 CSE 326 REPORT.pdf • 29 November 2023  
1377 words (9649 characters)

Essentially Human 3%

The text is written almost entirely by a human, with tiny to no AI assistance.

AI weightage		Content weightage	Sentences
<div>H</div>	Highly AI written	0% Content	0
<div>M</div>	Moderately AI written	1% Content	1
<div>L</div>	Lowly AI written	1% Content	1

# Personal Finance Management System

As a Project Work for Course

## INTERNET PROGRAMMING LABORATORY (CSE 326)

By

Sr. No.	Registration No	Name of Students	Roll No
1	12323287	Asmirandah	RK23RKA10
2	12324556	Minhaj Uddin Ahmad	RK23RKA11
3	12301199	Utkarsh Pandey	RK23RKA12



**L**OVELY  
**P**ROFESSIONAL  
**U**NIVERSITY

---

*Transforming Education Transforming India*

**Submitted To:** Mr. Muhammed Rafeeq War

Lovely Professional University Jalandhar, Punjab, India.

# INDEX

S.no	Table of Contents	Page. No
1.	INTRODUCTION	3
2.	TECHNOLOGIES USED	4
3.	MODULES	5
4.	SNAPSHOTS	11
5.	CODE SNAPSHOT	14
6.	REFERENCES	16

# *Introduction*

In the ever-evolving landscape of personal finance, the need for effective money management has never been more crucial. Enter "Hissab," your trusted companion in the journey toward financial mastery. Hissab, which translates to "account" in several languages, encapsulates the essence of our platform: a dedicated space designed to help individuals take command of their financial accounts, aspirations, and, ultimately, their futures.

Hissab is more than just a website; it is a comprehensive resource hub meticulously crafted to demystify the complexities of personal finance. Whether you're a seasoned investor, a recent graduate navigating your first paycheck, or someone looking to rebound from financial setbacks, Hissab is here to provide tailored guidance for every stage of your financial journey.

At Hissab, we understand that financial well-being is a dynamic and deeply personal pursuit. Our platform is not a one-size-fits-all solution but rather a versatile toolkit that adapts to your unique circumstances. From budgeting basics to advanced investment strategies, Hissab offers a spectrum of resources, articles, and tools to empower you to make informed financial decisions.

# *Technologies Used*

The foundation of this project rests on HTML, CSS, and JavaScript, each playing a pivotal role in creating a seamless and dynamic user experience.

## JavaScript:

The core functionality of the application is powered by JavaScript, facilitating dynamic interactions and enabling real-time data manipulation on the client side. This not only enhances user engagement but also contributes to the fluidity of the overall experience.

## HTML:

HTML serves as the structural backbone of the webpage. It not only creates the essential framework but also utilizes attributes from various tags to trigger JavaScript functions, seamlessly integrating the logic and presentation layers.

## CSS:

CSS takes the user interface to the next level, not merely as a styling tool but as a means to enhance the webpage's responsiveness. By implementing responsive design techniques, the webpage becomes compatible with a diverse array of devices, offering an aesthetically pleasing and intuitive interface.

## JavaScript Functions:

The majority of the webpage's functionality is executed through JavaScript functions, contributing to a more modular and efficient code structure. This approach minimizes the code clutter in the HTML file, enhancing maintainability and overall system performance.

# Modules

## 1. *Variable Declaration*

The project begins by declaring the variables for the home page and navigation bar. The navbarMenu variable refers to the navigation menu, while hamburgerBtn and hideMenuBtn represent the hamburger button and close button within the menu, respectively. Additionally, showPopupBtn is associated with a button triggering the display of a popup, and formPopup corresponds to the entire popup form. The hidePopupBtn variable is assigned to the close button within the popup, and signupLoginLink captures an array of links within the popup form designed for signup and login purposes

```
// Variable Declaration
const navbarMenu = document.querySelector(".navbar .links");
const hamburgerBtn = document.querySelector(".hamburger-btn");
const hideMenuBtn = navbarMenu.querySelector(".close-btn");
const showPopupBtn = document.querySelector(".login-btn");
const formPopup = document.querySelector(".form-popup");
const hidePopupBtn = formPopup.querySelector(".close-btn");
const signupLoginLink = formPopup.querySelectorAll(".bottom-link a");
```

## 2. *Navigation Bar and Popup Bar*

The code features two modules: the Navbar Module and the Popup Module. The Navbar Module is always available on top of the page. The Popup Module responds to user interactions with the popup form, toggling the visibility of the entire page body (document.body) by adding or removing the "show-popup" class upon clicking the designated button (showPopupBtn). Additionally, it dynamically manages the display of the signup section within the form based on link clicks.



