# **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הגשת פרויקט**

**Personal Finance App Manger A5** קבוצה מספר 4

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
| **עדן בטו** | 207529959 |
| **עבדאללה אבו אלהיגא** | **211575790** |
| **ראזי מוגרבי** | **323016535** |

***תקציר:*** האפלקציה שלנו היא אפליקציית ניהול פיננסי אישי, היא מאפשרת למשתמש לעקוב אחרי ההוצאות שלו, הצבת יעדים פיננסים, והצבת תכנון הוצאות(budget)

***טכנולוגיות מרכזיות:***

Frontend : React, HTML

Backend: JavaScript, Express JS

Database: MongoDB

Styling: Tailwind CSS

***לינק לMTW:***

<https://www.morethanwallet.com/app/695>

***לינק לגיט צד לקוח:***

<https://github.com/razimograbi/ReactPersonalFinaceWeb.git>

***לינק לגיט צד שרת:***

<https://github.com/razimograbi/PartialBackendForWeb.git>

***לינק לאתר:***

<https://react-personal-finace-web.vercel.app/>

Username: [john@example.com](mailto:john@example.com)

Password: 123456

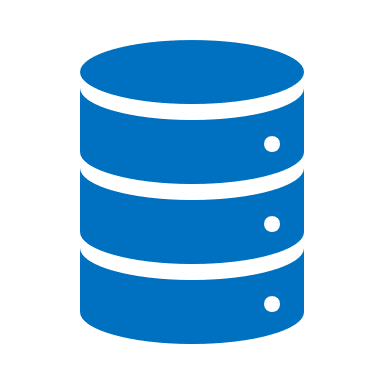
|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו (סבב עבודה מספר 3)** | **משימות שהושלמו** |
| עדן בטו | המרת הפרויקט ל React  הוספת עוד שני פיצ'רים בדף goals  מימוש קומפוננט Modal ושימוש בו במספר מקומות בפרויקט  תיקוני באגים  פינישים ותוספות לממשק משתמש | הכל |
| עבדאללה אבו אלהיג'א  מהנדס מערכת | המרת הפרויקט ל React  שליפת נתונים מMongo DB והצגת אותם בדף הראשי  תיקון כמה פיצ'רים שהיו תקולים | הכל |
| ראזי מוגראבי | המרת הפרויקט ל React  מימוש ודיפלוי SERVER  תיקוני באגים | הכל |

2. הציגו רשימת דרישות פונקציונליות ולא פונקציונליות (לסווג לא פונקציונליות לפי wikipedia NFR).

* **Functional and non-Functional requirements:**

|  |  |  |
| --- | --- | --- |
|  | functional requirement | Non-functional requirement |
| 1. | Users should be able to create an account. | The account creation process should take no longer than 1minute to complete.(Performance) |
| 2. | Users should be able to create and customize budgets for different expense categories. | The system should maintain at least 99.9% uptime, ensuring that it is accessible to users 24/7 ( Availability) |
| 3. | Users should be able to manually enter expenses, specifying details such as the amount, date, category. | Loading and saving budget configurations should be near-instantaneous, with minimal latency.(Performance) |
| 4. | Users should be able to set specific financial goals. | The features across the website should work seamlessly across different devices and screen sizes, including desktops, laptops, tablets, and smartphones.(Usability and Adaptability) |
| 5. | Users should be able to view their expenses | The expense entry form should be user-friendly, with clear labels and input fields to facilitate accurate data entry. (Usability) |
| 6. | The website should include a user toggle or setting that allows users to switch between light mode and dark mode. | The system should be designed to accommodate future changes in requirements without requiring extensive redevelopment or disruption to existing functionalities.(Flexibility) |
| 7. | Users should be able to see charts depicting their expenses | Regular backups of user account data should be performed to ensure data integrity and availability in the event of system failures or data loss incidents. (Backup and Data integrity) |
| 8. | The system should provide users with a confirmation prompt when they’re about to take an action | The system allows for easy maintenance and updates, with modular code structures and clear documentation for developers (maintainability). |

הציגו ארכיטקטורה מעודכנת של האתר (תרשים הכולל את האלמנטים המרכזיים)



Web Client

Frontend – React.js, HTML, Tailwind, JavaScript

User

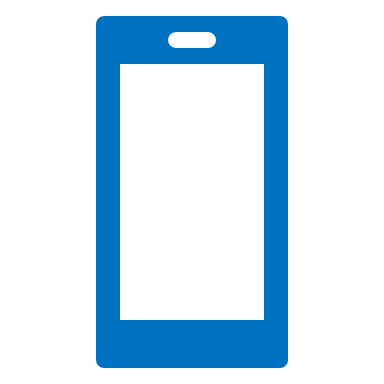
Display results

Web Server- Express JS

Backend

App Logic – JavaScript, Node.js

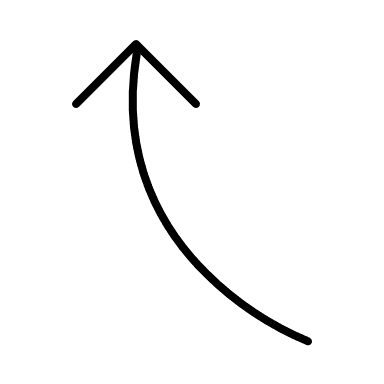
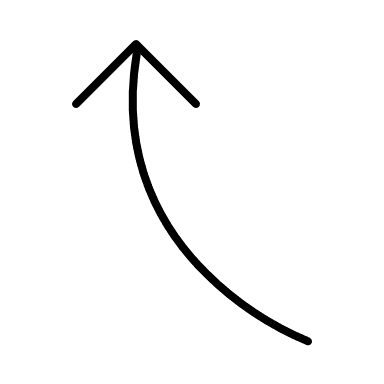
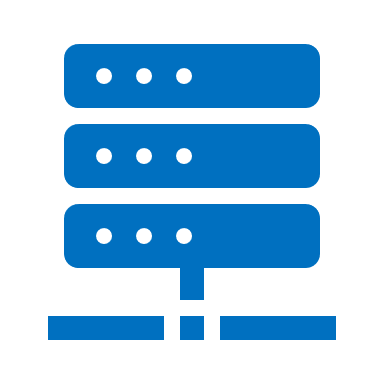
Database - MongoDB



