Getting Started" page as shown below:



4. Once account is added, user is redirected to the Dashboard. For subsequent login user will be redirected to this page directly.



Dropdowns:



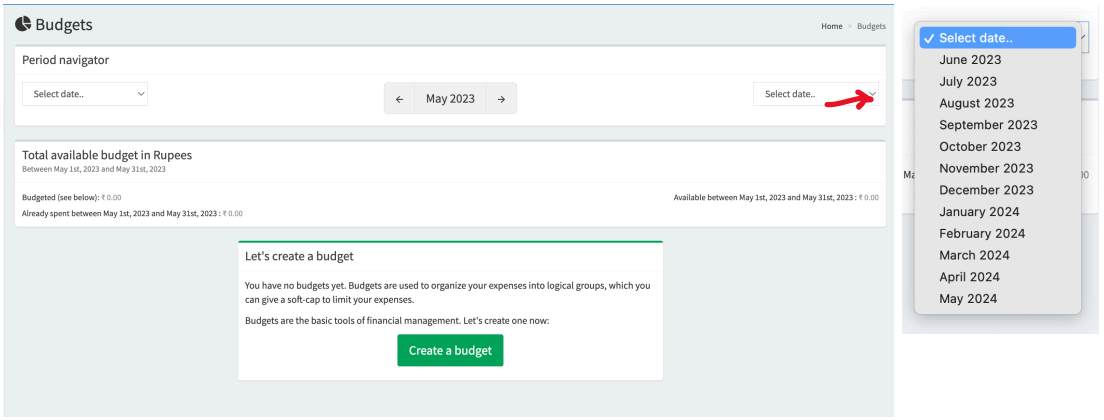
5. Budgets

Once you start creating transactions you start to realise that in a month, the same kind of stuff always comes back. For example:

- Bills
- Groceries
- Cigarettes
- Going out for drinks
- Clothing

Likewise, you should start to notice that you always spend the same amount of money on these things. That amount may be too high for your tastes, and you may want to change that. Or at least, keep track of it.

These things are budgets. Budgets are a kind of "category" that come back every single month. Bills are recurring (rent, water, electricity). You buy groceries every day. You need to pay rent every month.



Adding money to a budget

Automatic budgeting

Edit or create a budget and pick from the following options:

a. Fixed amount

   For example amount is set to 2000 monthly will give you an automatic budget amount of 2000, valid for one month, every month. This will happen automatically. Other periods are explained further ahead.

b. Rollover ("Add an amount every period")

Rollover budget amounts can be used to "save up" money in a budget. Application will take the budget left from the previous period and add the configured amount to the budget.

If you set it to "monthly" (2500):

January, the budget will be set to 2500.

February, the budget will be set to 5000.

March, 7500 etc.

If you spend money in your budget, this will be reflected in the budget. For example, with the example budget now at 7500, this is what happens when you spend 2000:

April, the budget will be set to 8000: 7500 + 2500 - 2000.

May, 10500. Same logic.

If at any point you spend more than the amount in the budget, the routine will start over. So if you spend 12000(which is more than 10500):

June, the budget will be set to 2500 again.

c. Adjusted ("... and correct for overspending")

If you set an adjusted budget amount, the budget amount will be set every period to the amount you specify. If you have overspent this amount in the previous period, this will be corrected in the current period.

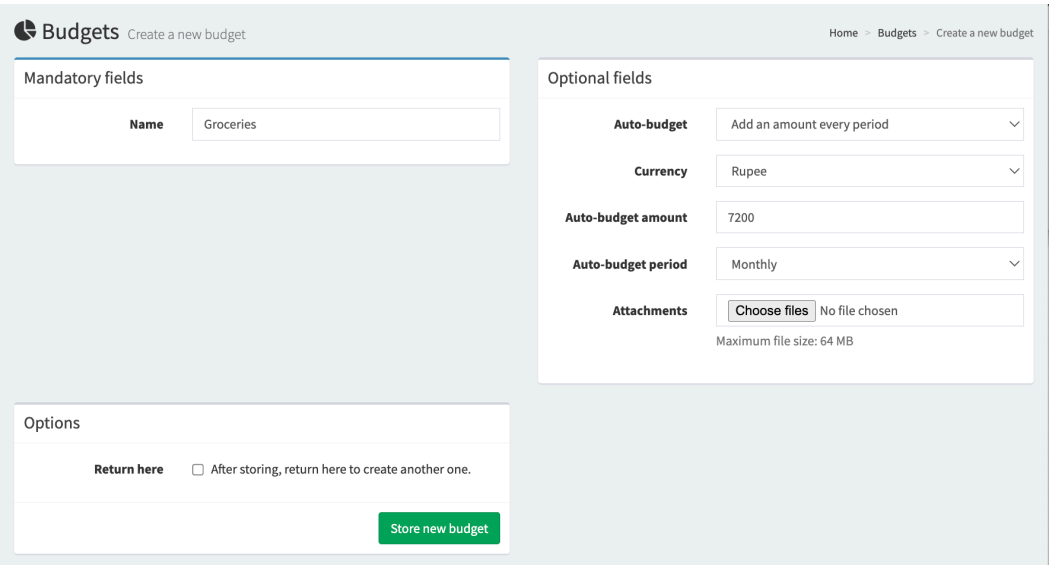
Example: you have a monthly budget of 500 for clothes.

