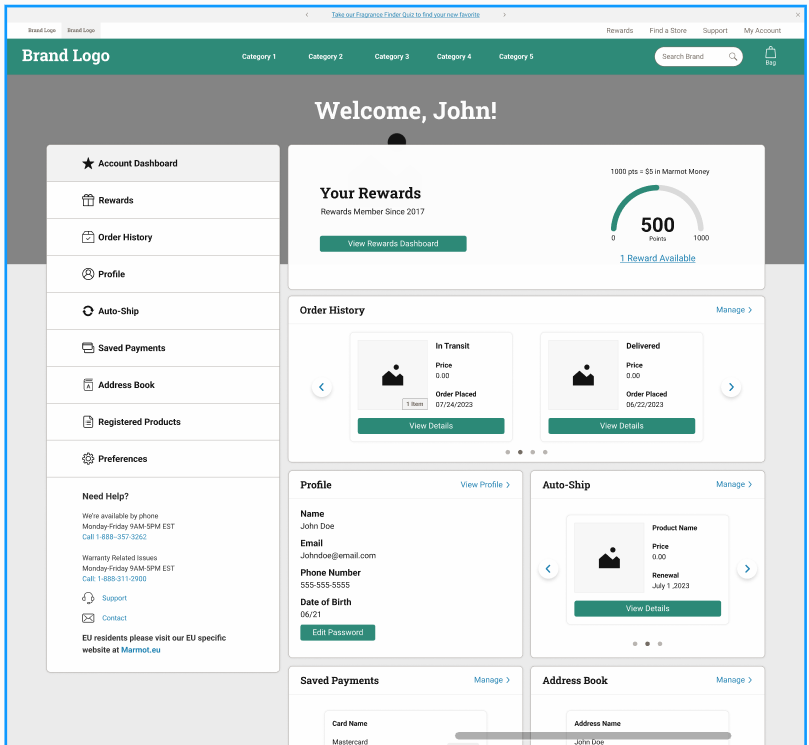
**Account Dashboard (Marmot)**

**Introduction**

The Account Dashboard is a page component where customers can easily view and manage important account information. This includes updating personal profile details, checking past orders, tracking rewards, managing shipping details, managing addresses, handling payment options, etc.

**UI comprises of the following:**

1. A responsive navigation bar that navigates users smoothly to different sections such as profile, orders, payment, shipping, and address-related information.
2. For each section in the navigation bar, there is a brief summary providing an overview. This helps users to view summarized information of that section, also they can view more details by clicking on the “manage” button or through a navigation bar that will navigate them to the secondary page.



This page can be accessed on brand website by url :<https://test2.marmot.com/account?abtest=A>.

The “/account” is a pathname for accessing the account dashboard page.

**Steps to Configure Routing for Account Dashboard**

1. Find the routes.jsx file in src directory.
2. In the routes.jsx file, routes are defined as objects in an array. Each object represents a route.
3. For the Account Dashboard, the object might look something like this:

{

path: "/account",

component: Account,

exact: true,

}

1. **path:** The URL path for the route ('/account' for the Account Dashboard).
2. **component:** The Account component to render when this route is accessed .
3. Add the route object to the routes array in routes.jsx.

The source code for the component is in **pages/account/index.jsx**.

This component has a **shouldGetProps** function from the PWA kit that **returns “true” on change in the pathname or query params**. It also includes a **getProps** function that runs again when there's a change in these parameters.

The component gets information from site preferences, specifically “**accountDashboardTiles**”, to determine which sections are visible on the page.

The expected JSON format for site preference is:

["accountDashboard", "profile", "orderHistory", "password", "savedPayments", "subscription", "addressBook", "preferenceCenter”]

The sections displayed on the page will match the order and content of the accountDashboardTiles array.

