



Budget Wise App

One Stop Shop for all your financial needs

Problem Description

According to a report generated by Statistics Canada, the inflation is at all time high. It has risen 7.7% over the year [1]. With inflation, many Canadians are finding it hard to budget their financial resources. This could be due to excessive or frivolous spending, never ending loaned payments, not having a financial plan to manage their capital. Our team has decided to put together a web application for users to efficiently manage their money using modern algorithms, see analytics and create an organized plan to beat inflation.

Our team is proposing an idea of developing a self finance management web application called Budget Wise. This web application will let users manage their expenses by providing them with several analytics on their money expenditure

Architecture

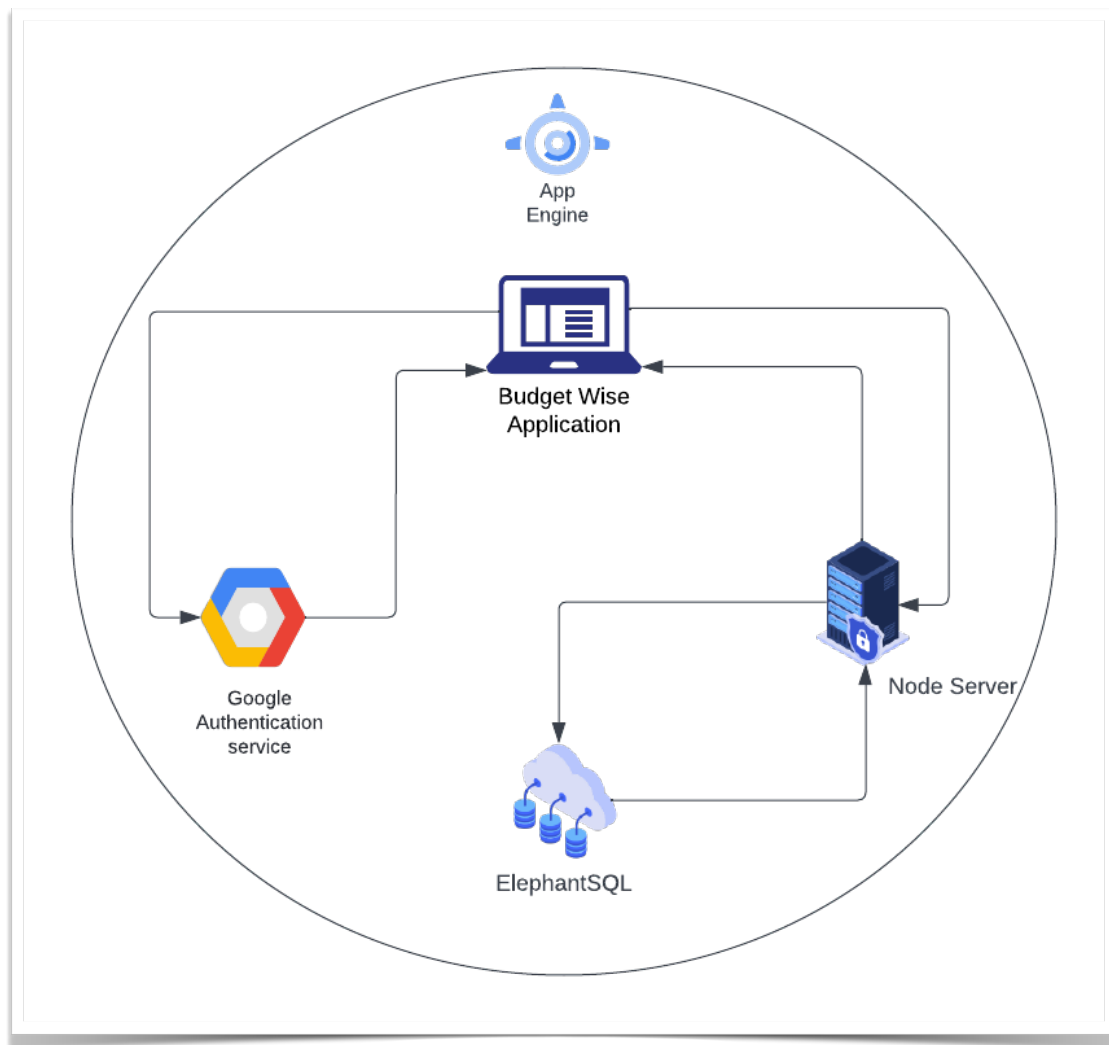
The application is built using the PERN architecture.

