

# Project: Home Budget Economy (Web Development)

## Deployment

**Heroku** is used for deploying the server.

**Netlify** is used for deploying the frontend.

Website Link: [Home Budget Economy](#)

Server Link: [Server - Home Budget Economy](#)

Netlify: [Netlify](#)

Heroku: [Heroku](#)

## Description

### Problem

Managing one's income has been the key resource to one's financial stability which not only affects himself, but also his/her family. Apart from insurance policies, savings has always been an important factor in situations such as emergencies, investment in shares, property or starting a business. It is very important for a person to keep his/her family financially stable and if one is unable to achieve it by increasing his/her income, he/she can do so by decreasing their expenditure.

Nowadays, home economics has become a serious issue as money transactions are advancing with newer technologies. The process of transactions that gets abstracted by the swipe of a card or by a few clicks of a button. This degrades the essential critical thinking that occurs when spending money. This is especially true when numerous smaller transactions occur in a small period of time. This becomes evident only when the person checks their bills and balance at the end of the month and notices the apparent loss of money. This calls for a solution where the user can effectively analyse and manage their expenditures.

### Solution

In order to tackle this problem, I have created a website that allows the user to add entries regarding their income and expenditure. Even the smallest expenditures need to be added.

The website would suggest the user to set a reasonable goal which will finally be their monthly savings. If the monthly savings are greater than the goal, then it will congratulate them. If the

monthly savings are less than the goal, then it will try to find the reason for it, such as a higher electricity bill than the average value, etc.

The expenditure will also be handled in a professional manner. Expenditure will be categorized as:

- Food
- Housing/Rent
- Periodic Bills
- Transportation
- Medical
- Insurance
- Taxes
- Short Purchases
- One-time Purchases/Consumer Durables
- Recreational
- Investment

The website would allow the user to add information about their income and expenses in a tabular format that will be time stamped. The user will have added, edit and delete access to their own table. Each table would be one month long summarizing the savings and expenses for that particular month. Based on the values inputted, the website would suggest solutions or celebrate the monthly savings achieved. This information can also be viewed in a graphical format allowing the user to analyse their situation.

Release 2 is made according to the needs for a general office worker, i.e., the expenditures groups are typically present most in their lives. But in the next releases, the website could include presets that could be used by different occupations. Furthermore, we can add a customizable option so that the people themselves can add or subtract a group from the list of groups.

## Existing solutions

The existing solutions for this problem are various home budget planning calculators such as [bankrate.com](http://bankrate.com), [moneycontrol.com](http://moneycontrol.com), [smartasset.com](http://smartasset.com), etc. The main problem with some of these websites is that they have outdated UI and conceptual advice which is essential but difficult to understand. Most importantly, they are centred towards promoting investments and provide advice for the same.

## Main Packages/Dependencies used

### Backend

- [express](#)
- [mongoose](#)
- [bcryptjs](#)
- [jsonwebtoken](#)
- [cors](#)

- [concurrently](#)
- [config](#)
- [dotenv](#)

devDependencies:

- [mocha](#)
- [nodemon](#)

## Frontend

- [react](#)
- [react-dom](#)
- [axios](#)
- [redux](#)
- [react-redux](#)
- [redux-thunk](#)
- [jwt-decode](#)
- [react-router-dom](#)
- [@mui/icons-material](#)
- [@mui/material](#)
- [chart.js](#)
- [react-chartjs-2](#)
- [web-vitals](#)

## How to use?

These are some details that will help you understand the control flow in the project.

## Prerequisites

- Node.js should be installed on your PC.
- Backend uses commonJS imports(26/10/2021). So, check for dependency issues since this will be soon deprecated.
- Please `npm install` all packages and dev packages for server and client side before running anything. Refer the Important Commands below for instructions.

## Important Commands

- Server(Backend). Refer `./economy/package.json`. Use in `./economy/` directory:
  - `npm run start` : Starts the server using node.
  - `npm run server` : Starts the server using nodemon.

- `npm run dev` : Uses concurrently package to run both server and client concurrently.
  - `npm run client-install` : Install the necessary dependencies of client react app from this directory.
  - `npm run client` : Starts the client react app in client from this directory.
- **Client(Frontend). Refer `./economy/client/package.json`. Use in `./economy/client/` directory:**
- `npm start` : Starts the client react app.