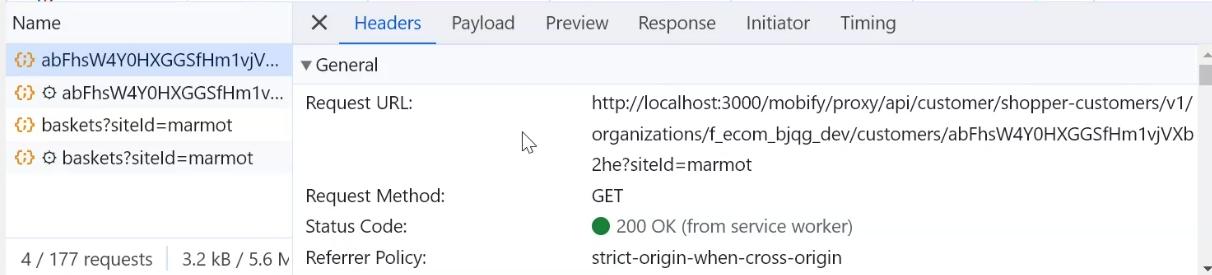
**PageId Prop and its Functionality:**

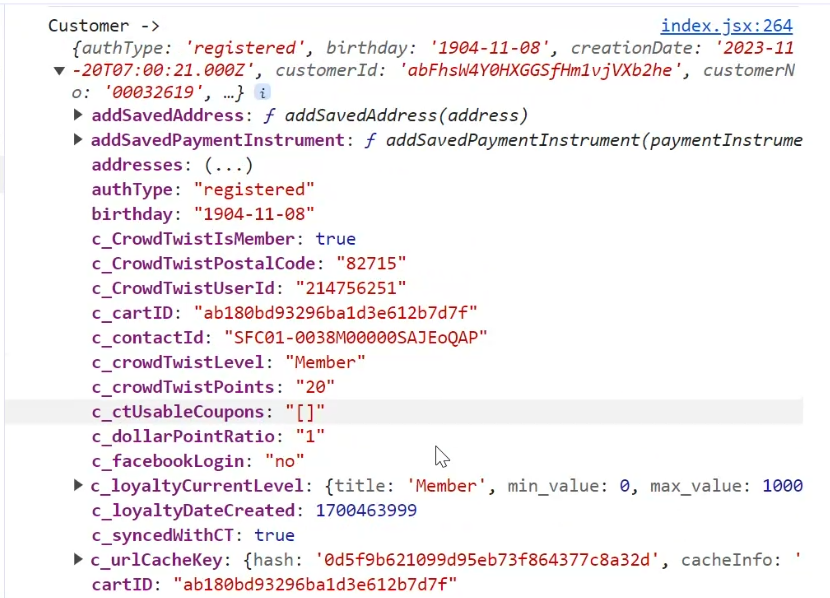
The page receives a “pageId” prop, which aligns with one of the items in the accountDashboardTiles from the site preferences. This prop identifies the active section of the Account Dashboard, whether it's the main page or some secondary page.

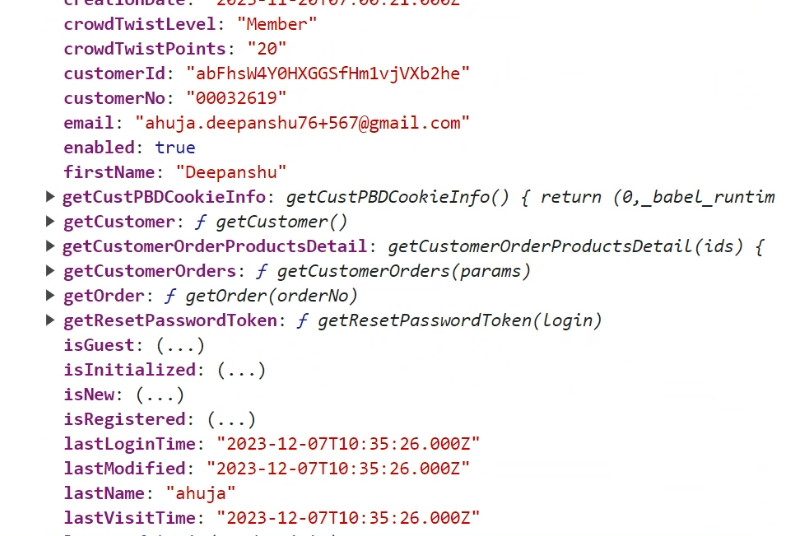
**Fetch Strategy**

**Customer Data:** For the Account page, customer data, including Crowd Twist information, is fetched through the useCustomer hook. This hook utilizes the customer SCAPI API and is triggered right after the user successfully logs in. It's also triggered when a logged-in user opens the app in a browser or a new tab, ensuring that the most current customer and loyalty program information is always available, regardless of how or when the user accesses their account.

**Order History Data:** The order history of the customer is retrieved on the client side using the Deck Commerce API ([Deck Commerce](https://newellbrands.atlassian.net/wiki/spaces/DTI/pages/3573776388/Deck+Commerce+-+SFCC+-+Proxy)).



****

****

**Integration with Complib Library Components:**

This page incorporates components from the complib library.

**Details on Complib Components:**

**AccountDashboardNavBar**

* **Name:** AccountDashboardNavBar
* **File Path:** src/acc-navbar/index.jsx (complib).
* **Description:** This is a responsive navigation bar designed to help users easily navigate to various sections of the Account Dashboard, like profile, orders, payments, shipping, etc.

**Props**

* **navStyles** (object): Custom styles for the navigation bar.
* **data** (Array): An array of navigation items.
* **selectedItem** (object): The currently selected navigation item.
* **Icons** (object): Icons to be displayed within the navigation bar.
* **isMobile** (bool): Indicates if the current screen size is mobile.
* **onNavItemClick** (func): Function to execute on navigation item click.
* **toggleMobileMenu** (func): Function to toggle the mobile menu's open/close state.
* **isMenuOpen** (bool): Indicates the open/close state of the mobile menu.