The server-side (backend) is built using **Node.js** (express) which connects to an external API **ElephantSQL** which provides database services in **postgreSQL**.

The database named db consists of five tables. The backend also uses an internal service from GCP for authentication called Google Sign-in. We have used a route-controller architecture for backend development. It eases connectivity to backend services and helps us in modular application development.

The front-end is built using **React.js**, and the CSS framework called **Antd**.

Architecture



Google Auth returns an object with valid user email which is encoded by in-house API to be saved as a unique key in database for identification. If email encoding matches, the user can use Budget Wise Application and send requests to/from server connected to Cloud ElephantSQL service for finance management

Database

The application database is developed using **PostgreSQL** connected using third party **ElephantSQL**. Budget Wise make calls to ElephantSQL service to handle database queries.



The app database consists of 5 tables:

1. users - consists of user unique token, email id and general user column
2. credicards - consists of user unique token, card name, card number, expiry date and amount
3. budgettable - consists of user unique token, budget name, total amount spent and a max amount
4. expensetable - consists of user unique token, budget name, expense description, amount, date and a unique serial id for row
5. owings - consists of email which helps in recognizing the user money was sent from, email for user the amount was sent to, amount, date and a unique serial id.

Each of these tables have their own constraints, for example primary keys, foreign keys, on delete cascade etc. We have added documentation in readme for external members to have a look at our database structure.

Authentication:

Authentication is done using the Google-Sign Service which is used as Internal service. It is provided by Google Cloud Platform. A unique token is generated and then saved in the users table to identify each user.

Version Control:

Gitlab is used for version control. Our team worked on feature branches to deliver this app in incremental development cycles. It enabled us to perform code reviews and finish our app development on time. In order to view the version control use the [“dev”](#) branch to view budget wise app's source code.

App Workflow/Features

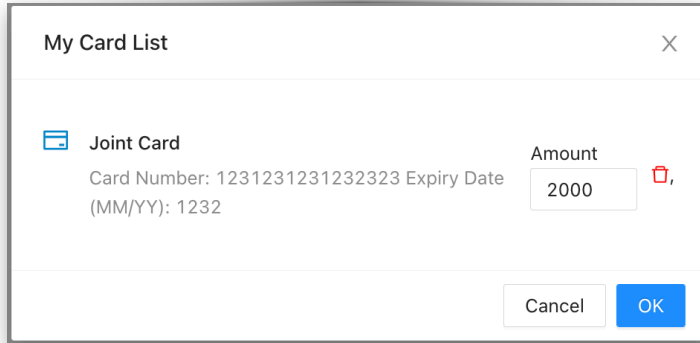
Login/Signup using Google Authentication

Login and sign up has been built using the Google Authentication Service. This is an Internal API that we used for this application. Simply Sign up with your user account and login to the application. The user will land on the Budget portfolio tab. The user email will be displayed and the user has the ability to log out.

Add Credit/Debit Card

The user can now proceed to add credit card or debit cards. The react-credit card library auto detects the type of card used by user. The library can detect different types of credit cards, debit cards, etc. Add your card details in the modal to add credit for managing your expenses.

The image displays two side-by-side screenshots of the 'Add Credit/Debit Card' modal. Both modals have a title bar with a close button (X). The left modal features a Mastercard template with a grey background, a yellow chip, and the Mastercard logo. The right modal features a Visa template with a blue background, a yellow chip, and the Visa logo. Both modals contain the following form fields: 'Card Number' (with a placeholder '54' and a dropdown arrow), 'Card Nick Name' (with a placeholder 'Joint Visa/Mastercard'), 'Expiry' (with a placeholder 'mmYY'), and 'Amount' (with a placeholder 'Amount'). At the bottom right of each modal are 'Cancel' and 'OK' buttons.