#### 4. *Utilities (utils)*

This module encapsulates a set of versatile functions for formatting and unformatting various data types. The **formatCurrency** function ensures the proper display of numeric values as currency, incorporating considerations for positive and negative amounts and converting them to the Indian Rupees (INR) format. The **formatDescription** function capitalizes the initial letter of a given description. Meanwhile, **formatAmount** standardizes numeric values by multiplying them by 100 and rounding them. On the unformatting side, the **unformatAmount** function reverses the formatting process for amounts, while **unformatDate** and **unformatCurrency** reverse the formatting for dates and currency, respectively. These utility functions collectively contribute to consistent and user-friendly data handling within the application.

```
// Module 1: Utilities (utils)
const utils = {
  >   formatCurrency(value) { ...
    },           // ... (formatting currency)
  >   formatDescription(description) { ...
    },           // ... (formatting description)
  >   formatAmount(value) { ...
    },           // ... (formatting amount)
  >   formatDate(date) { ...
    },           // ... (formatting date)
  >   unformatAmount(value) { ...
    },           // ... (unformatting amount)
  >   unformatDate(date) { ...
    },           // ... (unformatting date)
  >   unformatCurrency(value) { ...
    },           // ... (unformatting currency)
};
```

#### 5. *Modal*

This module is designed to facilitate the control of modal overlays within the application. Its functions provide essential features for managing the visibility and interactions with modals. The `'toggleState'` function dynamically adjusts the modal's visibility based on a specified mode, simultaneously ensuring the clearing of form fields through



`Form.clearFields()`. The `toggleScreen` function handles the display of a confirmation modal overlay and, in the case of a positive confirmation ('yes'), triggers the removal of a transaction via `Transaction.remove(index)`. Additionally, the `eventListener` function sets up event listeners to detect user interactions with the modal overlay and its underlying elements. It enables the toggling of the modal state when clicking outside the modal, while carefully managing the event status to prevent immediate toggling during modal interactions. Overall, this module plays a crucial role in enhancing the user interface and experience by providing a responsive and interactive modal system.

```
// Module 2: Modal (Modal)

const Modal = {
  toggleState(mode) {...
  },          // ... (toggling modal state)

  toggleScreen(index, confirm) {...
  },          // ... (toggling confirmation modal)

  eventListener(eventStatus = false) {...
  },          // ... (event listeners for modal)
};
```

## 6. Storage

This module serves as a crucial component in the application's data management. The `get` function is responsible for retrieving data from the browser's local storage, specifically targeting the key `hide.finances.transactions`. It intelligently parses the retrieved data from JSON format, returning an empty array if no data is found. On the storage side, the `set` function efficiently stores the provided transaction data into the local storage, ensuring its persistence under the designated key. By leveraging the capabilities of local storage, this module enables the application to seamlessly retrieve and store transactional data, contributing to a smooth and persistent user experience.

```
// Module 3: Storage (Storage)

const Storage = {
  get() { ...
  },          // ... (retrieving data from localStorage)

  set(transactions) { ...
  },          // ... (setting data in localStorage)
};
```

## 7. *Transaction*

This module plays a central role in managing transactional data within the application. The `add` function handles the addition or modification of transactions based on the form mode ('new' or 'edit'). It ensures the proper updating of the transaction list and triggers a reload of the application through `App.reload()`. The `remove` function removes a transaction at a specified index, also prompting a reload. Additionally, the module provides functions for calculating and formatting total income, total expenses, and the overall balance. The visual representation of the total balance dynamically adjusts based on whether it is positive, negative, or zero. This module acts as a crucial engine for maintaining, updating, and analyzing financial transactions, contributing to a comprehensive and responsive user experience.

```
//Module 4: Transaction (Transaction)

const Transaction = {
  all: Storage.get(),
  add(transaction) { ...
  },          // ... (adding or updating transaction)

  remove(index) { ...
  },          // ... (removing transaction)

  incomes() { ...
  },          // ... (calculating and formatting total income)

  expenses() { ...
  },          // ... (calculating and formatting total expenses)

  total() { ...
  },          // ... (calculating and formatting total balance)
};
```