All items o**f AccountDashboardNavBar** will be the same as the items in **accountDashboardTiles from site preferences.**

**Example:**

<AccountDashboardNavBar

navStyles={customStyles}

data={[

{ label: 'Account Dashboard', value: 'accountDashboard' },

{ label: 'Profile', value: 'profile' },

{ label: 'Order History', value: 'orderHistory'}

]}

selectedItem={{ label: 'Account Dashboard', value: 'accountDashboard' }}

Icons={Icons}

isMobile={true}

onNavItemClick={handleNavItemClick}

toggleMobileMenu={handleToggleMobileMenu}

isMenuOpen={isMenuOpen}

/>

**Menu and Icon Management:**

* **isMenuOpen:** This is managed within the page component using useState, controlled via **toggleMobileMenu**.
* **Icons:** The Icons prop should correspond with each element's value in **accountDashboardTiles**, representing it with an SVG icon.

{

profile: profileSVGIcon,

orderHistory: orderHistorySVGIcon,

address: addressSVGIcon,

payment: paymentSVGIcon

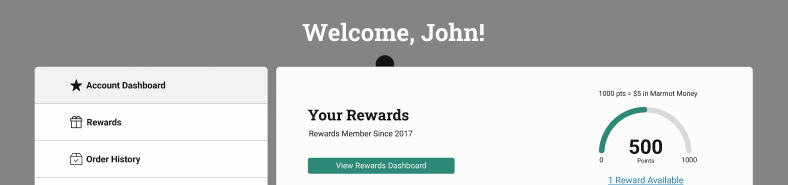
}

**Determining the Selected Item:**

* The selectedItem prop is derived based on the **pageId**, ensuring that the navigation bar accurately reflects the currently active section of the dashboard.

**HeaderWithBackground**

* **Name:** HeaderWithBackground
* **File Path:** src/header-with-background/index.jsx (complib).
* **Description:** This component displays a header with a custom background image and text with a welcome message on an image that adjusts to fit any screen size.

****

**Props**

* **welcomeMessage** (string): The welcome message text displayed over the background image.
* **background** (Object):
  + **desktopImage** (string, required): The image URL for desktop view.
  + **tabletImage** (string, optional): The image URL for tablet view. Defaults to desktopImage if not provided.
  + **mobileImage** (string, optional): The image URL for mobile view. Defaults to desktopImage if not provided.
  + **imageAltText** (string, required): Alternative text for the image, enhancing accessibility.
* **styles** (Object, optional): Custom styles that can be applied to the component for specific customization needs.

**Example**:

<HeaderWithBackground

welcomeMessage="Welcome to Our Store!"

background={{

desktopImage: "url-to-desktop-image.jpg",

tabletImage: "url-to-tablet-image.jpg",

mobileImage: "url-to-mobile-image.jpg",

imageAltText: "A welcoming storefront",

}}

styles={customStyles}

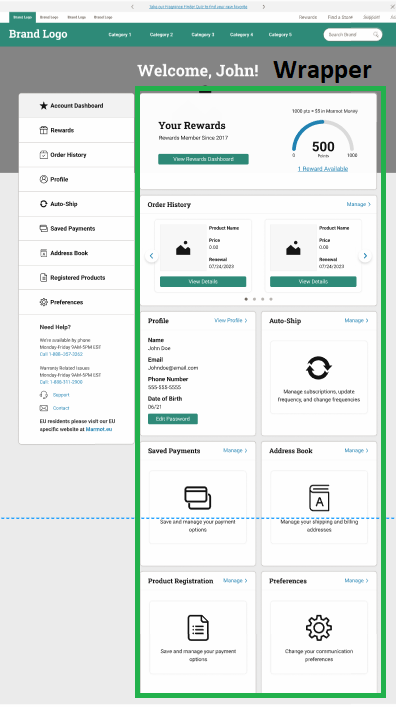
/>

**Data Source**

In the current implementation, this component receives mock data from “**accountHeaderMock**” located in “**mockData.js**”.

**3. *AccountDashboardWrapper***

* **Name:** AccountDashboardWrapper
* **File Path:** src/acc-dashboard-wrapper/index.jsx (complib).
* **Description:** This component serves as a wrapper for various sections of the account dashboard. It dynamically renders different dashboard sections based on the provided configurations and data.

****

**Props**

* **accountDashboardConfig** (Object): Configuration object defining components to render for each section.
* **orderedSections** (Array): An array of objects detailing the order and configuration of the dashboard sections.
* **Icons** (Object): Object containing icons used in the component.