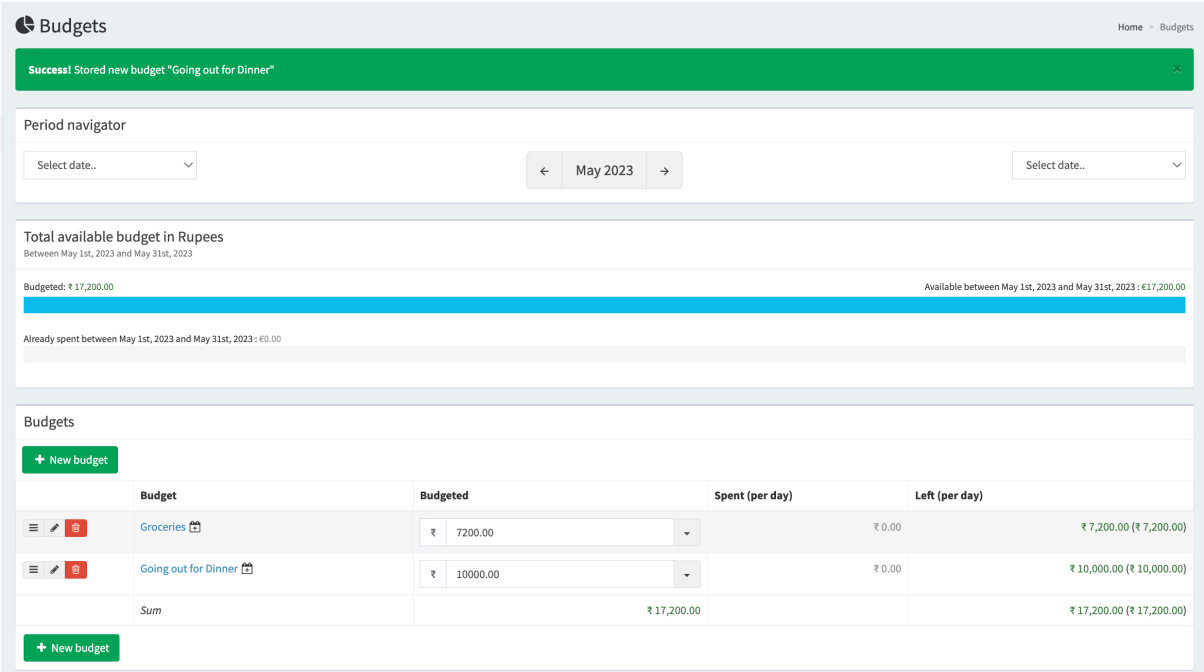
January, the budget amount will be set to 500

February, the budget amount will be set to 1000

You spend 1250 in February.

March, the budget amount will be set to 250 (1000 + 500 - 1250)





Bills
You can keep an eye on your expected recurring expenses by creating bills. Things like rent and utilities must be paid every month and Firefly III can keep track of such things.

When you create a bill, you tell Firefly III in what range you expect the bill to be. You also input the title of the bill, and how often the bill is expected to be paid.

The name is descriptive only and is not used internally.

