**Full Stack Development with MERN**

**Frontend Development Report**

|  |  |
| --- | --- |
| Date | 19 July 2024 |
| Team ID | SWTID1721040922 |
| Project Name | WalletWatch – The Expense Tracker app |
| Maximum Marks | 10 |

**Project Title: WalletWatch-Expense Tracker app**

Date: 19th July 2024

Prepared by: M.Madhuri Nagalakshmi

**Objective**

The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the Walletwatch-The expense Tracker project.

**Technologies Used**

* **Frontend Framework:** React.js
* **State Management:** Redux API
* **UI Framework/Libraries:**Bootstrap
* **API Libraries:** Axios

**Project Structure**

Give Screenshot of Frontend Project with explanation

****

**Key Components**

**public/**

* **index.html: The main HTML file that serves the React app.**
* **favicon.ico: The favicon for the app.**
* **manifest.json: The web app manifest file for Progressive Web Apps (PWAs).**

**src/**

* **The main source directory for the React application.**

**src/assets/**

* **images/: Contains image files used in the app.**
* **styles/: Contains global styles for the app.**
  + **main.css: The main CSS file for global styles.**

**src/components/**

* **Auth/: Contains components related to authentication.**
  + **Login.js: The login form component.**
  + **Register.js: The registration form component.**
  + **Logout.js: The logout component.**
* **Expenses/: Contains components related to expenses.**
  + **ExpenseList.js: Displays a list of expenses.**
  + **ExpenseItem.js: Displays a single expense item.**
  + **AddExpense.js: Form to add a new expense.**
  + **EditExpense.js: Form to edit an existing expense.**
* **Categories/: Contains components related to categories.**
  + **CategoryList.js: Displays a list of categories.**
  + **CategoryItem.js: Displays a single category item.**
  + **AddCategory.js: Form to add a new category.**
  + **EditCategory.js: Form to edit an existing category.**
* **Budgets/: Contains components related to budgets.**
  + **BudgetList.js: Displays a list of budgets.**
  + **BudgetItem.js: Displays a single budget item.**
  + **AddBudget.js: Form to add a new budget.**
  + **EditBudget.js: Form to edit an existing budget.**
* **Reports/: Contains components related to reports.**
  + **ReportSummary.js: Displays a summary report.**
  + **CategoryReport.js: Displays a report for a specific category.**
  + **DateRangeReport.js: Displays a report for a specific date range.**
* **Notifications/: Contains components related to notifications.**
  + **NotificationList.js: Displays a list of notifications.**
* **Layout/: Contains layout components.**
  + **Header.js: The header component.**
  + **Footer.js: The footer component.**
  + **Sidebar.js: The sidebar component.**

**src/context/**

* **Contains context providers for managing global state.**
  + **AuthContext.js: Context provider for authentication state.**
  + **ExpenseContext.js: Context provider for expense state.**
  + **CategoryContext.js: Context provider for category state.**
  + **BudgetContext.js: Context provider for budget state.**
  + **NotificationContext.js: Context provider for notification state.**

**src/hooks/**

* **Contains custom hooks for reusable logic.**
  + **useAuth.js: Custom hook for authentication logic.**
  + **useExpense.js: Custom hook for expense logic.**
  + **useCategory.js: Custom hook for category logic.**
  + **useBudget.js: Custom hook for budget logic.**
  + **useNotification.js: Custom hook for notification logic.**

**src/pages/**

* **Contains page components for different routes.**
  + **HomePage.js: The home page component.**
  + **LoginPage.js: The login page component.**
  + **RegisterPage.js: The registration page component.**
  + **ExpensePage.js: The expense management page component.**
  + **CategoryPage.js: The category management page component.**
  + **BudgetPage.js: The budget management page component.**
  + **ReportPage.js: The report generation page component.**

**src/services/**

* **Contains service modules for making API requests.**
  + **authService.js: Service for authentication-related API requests.**
  + **expenseService.js: Service for expense-related API requests.**
  + **categoryService.js: Service for category-related API requests.**
  + **budgetService.js: Service for budget-related API requests.**
  + **notificationService.js: Service for notification-related API requests.**