**Example**

<AccountDashboardWrapper

accountDashboardConfig={dashboardConfig}

orderedSections={sections}

Icons={dashboardIcons}

/>

The **dashboardConfig** is a configuration object within the Account Dashboard, where each value from **accountDashboardTiles** is mapped with a specific component.

**For example**: For **'profile'** or **'orderHistory'**, a respective component (e.g., profileComponent, orderHistoryComponent) is assigned as a value. These components are responsible for rendering the respective sections on the dashboard.

{

profile: profileComponent,

orderHistory: orderHistoryComponent,

}

The `**orderedSections**` array contains objects, each representing a specific section of the Account Dashboard.

The structure of these objects is as follows:

1. **label**: The text label displayed for each section.

2. **header** (boolean): Determines if the label should be shown, based on the truthiness of the header.

3. **value**: This should correspond to one of the values in `accountDashboardTiles`.

4. **data**: Contains the necessary prop information for the respective section components defined in `dashboardConfig`.

5. **onActionLabelClick**: Function to be executed on clicking “**manage”** cta link of section tile. In the current implementation, this function will navigate to the secondary page.

**NOTE**: The keys in the `**dashboardConfig**` object and the `**value**` in each `**orderedSections**` object must align with the values specified in `**accountDashboardTiles**`. This alignment ensures that the dashboard sections are correctly matched and rendered with their respective components.

**AccountDashboardNeedHelp**

* **Name:** AccountDashboardNeedHelp
* **File Path:** src/acc-dashboard-need-help/index.jsx (complib).
* **Description:** This component renders the "Need Help" section of the account dashboard page. It provides users with information and resources for assistance, including customer service availability, contact options, and relevant website links.

**Props**

* **title** (string): The title of the 'Need Help' section.
* **services** (Array): An array of service objects, each containing details about available services (text, days, time, call).
* **reachUs** (Array): An array of objects providing information on how to reach out, including icons, text, redirect URLs, and targets for the links.
* **websites** (Array): An array of objects representing available websites, including text, website name, URL, and target for the link.

**Example**:

<AccountDashboardNeedHelp

title="Need Assistance?"

services={[

{

text: "Customer Service",

days: "Mon-Fri",

time: "9am-5pm",

call: "1234567890",

},

]}

reachUs={[

{

icon: "icon-url",

text: "Email Us",

redirectURL: "mailto:example@example.com",

target: "\_blank",

},

]}

websites={[

{

text: "Visit our FAQ",

name: "FAQ",

url: "https://example.com/faq",

target: "\_blank",

},

]}

/>

**Data Source**

In the current implementation, this component receives mock data from “**needHelpData**” located in “**mockData.js**”.

**Rewards (Section Component)**

The Rewards section on the Account Dashboard becomes visible when the “**crowdTwistEnabled**” feature is activated in site preferences. This is determined through the **useSitePreferences**('**CrowdTwistEnabled**') hook.

const CrowdTwistEnabled = useSitePreferences('CrowdTwistEnabled’)

This reward section comprises of two scenarios :

* **Enrolled in Rewards:** When the user is part of the rewards program.
* **Not Enrolled in Rewards:** When the user is not a participant in the rewards program.

**Data Source**

The data for this section, including the user's status in the rewards program, is sourced from the customer object obtained from the useCustomer hook.

This object contains various details about the customer, including their CrowdTwist membership status and related information.

customer: {

…some info,

"c\_CrowdTwistIsMember": true,

"c\_CrowdTwistPostalCode": "82715",

"c\_CrowdTwistUserId": "214756251",

"c\_contactId": "SFC01-0038M00000SAJEoQAP",

"c\_crowdTwistLevel": "Member",

"c\_crowdTwistPoints": "20",

"c\_ctUsableCoupons": "[]",

"c\_syncedWithCT": true,

"c\_loyaltyDateCreated": 1700463999,

"c\_loyaltyCurrentLevel": {

"title": "Member",

"min\_value": 0,

"max\_value": 100000000,

"image\_url": null,

"current": true,

"progressBarWidth": "0.00%",

"prevTier": false,

"nextTier": null,

"displayInMobile": true,

},

…more

}

For instance, when **customer.c\_CrowdTwistIsMember** is **true**, it triggers the display of the “**Enrolled in Rewards**” component.

If this value is **false**, the “**Not Enrolled in Rewards**” component is shown.

**AccountDashboardRewards (** enrolled in rewards **)**

* **Name:** **AccountDashboardRewards**
* **File Path:** src/acc-dashboard-rewards/index.jsx (complib).
* **Description:** This component is utilized within the account dashboard to provide users with a quick overview of their rewards information, including points progress, and available rewards.