- Minimum amount: 7000
- Maximum amount: 8000
- Description: Monthly rent

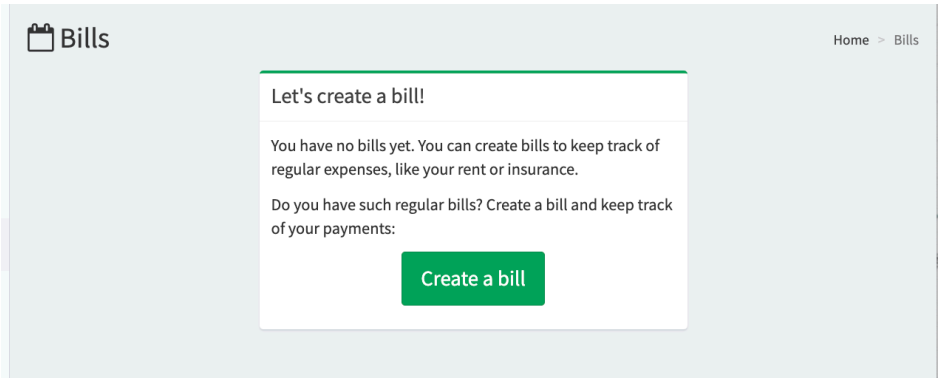
- Repeats every month

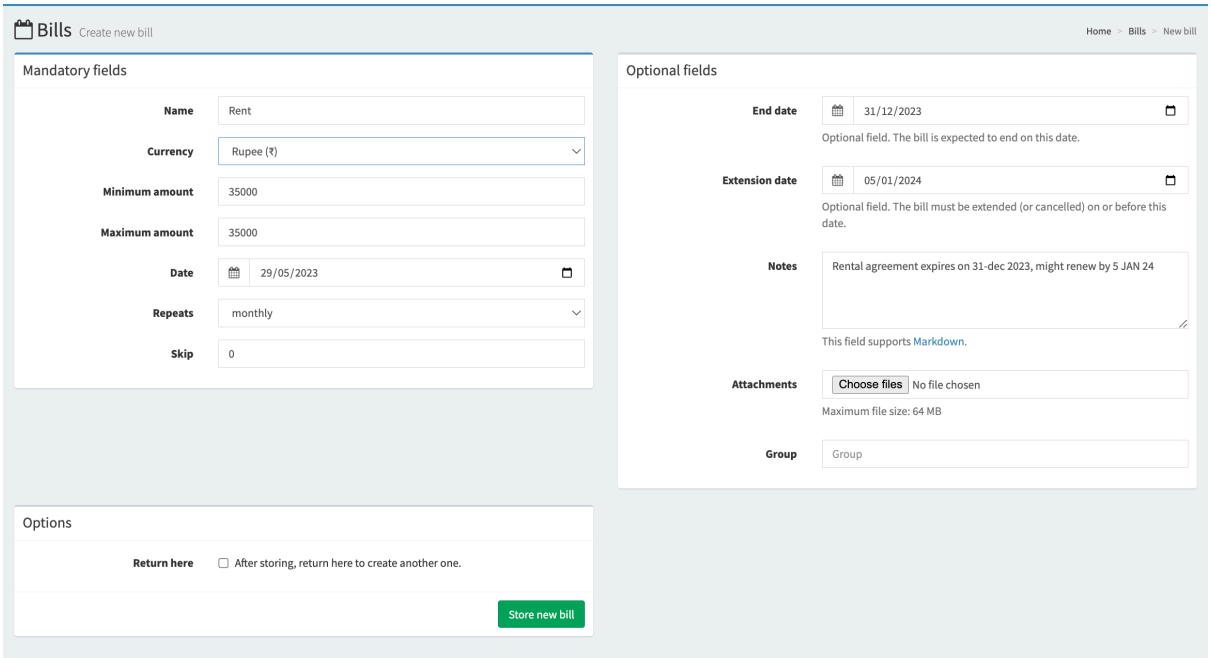
You can also set the end date and the extension date.

