

INTER DISCIPLINARY PROJECT REPORT

At

**Sathyabama Institute of Science and Technology
(Deemed to be University)**

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

**Som Mahanti Sai Vinit
(Reg. No. 40731099)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

April 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Som Mahanti Sai Vinit (40731099)** who carried out the project entitled "**Expense Tracker**" under my supervision from Jan 2023 to April 2023.

Internal Guide

Dr.A.Annie Micheal ME., Ph.D.

Head of Department

Dr. S. Vigneshwari M.E., Ph.D.

Submitted for Viva voce Examination held on 17-04-2023

Internal Examiner

External Examiner

DECLARATION

I **Som Mahanti Sai Vinit** hereby declare that the Project Report entitled “**Expense Tracker**” done by me under the guidance of **Dr.A.Aannie Micheal ME., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 17-04-2023

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.Annie Micheal M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

An Expense Tracker project, which is designed to help users track their income, expenses, and available funds. The user interface consists of a header with two images and a container with several sections.

The budget numbers section displays the money earned, available, and spent, with the amounts displayed in Indian rupees. The input area allows users to enter a description and amount for an income or expense item. The buttons area includes two buttons to add an income or expense item. The items container section displays the income and expense items added by the user, with each item including a title, date, and amount.

Additionally, the document includes a chart section that displays a graphical representation of the income and expense items. The chart is created using the Chart.js library.

Overall, the HTML document provides a functional and visually appealing interface for users to track their income and expenses. With its user-friendly design and efficient tracking features, the Expense Tracker project can be a valuable tool for managing personal finances.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE NO
	ABSTRACT	I
	CONTENTS	II
	LIST OF FIGURES	IV
1	INTRODUCTION	1
	1.1 EXPENSE TRACKER	1
	1.2 ESTIMATING INFORMATION	1
	1.3 AREA OF RESEARCH	2
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	6
	2.1 AIM	6
	2.2 SCOPE	6
3.	MATERIALS AND METHODS, ALGORITHMS USED	8
	3.1 SOFTWARE REQUIREMENTS	8

	3.2. USER INTERFACE	9
	3.3. LIBRARIES USED	9
	3.4. IMPLEMENTATION OF METHODS	13
4.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	16
5.	SUMMARY AND CONCLUSIONS	17
B	SCREENSHOTS	19
C	SOURCE CODE	26
D	REFERENCES	31

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	Ideation map	8
3.2	Admin login	23
3.3	System architecture	6
3.6	Web interface	26
4.1	Result	40

CHAPTER 1

INTRODUCTION

Managing personal finances can be a challenging task, but it is essential for achieving financial stability and reaching long-term financial goals. An expense tracker can be a valuable tool to help individuals monitor their income, expenses, and available funds. The Expense Tracker project is designed to provide an efficient and user-friendly solution for tracking personal finances.

The Expense Tracker project uses HTML, CSS, and JavaScript to create a web-based application that allows users to input and track their income and expenses. The interface includes various features, including a budget numbers section that displays the user's money earned, available, and spent. The input area allows users to enter a description and amount for income and expense items, while the buttons area provides a straightforward way to add income or expense items.

The items container section displays the income and expense items added by the user, including a title, date, and amount. Finally, the chart section provides a graphical representation of the income and expense items, enabling users to visualize their financial situation.

By using the Expense Tracker project, individuals can effectively manage their finances, stay within their budget, and make informed financial decisions. The project is a valuable tool for anyone who wants to take control of their personal finances and achieve their financial goals.

1.1 AREA OF RESEARCH

In the area of web development, one area of research for an expense tracker project could be the implementation of real-time data synchronization between multiple devices. This would allow users to seamlessly access and update their expense data across different devices, such as their desktop computer, laptop, tablet, or mobile phone.

Another area of research could be exploring the use of web-based data visualization tools to present users' expense data in an easy-to-understand and engaging manner. By using tools such as charts, graphs, and infographics, developers could help users better understand their spending patterns and identify areas where they can save money.

Additionally, researchers could investigate the use of artificial intelligence and natural language processing to provide users with more personalized and interactive expense tracking experiences. For example, an expense tracker could use machine learning algorithms to analyze users' transaction data and provide customized spending advice or alerts based on their unique financial situation and goals.

Finally, another area of research could be exploring the impact of user interface design on users' adoption and usage of an expense tracker. By conducting user research and usability testing, developers could identify the most effective ways to design an intuitive and user-friendly interface that encourages users to track their expenses and achieve their financial goals.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM OF THE PROJECT

The aim of the Expense Tracker project is to develop a web application that helps users keep track of their expenses and manage their budgets effectively. The application will allow users to enter their income and expenses, categorize them, and view reports and visualizations of their spending habits. The main goal of this project is to provide users with a simple and intuitive tool that can help them gain insights into their financial situation, identify areas where they can reduce expenses, and make better financial decisions.

2.2 SCOPE OF THE PROJECT

The scope of the Expense Tracker project is to develop a web application that provides users with the ability to manage their finances efficiently. The application will allow users to add, edit, and delete income and expense items, categorize them, and view reports and visualizations of their spending habits. The project will also include the development of a user authentication system, which will ensure that only authorized users can access and modify their financial information.

The application will be developed using web development technologies such as HTML, CSS, and JavaScript, and will utilize a backend database to store user information securely. The project will also involve the use of third-party libraries and APIs, such as Chart.js, for data visualization.

CHAPTER 3

MATERIALS AND METHODS, ALGORITHMS USED

An expense tracker project is a powerful tool for individuals who want to take control of their finances and manage their expenses effectively. It provides users with features such as expense tracking, budget setting, analytics and insights, goal setting, data security, and customization, all aimed at helping users make informed financial decisions and achieve their financial goals. By using an expense tracker, individuals can gain a better understanding of their spending patterns, save money, and ultimately improve their financial well-being.

Managing personal finances and expenses is a crucial aspect of maintaining financial stability and achieving financial goals. An expense tracker project is a digital tool or application designed to help individuals track and manage their expenses effectively. It allows users to monitor their spending, set budgets, analyze spending patterns, and gain insights into their financial habits.