****

**Props**

* **ishideTier** (boolean): Indicates whether the tier information should be hidden.
* **enrolledYear** (number): The year the user enrolled in the rewards program.
* **pointsProgressProps** (Object): Properties for the points progress bar, including value, max, size, filledStroke, stroke, strokeLinecap, and strokeWidth.
* **tiersInfo** (Object): Information about the available tiers and their expiration dates.
* **totalRewards** (number): The total number of rewards available to the user.
* **paths** (Object): Object containing navigation paths like rewardsAvailable, howToEarnPoints, and viewRewardsBtn.
* **onViewRewards** (function): Callback function triggered when the "View Rewards" button is clicked.
* **rewardsText** (Object): Object containing text and labels for various elements in the component.

In current implementation, paths props are navigating to rewards secondary page.

**ishideTier** is true for “**marmot**” and hence no tier information will be displayed.

**Example**:

<AccountDashboardRewards

data={{

ishideTier: true,

enrolledYear: 2021,

pointsProgressProps: {

value: 75,

max: 100,

size: 200,

filledStroke: "#000",

stroke: "#ddd",

strokeLinecap: "round",

strokeWidth: 10,

},

totalRewards: 5,

paths: {},

onViewRewards: () => {},

variant: "default",

rewardsText: {},

}}

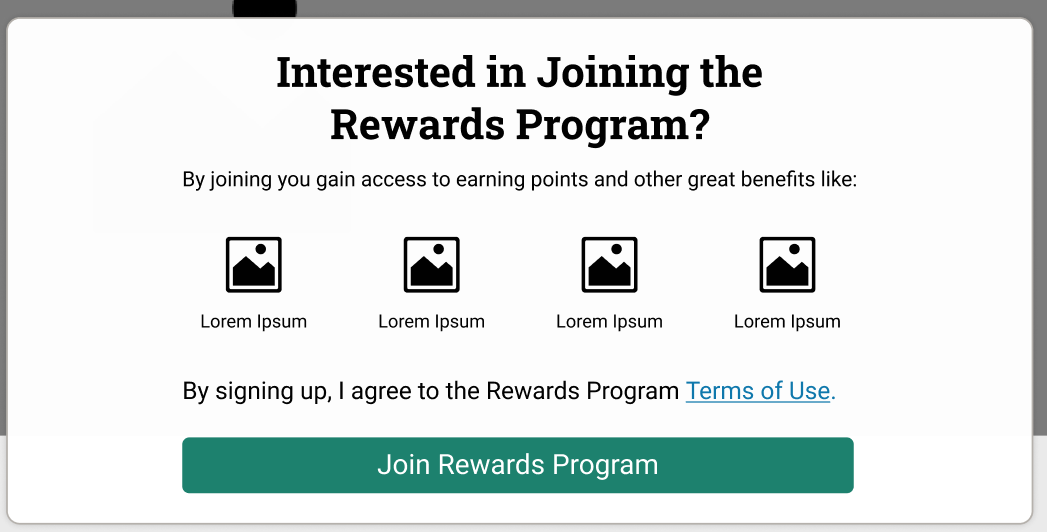
/>

**Data Sources:**

1. Data is available in customer from useCustomer hook
2. Static text and labels for the component are sourced from translationMessages via the **getTranslations** function from utils.js.

**AccountDashboardNoRewards (** notenrolled in rewards **)**

* **Name:** AccountDashboardNoRewards
* **File Path:** src/acc-dashboard-no-rewards/index.jsx (complib).
* **Description:** This component would be displayed for non loyalty members and it is utilized within the account dashboard to provide users the information about a brand’s loyalty program and be given the ability to sign up, so that they can collect rewards.

****

**Props**

* **title** (string): The title of the no rewards section.
* **benefitsIntroDesktop** (string): The introductory text for desktop view.
* **benefitsIntroMobile** (string): The introductory text for mobile view.
* **benefits** (Array): An array of benefit items to be displayed.
* **legalDisclaimer** (Object): Contains the legal disclaimer text and related properties.
* **cta** (Object): Call to action information including the button title and click handler.
* **breakpoints** (Object): Breakpoint conditions for responsive design.

**Example**:

<AccountDashboardNoRewards

title="No Rewards Available"

benefitsIntroDesktop="Explore the benefits you can earn"

benefitsIntroMobile="Explore benefits"

benefits={[...]}

legalDisclaimer={{ consent: "consent", link: "/link", tnc: "Terms & Conditions" }}

cta={{ title: "Learn More", onActionLabelClick: handleLearnMoreClick }}

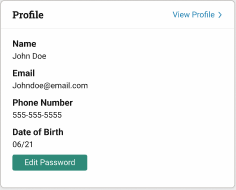
/>

**Data Source**

In the current implementation, this component receives mock data from “**noRewardsMock**” located in “**mockData.js**”.