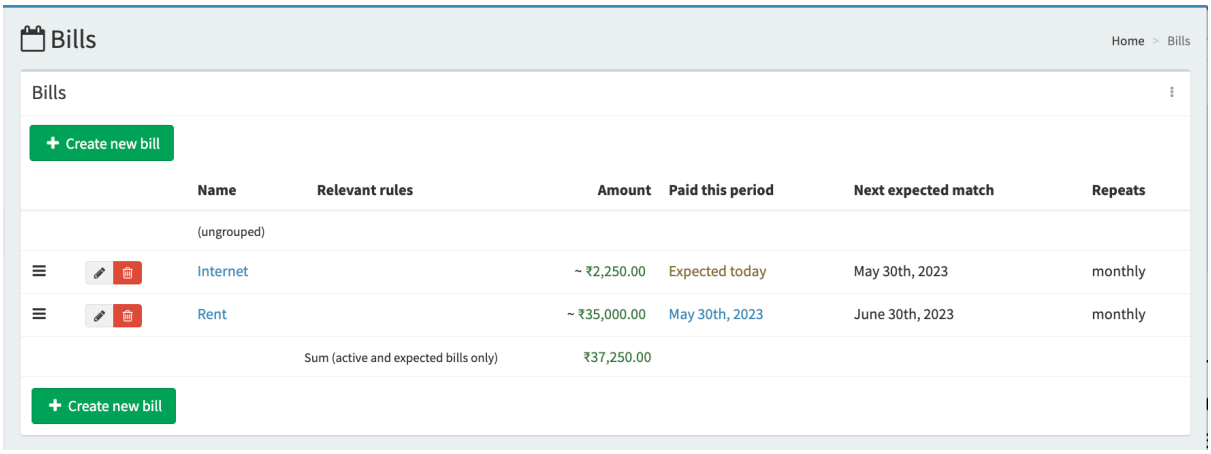
These properties by themselves are mostly cosmetic. They allow application to predict for you how much you should expect to spend on these bills. On the frontpage, a little box will tell you how you're doing.

If you enter a number in the "skip" field, the bill will be automatically skipped every X times; a bill that arrives every 3 months can be entered by filling in "2".

If you edit a bill and change the amount, the rule will not be automatically updated to match. When you delete the bill, transactions associated with the bill will lose this association but will not be deleted.

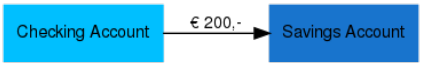


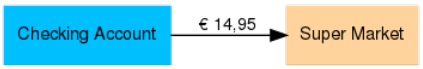




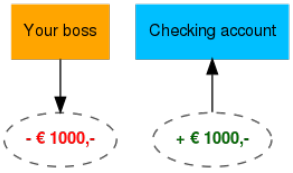
Transactions

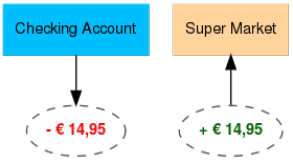
Money moves from A to B. It doesn't matter if this is an expense, your salary or you moving money around: *money moves from A to B*:



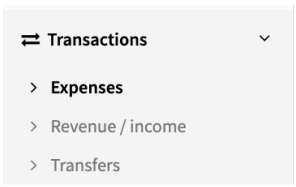


You get money and your boss loses it. You spend money and the Supermarket "earns" it:

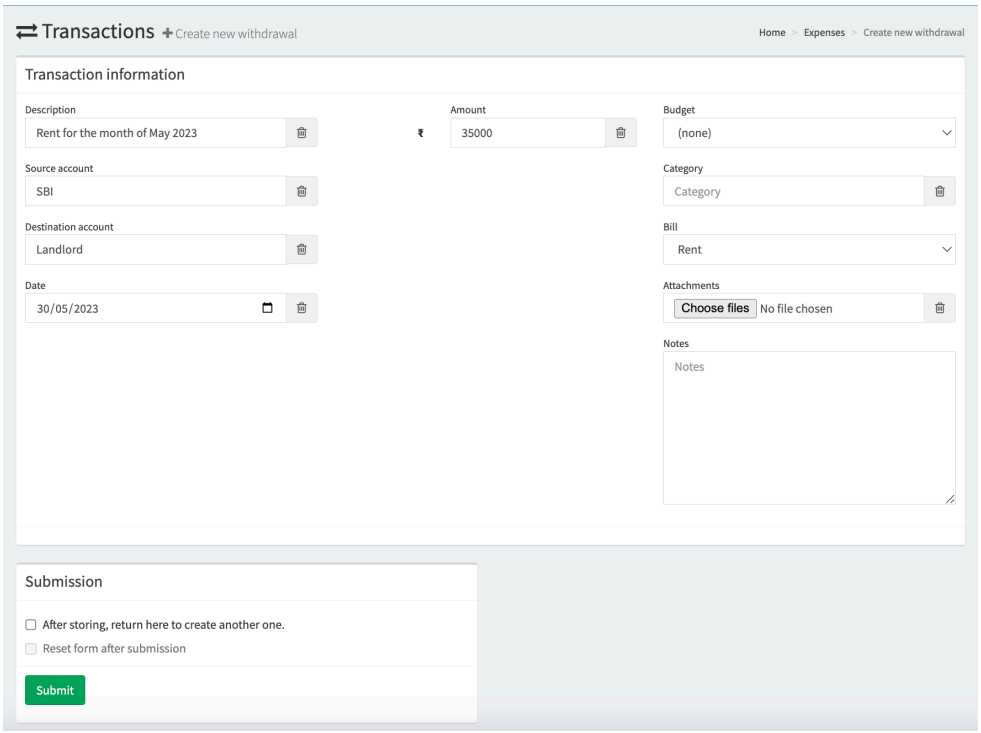




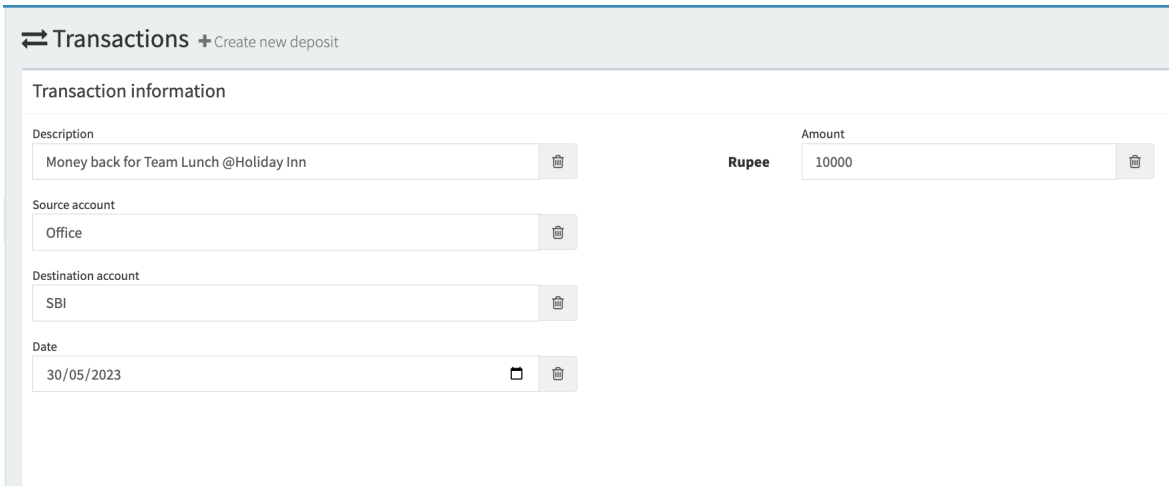
Each transaction is stored twice. Once as a loss (for one party), and once as a profit (for the other party). This seems pretty pointless, and technically it is. But it was designed back when clerks could be fraudulent and this double-entry system stopped fraud.



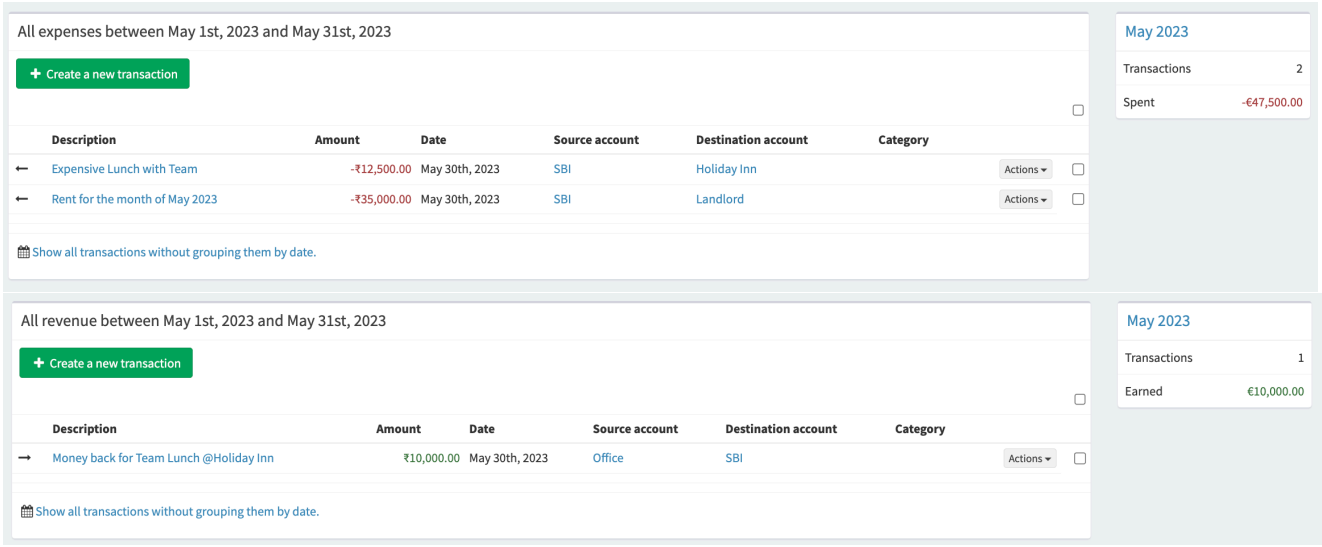
Expenses:



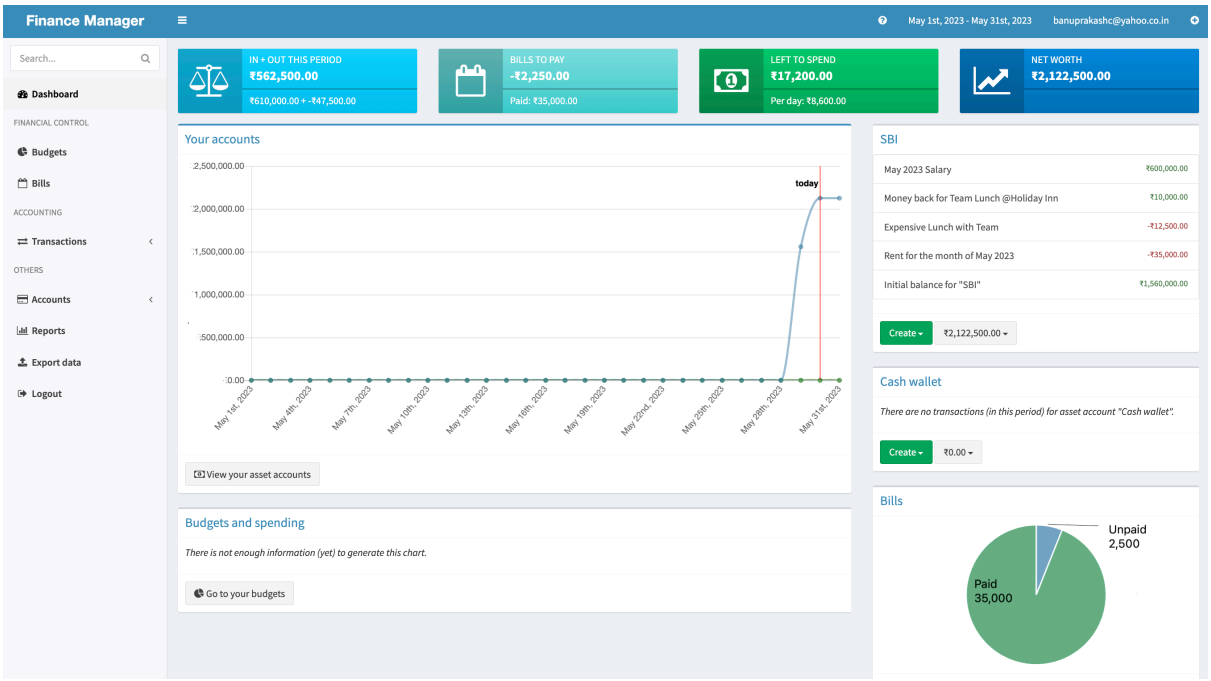
Deposits
When you wish to create a deposit, select a revenue account first, as the source account. When it doesn't exist yet, free-type your own input, and it will be created for you. Then, select an asset account or liability as the destination account. If the source account already exists the form will recognize that you're creating a deposit, and the "budget"-selector will disappear.



Transactions:



Dashboard:



Reports:
Default Financial Report from Jan 1st to Dec 31st

1. Bar Char [Income vs expenses]
2. Line Chart [Net worth with Month on X axis and Net worth Y axis]
3. Tabular data for Revenue/Income and one for expenses

Technology Stack to be used for this Project:

- Building REST Apis
  - Build REST APIs using Spring Boot
  - Use MySQL for Persistence
  - All RestControllers should be unit tested using JUnit and Mockito
  - Use OpenAPI/ Swagger for Documentation
  - Use RestTemplate/PostMan for testing APIs
- Building UI components
  - Build UI components using REACT
  - Use Redux for State management
  - Use React-Router-DOM for routes
  - Use React Testing Library for Unit testing React components
  - Use Cypress for E2E testing
- Fake RESTful web services using json-placeholder during development stage
- Use Bootstrap/Bulma/Tailwind for Responsive Web design
- Use Font-awesome for icons
- Security
  - Spring Security with JWT for authorization