Purpose: The purpose of the expense tracker project is to provide users with a convenient and efficient way to track their expenses and gain better control over their finances. By using an expense tracker, users can monitor their spending in real-time, identify areas where they may be overspending, and make informed financial decisions based on their spending patterns and budget goals.

Expense tracking: The core feature of the expense tracker project is the ability to track expenses. Users can input their expenses manually or import them automatically from linked bank accounts or credit cards. Expenses can be categorized into different categories such as groceries, entertainment, transportation, etc., making it easy to see where the money is being spent.

Budget setting: The expense tracker project allows users to set budgets for different expense categories. Users can specify the amount they want to allocate to each category and track their progress towards meeting those budget goals. The tracker

sends notifications or alerts when the user is close to exceeding their budget in a particular category, helping them to stay on track.

Analytics and insights: The expense tracker project provides users with detailed analytics and insights into their spending habits. Users can view graphical representations of their spending patterns, see which categories they are spending the most on, and identify areas where they can cut back on expenses. This feature helps users make informed financial decisions and adjust their spending habits accordingly.

Goal setting: The expense tracker project allows users to set financial goals, such as saving for a vacation, paying off debt, or building an emergency fund. Users can input their goals, set a target amount, and track their progress towards achieving those goals. This feature provides motivation and helps users stay focused on their financial objectives.

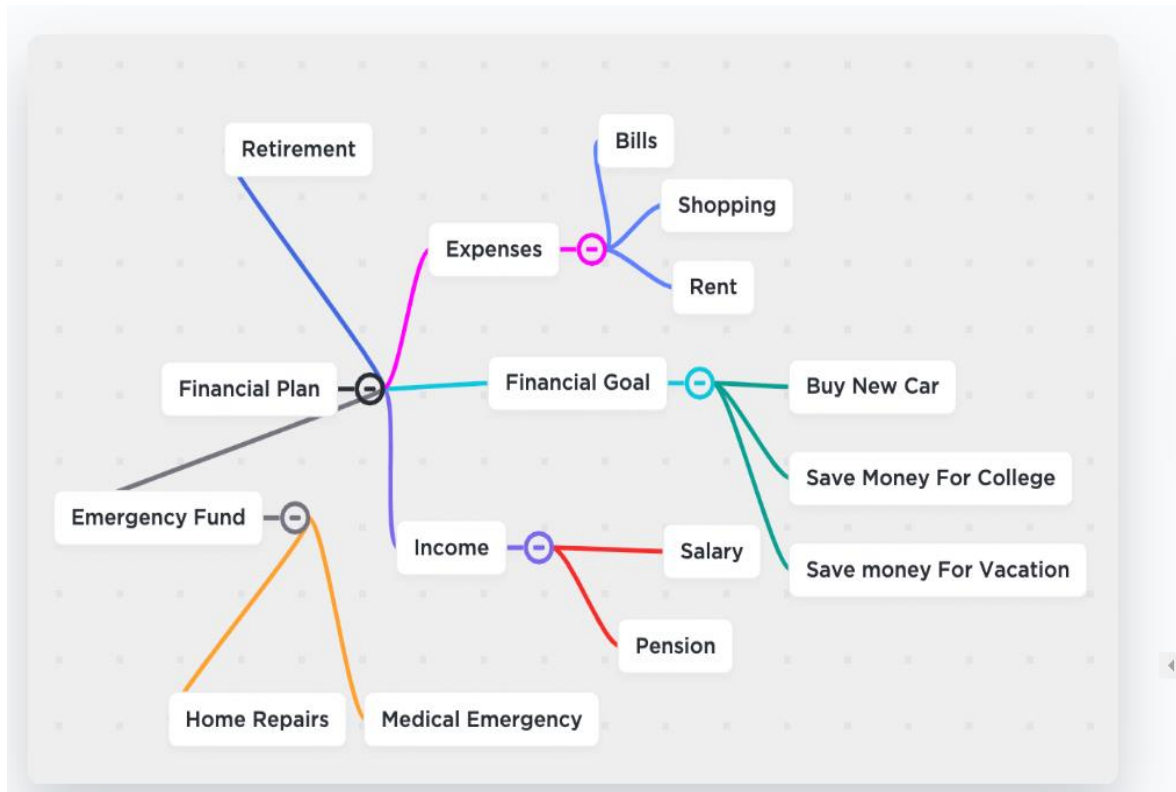


Fig : 3.1 Ideation map

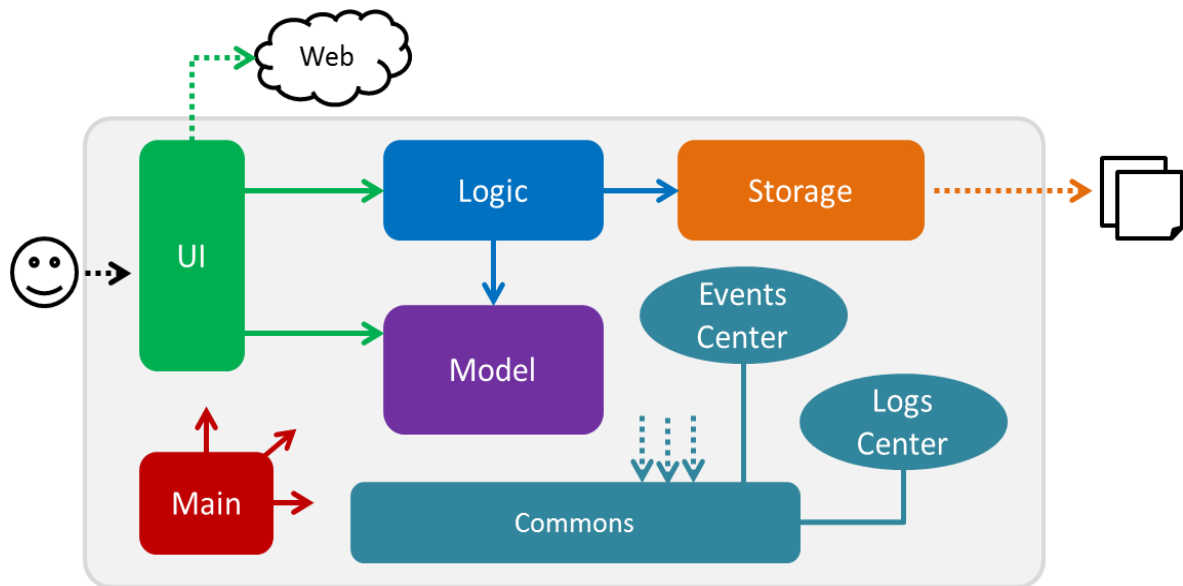


Fig 3.2 System architecture

The *Architecture Diagram* given above explains the high-level design of Expense Tracker. Given below is a quick overview for each component

Main has only one class called MainApp. It is responsible for:

- Initialising the components in the correct sequence and connecting them up with each other when the app is launched.
- Shutting down the components and invoking cleanup methods where necessary when the app is shut down.