**AccountDashboardProfile**

* **Name:** AccountDashboardProfile
* **File Path:** src/acc-dashboard-profile/index.jsx (complib).
* **Description:** This component is used for displaying user profile information within the account dashboard. It shows details like name, email, phone number, birthday, and provides links to edit these details. In current implementation, all links will navigate to “**profile**” secondary page

****

**Props**

* **textInfo** (Object): Contains text and labels for various elements in the component. It includes labels like name, email, phoneNumber, addPhoneNumber, birthday, addBirthday, and editPassword.
* **data** (Object): Customer's profile data to be displayed.

Includes fields such as name, email, phoneNumber, and birthday.

**links** (Object): Paths for editing user information.

Contains URLs for editing phone number, birthday, and password.

**Example**:

<AccountDashboardProfile

textInfo={{

name: "Name",

email: "Email",

phoneNumber: "Phone Number",

addPhoneNumber: "Add Phone Number",

birthday: "Birthday",

addBirthday: "Add Birthday",

editPassword: "Edit Password",

}}

data={{

name: "John Doe",

email: "johndoe@example.com",

phoneNumber: "123-456-7890",

birthday: "01/01/1990",

}}

links={{

phoneNumber: "/edit-phone",

birthday: "/edit-birthday",

editPassword: "/edit-password",

}}

/>

**Data Sources:**

1. Data is available in customer from useCustomer hook.
2. Static text and labels for the component are sourced from translationMessages via the **getTranslations** function from utils.js.

customer: {

"authType": "registered",

"birthday": "1904-11-08",

"creationDate": "2023-11-20T07:00:21.000Z",

"customerId": "abFhsW4Y0HXGGSfHm1vjVXb2he",

"customerNo": "00032619",

"email": "ahuja.deepanshu76+567@gmail.com",

"enabled": true,

"firstName": "Deepanshu",

"lastLoginTime": "2023-11-30T08:50:35.000Z",

"lastModified": "2023-11-30T09:53:41.000Z",

"lastName": "Ahuja",

"lastVisitTime": "2023-11-30T09:53:40.997Z",

"login": "ahuja.deepanshu76+567@gmail.com",

"phoneMobile": "1234447890",

…more data

}

**Order History**

* **Name:** AccountDashboardOrderHistoryCrousel
* **File Path:** src/acc-dashboard-order-history-crousel/index.jsx (complib).
* **Description:** The Order History section of the Account Dashboard is designed to show a customer's order history.

It includes two scenarios:

1. **For Customers with Past Orders**: Customers will see their order details presented in a carousel format for easy browsing and review of their purchase history.
2. **For Customers without Orders**: Customers will see the product recommendations.

**Data Sources:**

* **Order History**: This is obtained by fetching the customer's previous order information from the Deck Commerce API. More details can be found at the [Deck Commerce - SFCC - Proxy](https://newellbrands.atlassian.net/wiki/spaces/DTI/pages/3573776388/Deck+Commerce+-+SFCC+-+Proxy) documentation.
* **Product Recommendations**: For customers without order history, product recommendations are sourced from Einstein's API, specifically using the "**best-selling-prod-in-all-categories**" recommender.

**Fetch Strategy:**

* When the page is rendered, the system fetches the logged-in customer's order details from Deck Commerce on the client side.
* If the customer has past orders, the response from Deck Commerce is formatted and stored in a state variable for display.
* In cases where there are no past orders, product recommendations are fetched from Einstein's API, with the response similarly formatted and stored.
* This state variable is then utilized to present the relevant information to the logged-in customer.