Request

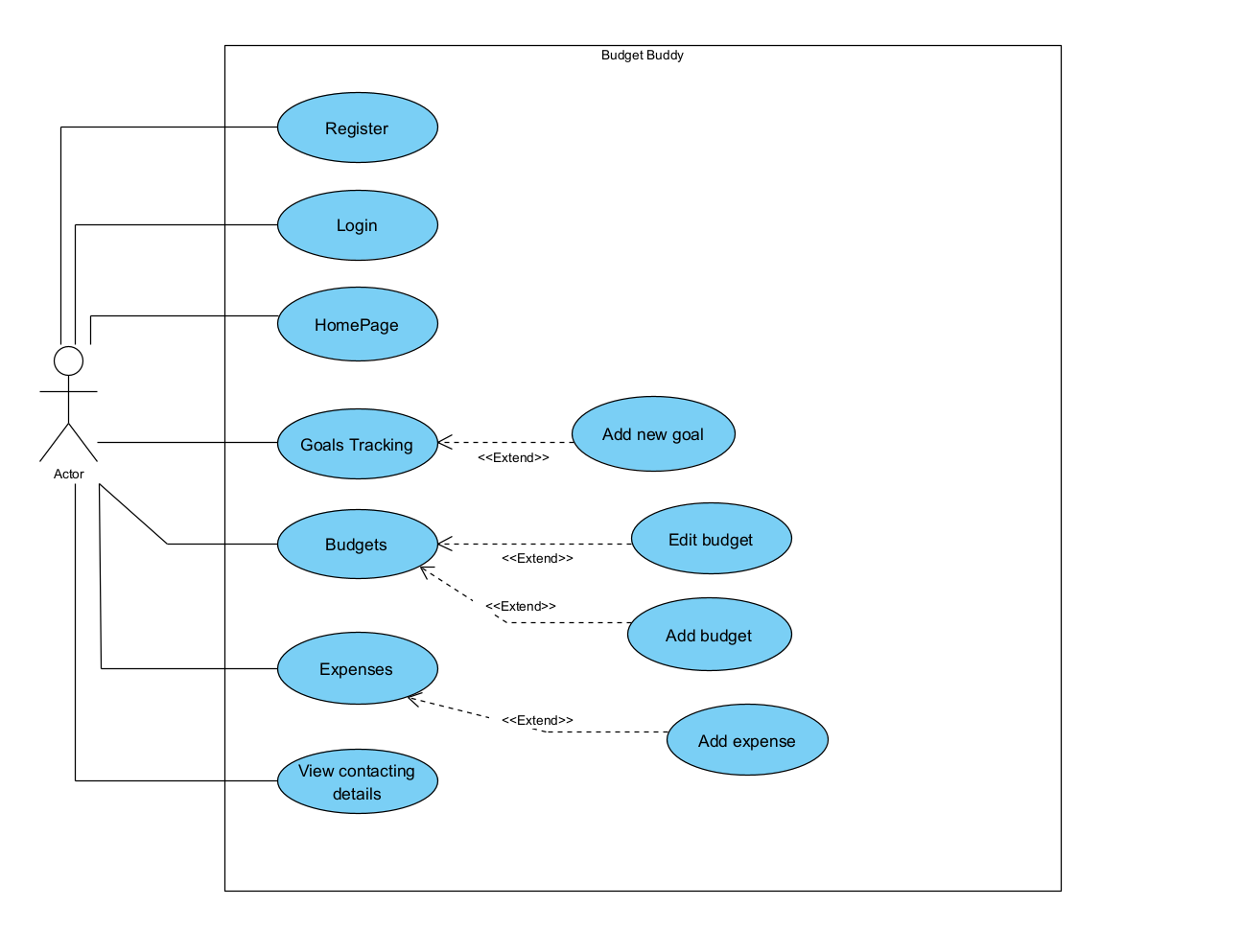
Response

JWT authentication

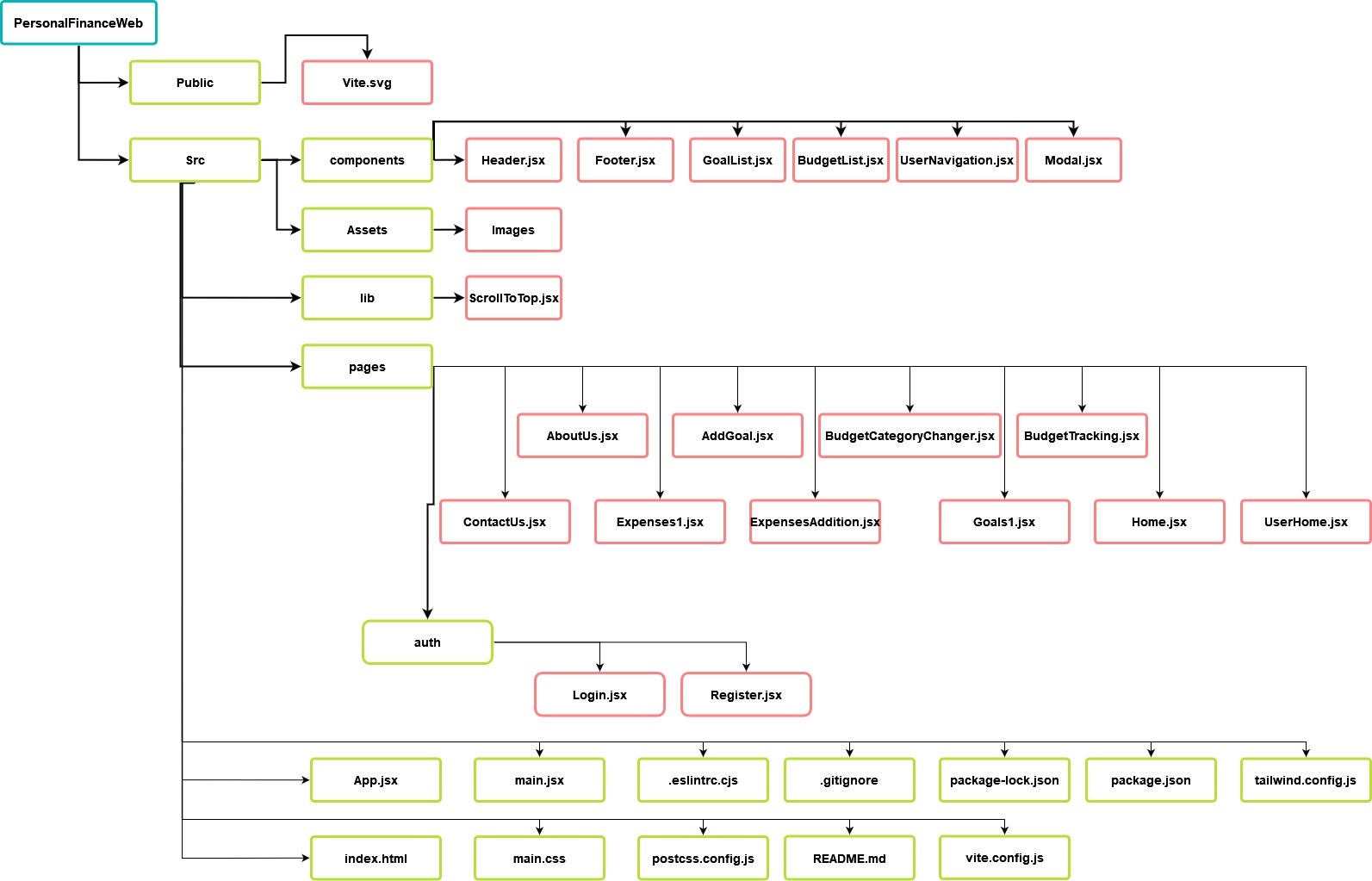


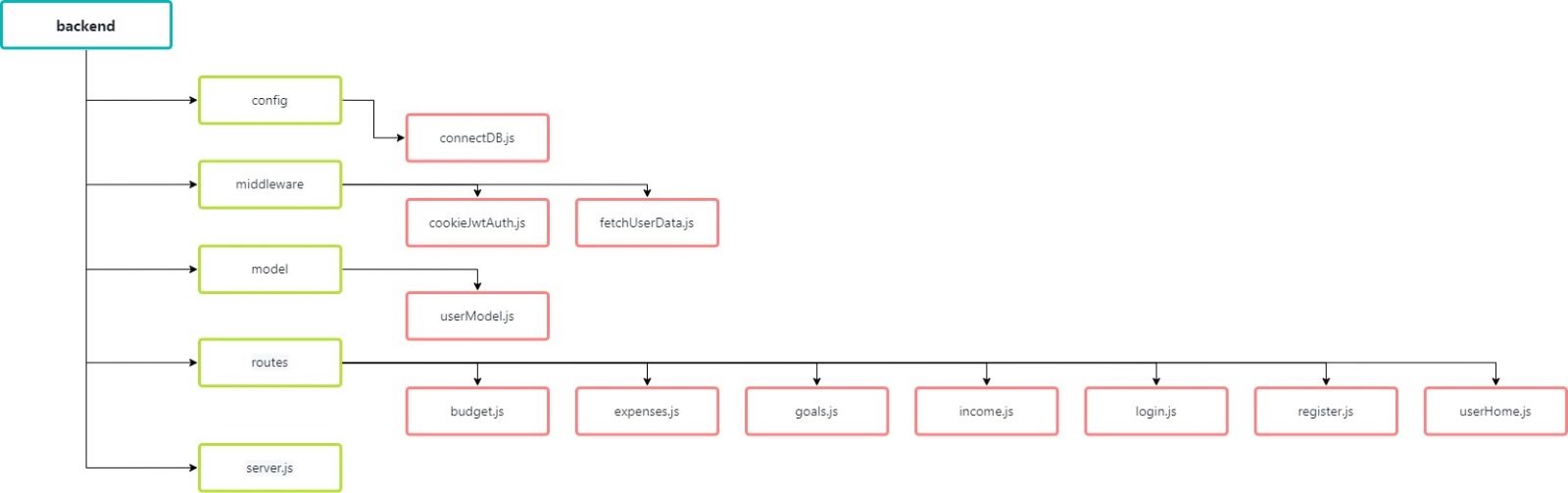
* Node.js played a pivotal role in our project on both the client and server sides. On the client side, Node.js served as the runtime environment for tools like npm, enabling the installation and management of frontend dependencies such as React and Tailwind CSS. It provided a platform for executing scripts and running development servers. On the server side, Node.js powered the backend logic, facilitating the execution of server-side JavaScript code via frameworks like Express.js. It handled incoming HTTP requests, processed data, and interacted with databases or external services, serving as the backbone of your server infrastructure.
* JWT (JSON Web Token) is a compact, URL-safe means of representing claims between two parties. It's used for authentication and authorization in our web applications where the token renews every hour for the user (so the account isn’t forgotten logged-in and someone else might have access to sensitive information).
* we utilized Express.js for server-side development, simplifying tasks like HTTP request handling and routing. Its minimalist framework enabled rapid development and maintenance of server components, ensuring scalability and flexibility. Additionally, leveraging its extensive ecosystem of plugins and middleware enhanced the project's functionality, contributing to its success.
* For our database we used MongoDB
* For styling we used Tailwind CSS which made the UI development easier.
* Deployment of the client side was via Vercel, deployment of server side was via Render

3. הציגו דיאגרמת use case המתארת את השימוש באתר.



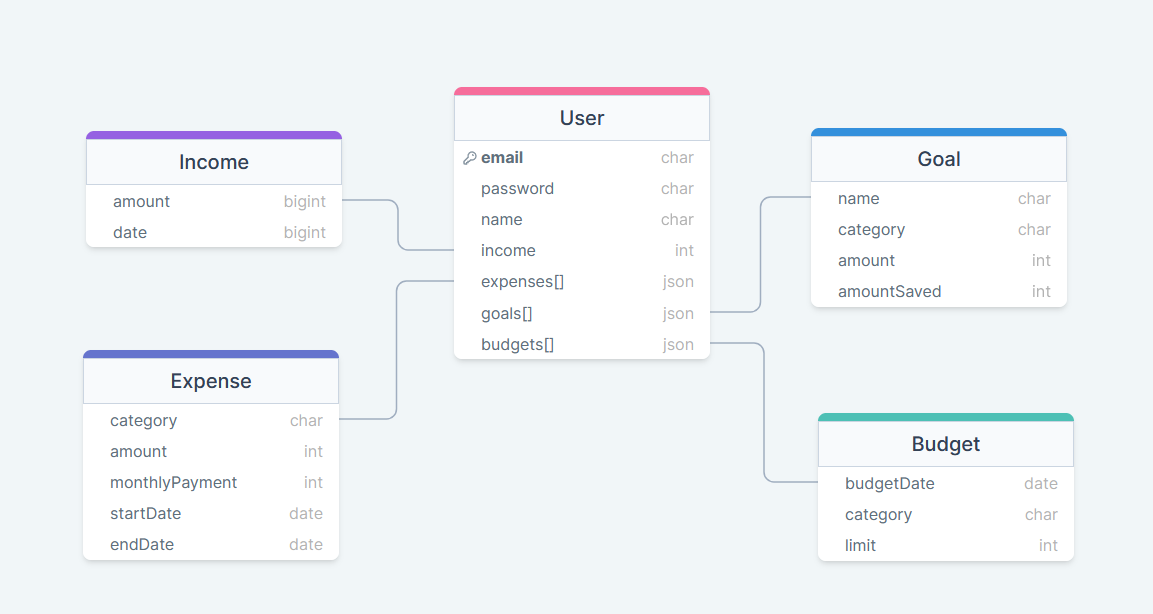
4. יש להציג מבנה סופי של האתר שלכם:   
א. האתר ימומש ב -react/preact, וכן שימוש ב Tailwind – נא להציג דיאגרמה המתארת את התיקיות והקבצים השונים.

