PIF Client: Payment Process and Purchase History Documentation

---

Booking Page Workflow

Overview

The **Booking Page** handles fetching event data, managing referral codes, and processing payments.

Workflow Steps:

1. **On Page Load:**
    - **Fetch Event Details:**
        - Extract `eventId` from URL parameters.
        - Use `useEventStore()` to fetch and set the event data in the component's state.
    - **Fetch Shared Referral Code:**
        - Use `usePaymentStore()` to call `fetchSharedReferralCode()`.
        - Set the referral code data to state if available.
2. **Referral Code Handling:**
    - **Input Field:**
        - A text input allows users to enter a referral code.
        - If a shared referral code exists, it auto-fills the input field.
    - **On Input Change:**
        - The referral code is updated in real-time using the `setReferralCode` function.
    - **Confirmation:**
        - On clicking the Confirm button:
            - `handleReferralCode` function is called.
            - Checks if the referral code is valid:
                - **If valid:**
                    - Sets the state `isReferralCode` to `true`.
                    - Stores referral information in `localStorage` under the key `referralInfo`.
                - **If invalid:**
                    - Displays an error message or prompt.

3. **Payment Handling:**

    - **Terms and Conditions:**
        - Payment button is enabled only if the user agrees to the terms and conditions.
    - **Initiate Payment:**
        - On clicking the Make Payment button:
            - `handlePayment` function is called with arguments: `eventTitle`, `price`, `eventId`, `userId`, and `description`.
            - Processes the payment and redirects to the **Payment Success Page** upon success.
4. **Discount Calculation:**

    - If a valid referral code is applied:
        - Calculate the discount based on the code.
        - Update the discounted price in the state.

Payment Success Page Workflow

Overview

The **Payment Success Page** verifies payment details, processes additional actions such as calendar creation, and displays the invoice.

Workflow Steps:

1. **On Page Load:**

    - Extract `eventId` and `sessionId` from the URL parameters.
    - Retrieve `eventDetails` from cookies and fetch event data using `eventId`.
    - Use `useImmerReducer` to set all data to the component's state.
2. **Verify Payment:**

    - If `userId` and `username` are present:
        - Call the `verifyPayment` function:
            1. **Session Verification:**
                - Call `checkSession` with:
                    - `sessionId`, `userId`, `eventId`, `username`, and `userDocumentId`.
                - On success:

2. **Calendar Creation:**
    - Call `createCalendarData` with:
        - `eventDocumentId`, `userDocumentId`, and `eventDate`.
3. **User Event Data Creation:**
    - Call `createUserEventData` with:
        - `eventDocumentId`, `userDocumentId`, `eventStatus`, `referral` (or `null` if unavailable), `bookDate`, `startTime`, and `endTime`.
4. **Referral Handling:**
    - If `referralInfo` is available:
        - Call `createCommission` with:
            - `discountPrice`, `eventId`, `referralUsername`, and `userDocumentId` for the referrer.
        - Call `updateUserBySharedReferral` with:
            - `userId` and `sharedReferralId`.

3. **Invoice Display:**
    - Show an invoice modal and render the invoice as a PDF using the `react-pdf` render package.

Purchase History Component Documentation

Overview

The Purchase History component is a React functional component designed to display a user's payment history for events. It provides various features such as filtering payment history, pagination, viewing invoices, and downloading them as PDF files.

Features

- **Fetch Payment History:**
    - Retrieves payment history for a specific user by invoking the `getSpecificUserPaymentHistories` function.
    - Utilizes the `usePaymentHistoryStore` Zustand store for state management.
- **Filtering Payment History:**

    - Filters history by time periods:
        - Today, This Week, Last 2 Weeks, This Month, Last 6 Months, All Time.
    - Displays results dynamically based on the selected filter.
- **Pagination:**
    - Handles pagination to load additional pages of payment history.
    - Uses "Next" and "Previous" buttons for navigation.
- **Invoice Handling:**
    - Displays invoice details in a modal.
    - Allows users to download the invoice as a PDF.
- **Event Status and Payment Status Display:**
    - Displays the event's progress status (e.g., "Not Started," "In Progress," or "Completed").
    - Displays payment status (e.g., "Paid," "Unpaid," or "Partial") with color-coded text.

Dependencies

- **State Management:**
    - Zustand: `usePaymentHistoryStore` and `useAuthStore` for managing payment histories and user data.
    - `useImmerReducer` for managing the local state with an immer reducer.
- **Third-Party Libraries:**
    - React Icons: For icons such as chevrons and calendar icons.
    - `@react-pdf/renderer` and `file-saver`: For generating and downloading PDF files.
    - `dayjs`: For date manipulation and comparison.