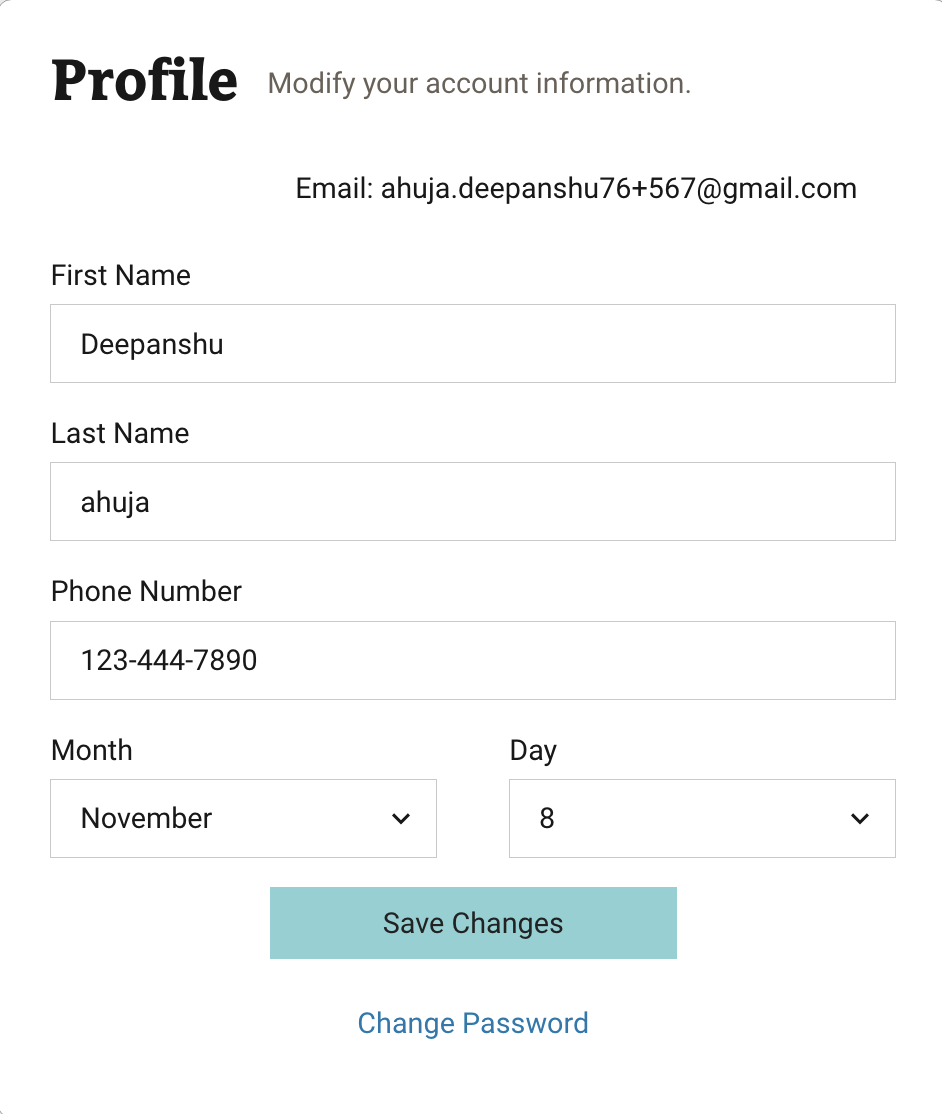
**Secondary pages**

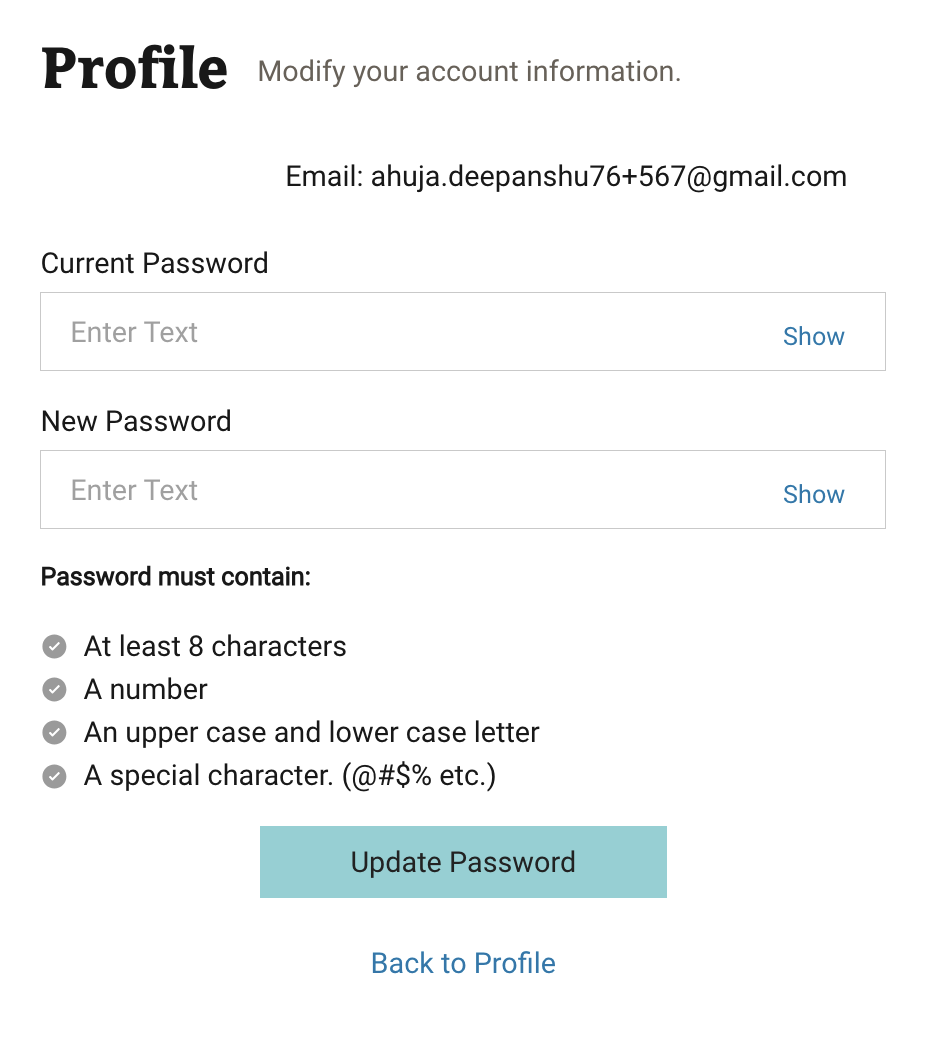
**1. Profile**

When the **pageId** is set to **'profile'**, the Profile secondary page is shown on the Account Dashboard. This page is where users can update their personal information and also change their password, ensuring users have easy access to modify their information or update their password as needed.

**Component Overview**

* **Name:** AccountDashboardEditProfile
* **File Path:** src/acc-dashboard-edit-profile/index.jsx (complib).
* **Description:** A component that renders forms for editing customer profile details and changing the password. It toggles between the edit profile form and the change password form based on user interaction.

****

****

**Props**

* **textInfo** (Object): Contains labels, placeholders, page titles, descriptions, and error messages for form fields.
* **daysInMonths** (Object): An object mapping month names to their numeric values and days count, used for date inputs.
* **customer** (Object): Customer data to pre-populate in the form.
* **isMobileView** (boolean): Flag to indicate if the view is on a mobile device.
* **onUpdateProfile** (Function): Callback function for profile update form submission.
* **Icons** (Object): Icons used in the component.
* **isLoyaltyMember** (boolean): Indicates if the user is a loyalty program member.
* **rules** (Object): Validation rules for form fields.
* **onUpdatePassword** (Function): Callback function for password update form submission.
* **showChangePasswordForm** (boolean): Flag to toggle between the edit profile form and change password form.
* **onEditProfileViewChange** (Function): Function to handle the view change between profile and password forms.

**Example**

**<AccountDashboardEditProfile**

**textInfo={{**

**labels: {**

**firstName: "First Name",**

**lastName: "Last Name",**

**phoneNumber: "Phone Number",**

**month: "Month",**

**day: "Day",**

**currentPassword: "Current Password",**

**newPassword: "New Password",**

**saveButton: "Save",**

**changePasswordLink: "Change Password",**

**updatePassword: "Update Password",**

**editProfile: "Edit Profile"**

**},**

**placeholder: {**

**firstName: "Enter First Name",**

**lastName: "Enter Last Name",**