Commons represents a collection of classes used by multiple other components.

Logic is the command executor. It defines its API in the Logic.java interface and exposes its functionality using the LogicManager.java class. Read Logic for more details.

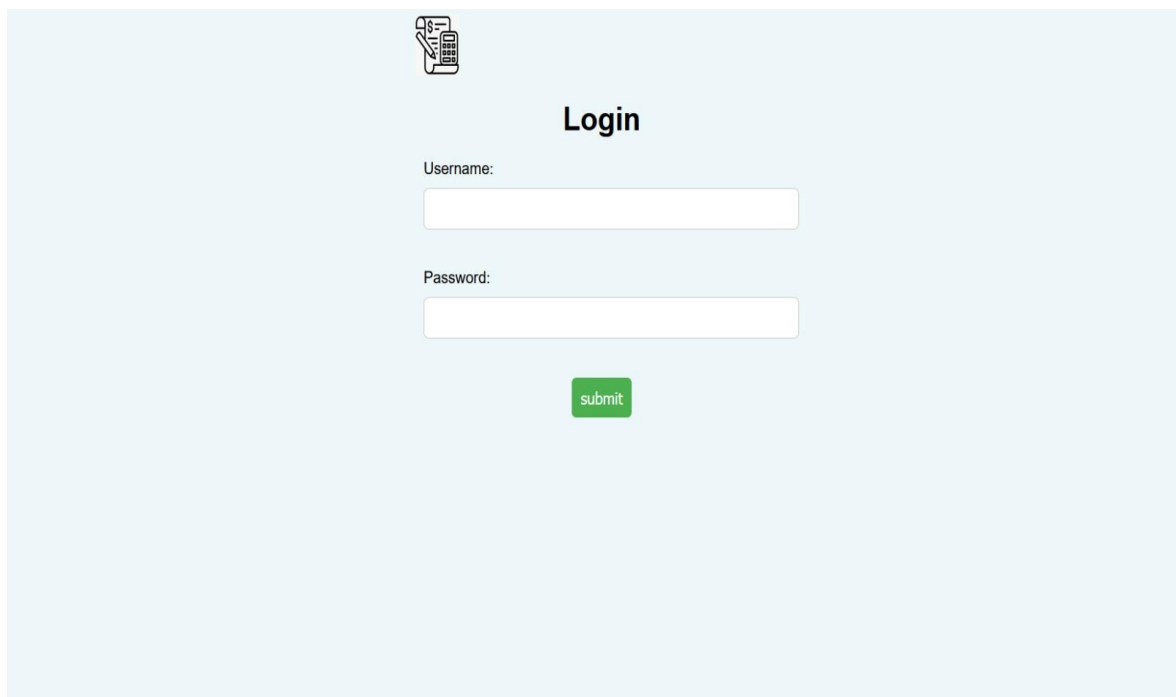
UI is responsible for the UI of the App. It defines its API in the Ui.java interface and exposes its functionality using the UiManager.java class. Read UI for more details.

Model holds the data of the App in-memory. It defines its API in the Model.

Storage reads data from, and writes data to, the hard disk. It defines its API in the Storage.java interface .

3.1 SOFTWARE REQUIREMENTS:

- Front-end development: HTML, CSS, JavaScript
- Operating system



The image shows a login form titled "Login" on a light blue background. At the top left is an icon of a hand holding a calculator. Below the title, there are two input fields: one for "Username:" and one for "Password:". Below the password field is a green "submit" button.

Fig 3.3: Admin Login

3.2 User Interface

The user interface for authentication in an expense tracker project typically includes a login page where users can enter their credentials to access their accounts. The authentication process verifies the user's identity based on the information they provide, such as username and password, to determine if they are authorized to access the expense tracker application.

The login page usually consists of the following elements:

1. Username and password input fields: These are text input fields where users can enter their username and password. The fields may be labeled as "Username" and "Password" respectively, with appropriate placeholders or labels to guide users in entering the correct information.
2. Submit button: This is a button that users can click to submit their username and password for authentication. Once clicked, the authentication process is triggered, and the entered credentials are checked against the stored user credentials in the system.
3. Error message: This is a section or an alert that displays an error message if the entered credentials are invalid or incorrect. It may indicate that the username or password is incorrect, and prompt users to re-enter the correct information.

3.3 CODE FOR THE AUTHENTICATION:

```
<!DOCTYPE html>

<html>

<head>

    <title>Login</title>

<style>

body {
```

```

font-family: Arial, sans-serif;

/*background-image: url("kaka.jpg");*/

background-color: #edf6f9;

background-size: cover;

}

h1 {

text-align: center;

margin-top: 20px;

}

form {

max-width: 400px;

margin: 0 auto;

text-align: center;

}

label {

display: block;

margin-bottom: 10px;

text-align: left; /* align label text to left */

}

input[type="text"], input[type="password"] {

display: block;

margin: 0 0 20px 0; /* move input fields to the left */

padding: 10px;

width: 100%;

```

```

border-radius: 5px;

border: 1px solid #ccc;

font-size: 16px;
}

input[type="submit"] {

background-color: #4CAF50;

color: #fff;

padding: 10px;

border-radius: 5px;

border: none;

font-size: 16px;

cursor: pointer;

}

input[type="submit"]:hover {

background-color: #3e8e41;

}

.alert {

color: red;

margin-top: 20px;

}

</style>

</head>

<body>