My Card List

Joint Card

Card Number: 1231231231232323 Expiry Date (MM/YY): 1232

Amount: 2000

Cancel OK

Cards Added

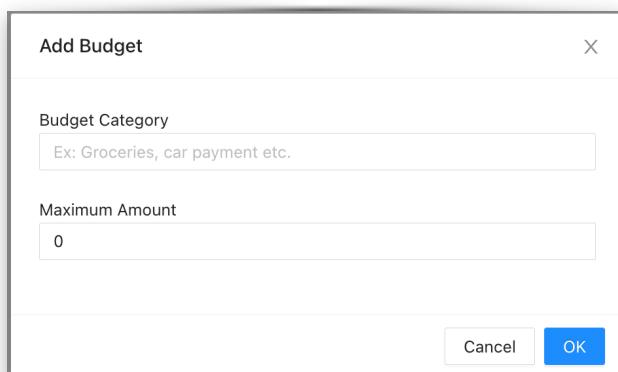
Users also have the ability to view their added cards. This feature also extends to deleting and editing their cards.

Budget Management

Once the card is added, the user is ready to add different budgets they want to manage.

Steps:

- i) The user will use the **Add Budget** feature to add different budget categories. This will signify the areas the user spent their money in. For example fuel, groceries, etc. Then the user specifies the max amount they want to spend for unique category.
- ii) After adding budget categories, the user can proceed to add expenses for their budget categories. They will provide a small description for their expenses. For example adding an expense for groceries can contain “eggs”, “milk” etc. with expense amount.
- iii) Other features to note on this screen is the ability to edit the max amount from budget cards, delete a budget or delete all budgets. User can also see their available credit which is the total amount in cards - all their expenses.



Add Budget

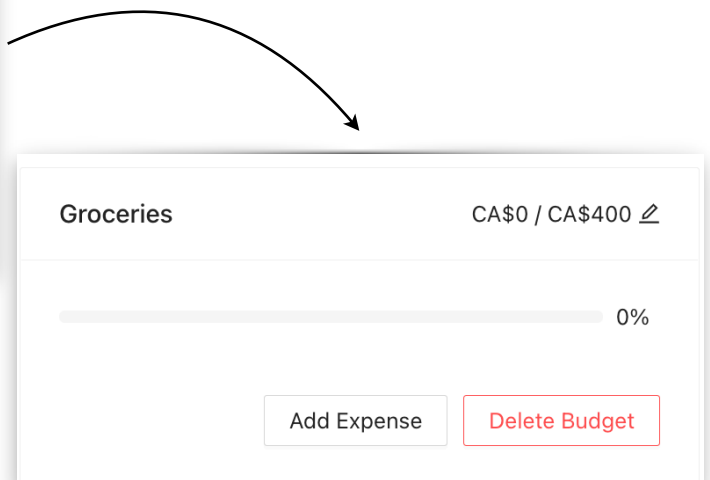
Budget Category

Ex: Groceries, car payment etc.