**phoneNumber: "Enter Phone Number",**

**},**

**pageTitle: "Edit Profile",**

**description:"description",**

**passwordRequirements: {**

**// Password requirements text**

**},**

**errorMessages: {**

**firstName: "First name is required",**

**lastName: "Last name is required",**

**phoneNumber: "Invalid phone number",**

**day: "Day is required",**

**passwordEmpty: "Password cannot be empty",**

**invalidNewPassword: "New password does not meet requirements"**

**}**

**}}**

**daysInMonths={{**

**January: 31,**

**February: 29,**

**March: 31,**

**}}**

**customer={{**

**name: "Deepanshu Ahuja",**

**email: "deepanshu@example.com",**

**phoneNumber: "123-456-7890",**

**birthday: "1990-01-01"**

**}}**

**isMobileView={true}**

**onUpdateProfile={handleUpdateProfile}**

**Icons={{**

**checkIcon: CheckIcon,**

**fallback: FallbackIcon**

**}}**

**isLoyaltyMember={true}**

**rules={{**

**maxLength: {**

**firstName: { length: 40, errorMessage: "First name too long" },**

**lastName: { length: 40, errorMessage: "Last name too long" },**

**}**

**}}**

**onUpdatePassword={handleUpdatePassword}**

**showChangePasswordForm={false}**

**onEditProfileViewChange={handleViewChange}**

**/>**

**Data Sources:**

* Data is available in customer from useCustomer hook
* Static text and labels for the component are sourced from translationMessages via the getTranslations function from utils.js.

If a customer is a member of the CrowdTwist program, represented by `**customer.c\_CrowdTwistIsMember**` being **true**, then the phone number and birthday input fields are displayed on the user interface.

**Regarding the functionality based on the browser URL:**

1. When the URL includes `**?mode=editPassword**`, the page shows a form for changing the password.

2. If this parameter is not present, the default display is the edit profile form.

**For the actions on this page:**

* The `**updateCustomer**` function from the `**useCustomer**` hook is used by “**handleUpdateProfile”** function for updating profile details.
* The `updatePassword` function, also from the `useCustomer` hook is used by the “**handleUpdatePassword”** function for changing the user's password.

On successfully changing their password, users are redirected back to the main Account Dashboard page.