*****Client side files and directories diagram***

***Server side files and directories diagram***

ב.

DB Diagaram – we have used non-relational database (MongoDB) , the relational structure in the diagram is solely for illustrating architecture.



יש להגיש תיק למתכנת:

1. יש לתעד ולהסביר את כל הפונקציות המרכזיות בקוד שלכם.

We created our frontEnd using ReactJs , here are the main reusable components that we used (in later pages, there the components that make up our site pages)

1. BudgetList.jsx
2. Footer.jsx
3. GoalList.jsx
4. Header.jsx
5. Modal.jsx
6. UserNavigation.jsx
7. **BudgetList.jsx :**

function BudgetList({ currentlySelectedMonth, currentlySelectedYear }) {...}

**Explanation** : its a react component that displays a list of budgets based on a selected Month and year, also allowing the user to edit existing budgets and track new budget categories.

**Key Methods:**

1. fetchData : Fetches budget data from the API based on the currently selected month and year.
2. displayBudgets : Processes the fetched budget data and expenses to calculate and display spent percentages.
3. getExpensesBasedOnMonthAndYear : Fetches expense data for the given month and year.
4. calculateSpentPercentages: Calculates the percentage of the budget spent for each category.
5. getToken: Retrieves the authentication token from local storage.
6. handleAddBudget: Handles the addition of a new budget by using a POST request to the API.
7. handleBudgetEdit : Handles the update of an existing budget by using a PUT request to the API.
8. **Footer.jsx :**

const Footer = () => {...}

**Explain**: its a responsive footer component which we usually add at the bottom of the page.

1. **GoalList.jsx :**

const GoalList = () => {...}

**Explain :** its a react component that displays a list of financial goals, each with options to add money toward the goal or delete the goal.