```

```

<h1>Login</h1>

<form id="login-form">

    <label for="username">Username:</label>

    <input type="text" id="username" name="username"><br>

    <label for="password">Password:</label>

    <input type="password" id="password" name="password"><br>

<input type="submit" value="submit">

<p id="warning" style="color: red;"></p>

</form>

<script>

// Add event listener to the login form

    document.getElementById("login-form").addEventListener("submit",
function(event) {

// Prevent the form from submitting

    event.preventDefault();


// Get the values of the username and password input

    Var username = document.getElementById("username").value;

    Var password = document.getElementById("password").value;


console.log("Username: " + username);

console.log("Password: " + password);

// Check if the entered username and password match the predefined ones

if (username === "vinit" && password === "vinit") {

    // Redirect the user to another page

```

```

        window.location.href = "index1.html ";

        // Display an error message

    } else{

        /*alert("Incorrect username or password");*/

        var warning = document.getElementById("warning");

        warning.innerHTML = "Incorrect username or password"

        }

    });

</script>

</body>

</html>

```

3.4 CSS STYLE

```

@import
url('https://fonts.googleapis.com/css2?family=Andika+New+Basic&display=swap');

*{

    box-sizing: border-box;

    margin: 0;

    padding: 0;

}

:root{

    --green: #0f7556;

    --red: #bc4749;

}

```

```

body{

    background-color: #cbdf6;

    font-family: 'Andika New Basic', sans-serif;

    color: #2f3e46;

}

.container{

    width: 100%;

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-items: center;

    padding-top: 3rem;

    padding-bottom: 3rem ;

}

.budget__numbers{

    display: flex;

    justify-content: space-between;

    width: 70%;

    max-width: 700px;

    margin: 1rem 0 3rem;

}

.number{

    display: flex;

```

```

    flex-direction: column;

    justify-content: center;

    align-items: center;

    background-color: white;

    padding: 10px;

    border-radius: 15px;
}

.amount__container{

    display: flex;

}

#amount_earned,#earned_symbol{color: var(--green);}

#amount_spent,#spent_symbol{color: var(--red);}


.input__area input{

    border: none;

    padding: 10px;

    border-radius: 15px;

    outline: none;

}

.input__area #description

{

    width: 330px;

}

.input__area #amount{

```

```

width: 100px;

text-align: center;

}

/* remove arrows from number input */

.input__area input::-webkit-outer-spin-button,

.input__area input::-webkit-inner-spin-button{

    -webkit-appearance: none;

}

.buttons__area{

    margin: 10px 0 3rem;

}

.buttons_area #add_income,

.buttons_area #add_expense{

    border: none;

    padding: 10px 20px;

    color: white;

    border-radius: 15px;

    outline: none;

    cursor: pointer;

}

#add__income{background-color: var(--green);}

#add__expense{background-color: var(--red);}

.items__container{

    display: flex;

```

```

    justify-content: space-between;

    align-items: flex-start;

    width: 70%;

    max-width: 700px;

}

#income__container h2{color: var(--green);}

#expenses__container h2{color: var(--red);}


.items__container .container{

    width: 49%;

    border-radius: 15px;

}

.container .item{

    width: 100%;

    background-color: white;

    border-radius: 15px;

    margin: 5px 0;

    display: flex;

    justify-content: space-between;

    padding: 5px 40px 5px 30px;

    position: relative;

}

.item__income,

.item__expense{

```

```

    display: flex;
}

.income span, .income p{color: var(--green);}

.expense span, .expense p{color: var(--red);}

.item .far{

    position: absolute;

    top: 0;

    right: 0;

    height: 100%;

    width: 30px;

    display: flex;

    justify-content: center;

    align-items: center;

    color: white;

    background-color: var(--red);

    border-top-right-radius: 15px;

    border-bottom-right-radius: 15px;

    cursor: pointer;

    transform: translateX(30px);

    opacity: 0;

}

.item:hover .far{

    animation: showDelete;

    animation-duration: 0.5s;

```

```

        animation-fill-mode: forwards;
    }
    @keyframes showDelete{
        0%{
            transform: translateX(30px);
            opacity: 0;
        }
        100%{
            transform: translateX(0);
            opacity: 1;
        }
    }
    .income__date {
font-size: 12px; /* adjust the font size as needed */
        display: inline-block;
        margin-right: 10px;
        margin-top: 5px;
        padding-left: 5px;
        padding-right: 5px;
    }
    #myChart {
        width: 60px;
        height: 60px;
        margin: 0 auto;
    }

```

```
}
```

3.5 FUNCTIONAL METHODOLOGY

Implementing a simple income and expense tracker using JavaScript. It follows the module pattern with two separate modules, namely the **itemCtrl** and **UICtrl**, which handle the data and UI manipulation respectively. The **itemCtrl** module handles the creation, deletion, and retrieval of income and expense items, while the **UICtrl** module handles the manipulation of the UI elements to display the items and update the total earned, spent, and available amounts.

The **itemCtrl** module has a data structure to store the income and expense items, with methods for adding items, creating unique IDs, getting item IDs, and deleting items from the data structure.

The **UICtrl** module has methods for getting input values from the description and amount fields, adding income and expense items to the UI with appropriate HTML markup, clearing input fields, and updating the total earned, spent, and available amounts displayed on the UI.

The code seems to be missing some parts, such as event listeners to trigger the methods in the **UICtrl** module and update the data in the **itemCtrl** module. Additionally, there are some minor issues, such as incorrect formatting of the total spent amount in the **updateSpent** method of the **UICtrl** module. However, overall, the code provides a basic structure for building an income and expense tracker with JavaScript and HTML.

