

External Project Report on Web Technology Workshop-2 (CSE 4948)

A Grocery Bill generator using React



Submitted by

Name :Isita Ray	Reg. No.:2141018145
Name :Aditi Raj	Reg. No.:2141018143
Name :Aditya Ranjan Behera	Reg. No.:2141016404
Name:Kirtikanta Patra	Reg.No.:2141018141

B. Tech. **CSIT** 6th Semester (Section **D,Group11**)

INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH
(FACULTY OF ENGINEERING)

SIKSHA 'O' ANUSANDHAN (DEEMED TO BE UNIVERSITY), BHUBANESWAR,
ODISHA

Declaration

We, the undersigned students of B. Tech. of **CSIT** Department hereby declare that we own the full responsibility for the information, results etc. provided in this **PROJECT** titled **“A Grocery Bill generator using React”** submitted to **Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar** for the partial fulfillment of the subject **Web Technology Workshop-2 (CSE 4948)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

Isita Ray

Registration No.:2141018145

Aditya Ranjan Behera

Registration No.:2141016404

Aditi Raj

Registration No.:2141018143

Kirtikanta Patra

Registration No.:2141018141

DATE:27.5.24

PLACE:BBSR

Abstract

The Grocery Bill Generator Application using React and PDF Generation is a web-based project aimed at demonstrating the integration of user interface development and dynamic document creation. The project allows users to input grocery items, including names and prices, and generate a formatted PDF receipt. This receipt includes the current date in **dd/mm/yyyy** format, the time of generation, a list of items with prices in rupees, and the total amount. The application leverages React for building the user interface and `@react-pdf/renderer` for creating and downloading the PDF.

The user interface comprises a form for entering item details, a display list for reviewing added items, and buttons for adding items and generating the PDF bill. The bill maintains a clear and readable layout, ensuring an enhanced user experience. The application emphasizes user-friendly interaction, enabling users to easily input and manage their grocery items.

This project showcases fundamental principles of modern web development, including component-based architecture, state management, and third-party library integration. It serves as a practical introduction to essential aspects of web applications, such as dynamic user input handling and document generation. Suitable for educational purposes, this project provides insights into React's capabilities and the practical implementation of PDF generation within a web environment.

Contents

Serial No.	Chapter No.	Title of the Chapter	Page No.
1.	1	Introduction	1
2.	2	Problem Statement	2
3.	3	Methodology	3
4.	4	Implementation	5
5.	5	Results and interpretation	7
6.	6	Conclusion	8

1. Introduction

The Grocery Bill Generator Application using React and PDF Generation serves as a practical illustration of modern web development techniques. This project highlights essential principles such as component-based architecture, state management, and dynamic document creation. Users can input grocery items with their names and prices, and generate a well-formatted PDF receipt that includes the current date, time, list of items with prices in rupees, and the total amount.

In this application, React plays a crucial role in constructing a responsive and intuitive user interface. The form component allows users to add items, while the item list component displays the added items in a clear manner. A button component enables the generation of a PDF, showcasing the seamless integration of third-party libraries like `@react-pdf/renderer` for document creation and download.

This project delves into the core concepts of interactive web applications, emphasizing the management of user input and state. The integration of PDF generation demonstrates the capability to produce dynamic, print-ready documents directly from web applications. The focus on user-friendly interaction ensures that users can easily input, review, and generate their grocery bills.

Overall, this undertaking provides a comprehensive introduction to the practical aspects of modern web development, offering valuable insights into the use of React for creating interactive interfaces and the practical implementation of PDF generation in a web context. This project is particularly beneficial for those looking to understand the collaborative interplay of user input handling and dynamic document creation in web applications.

2. Problem Statement

The Grocery Bill Generator Application using React and PDF Generation aims to streamline the process of creating detailed grocery bills. This project focuses on key aspects of user input handling, dynamic display, and document generation. The application allows users to input item names and prices, displaying them in real-time on the webpage, and generate a formatted PDF containing this information.

To accurately reflect user inputs, the application captures item details and stores them in the state, ensuring real-time updates and correct data representation. The generated PDF includes the current date in dd/mm/yyyy format, time of generation, itemized list with prices in rupees, and the total amount.

