Budgeted Project Proposal

by

CS 157A - Team 12

Tuong Nguyen
Zehua Liu
Marcus Zhou

Project Overview

Our group is going to develop a robust B2C budgeting app. People nowadays oftentimes have difficulties managing their budget as well as income. Services such as Netflix, Spotify, Amazon Prime and various mortgage payments are mostly monthly recurring, and they are becoming a huge portion of people's spending. With so many transactions taking place in monthly basis, people fail to keep track of all of their monthly recurring spending. Without insightful and informative understanding of their own personal finance, people tend to make more poor financial decision. As a result, people are gradually getting into worse financial situations. To provide a solution, we are creating the app *Budgeted* that will keep track of users' incomes, expenses, and savings. It is able to utilize these data to generate visual graph displaying the ratios between income, saving and expense which ultimately aims to easily make more insightful spending decisions.

System Environment

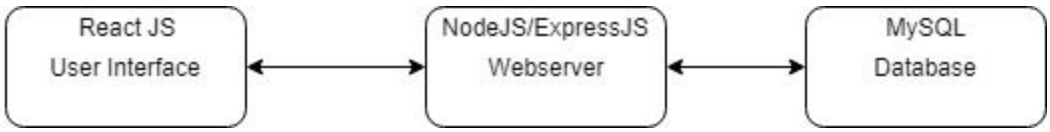

Figure 1: Budgeted App Architecture

- HW used: local machine/deployed on AWS/heroku
- SW used:
    - Version control: Git/Github
    - Front end: React

- Middleware: NodeJS/ExpressJS
- RDBMS: MySql
- Team communication: Slack/zoom meeting
- Misc: SQL workbench, Atom/Sublime text

Functional Requirement

User category includes end users who utilize our database application to make transactions to store their data, and administrator who have privilege-access to create and maintain user accounts.

Users will be able to enter their income recurring spending as well as how much money they want to save. The app will display how much of a "budget" a user has based on their input. User then can log and categorize each daily spendings as well as categorize.

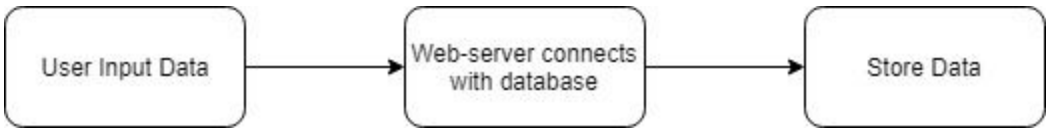

Figure 2: Functional Flow Overview

Non-Functional Requirement

Users will be able to login and securely register for an account. Each account password is hashed/salted and placed inside the DB. This will allow users to be authenticated/authorized to access their own data using Json-web-token(JWT)/0auth.

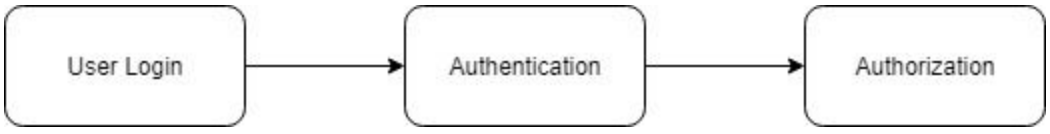

Figure 3: Access Control Overview

GUI user interface allows users to input data and display related information in visual forms using React and interact with the system.