3.6 JAVA SCRIPT FUNCTION AT BACK END

//ITEM CONTROLLER

```
const itemCtrl = (function(){  
  
    //item contructor  
  
    const Item = function(id, description, amount, date){
```

```

    this.id = id;

    this.description = description;

    this.amount = amount;

    this.date = date;
}

//data structure
const data = {
    items:[]
}

//public method
return{
    logData: function(){
        return data;
    },
    addMoney: function(description, amount){
        //create random id
        let ID = itemCtrl.createID();

        let date = new Date()

        //create new item
        newMoney = new Item(ID, description, amount, date);

        //push it into the array
        data.items.push(newMoney);

        return newMoney;
    }
}

```

```

},
createID: function(){
    //create random id number between 0 and 10000
    const idNum = Math.floor(Math.random()*10000);
    return idNum;
},
getIdNumber: function(item){
    //get the item id
    const amountId = (item.parentElement.id);
    //break the id into an array
    const itemArr = amountId.split('-');
    //get the id number
    const id = parseInt(itemArr[1]);

    return id;
},
deleteAmountArr: function(id){
    //get all the ids
    const ids = data.items.map(function(item){
        //return item with id
        return item.id
    });
    //get index
    const index = ids.indexOf(id)

```

```

        //remove item

        data.items.splice(index, 1);
    },

}

})();

//UI CONTROLLER

const UICtrl = (function(){

    //ui selectors

    const UISelectors = {

        incomeBtn: '#add__income',

        expenseBtn: '#add__expense',

        description: '#description',

        amount: '#amount',

        moneyEarned: '#amount__earned',

        moneyAvailable: '#amount__available',

        moneySpent: '#amount__spent',

        incomeList: '#income__container',

        expensesList: '#expenses__container',

        incomeItem: '.income__amount',

        expenseItem: '.expense__amount',

        itemsContainer: '.items__container'
    }

```

```

}

//public methods

return{

  //return ui selectors

  getSelectors: function(){

    return UISelectors

  },

  getDescriptionInput: function(){

    return {

      descriptionInput: document.querySelector(UISelectors.description).value

    }

  },

  getValuelInput: function(){

    return{

      amountInput: document.querySelector(UISelectors.amount).value

    }

  },

  addIncomeItem: function(item){

    //create new div

    const div = document.createElement('div');

    //add class

    div.classList = 'item income'

    //add id to the item

    div.id = `item-${item.id}`
  }
}

```

```

//add html

div.innerHTML = `

<h4>${item.description}</h4>

<div class="item__income">

  <span                                class="income__date">${new
Date(item.date).toLocaleString()}</span>

  <p class="symbol">₹</p>

  <span class="income__amount">${item.amount}</span>

</div>

<i class="far fa-trash-alt"></i>

`;

//insert income into the list

document.querySelector(UISelectors.incomeList).insertAdjacentElement('beforeend', div);

},

clearInputs: function(){

  document.querySelector(UISelectors.description).value = ""

  document.querySelector(UISelectors.amount).value = ""

},

updateEarned: function(){

  //all income elements

  const allIncome = document.querySelectorAll(UISelectors.incomeItem);

  //array with all incomes

```

```

const incomeCount = [...allIncome].map(item => +item.innerHTML);

//calculate the total earned

const incomeSum = incomeCount.reduce(function(a,b){

    return a+b

},0);

//display the total earned

const earnedTotal =
document.querySelector(UISelectors.moneyEarned).innerHTML =
incomeSum.toFixed(2);

},

addExpenseItem: function(item){

    //create new div

    const div = document.createElement('div');

    //add class

    div.classList = 'item expense'

    //add id to the item

    div.id = `item-${item.id}`

    //add html

    div.innerHTML = `

    <h4>${item.description}</h4>

    <div class="item__expense">

    <span class="income__date">${new
Date(item.date).toLocaleString()}</span>

    <p class="symbol">₹</p>

```

```

        <span class="expense__amount">${item.amount}</span>

    </div>

    <i class="far fa-trash-alt"></i>

    `;

    //insert income into the list

document.querySelector(UISelectors.expensesList).insertAdjacentElement('before
end', div);

},

updateSpent: function(){

    //all expenses elements

    const allExpenses = document.querySelectorAll(UISelectors.expenseItem);

    //array with all expenses

    const expenseCount = [...allExpenses].map(item => +item.innerHTML)

    //calculate the total

    const expenseSum = expenseCount.reduce(function(a, b){

        return a+b

    },0)

    // display the total spent

    const expensesTotal =

document.querySelector(UISelectors.moneySpent).innerHTML = expenseSum;

},

updateAvailable: function(){

    const earned = document.querySelector(UISelectors.moneyEarned);

    const spent = document.querySelector(UISelectors.moneySpent)

```

```

        const available = document.querySelector(UISelectors.moneyAvailable);

        available.innerHTML = ((+earned.innerHTML)-
(+spent.innerHTML)).toFixed(2)

    },

    deleteAmount: function(id){

        //create the id we will select

        const amountId = `#item-${id}`;

        //select the amount with the id we passed

        const amountDelete = document.querySelector(amountId);

        //remove from ui

        amountDelete.remove();

    },

}

})0;

```

//APP CONTROLLER

```

const App = (function(){

    //event listeners

    const loadEventListeners = function(){

        //get ui selectors

        const UISelectors = UICtrl.getSelectors();

        //add new income

        document.querySelector(UISelectors.incomeBtn).addEventListener('click',
addIncome);

```

```

        //add new expense

        document.querySelector(UISelectors.expenseBtn).addEventListener('click',
addExpense);

        //delete item

document.querySelector(UISelectors.itemsContainer).addEventListener('click',
deleteItem);

    }

//add new income

const addIncome = function(){

    //get description and amount values

    const description = UICtrl.getDescriptionInput();

    const amount = UICtrl.getValueInput();

    //if inputs are not emptyy

    if(description.descriptionInput !== "" && amount.amountInput !== ""){

        //add new item

        const newMoney = itemCtrl.addMoney(description.descriptionInput,
amount.amountInput);

        //add item to the list

        UICtrl.addIncomeItem(newMoney);

        //clear inputs

        UICtrl.clearInputs();

        //update earned

        UICtrl.updateEarned();

        //calculate money available