Maximum Amount

0

Cancel OK

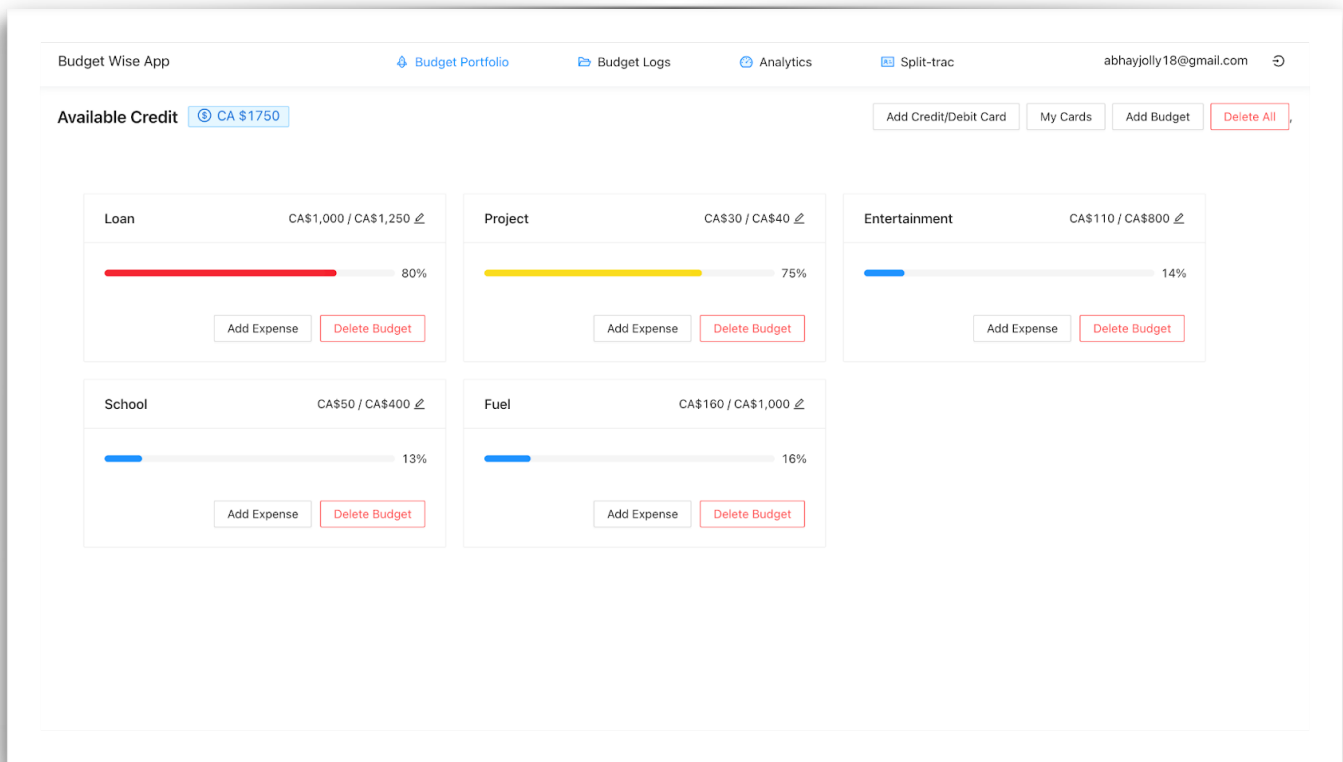


Groceries CA\$0 / CA\$400

0%

Add Expense Delete Budget

Home Screen



Budget Logs

When the user clicks on the Budget Logs tab. They can view all the logs of expenses they have added. This screen lets them “edit” the description and amount for their expenses. Users can also sort the budget logs by specific categories.

Budget Wise App				
Budget Portfolio Budget Logs Analytics Split-trac abhayjolly18@gmail.com				
Budget Logs Check your expenditure All				
Budget Category	Amount	Description	Date	Action
Loan	CA\$1,000	Studies	2022-6-16	Edit Delete
Project	CA\$30	CMPT 372	2022-6-17	Edit Delete
Entertainment	CA\$20	Movie	2022-6-20	Edit Delete
School	CA\$20	Tuition	2022-6-25	Edit Delete

Budget Category	Amount	Description	Date	Action
Entertainment	CA\$20	Movie	2022-6-20	
Entertainment	CA\$30	Netflix	2022-6-26	
Entertainment	CA\$30	Amazon	2022-6-27	
Entertainment	CA\$30	Game	2022-6-16	

Filter by Categories

Split Trac Modal

Split Trac

- i) Split-trac allows users to add the amount that they owe someone or someone owes them. Users can add people by their emails, provide a description and amount. If the other user is a Budget-Wise application user, these amounts will be added for them.
- ii) The user can view these logs, edit them, delete them or sort them to see what they owe or what someone owes them. They can also view total owings which is what everyone owes them - they owe everyone.