**Key Methods :**

1. handleDeleteGoal : Sends a DELETE request to the API to remove a specified goal.
2. handleAddMoney : Sends a PUT request to update the saved amount of a specified goal.
3. fetchGoalsData : Fetches the current list of goals from the API.
4. **Header.jsx:**

**Explanation**: its a react component that we use to navigate between pages , its also responsive, only inside the homeScreen.

1. **Modal.jsx :**

const Modal = ({ isOpen, handleModal, content, handleSubmit ,positiveLabel, negativeLabel }) => {…}

**Explanation** : it’s a react component designed to display dynamic content and provide a mechanism for user interaction through buttons that trigger specific actions(like confirm/decline) .

It appears centered on the screen and can be shown or hidden based on its state.

**Key Properties:**

1. isOpen (boolean): Controls the visibility of the modal. If true, the modal is shown; if false, the modal is hidden.
2. handleModal(function): A function to be called when the modal needs to be closed, triggered by clicking a negative (cancel) button or the modal backdrop.
3. Content(JSX) : The content to be displayed within the modal.
4. handleSubmit(function) : A function to be called when the user clicks the positive (confirm) button.
5. positiveLabel(String) : The text to display on the positive (confirm) button.
6. negativeLabel(String) : The text to display on the negative (cancel) button.
7. **UserNavigation.jsx :**

const UserNavigation = () => {…}

**Explanation**: it’s a react component that serves as the navigation bar, It provides links to various sections of the application such as Home, Budgets, Goals, and Expenses, and features a user profile dropdown menu for accessing settings and account management options and also to apply **dark mode** .

**Key Methods :**

1. toggleProfilePicDropdown: Toggles the visibility of the dropdown menu associated with the user profile picture. This allows users to access additional options like settings or sign out or dark mode.
2. toggleTheme: Switches the applications theme between light and dark modes. This setting is saved in local storage.

*Explanation about the components that make up the site’s pages :*

Lets start with the **App.jsx** :We used React-router to manage switching between each page and maintaining a single-page application.

**Login.jsx:** it handles/displays the login of the user . We used axios to send http requests to the server. Also in the login we store the token that we get from the server in the localStorage so we can access it later.

**Register.jsx :** this page handles registration for a new user. If an existing user tries to register it wont let him do that.

**AboutUs.jsx:** a page explaining about the journey of designing and creating this website

**AddGoal.jsx** : Handles adding a new goal to a user. (uses the modal.jsx component)

**BudgetCategoryChanger.jsx** : Change the budget category for an existing budget .

**BudgetTracking.jsx** : In this page we let the user view his budgets and spending based on a specific month and year that he chooses. We display the budget and category of that budget and underneath each budget we will show how much money did the user spend, to give the user information about how much did he want to spend versus how much did he actually spend.

**ContactUs.jsx** : A dummy contact us page to contact developers.

**ExpensesAddition.jsx** : A page where the user can Add a new expenses based on many categories like (loan, Food, Rent and more). Uses the Modal.jsx component.

**Goals1.jsx** : A page where the user can manage his financial goals (add/delete goal). Uses the Modal.jsx component .

**Expenses1.jsx** : A page where the user can view his expenses / add expenses, the page displays the expenses in a table.

**Home.jsx** : When the user first enters the website he will land in this page, this page is designed to be minimalist with a parallax effect when the user scrolls down, The user needs to click on sign in to enter the main Website.

**UserHome.jsx** : After the user Signes in , He will land in this Page this page has a UserNavigation.jsx Component that lets the user navigate between pages, it also displays the Expenses of the user.

**תיק מתכנת SERVER**

We built our server using Express JS , It does not serve our html pages but it’s a JSON API that Retrieves /updated/deletes objects from the Database and it manages the users and authentication.

We implemented our server using an MVC approach.

Let’s Start with configuration server file called **server.js:**

This file sets up the Express server with necessary middleware for request parsing, authentication, and routes for handling user interactions such as login, registration, and financial data management (for budgets/goals/expenses).

We use dotenv, to safely store the (DB\_URL, PORT, SECRET\_KEY). The secret key is for creating a JWT token.

In the server.js file we have this method:

const startServer = async () => {…}

its a Function to start the server, connect to DB and handle connection issues.

**Middleware:**

**cookieJwtAuth.js**: Middleware to authenticate requests using JWT tokens stored in cookies.

**fetchUserdata.js**: Middleware to fetch user data from the database.

**Model:**

**userModel.js:** This file defines the User model using Mongoose for MongoDB interactions, The User schema includes fields for storing user details (email, name, password), and nested structures for managing personal financial data such as income, expenses, goals, and budget.

**Routes:**

Budget.js: this file has the methods for adding deleting editing (retrieving based on month and year) + retrieving overall budgets for the user.

Expenses.js: This file has the methods for adding (retrieving based on this current month) + retrieving overall expenses for the user.

Goals.js: This file has methods for adding/deleting/editing and retrieving financial goals for the user.

Income.js: This file has methods for adding and retrieving the user income.

**Login.js**: This route handles login requests, The route extracts email and password from the request body. If either is missing, it responds with a 400 status code and an error message indicating both fields are required. Next, It attempts to find a user in the database by the provided email and checking if the provided password matches that one in the DB. Next comes the **Token Creation and Response**, If the credentials are correct, it generates a JWT token using jsonwebtoken.sign. This token includes the users name and email, and it is set to expire in one hour.

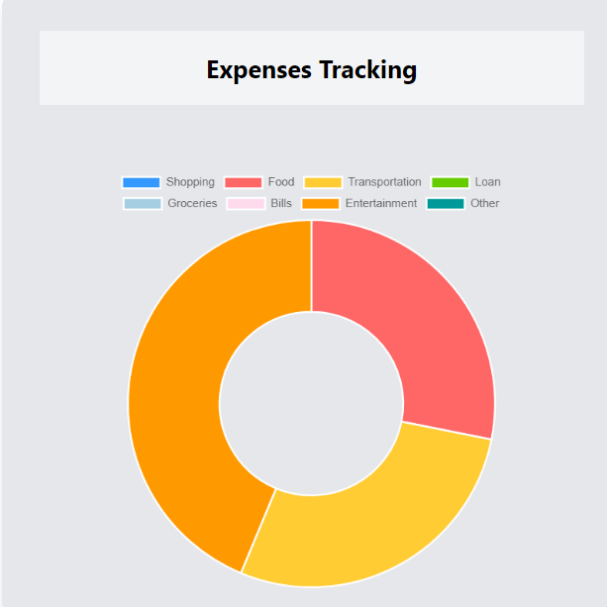
The token is then sent to the client as a HTTP cookie and included in the JSON response along with a success message and user details (email and name).

Register.js: this file handles a register request from the user. (taking the necessary info and checking if there is already a user with the same info if yes then it won’t allow him to register if no , then we will create a new user in the DB).

userHome.js: this route has one method that gets all the information about the user, (name, email, incomes, budgets, expenses, goals).

ב. יש לפרט את כל ה -API שהשתמשתם בהם, קישור ל- DB,סביבות מיוחדות שהתקנתם, קטעי קוד מיוחדים שלקחתם ממקורות שונים (כולל רפרנסים) , פרומפטים שהשתמשתם בהם בכלי AI.

* We have used the tools available at chartjs site in order to help us create charts across our website. <https://www.chartjs.org>



A graph with red bars

Description automatically generated

* We have used Mongo DB to save our data, as far as our search went, MongoDB does not give the option to share a public link (at least we couldn’t figure it out. if there’s a way we are unaware of, send us an email and we will happily provide)
* We have installed React.js, Tailwind CSS, Node.js in order to create our project.
* We used multiple code sections from different sources including: <https://tailwindcss.com/>, https://flowbite.com/ .