```

```

    UICtrl.updateAvailable();

    const getCurrentDateAndTime = function() {
const now = new Date();
const date = now.toLocaleDateString();
const time = now.toLocaleTimeString();
return `${date} ${time}`;
};

    }
}

//add new expense
const addExpense = function(){
    //get description and amount values
    const description = UICtrl.getDescriptionInput();
    const amount = UICtrl.getValueInput();
    //if inputs are not empty
    if(description.descriptionInput !== "" && amount.amountInput !== ""){
        //add new item
        const newMoney = itemCtrl.addMoney(description.descriptionInput,
amount.amountInput);
        //add item to the list
        UICtrl.addExpenseItem(newMoney);
        //clear inputs
        UICtrl.clearInputs();
    }
}

```

```

//update total spent
UICtrl.updateSpent();

//calculate money available
UICtrl.updateAvailable();

const getCurrentDateAndTime = function() {
    const now = new Date();
    const date = now.toLocaleDateString();
    const time = now.toLocaleTimeString();
    return `${date} ${time}`;
};

}

}

//delete item
const deleteItem = function(e){
    if(e.target.classList.contains('far')){
        //get id number
        const id = itemCtrl.getIdNumber(e.target)

        //delete amount from ui
        UICtrl.deleteAmount(id);

        //delete amount from data
        itemCtrl.deleteAmountArr(id);
    }
}

```

```

        //update earned
        UICtrl.updateEarned();

        //update total spent
        UICtrl.updateSpent();

        //calculate money available
        UICtrl.updateAvailable();
    }

    e.preventDefault()
}

//init function
return{
    init: function(){
        loadEventListeners();
    }
}

})(itemCtrl, UICtrl);

App.init();