## 8. *Form*

This module is instrumental in managing user input and handling transactional data within the application. It defines form elements corresponding to the description, amount, and date fields. The `getValues` function retrieves the current values from these fields, while `validateFields` ensures that essential fields are not left empty, throwing an error if necessary. The `formatValues` function processes and formats the input values using utility functions from the "Utilities" module (`utils`). Upon form submission, the `submit` function validates the fields, formats the values, saves the transaction using the "Transaction" module (`Transaction`), toggles the modal state, and clears the form fields. This module streamlines the interaction between users and the application, enforcing data integrity and contributing to a smooth and user-friendly experience.

```
// Module 5: Form (Form)

const Form = { ...
};
```

## 9. *Application*

This module acts as the central coordinator for the application's lifecycle. The `init` function is responsible for the initial setup, iterating through all transactions to display them and updating the overall balance. It also ensures the persistent storage of transactions through the "Storage" module (`Storage`) and sets up event listeners for the modal via the "Modal" module (`Modal`). Additionally, the module provides a `reload` function, which clears existing transactions and reinitializes the application. The final step involves the automatic initiation of the application upon page load through the `App.init()` call. This module encapsulates critical functionalities, ensuring a seamless and organized execution of the application's core processes.

```
// Module 6: Application (App)

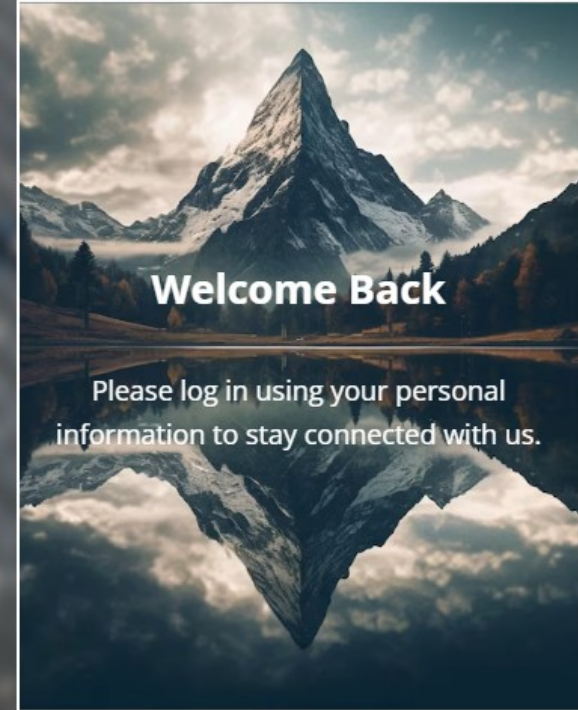
const App = { ...
};

// Initialization of the Application

App.init();
```

# Snapshots

Log in/Sign Up:



×

## LOGIN


Email

Password

[Forgot password?](#)

Log In

Don't have an account? [Signup](#)



