# Project Description Document

## Proposal:

A budget application that allows user to manage their income and expenses.

## Team Name: 6

## Team members:

* Kingsley I – Front end
* Daniel Rodriguez – Backend / Front end / Database
* Choun H. LEE - Backend
* QuangMinh Do – Backend / Front end / Database

## Mission Statement:

We are creating this budget management application to help users to manage their budget. User can add expenses and find out their balance for a specific month.

## Detailed description of your project:

User can create an account, login.

User can create, edit or delete expenses for the month, allocate fund to that expense

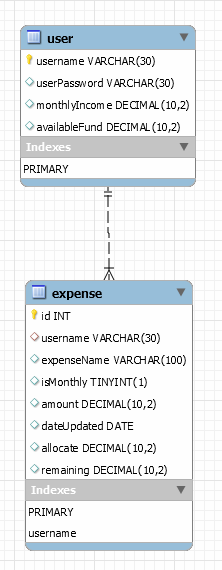
User can have overview of their expense for the current month viewing the pie chart.

## Database:

We will Use MySQL for data storage

Tables: 2

ERD:



## Backend (Java):

Use JDBC Template, connect to MySQL

**Interfaces / implementation**

Services to support controllers

**Classes**

Home Controller

MVC design/structure

Types of testing. (JUnit)

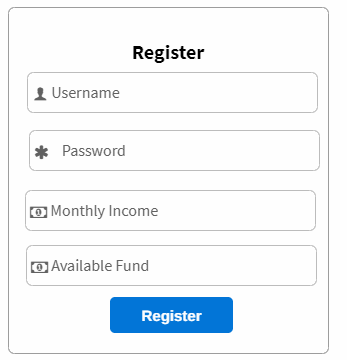
## Frontend React / JS:

### Flow of the App

**Register Page:**

User registers by providing username, password, monthly income and starting available fund.

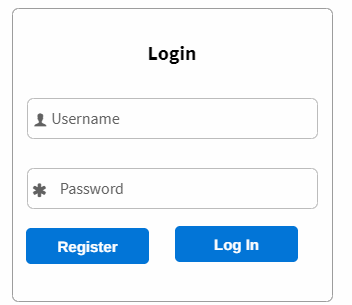
Once registering successfully, redirect to Login page.



**Login page**

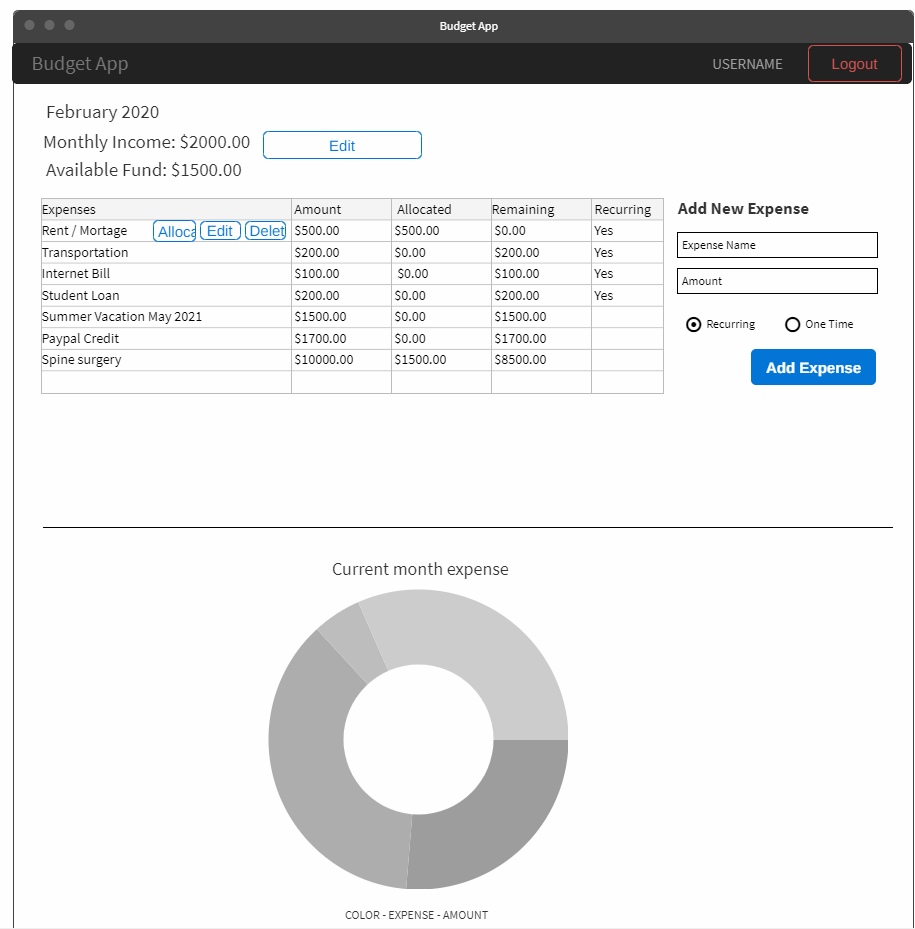
User has to use the created account info to login.