Constraints and Enhancements:

Current Constraints: The application supports basic item addition and PDF generation but lacks features for editing or deleting items. It currently handles simple lists without advanced functionalities like data persistence or error handling for invalid inputs.

Enhancement Potential: Improvements could include adding capabilities for editing and deleting items, implementing robust error handling to manage invalid data entries, and enhancing the user interface for better usability and accessibility.

By addressing these constraints and implementing potential enhancements, the Grocery Bill Generator Application can evolve into a comprehensive tool for managing and generating grocery bills. This project highlights essential web development practices, including state management, user interaction, and dynamic document generation, providing valuable insights into creating efficient and user-friendly web applications.

3. Methodology

1. Set Up Server:

- Start a server by typing `npm start`.

2. Handle Incoming Requests:

- Listen for incoming client connections.
- Upon connection, accept the client connection and initiate communication.

3. Client Communication:

- Read input from the client, such as adding items to the grocery list or generating the bill.
- Process the input and perform necessary operations, such as storing items or generating the bill.

4. Generate PDF:

- Utilize a PDF generation library to create a PDF document based on the grocery list and bill details.

5. Send Response to Client:

- Send the generated PDF back to the client for download.

6. Repeat for Multiple Clients:

- Implement concurrency to handle multiple client connections simultaneously.

7. Error Handling:

- Implement error handling to manage exceptions, such as handling `IOExceptions` or client disconnects gracefully.

Algorithm Overview:

1. The GroceryBillGenerator sets up a React application with components for managing user input and PDF generation.
2. The AddItemForm component handles user input for adding grocery items, including their names and prices.
3. The ItemList component displays the list of added items in a clear and organized manner.
4. The PDFButton component triggers the generation of a PDF receipt when clicked by the user.
5. When the PDFButton is clicked, the GroceryBill component collects the current date and time, along with the list of added items, and passes this data to the MyDocument component.
6. The MyDocument component uses `@react-pdf/renderer` to format the data into a PDF receipt, including the date in **dd/mm/yyyy** format, item names, prices in rupees, and the total amount.
7. The generated PDF is then downloaded by the user, providing a clear and structured representation of their grocery bill.
8. This system allows users to dynamically add items, generate a PDF receipt, and manage their grocery expenses in a user-friendly manner.

4. Implementation

Program

1. index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';
```

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
```



```
document.getElementById('root')  
);
```

2. App.css

```
.App {  
  font-family: Arial, sans-serif;  
  text-align: center;  
  padding: 50px;  
}
```

```
.add-item-form  
{ margin-bottom:  
  20px;  
}
```

```
.add-item-form input  
{margin-right: 10px;  
  padding: 5px;  
}
```

```
.add-item-form button  
{padding: 5px 10px;  
}
```

```
.item-list {  
  margin-bottom: 20px;  
}
```

```
.item-list ul {  
  list-style-type: none;  
  padding: 0;  
}
```

```
.item-list li  
{ padding: 5px  
  0;  
}
```

```
.pdf-button { padding: 10px 20px;  
background-color: #4CAF50;color: white;  
  border: none;  
  cursor: pointer;  
}
```

```
.grocery-bill
{ padding: 20px;
max-width: 600px;
margin: 0 auto;
background-color: #f7f7f7;
border-radius: 10px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
```

App.jsx

```
import React from 'react';
import GroceryBill from './components/GroceryBill';
import './App.css';
```

```
const App = () =>
{
  return (
    <div className="App">
      <GroceryBill />
    </div>
  );
};
```

```
export default App;
```

3. AddItemForm.jsx

```
import React, { useState } from 'react';
```

```
const AddItemForm = ({ addItem }) => {
  const [itemName, setItemName] = useState("");
  const [itemPrice, setItemPrice] = useState("");
```

```
  const handleSubmit = (e) =>
  {
    e.preventDefault();
    addItem({ name: itemName, price: parseFloat(itemPrice) });
    setItemName("");
    setItemPrice("");
  };
};
```

```
  return (
    <form onSubmit={handleSubmit} className="add-item-form">
      <input
        type="text"
```

```

        placeholder="Item name"
        value={itemName}
        onChange={(e) => setItemName(e.target.value)}
        required
      />
      <input
        type="number"
        placeholder="Item price"
        value={itemPrice}
        onChange={(e) => setItemPrice(e.target.value)}
        step="0.01"
        min="0"
        required
      />
      <button type="submit">Add Item</button>
    </form>
  );
};