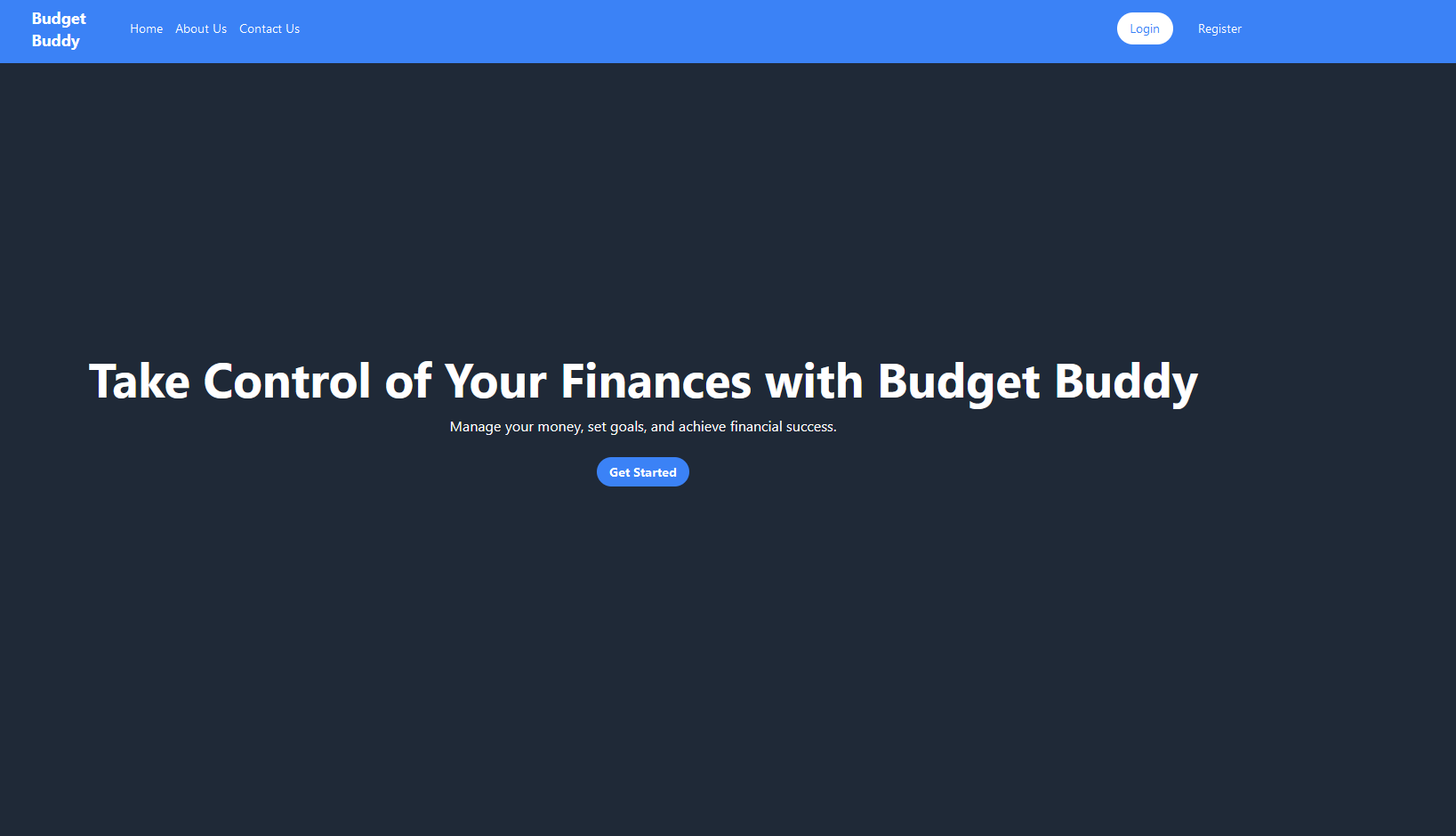
Code sections that used a base code from these sources were: The header, the footer, the contact us page, pages that had cards templates (such as login page, register page, add goal, add expense pages)

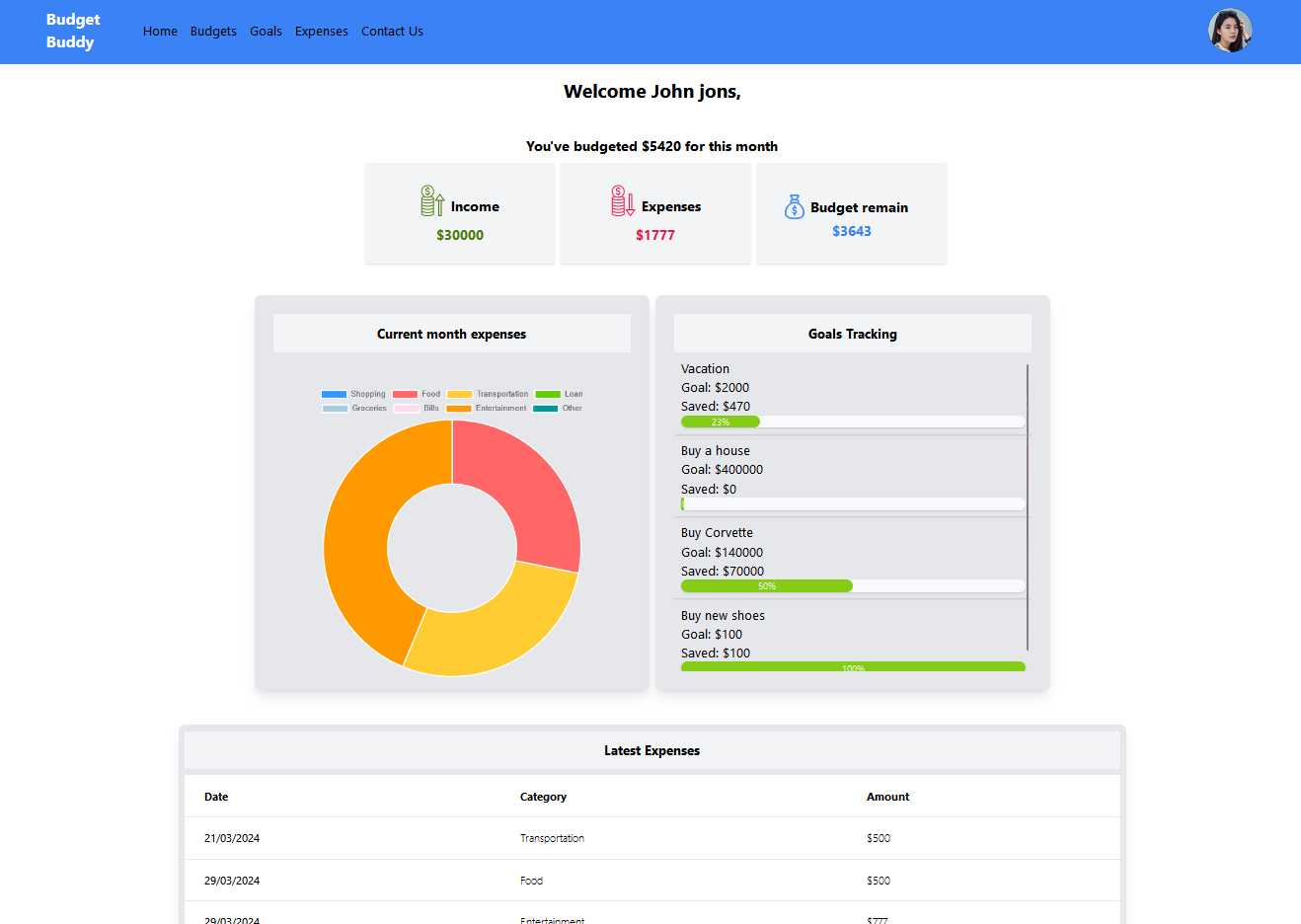
* A person holding a phone

  Description automatically generatedWe’ve used the help of various sources over the internet to implement the feature of “Parallax scrolling” in our home page were you can see a sliding window over a static image.
* We used ChatGPT to help us solve bugs and inquire about information/features/implementations we had previously no background of.(Sometimes it was helpful other times not so much)