## Website Navigation

- `/` : Home Page for basic information
- `/auth/` : Authentication Page for both sign in and sign up
- `/dashboard/budgeter/` : Budgeter Page for viewing, creating, editing and deleting user entries. Accessible after authentication.
- `/dashboard/analytics/` : Analytics Page for viewing consolidated data in area chart form in order to get an introspective overview over spending habits. Accessible after authentication.

## File Structure and Relations

### Backend

#### **ALL INSIDE ECONOMY FOLDER**

config

default.json

models

User\_Transaction.js

User.js

reference

routes

API

user\_transactions.js

users.js

middleware

admin.js

auth.js

.env

.env.example

.gitignore  
package.json  
Procfile  
server.js

- Main file is server.js
- It uses APIs defined in `./routes/API`
- APIs require model schema present in `./models`
- APIs also uses middleware functions defined in `./routes/middleware`
- Whole implementation uses some keys defined in `./config/default.json`

## Frontend

### **ALL INSIDE ECONOMY/CLIENT FOLDER**

public

index.html

src

App.js

index.js

actions

auth.js

sidebarDate.js

transactions.js

api

index.js

constants

actions.js

Functions

date.js

Images

body\_background.png

body\_background.svg home\_background.png

home\_background.svg

home\_how\_to\_use.jpg

reducers

index.js

auth.js

sidebarDate.js

transactions.js

Components

package.json

- Components dispatch actions upon clicking buttons in UI.
- actions use axios to use APIs present in backend.
- actions also dispatch type and payload to reducers.
- reducers update the state.
- The state is reused in Components.
- constants are used in both actions and reducers.

Components:

Analytics

Authentication

Budgeter

index.js

DataRows

Datarows.js

TransactionModal

Transactions

Dashboard

Home

Navbar

- App.js is included within index.js
- Navbar, Home, Authentication, Budgeter and Analytics is included within App.js and routed
- Dashboard is included within Budgeter and Analytics

## Release 1 features

- APIs are made for users and user\_transactions data base and both are provided with admin privileges.
- Adequate APIs are built for the **Backend** for existing features.
- Authentication API is also built for the user\_transactions data base.
- Client side is a **React-app**.
- Currently (26/10/2021), 4 webpages are present in the website: Home, Authentication(Sign in, Sign up), Budgeter and Analytics (Analytics is still incomplete).
- Authentication Backend and UI requires no fixes.
- User can view, add, edit and delete entries/transactions in Budgeter.

## Release 2 features

- Improve the verifier for sign up authentication. Should recognize valid email IDs. Should have strong password. (Added but commented)
- Modularized Budgeter into Modal and Grid Components.
- Reformatted overall UI.
- Add Media queries for existing features. (Not implemented)
- Added Analytics.
- Added features to Budgeter such as date sidebar and collapsible groups.

## Future tasks

- User profile section.
- Have an optional option to fill up a form to know user's goal and how to achieve it.
- Add a customer feedback section which should be visible to all users.
- Add a manual section which will be a detailed walkthrough for the website.
- Reinforcement learning agent to make the customer experience more interactive. Can also use it for future projection or learning spending habits.
- Add presets of groups to suit various occupations.
- Add a feature to add or delete groups. (To personalize groups further.)

## Extra Notes (For Devs)

- [Github URL](#)
- Important keys are present in `./economy/config/default.json`:
  - `mongoURI` : My MongoDB Atlas URI
  - `jwtSecret` : Key for json web token
  - `adminPassword` : Password for admin privileges
- Server runs on PORT:5000
- Client runs on PORT:3000

## Resources

### Overview

- Node.js
- Database: mongoDB, MongoDB Atlas
- API: express.js, mongoose
- Central Data Store for state: Redux
- Frontend: React.js

### For Developers

- [Github](#)
- [R1 Video](#)
- [R2 Video](#)

## Contact

Sagar Singh

[cs19b038@iittp.ac.in](mailto:cs19b038@iittp.ac.in)

Indian Institute of Technology, Tirupati.