**src/utils/**

* **Contains utility modules for reusable functions.**
  + **api.js: Utility for configuring and making API requests.**
  + **validators.js: Utility for data validation functions.**

**src/App.js**

* **The main app component that sets up the application structure and routing.**

**src/index.js**

* **The entry point of the React application, where the app is rendered to the DOM.**

**src/routes.js**

* **Defines the routing configuration for the application.**

**Project Root**

* **.env: Environment variables for configuration.**
* **.gitignore: Specifies files and directories to be ignored by Git.**
* **package.json: Lists all the dependencies and scripts for the project.**
* **README.md: Contains an overview and documentation of the project.**

**Routing**

Routing is managed using React Router. Here are the main routes:

* **/home** - Landing page of the application.
* **/dashboard** - Dashboard displaying user data and statistics.
* **/profile** - User profile management.

**Integration with Backend**

The frontend communicates with the backend APIs hosted on [backend URL]. Key endpoints include:

1. **Authentication Endpoints**
   * **POST /api/register**: Register a new user.
   * **POST /api/login**: Log in a user and return an authentication token.
   * **POST /api/logout**: Log out the user.
2. **User Profile Endpoints**
   * **GET /api/profile**: Retrieve the user profile information.
   * **PUT /api/profile**: Update user profile information.
3. **Expense Endpoints**
   * **GET /api/expenses**: Retrieve a list of all expenses.
   * **POST /api/expenses**: Add a new expense.
   * **GET /api/expenses/{id}**: Retrieve details of a specific expense.
   * **PUT /api/expenses/{id}**: Update an existing expense.
   * **DELETE /api/expenses/{id}**: Delete an expense.
4. **Category Endpoints**
   * **GET /api/categories**: Retrieve a list of all categories.
   * **POST /api/categories**: Add a new category.
   * **GET /api/categories/{id}**: Retrieve details of a specific category.
   * **PUT /api/categories/{id}**: Update an existing category.
   * **DELETE /api/categories/{id}**: Delete a category.
5. **Budget Endpoints**
   * **GET /api/budgets**: Retrieve a list of all budgets.
   * **POST /api/budgets**: Add a new budget.
   * **GET /api/budgets/{id}**: Retrieve details of a specific budget.
   * **PUT /api/budgets/{id}**: Update an existing budget.
   * **DELETE /api/budgets/{id}**: Delete a budget.
6. **Report Endpoints**
   * **GET /api/reports/summary**: Retrieve a summary report of expenses.
   * **GET /api/reports/category/{id}**: Retrieve a report for a specific category.
   * **GET /api/reports/date-range**: Retrieve a report for a specific date range.

**User Interface (UI) Design**

The UI design follows a set of principles aimed at providing a user-friendly and intuitive experience. These principles include:

1. **Consistency**: Ensuring uniformity in design elements like colors, fonts, and button styles across the application for a cohesive look and feel.
2. **Simplicity**: Keeping the interface clean and straightforward to avoid overwhelming users. This includes using clear labels, icons, and minimalistic layouts.
3. **Accessibility**: Designing with all users in mind, including those with disabilities. This involves using proper contrast ratios, keyboard navigability, and screen reader compatibility.
4. **Responsiveness**: Ensuring the app works well on all devices, from mobile phones to desktops, by employing responsive design techniques.
5. **Feedback**: Providing immediate and clear feedback to user actions, such as button clicks and form submissions, to enhance user interaction and satisfaction.
6. **Visual Hierarchy**: Using size, color, and placement to prioritize elements, guiding users to the most important actions and information first.

**UI Framework/Library**

The frontend is implemented using **React**, a popular JavaScript library for building user interfaces. React's component-based architecture allows for modular and reusable UI components, which enhances development efficiency and maintainability.

**Additional Libraries and Tools:**

1. **React Router**: For handling client-side routing and navigation within the app.
2. **Context API**: For state management, allowing for a predictable and centralized way to manage application state.
3. **Axios**: For making HTTP requests to the backend APIs.
4. **CSS Modules**: For styling components, enabling scoped and maintainable CSS.
5. **Bootstrap**: For pre-built components and responsive design, speeding up the development process and ensuring a consistent UI.