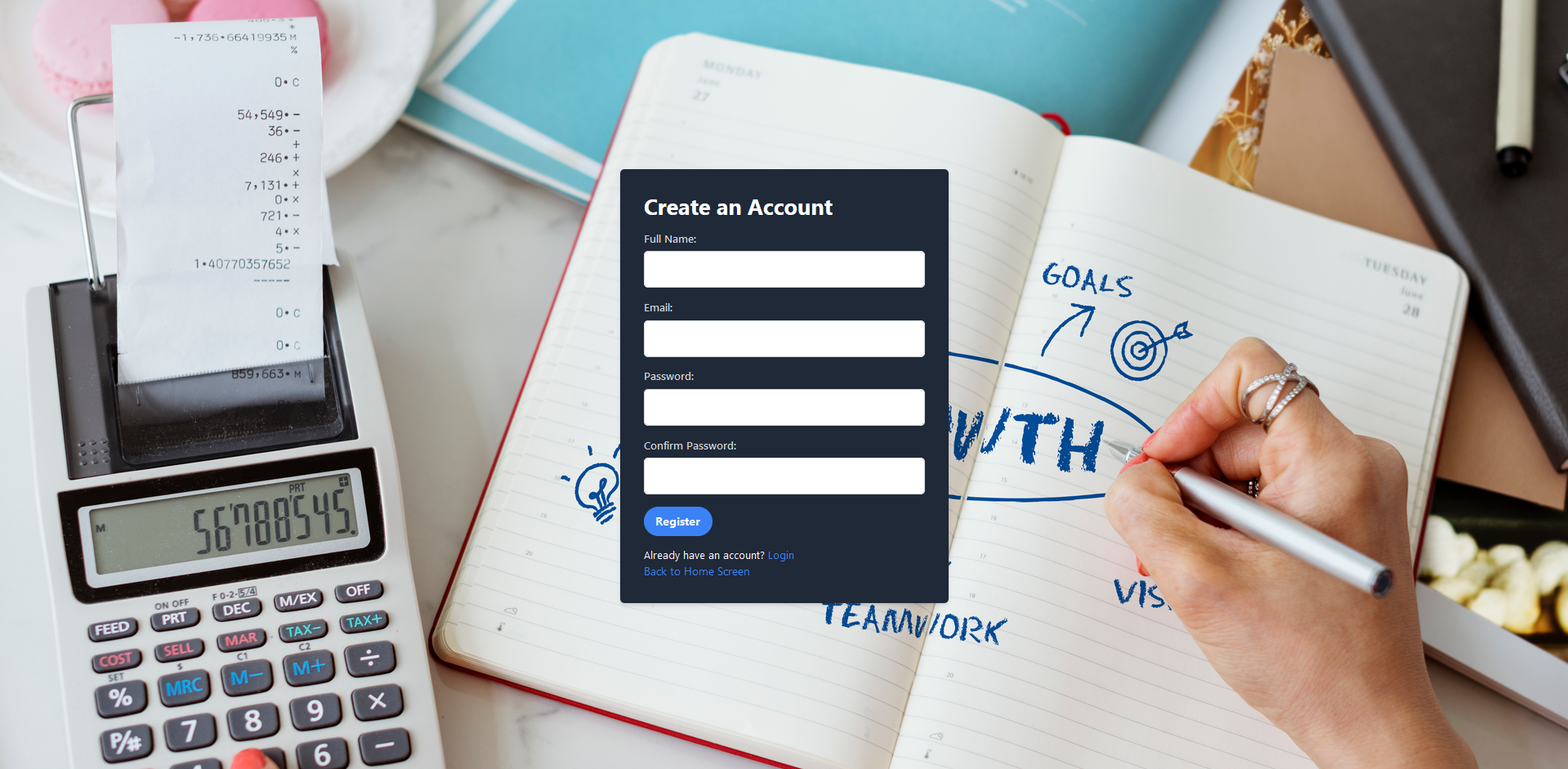
6. יש להגיש תיק משתמש הכולל צילומי מסך והסברים כיצד להשתמש באתר שלכם.