Split Trac Log

Budget Wise App Budget Portfolio Budget Logs Analytics Split-trac abhayjolly18@gmail.com

Total Owings ③ -CA\$25 Add Owning All

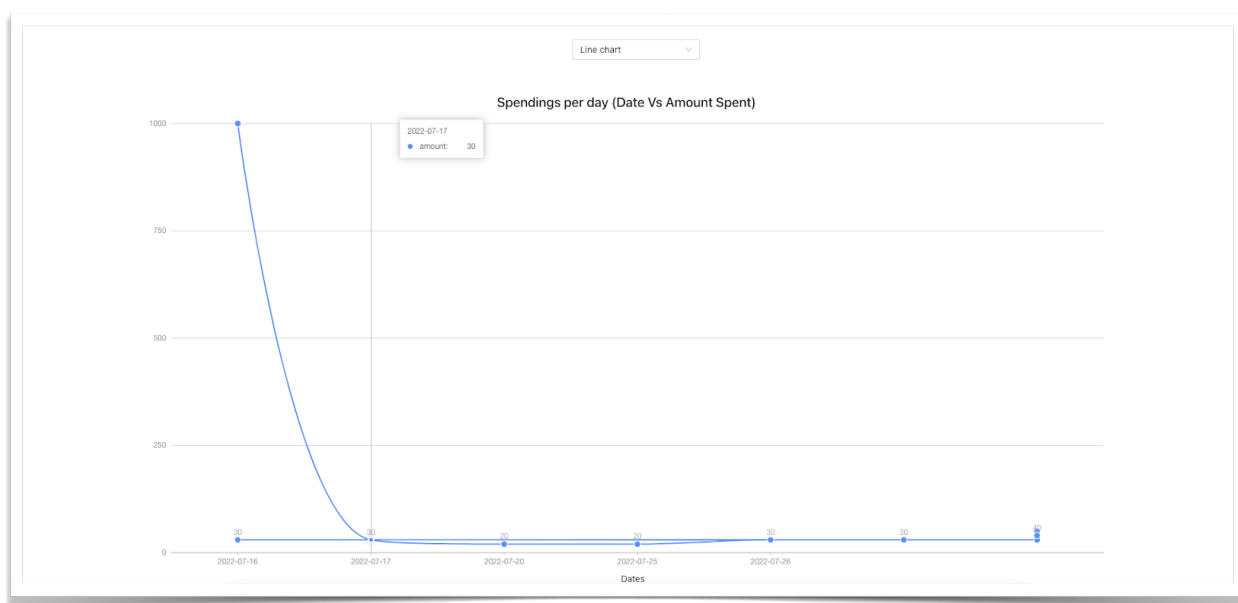
Sender	Receiver	Amount	Description	Date	Action
abhayjolly18@gmail.com	dima@gmail.com	CA\$30	class	2022-6-28	✎ ✖
dima@gmail.com	abhayjolly18@gmail.com	CA\$55	new test	2022-6-28	✎ ✖
abhayjolly18@gmail.com	raman.singh654@gmail.com	CA\$50	Project	2022-6-28	✎ ✖
raman.singh654@gmail.com	abhayjolly18@gmail.com	CA\$5	Coffee	2022-6-28	✎ ✖
abhayjolly18@gmail.com	dima.baranov.2016@inbox.ru	CA\$10	Coffee	2022-6-28	✎ ✖
dima.baranov.2016@inbox.ru	abhayjolly18@gmail.com	CA\$5	Snacks	2022-6-28	✎ ✖