Props

This component does not accept any props directly. It relies on Zustand stores for data.

State Management

The local state is managed using `useImmerReducer`. Below are the state variables and their purpose:

- **filter (Filter):** Determines the current filter applied to the payment history.
- **page (number):** Tracks the current page in the pagination.
- **showModal (boolean):** Indicates whether the invoice modal is visible.
- **invoiceData (InvoiceData | null):** Stores the invoice data for the modal.

Reducer

The reducer manages state transitions based on dispatched actions:

- **Actions:**
  - `SET_FILTER`: Updates the filter and resets the page to 1.
  - `NEXT_PAGE`: Increments the page number.
  - `PREV_PAGE`: Decrements the page number but ensures it does not go below 1.
  - `RESET_PAGE`: Resets the page to 1.
  - `TOGGLE_MODAL`: Shows or hides the invoice modal.
  - `INVOICE`: Sets the invoice data to be displayed.

Key Functions

- **useEffect Hook:**
  - Fetches user-specific payment histories when filter, page, or `documentId` changes.
- **handleFilterChange:**
  - Dispatches the `SET_FILTER` action to update the filter.
- **handleNext and handlePrev:**
  - Dispatches `NEXT_PAGE` and `PREV_PAGE` actions for pagination.
- **toggleModal:**
  - Shows or hides the invoice modal.
- **handleDownload:**
  - Generates and downloads a PDF of the invoice using `@react-pdf/renderer` and `file-saver`.

UI Details

- **Header Section:**
  - Displays a summary of events for the current month:
    - Total events
    - Attended events
  - Filter buttons for selecting the time range.
- **Table Section:**
  - Displays a table with the following columns:

        - Event Name
        - Purchase Date
        - Invoice Number (clickable for modal and download)
        - Price
        - Payment Mood (Paid, Unpaid, Partial)
        - Event Status (Not Started, In Progress, Completed)
        - Event Date
- **Pagination:**
    - Pagination buttons for navigating between pages.

Utility Functions

- **getPaymentStatus:**
    - Maps payment statuses ("Paid", "Unpaid", "Partial") to color-coded text.
- **getEventStatus:**
    - Compares event dates with the current date to determine the status (Not Started, In Progress, Completed).
- **getButtonClass:**
    - Determines the button styling based on the selected filter.
- **getFilterDisplayName:**
    - Converts filter values into user-friendly labels.

Key Functions

1. `handleBook` (Event Details Page)

- **Purpose:** Sets event details into cookies and redirects to Booking Page.
- **Parameters:** None.
- **Workflow:**
    1. Sets `eventDetails` into browser cookies.
    2. Redirects using `router.push('/booking-page')`.

2. `fetchSharedReferralCode` (Booking Page)

- **Purpose:** Fetch shared referral code if available.
- **Workflow:**

    1. Uses `usePaymentStore()` to fetch the referral code.
    2. Sets data into the component's state.

3. `handleReferralCode` (Booking Page)

- **Purpose:** Validates and applies the referral code.
- **Workflow:**
    1. Checks if the entered referral code is valid.
    2. Updates state and `localStorage` with referral info if valid.

4. `handlePayment` (Booking Page)

- **Purpose:** Processes the payment.
- **Workflow:**
    1. Ensures terms and conditions are agreed upon.
    2. Sends payment data and initiates the payment process.
    3. Redirects to the Payment Success Page upon completion.

5. `verifyPayment` (Payment Success Page)

- **Purpose:** Verifies payment session, creates calendar entries, and handles user-event associations.
- **Workflow:**
    1. Calls `checkSession` to verify the session.
    2. Calls `createCalendarData` to create calendar events.
    3. Calls `createUserEventData` to log user-event details.
    4. Handles referral commissions and updates.

State Management

Booking Page State:

1. **Event Data:**
    - Set using `useEventStore()` based on `eventId`.
2. **Referral Code Data:**
    - Set using `usePaymentStore()` and updated via `handleReferralCode`.
3. **Discount Information:**
    - Calculated and updated in the state after referral code validation.

Payment Success Page State:

1. **Event Details:**
    - Retrieved from cookies and set using `useImmerReducer`.
2. **User and Session Details:**
    - Managed for verifying payment and processing related tasks.

Additional Notes

- **Referral Code Validation:**
    - Ensure backend APIs for referral validation are reliable.
    - Consider edge cases where referral codes may expire or be invalidated.
- **User Experience:**
    - Provide clear feedback on successful or failed referral code application.
    - Highlight discounts prominently after applying the referral code.
- **Invoice Generation:**
    - Use `react-pdf` to ensure cross-browser compatibility and PDF export functionality.