```

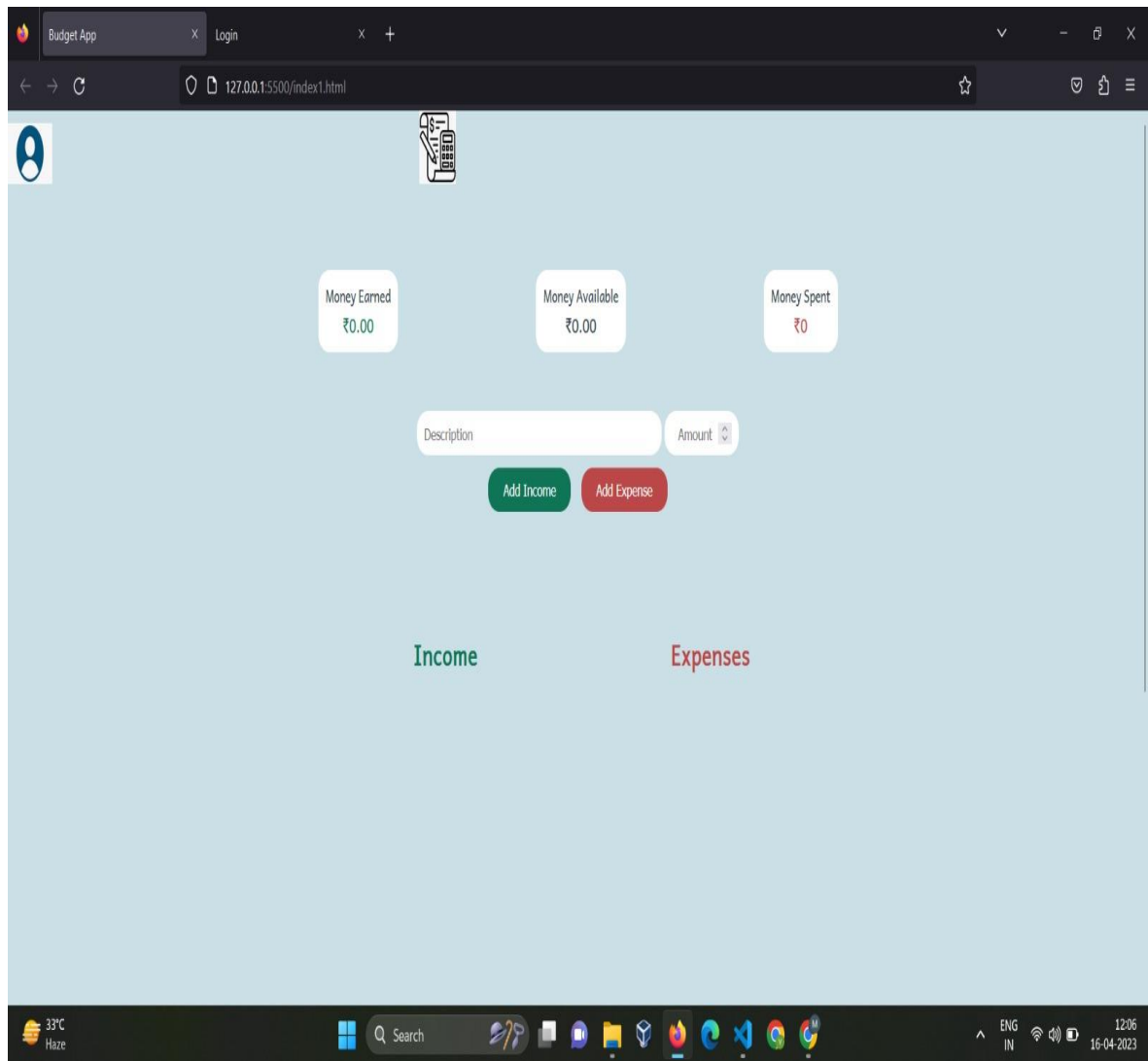


Fig 3.6 web interface

4.1 Result:

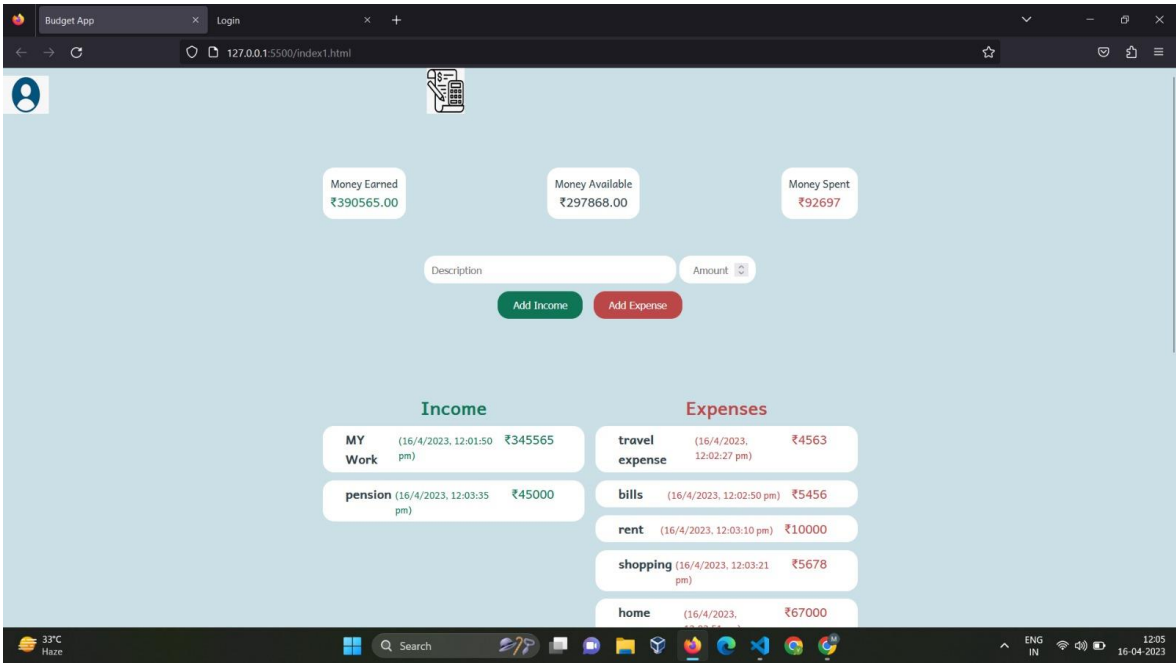
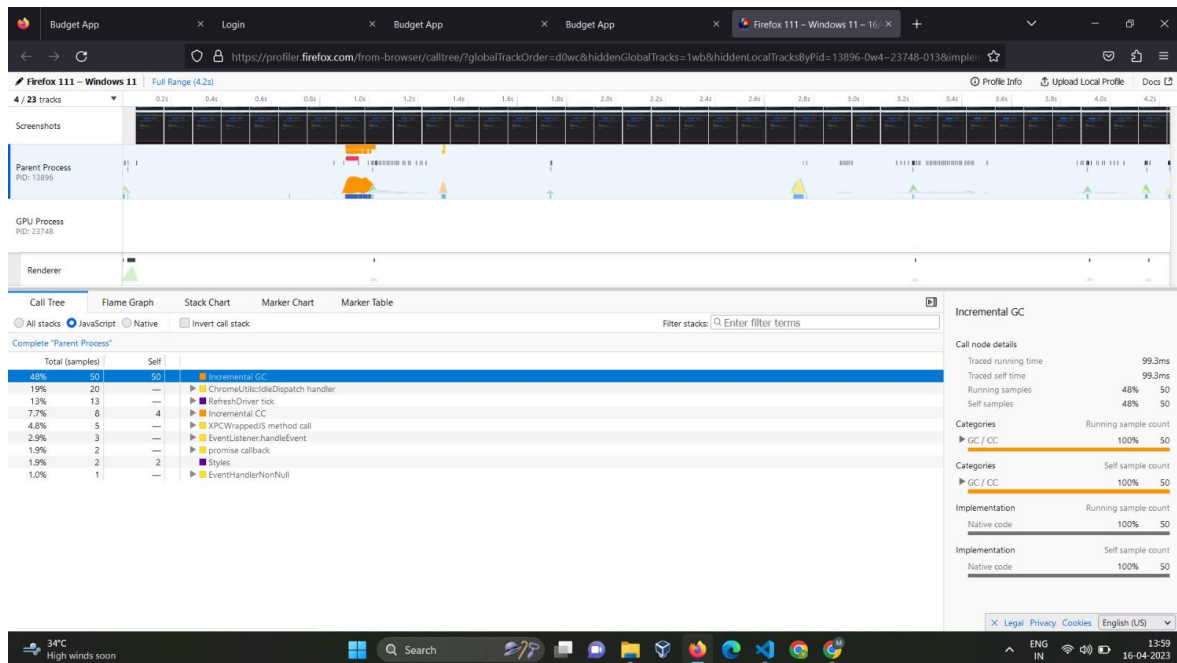


Fig 4.1 result

4.2 Performance Analysis:

Functionality:

The first aspect of performance analysis for an expense tracker project is its functionality. The key performance indicators (KPIs) for functionality may include the ability to add and manage expenses, categorize expenses, set budgets, generate reports, and provide a user-friendly interface. A thorough evaluation should be conducted to ensure that all the required.



Maintenance and Support:

The performance analysis should also consider the ease of maintenance and support for the expense tracker project. It should evaluate factors such as code maintainability, documentation, error logging, and technical support to ensure that the system can be effectively maintained and supported throughout its lifecycle.