Our landing page

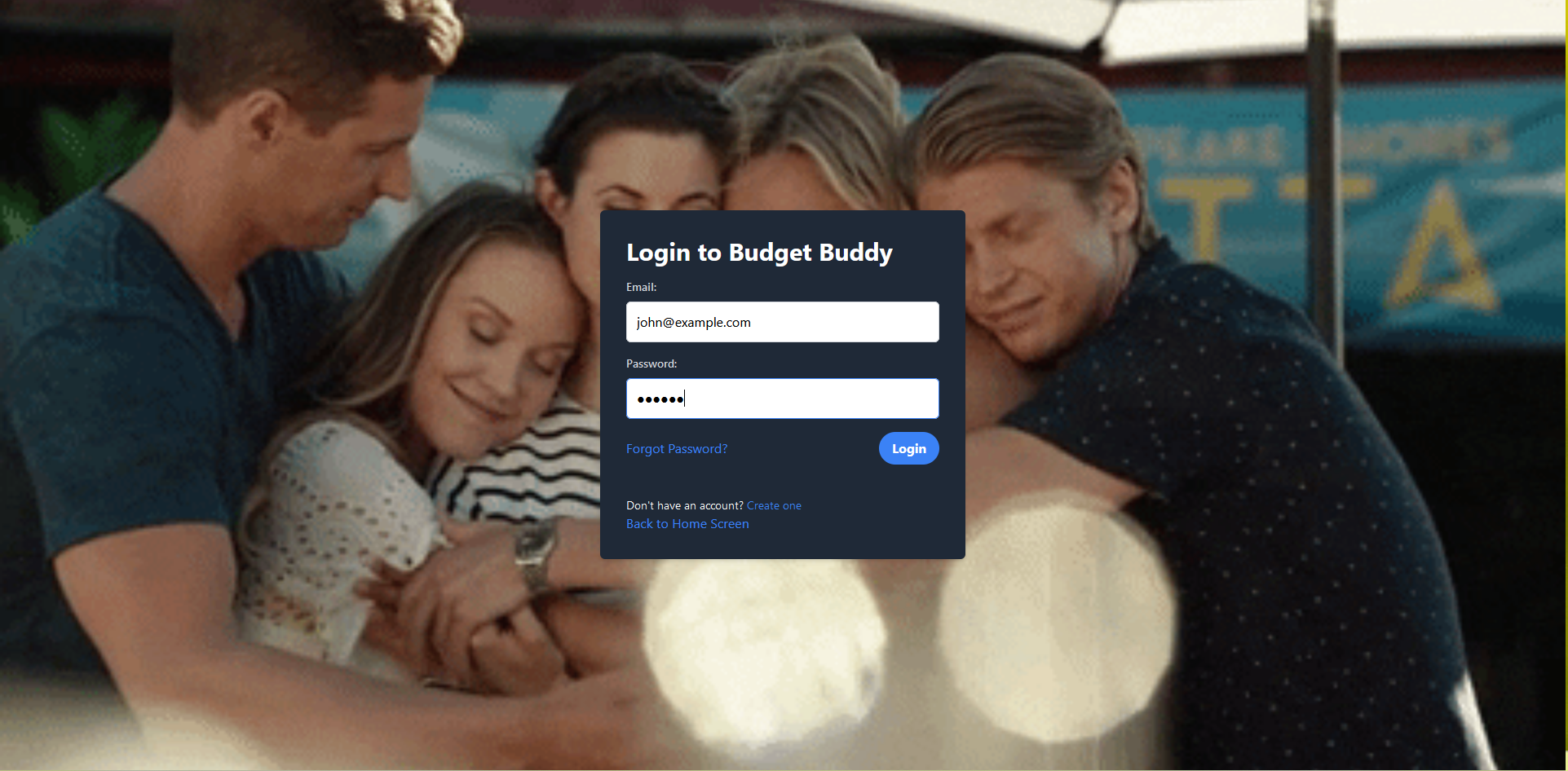


Users home page screen is a summary of the current month’s data

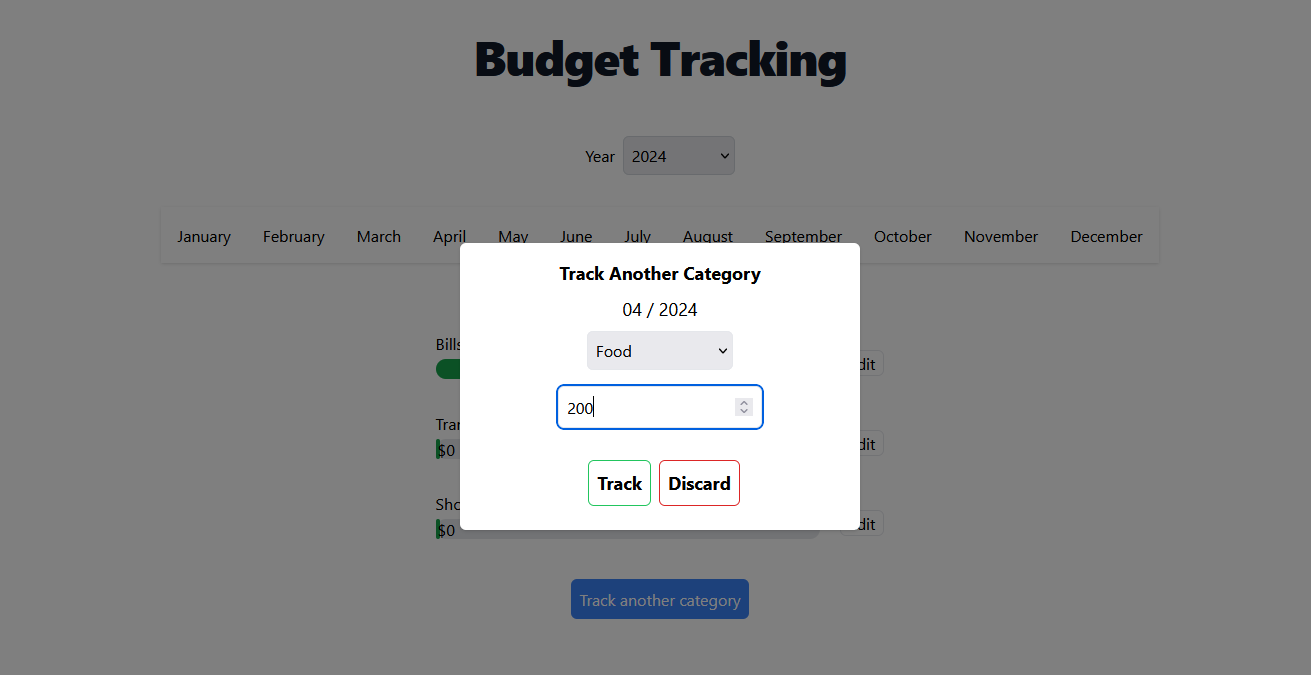
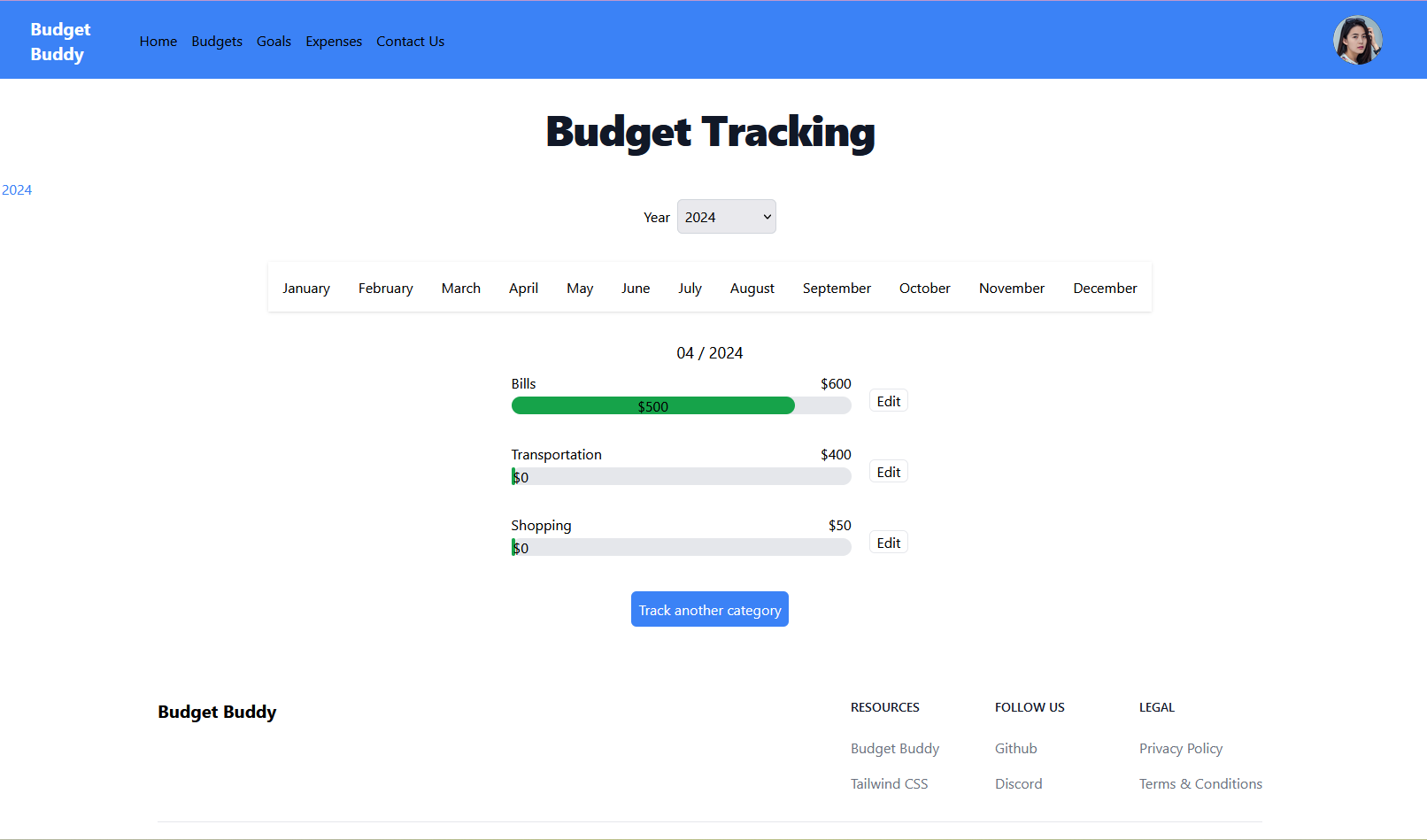
Register screen