×

## SIGNUP

Enter your email

Create password

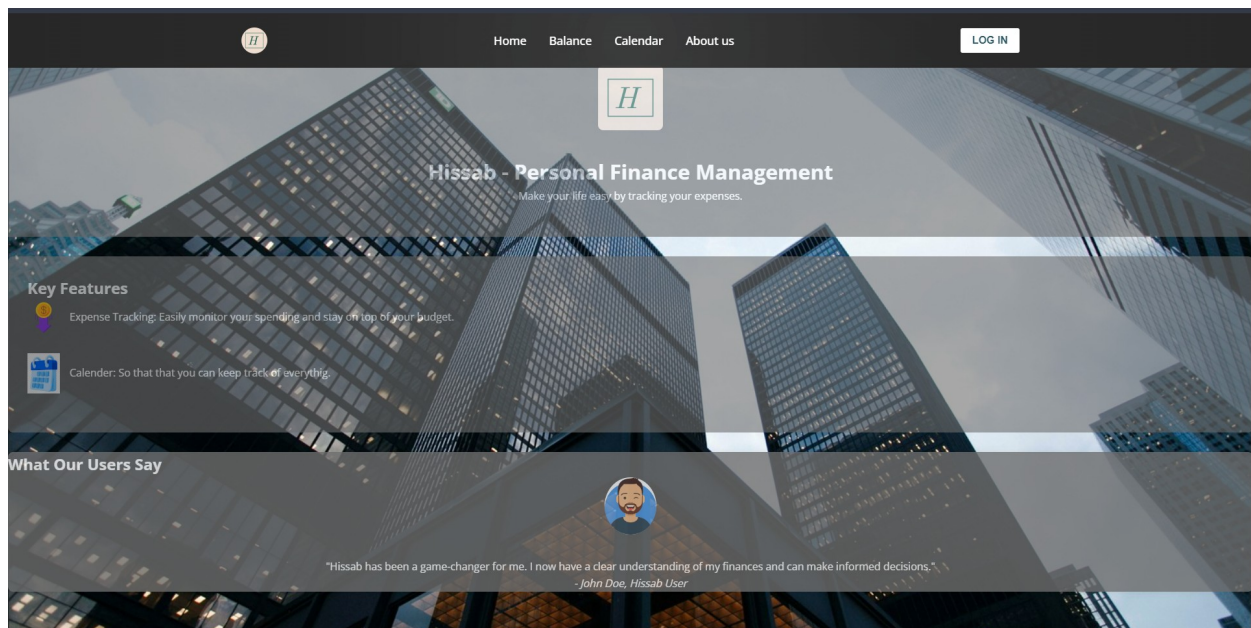
☐ I agree the [Terms & Conditions](#)

Sign Up

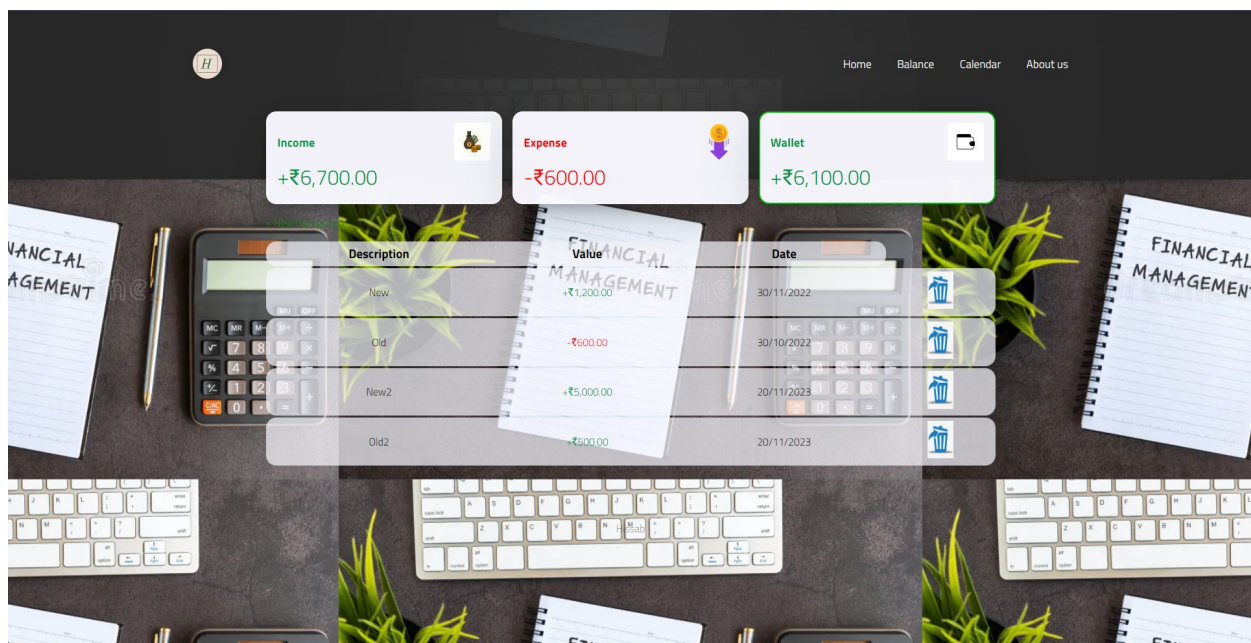
Already have an account? [Login](#)