```

```
export default AddItemForm;
```

4. GroceryBill.jsx

```

import React, { useState } from 'react';
import { Page, Text, Document, StyleSheet, PDFDownloadLink } from '@react-pdf/renderer';
import AddItemForm from './AddItemForm';
import ItemList from './ItemList';
import './App.css';

const styles =
  StyleSheet.create({page: {
    padding: 30,
    fontSize: 12,
  },
  title:
    { fontSize:
      18,
      textAlign: 'center',
      marginBottom: 20,
    },
  section:
    { marginBottom:
      10,
    },
  item: {
    display: 'flex',

```

```

    flexDirection: 'row',
    justifyContent: 'space-between',
  },
  total:
    { marginTop:
      20,
      fontSize: 16,
      textAlign: 'right',
    },
  });

```

```

const MyDocument = ({ items, date, time }) => (
  <Document>
    <Page style={styles.page}>
      <Text style={styles.title}>Grocery Bill</Text>
      <Text>Date: {date}</Text>
      <Text>Time: {time}</Text>
      <Text style={styles.section}>Items:</Text>
      {items.map((item, index) => (
        <Text key={index} style={styles.item}>
          {item.name}: ₹{item.price.toFixed(2)}
        </Text>
      ))}
      <Text style={styles.total}>
        Total: ₹{items.reduce((acc, item) => acc + item.price, 0).toFixed(2)}
      </Text>
    </Page>
  </Document>
);

```

```

const GroceryBill = () => {
  const [items, setItems] = useState([]);
  const date = new Date().toLocaleDateString('en-GB');
  const time = new Date().toLocaleTimeString();

```

```

  const addItem = (item) =>
    {setItems([...items, item]);
  };

```

```

  return (
    <div className="grocery-bill">
      <h1>Grocery Bill Generator</h1>
      <AddItemForm addItem={addItem} />
      <ItemList items={items} />
      <PDFDownloadLink

```

```

    document={<MyDocument items={items} date={date} time={time} />}
    fileName="grocery_bill.pdf"
  >
    {{{ loading }} =>
      loading ? 'Loading document...' : <button className="pdf-button">Generate PDF</button>
    }
  </PDFDownloadLink>
</div>
);
};

export default GroceryBill;

```

5. ItemList.jsx

```

import React from 'react';

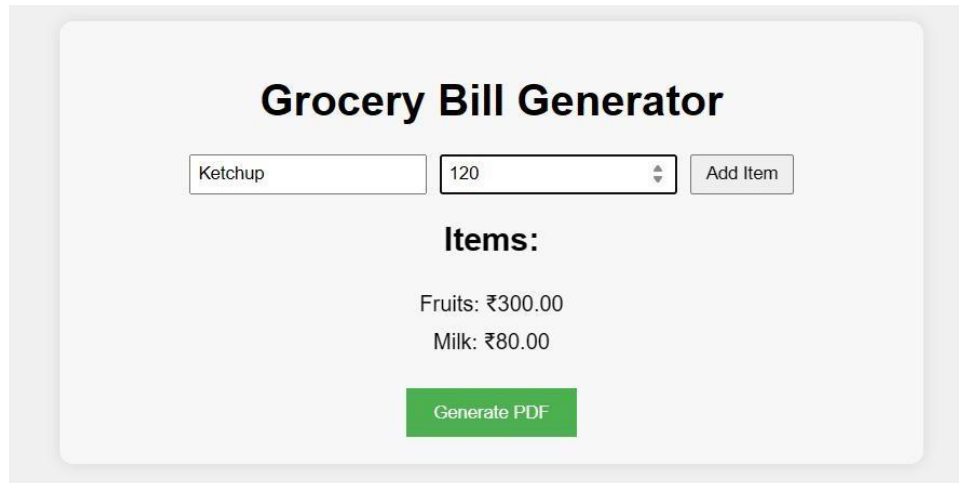
const ItemList = ({ items }) => (
  <div className="item-list">
    <h2>Items:</h2>
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item.name}: ₹{item.price.toFixed(2)}</li>
      ))}
    </ul>
  </div>
);

export default ItemList;