Once logged in, redirect to Home page



**Home page**

Here user can add, allocate, edit, delete expenses.   
View expenses table for the current month.  
Edit income and fund.



## React Components/structure

Login component  
Register component  
Navbar component

**Main component:**

* **Main Component State:** {message: string, month: Feb, monthlyIncome: decimal, availableFund: decimal,  
    
  data: [ {expense1}, {expense2}], newExpense: {expenseName: “”, amount: “”}
* **Main component Functions**: getExpenses(), getIncomeAndFund(), addExpense( when done setState(data)), editExpense (also handle allocation function), deleteExpense
* **Child Components:**
  + Add Expense component (passed in addExpense handler)
  + Expenses table component (render expenses here from the props passed in by the Main component)
  + Income and fund component
  + Edit income and fund component
  + Edit expense component
  + Chart component

## API Endpoints

|  |  |
| --- | --- |
| POST /register | Request:   * + - Type: JSON     - Body Param:  {username: string,  password: string income: decimal, availableFund: decimal }   Response: ERROR   * + - Type: JSON     - Status: 422     - Response:  {message: “Username already taken”}   Response: SUCCESS   * + - Type: JSON     - Status: 201     - Response:  {message: “Username created”, username: string} |
| POST /login  Handle Login | Request:   * + - Type: JSON     - Body Param:  {username: string,  password: string”}   Response: ERROR   * + - Type: JSON     - Status: 401     - Response:  {message: “Invalid username or password”}   Response: SUCCESS   * + - Type: JSON     - Status: 200     - Response:  {message: “Login successful”,  username: string} |
| GET /income | Request:   * + - Type: JSON     - Body Param:  {username: “username”, month: date}   Response: ERROR   * + - Type: JSON     - Status: 404     - Response:  {message: “Invalid request”}   Response: SUCCESS   * + - Type: JSON     - Status: 200     - Response:  {monthlyIncome: 2000,  availableFund: 1500} |
| PUT /income  Edit income | Request:   * + - Type: JSON     - Body Param:  {username: string, monthlyIncome: decimal, availableFund: decimal, month: date}   Response: ERROR   * + - Type: JSON     - Status: 404     - Response:  {message: “Invalid request”}   Response: SUCCESS   * + - Type: JSON     - Status: 200     - Response:  {message: “Update successfully”} |
| GET /expenses  Use for displaying expense table for that month  Also use for displaying chart | Request:   * + - Type: JSON     - Body Param:  {username: string, month: date}   Response: ERROR   * + - Type: JSON     - Status: 404     - Response:  {message: “Invalid request”}   Response: SUCCESS   * + - Type: JSON     - Status: 200     - Response:  { **~~month: date,~~ ~~data~~**~~:~~ [ expenseId: int,  expenseName: string, remaining: decimal,  amount: decimal,  allocated: decimal,  isMonthly: Boolean,  dateUpdated: dateString], [] , [] } |
| POST /expenses  handle Add One New Expense | Request:   * + - Type: JSON     - Body Param:  {username: string,   expenseName: string, amount: decimal isMonthly: boolean, month: date}  Response: ERROR   * + - Type: JSON     - Status: 404     - Response:  {message: “Invalid request”}   Response: SUCCESS   * + - Type: JSON     - Status: 201     - Response:  {message: “Expense created successfully”} |
| PUT /expenses  Handle edit and allocation function for one expense | Request:   * + - Type: JSON     - Body Param:  {username: string,   expenseId: int,  expenseName: string, amount: decimal,  **allocated: decimal**, isMonthly: Boolean, month: date}  Response: ERROR   * + - Type: JSON     - Status: 422     - Response:  {message: “Invalid request”}   Response: SUCCESS   * + - Type: JSON     - Status: 200     - Response:  {message: “Expense updated successfully”} |
| DELETE /expenses | Request:   * + - Type: JSON     - Body Param:  {username: string,   expenseId: int, month: date }  Response: ERROR   * + - Type: JSON     - Status: 422     - Response:  {message: “Invalid request”}   Response: SUCCESS   * + - Type: JSON     - Status: 200     - Response:  {message: “Expense delete successfully”, ~~expenseId: int~~ } |