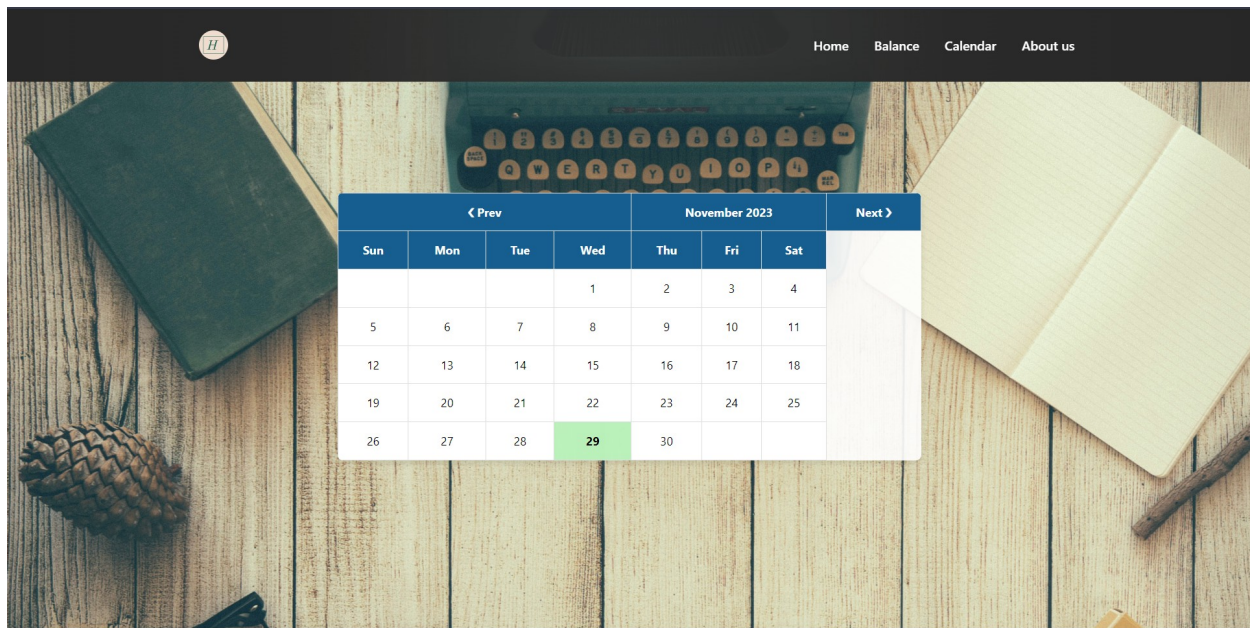
Home page of the website:



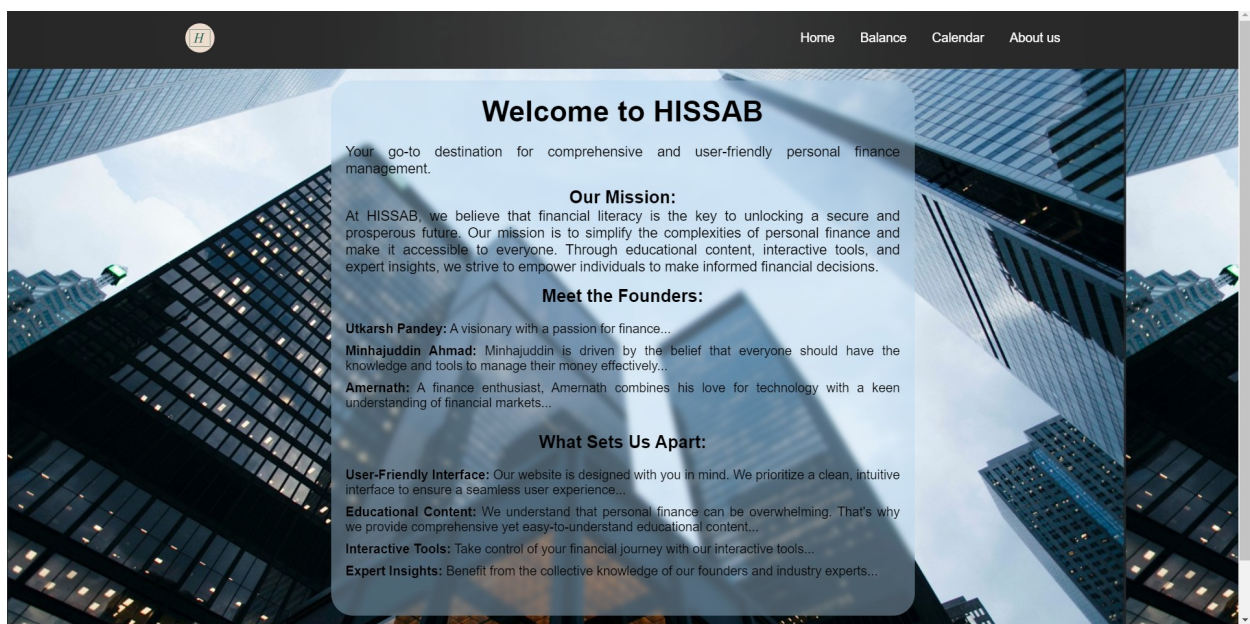
Balance page after adding transactions:



## Calendar(Showing the current date):



## About Us:



## Website URL:

<https://utkarsh1454.github.io/Hissab---Personal-Finance-Management/>



# Code Screenshots

## HTML

### Home Page Source Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Hissab - Personal Finance Management</title>
  <link rel="icon" href="logo.jpeg">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Rounded:opsz
  <link rel="stylesheet" href="index.css">
  <script src="index.js" defer></script>
</head>
<body>
  <header>
    <nav class="navbar">
      <span class="hamburger-btn material-symbols-rounded">Home</span>
      <a href="index.html" class="logo">
        
      </a>
      <ul class="links">
        <span class="close-btn material-symbols-rounded">close</span>
        <li><a href="index.html" target="_blank">Home</a></li>
        <li><a href="balance.html" target="_blank">Balance</a></li>
        <li><a href="calender.html" target="_blank">Calendar</a></li>
        <li><a href="about.html">About us</a></li>
      </ul>
      <button class="login-btn">LOG IN</button>
    </nav>
  </header>

  <div class="hero-section">
    
    <h1>Hissab - Personal Finance Management</h1>
    <p>Make your life easy by tracking your expenses.</p>
  </div>

  <div class="blur-bg-overlay"></div>
  <div class="form-popup">
    <span class="close-btn material-symbols-rounded">close</span>
    <div class="form-box login">
      <div class="form-details">
        <h2>Welcome Back</h2>
        <p>Please log in using your personal information to stay connected with us.</p>
      </div>
      <div class="form-content">
```

## Add Transaction Page

```
</body> </html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1.0" />
  <link rel="stylesheet" href="balance.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Titillium+Web:ital,wght@0,200;0,300;0,
    rel="stylesheet">
  <title>Balance | Hissab</title>
  <link rel="icon" href="logo.jpeg">
</head>

<body>
  <header>
    <p class="nav1">
      <nav class="navbar">
        <span class="hamburger-btn material-symbols-rounded">Home</span>
        <a href="index.html" class="logo">
          
        </a>
        <ul class="links">
          <span class="close-btn material-symbols-rounded">close</span>
          <li><a href="index.html" target="_blank">Home</a></li>
          <li><a href="balance.html" target="_blank">Balance</a></li>
          <li><a href="calender.html" target="_blank">Calendar</a></li>
          <li><a href="about.html" target="_blank">About us</a></li>
        </ul>
      </nav>
    </p>
  </header>

  <main class="container">
    <section id='balance'>
      <h2 class="sr-only">Balance</h2>

      <div class="card incomes">
        <h3>
          <span>Income</span>
          
        </h3>
        <p id="incomes-display">Rs 0.00</p>
      </div>

      <div class="card expenses">
```

(due to more lines and different types of code please refer the [github](#) for the full file)

# GITHUB LINK



# **References**

1. [https://github.com\](https://github.com/)
2. <https://www.w3schools.com>
3. <https://opoenai.com>