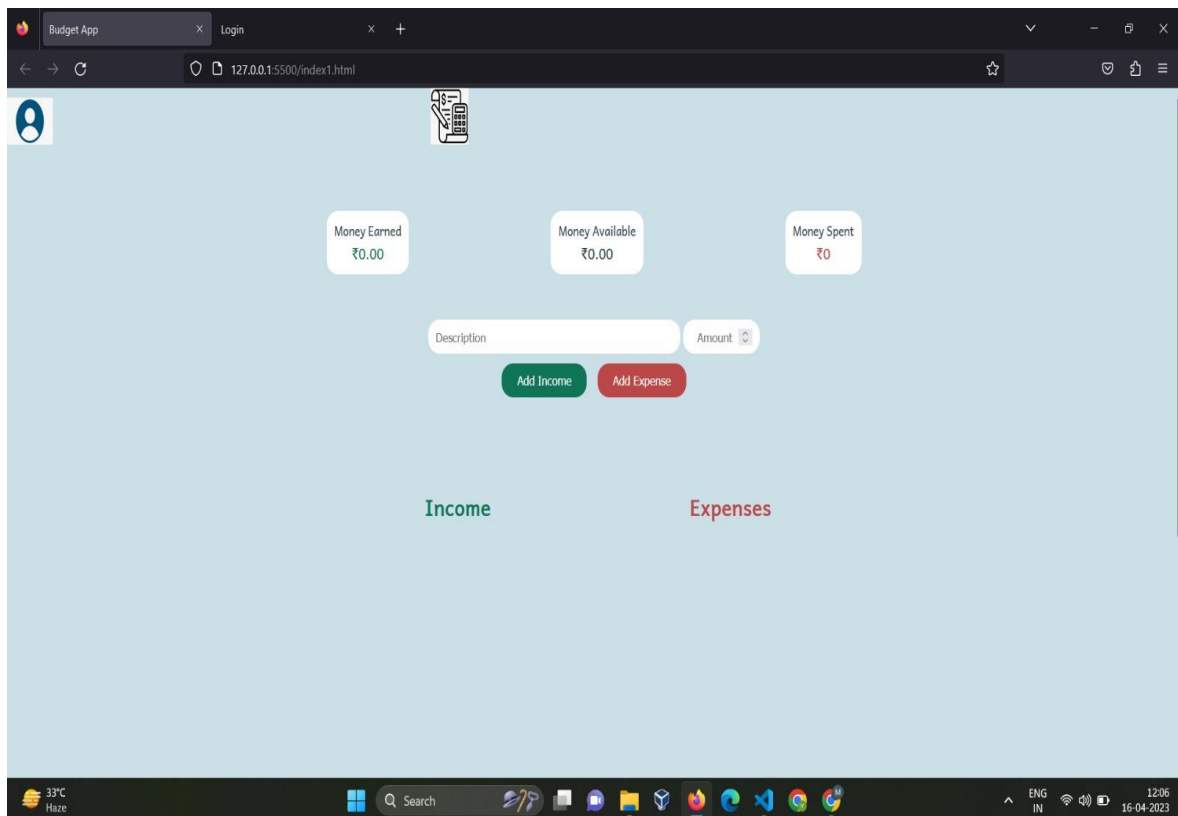
Summary:

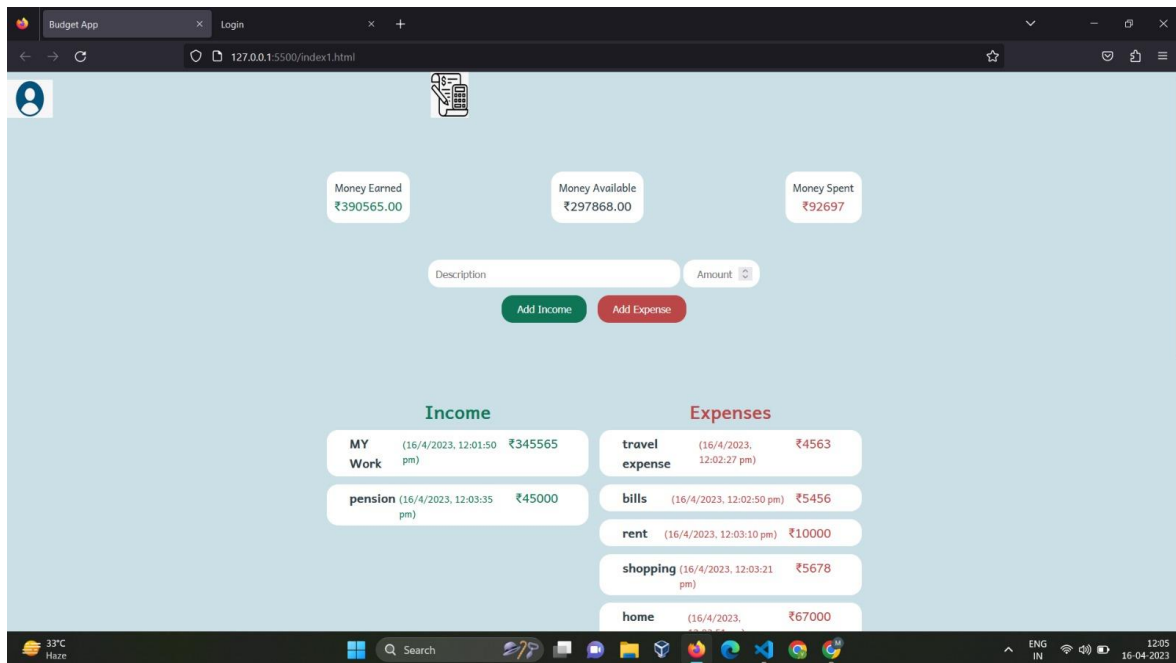
The expense tracker project is a software system designed to help users manage their expenses effectively. It typically includes features such as expense tracking, budgeting, reporting, and user-friendly interfaces. The project's performance can be analyzed based on various factors, including functionality, reliability, performance, security, usability, scalability, maintenance and support, and cost-effectiveness.

Conclusion:

The performance analysis of an expense tracker project is essential to ensure that the project meets its objectives and delivers value to users. By thoroughly evaluating the system's functionality, reliability, performance, security, usability, scalability, maintenance and support, and cost-effectiveness, project stakeholders can identify strengths and weaknesses, make necessary improvements, and ensure that the expense tracker project performs optimally. A well-performing expense tracker project can help users effectively manage their expenses, track their spending, and achieve their financial goals.

SCREENSHOTS:





SOURCE CODE:

```
addMoney: function(description, amount){
    //create random id

    let ID = itemCtrl.createID();

    let date = new Date()

    //create new item

    newMoney = new Item(ID, description, amount, date);

    //push it into the array

    data.items.push(newMoney);

    return newMoney;
}
```

```
deleteAmountArr: function(id){  
    //get all the ids  
    const ids = data.items.map(function(item){  
        //return item with id  
        return item.id  
    });  
    //get index  
    const index = ids.indexOf(id)  
    //remove item  
    data.items.splice(index, 1);
```

REFERENCES:

<https://www.mint.com/>

<https://www.personalcapital.com/>

<https://www.youneedabudget.com/>

<https://www.expensify.com/>

<https://www.pocketguard.com/>

<https://goodbudget.com/>

<https://wally.me/>

<https://www.spendee.com/>

<https://www.honeydue.com/>

<https://www.moneylover.me/>