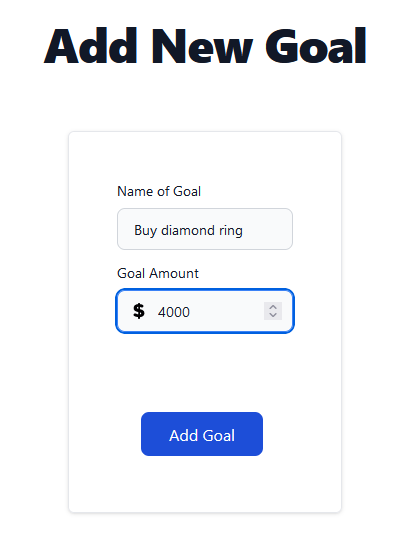
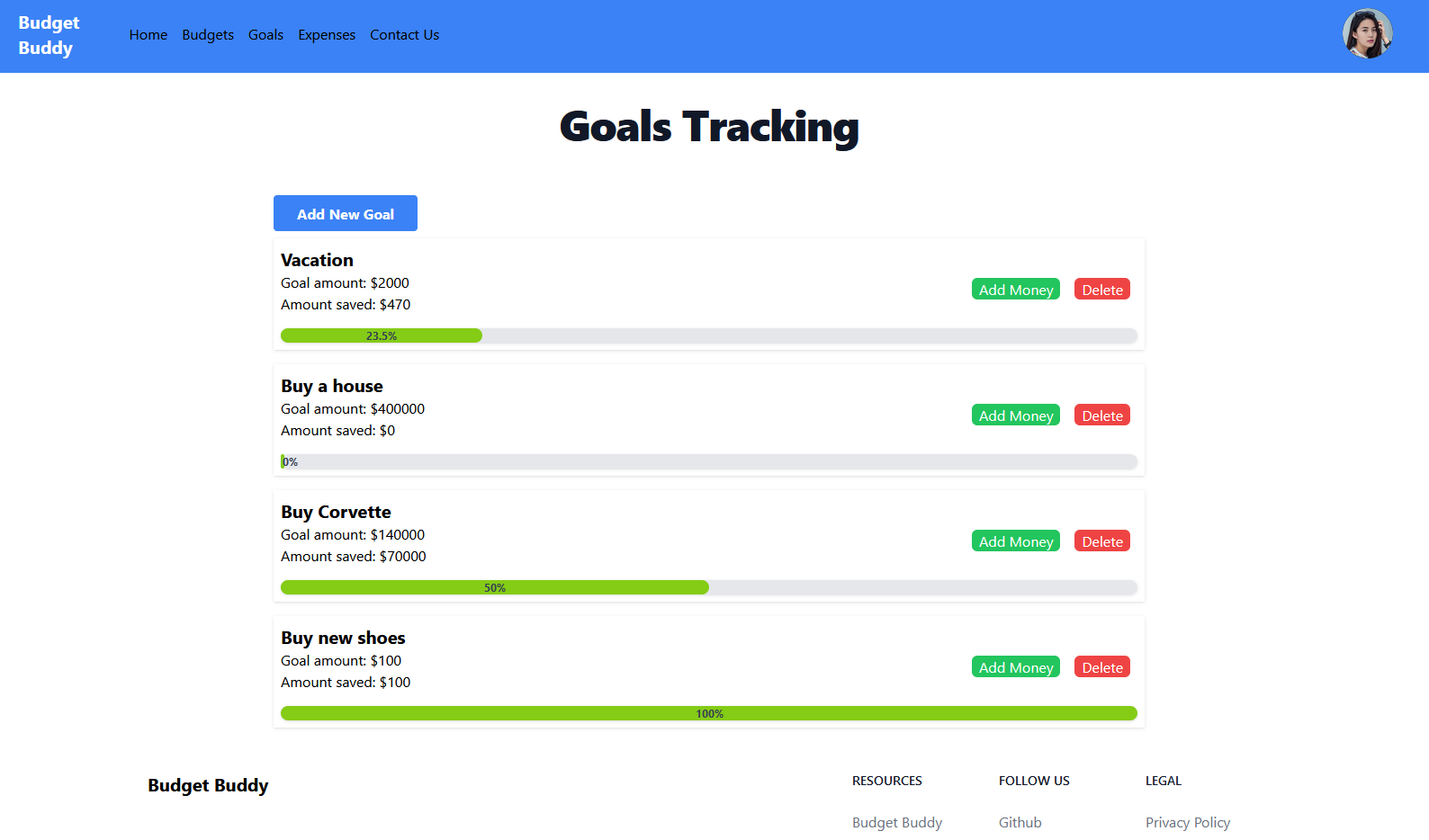
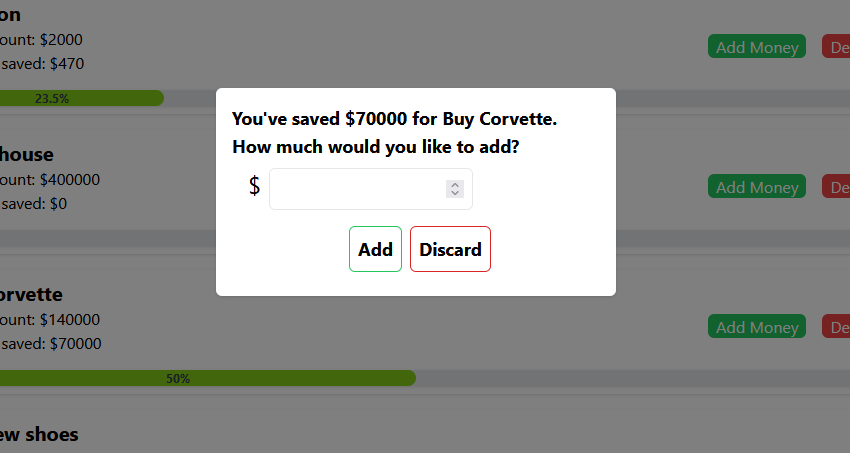
Our Login screen



Users can set their budget for different categories in certain month and year and track them based on their expenses in Budgets page



Users can view their financial goals, add new goals, or add to the saved amount of the goal



Users can track their expenses and view their income/expenses distribution across the months, and a doughnut chart displaying each category’s part in the expenses, and two tables displaying past expenses, and upcoming ones. Users can also add a new expense.