```

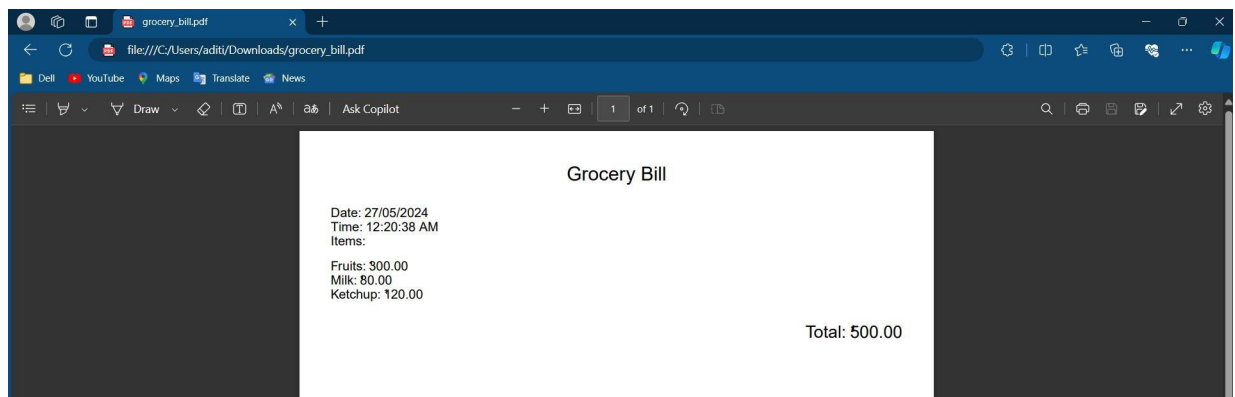
5. Results & Interpretation

In localhost:3000, the grocery app will open where we can add grocery items with their price.



The screenshot shows a web application titled "Grocery Bill Generator". It features a form with two input fields: one for the item name "Ketchup" and another for the price "120". To the right of the price field is a small up/down arrow icon. A button labeled "Add Item" is positioned to the right of the price field. Below the form, the text "Items:" is displayed, followed by a list of items and their prices: "Fruits: ₹300.00" and "Milk: ₹80.00". At the bottom of the form is a green button labeled "Generate PDF".

On clicking “Generate PDF”, a file will be downloaded and all of our items will be displayed in the pdf file.



6. Conclusion

In conclusion, the Grocery Bill Generator Application using React and PDF Generation has effectively addressed the challenge of creating a user-friendly system for managing grocery expenses and generating formatted receipts. The project has demonstrated the seamless integration of user interface development and document generation within a single-page application, providing users with a convenient solution for tracking their purchases.

By leveraging React for building the user interface and `@react-pdf/renderer` for creating PDF receipts, the application ensures a smooth and intuitive user experience. The dynamic addition of grocery items, along with the generation of a clear and structured PDF receipt, enhances the overall usability and functionality of the application.

The project's emphasis on user interaction and clear communication of information underscores its commitment to providing an efficient and user-friendly solution. The generated PDF receipt includes essential details such as the current date in ``dd/mm/yyyy`` format, item names, prices in rupees, and the total amount, ensuring clarity and accuracy.

This project serves as a valuable educational resource, offering insights into modern web development practices, including component-based architecture, state management, and third-party library integration. Furthermore, it provides a solid foundation for future enhancements, such as the addition of advanced features or integration with external APIs.

In essence, the Grocery Bill Generator Application demonstrates the successful implementation of a user-friendly solution for managing grocery expenses and generating formatted receipts, showcasing its potential for enhancing personal finance management in a digital environment.