< 1 >

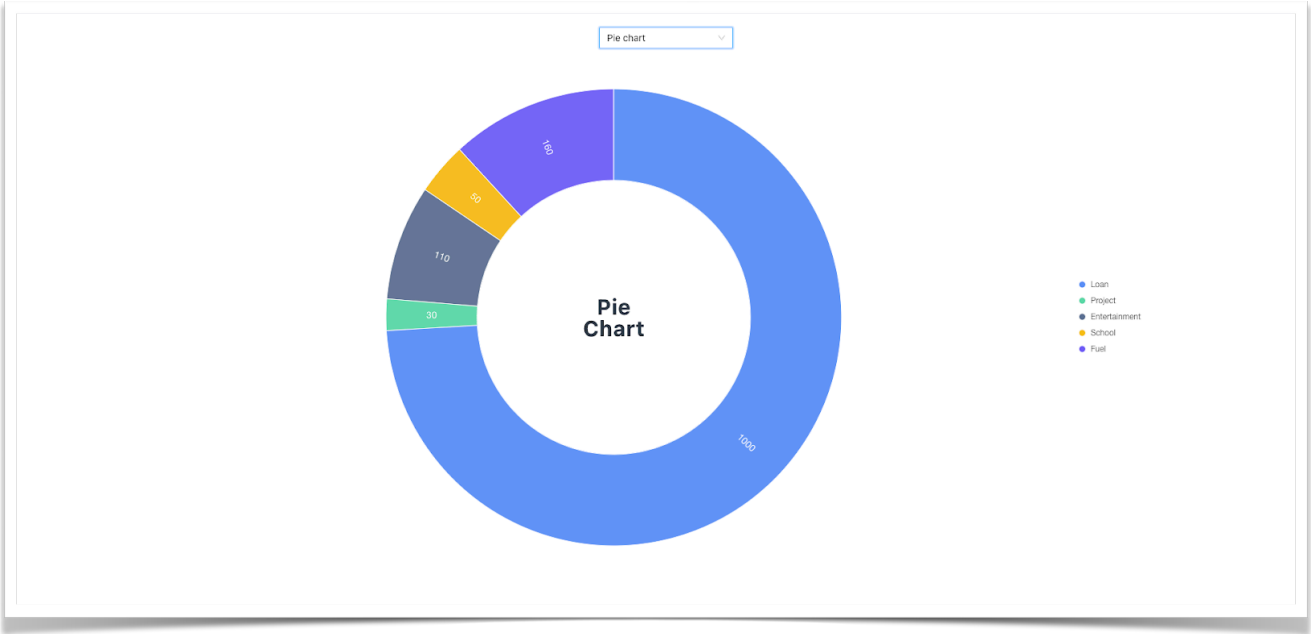
Analytics

The app allows user to visualize their expenses with different graphs

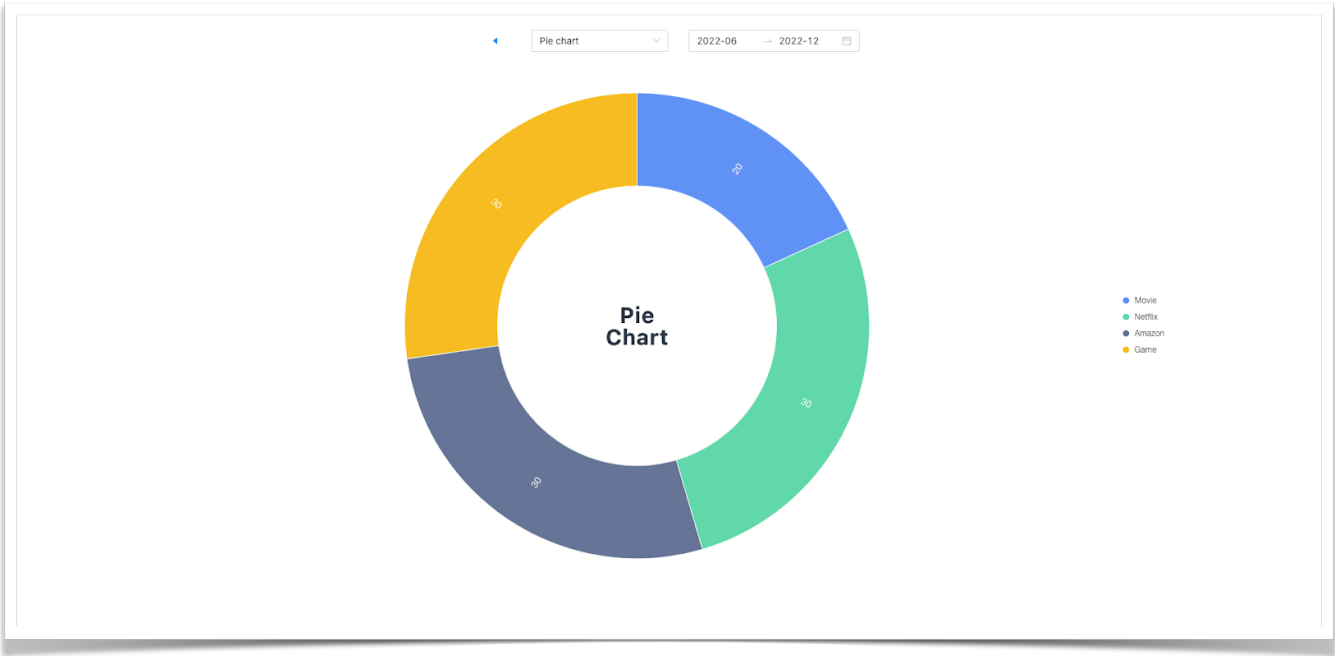
Line Chart



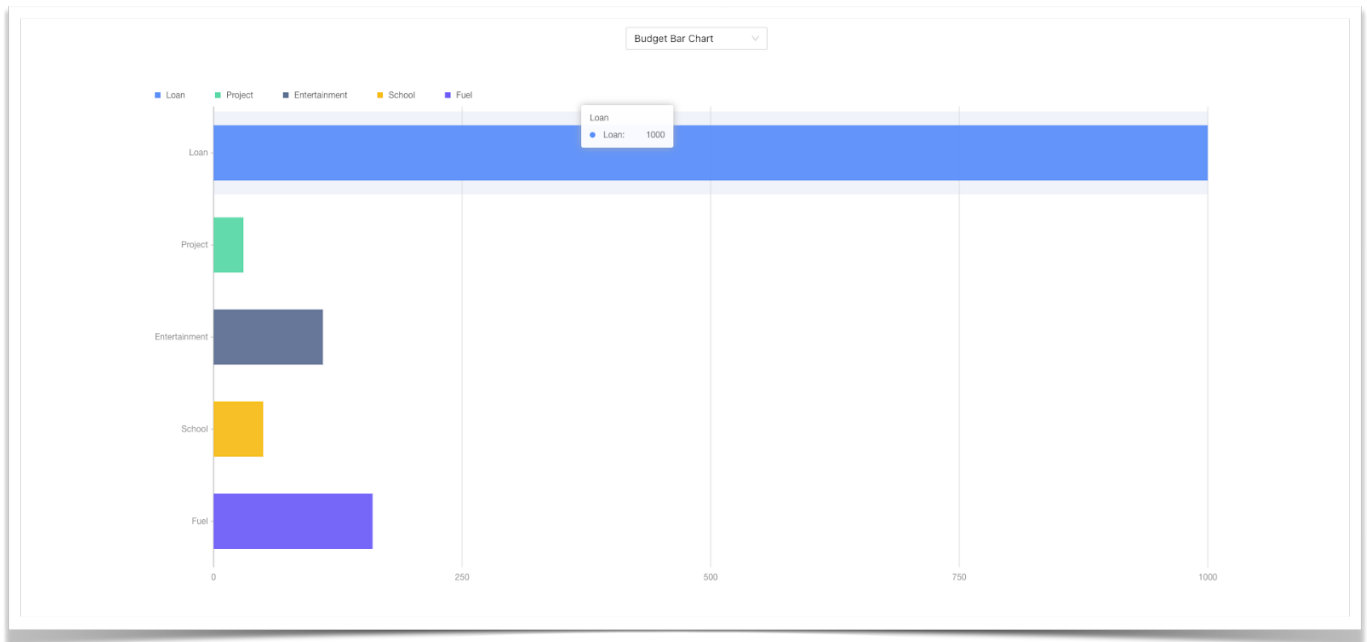
Pie Chart for Categories



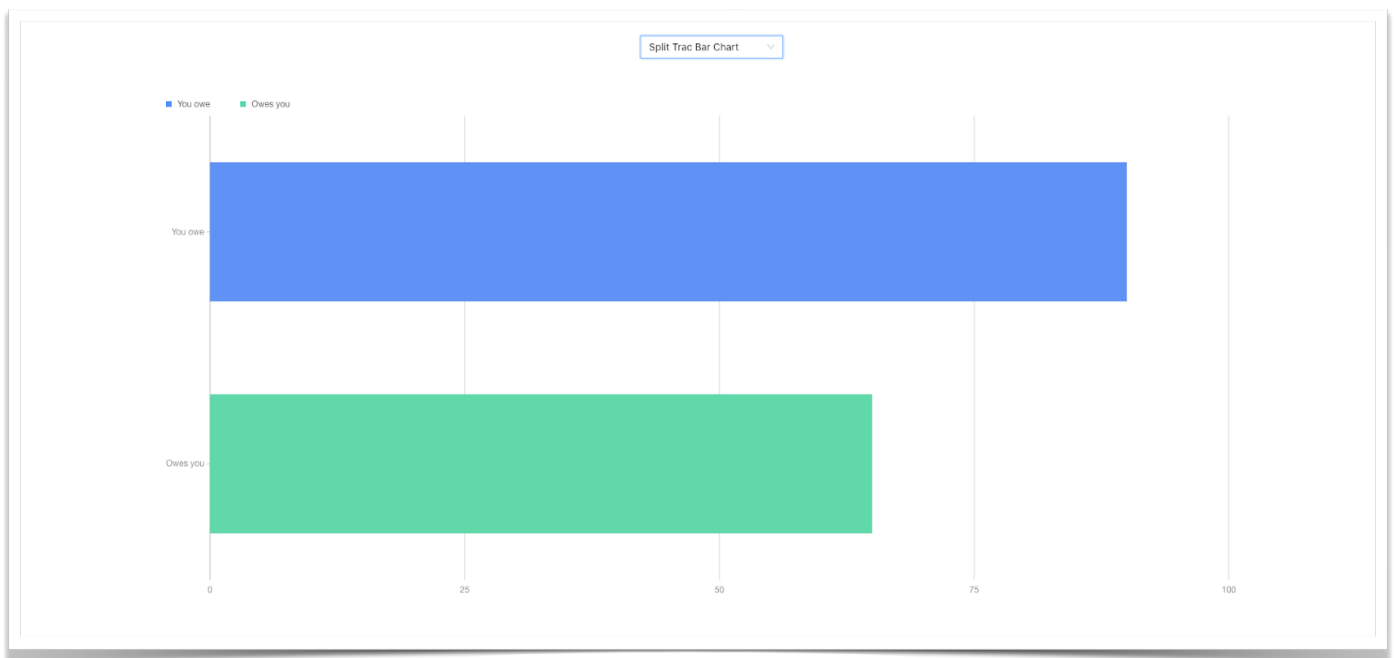
Pie Chart for each Budget Category



Bar Chart for each Category



Oweing Bar Chart (Info about how much you owe/someone owes you)



Testing

We used three approaches to perform tests:

1. UI testing
2. Regression testing
3. API testing using Postman

We worked on our separate branches and performed code reviews before merges to the “dev” branch. This helped us detect bugs earlier in the process. We also used logging to assure what results we get while we use the application and that helped us resolve some small bugs that might go undetected.

The project can also be extended to unit testing in the future using Mocha and Chai.

Future Vision

Everyone in the group showed great enthusiasm in building budget wise app which helped us in introducing more features than we imagined. Along the way we discovered more important features to add and therefore removed some unimportant ones that we thought about initially. For example, ability to add credit/debit card over having an option to change currency for expenses.

We made sure our existing features are well tested and therefore they all work properly. We have also introduced user input checks wherever necessary, also with status visibility using messages.

Future prospects may include using linear optimization to help users understand areas to save money in.

Some graphs in the application are clickable which helps view expenses for a budget category and between specific dates. All the graphs are interactive, so feel free to introduce more data to the application and understand your data deeply with the help of analytics.

You don't need a special user id and password to work with the application. Simply using your gmail account to sign up and login will be enough. If you are already signed up, you will be notified when signing up again.

Gitlab Link: <https://gitlab.com/dbaranov7658/budget-wise-app/-/tree/dev>

App Link: <https://app-356621.nn.r.appspot.com/>