Introduction

This document details the software design for the Salesbricks Order Builder, a multi-stage user interface designed to help sellers construct orders for buyers. The objective of the project is to provide an intuitive workflow that guides sellers through the entire order process

Goals and Objectives

a. Functional Goal: To deliver a complete 4 stage workflow that implements all required functionality, including: customer data entry, product and plan selection, price editing for plans, contract term definition with automatic date calculation, and a review screen with an accurate order summary
b. UX Goal: To provide an intuitive and accessible user experience. This is achieved through clear, logically grouped form inputs with clear labeling, non-obtrusive real time validation, and a design that ensures clarity and ease of use.
c. Technical Goal: To develop a high quality, professional codebase that is maintainable, extensible and performant. The state management ensures data consistency across all steps and component architecture will be modular to facilitate future development and debugging

Assumptions

a. All required data (e.g. products, plans, add-ons) will be hard coded in the site's code. There will be no asynchronous fetching.
b. The application will assume the seller's identity is already known.
c. The site will be built as "desktop" first. Mobile design was never mentioned in the requirements.
d. Features mentioned in the document (e.g. CRM, opportunity field, account rep, payment terms) are purposefully omitted as per the instructions.

Technology stack

1. React
    a. Availability of libraries
    b. Familiarity with the framework
    c. Component based allows for reusable UI components (buttons, inputs, forms)
    d. Built in state management
2. Built in React hooks for state management
    a. Local state using `useState`
    b. `useContext` For global form state

3. CSS and styling using Tailwind
    a. Allows for quick development
4. UI components by Shadcn
    a. Contains all of the needed components out of the box
    b. Considers accessibility by default
5. A combination of React Hooks Form and Zod to provide validation
    a. Using React Hooks Form to simplify writing the code for the form.
    b. Zod for declarative runtime validation

Data models

1. Main Order Model

```
interface Order {
  customerName: string;
  customerAddress?: {
    addressLine1: string;
    addressLine2?: string;
    city: string;
    state: string;
    zipCode: string;
  };
  selectedPlanId: string | null;
  customPlanPrice: number | null;
  startDate: string | null;
  contractPeriodInMonths: number;
  endDate: string | null;
  selectedAddOns: {
    addOnId: string;
    quantity: number;
  }[];
}
```

2. Product Model

```
interface Product {
  id: string;
  name: string;
  plans: Plan[];
}
```

3. Plan Model

```
interface Plan {
  id: string;
  name: string;
  defaultPrice: number;
  priceInterval: 'mo' | 'yr';
}
```

4. AddOn Model

```
interface AddOn {
  id: string;
  name: string;
  price: number;
  priceLabel: string;
}
```

Detailed implementation

1. Global state:
    a. The entire application will be wrapped in a React Context. `OrderProvider`, which provides access to the main `Order` data model and an `updateOrder` hook
2. Component structure:
    a. A top level `Order` component will manage the currently active stage
    b. A `StageBuilder` component will conditionally render the stage components depending on the currently active stage.
3. Form Management:
    a. Each stage component will be have it's own implementation of React Hook Form's `FormProvider` to manage the local form state
    b. Each stage component will contain a Zod schema defining the validation rules for form.
    c. Each stage will only progress to the next stage if the current form passes validation
4. Stage 1: Customer Information
    a. The `customerName` field will be marked as required.
    b. A checkbox for `prePopulateCustomerInformation` will have its state tracked by React Hook Form.

    c. If the `prepopulateCustomerInformation` field is checked, the address form appears and the address fields are all required except for `addressLine2`.
1. Stage 2:
    a. A `productLine` dropdown will contain all available product lines and is required.
    b. Selecting `productLine` updates the available plans available to be chosen. The form will also include details about each plan.
    c. `selectedPlanId` will be set by seller by selecting a radio button representing the chosen plan
    d. Under each plan is an inline “Edit price” field. On click will reveal an input field, for the `customPlanPrice` field. Modifying the price only applies if this plan is selected and not the stored price of the plan. Changes to the price will not persist on refresh
    e. Selecting a different plan will reset the price to the new plan’s default price
2. Stage 3: Contract terms
    a. The `endDate` will be automatically calculated and updated whenever the `startDate` or `contractPeriod` changes.
    b. A dropdown with values of 6, 12, 24 and 36 months and “custom” is available as the `contractPeriod` field.
    c. If “custom” is selected and input field will be rendered
    d. The Zod schema for this will enforce the custom duration to be a minimum of 1 month
3. Stage 4: Review and fine tune
    a. Add ons appear as a checkbox with an input field to specify the quantity for the add on
    b. The Zod schema will enforce the quantity for an add on cannot be less than 0
    c. Setting the quantity to 0 updates the state but will not remove the add on from the order. The add on will only be removed if the seller unchecks the checkbox to its left.
    d. The summary panel will be on the right side of the page, it is read only and renders data directly from the global `OrderContext`
    e. The summary panel includes a “Totals” row. This will be calculated by adding the `customPlanPrice` and the total of all add-ons.
    f. A “Finalize Order” button at the end of the form, on click will trigger a success message, completing the workflow

UI/UX Design Decisions

For this implementation, the primary goal is in providing a functionally complete application. This is why the approach will be favoring speed of development instead of following mockups.

Achieving this will be done by using the default `Shadcn/ui` components with its default theme and components. This provides a modern and accessible design system out of the box.

Though the final appearance may differ from the mockups, the goals written in the second section of this document will be followed.

Implementation timeline and task breakdown

This is the proposed project time allocation:

1. Software Design Document (SDD) - (1 hour, 45 minutes)
    a. Thorough planning, architecture design, and writing of this document to ensure all requirements are met and decisions are justified.
2. Code Implementation - (2 hours, 15 minutes)
    a. Initial Project Setup: (15 mins)
        i. Initializing the React application and installing all required dependencies (React Hook Form, Zod, Tailwind CSS, etc.).
    b. Global Component & State Setup: (25 mins)
        i. Creating the `OrderProvider` using React Context.
        ii. Defining the main `Order` state and data models.
        iii. Building the `StageBuilder` and overall application layout.
    c. Stage 1 & 2 Implementation: (35 mins)
        i. Building the UI and logic for customer information, product selection, and the "edit price" feature.
    d. Stage 3 & 4 Implementation: (40 mins)
        i. Building the UI and logic for contract terms, date calculations, and the add-ons selection.
    e. Summary View Implementation: (10 mins)
        i. Creating the dynamic summary panel.
    f. Final Testing & Buffer: (10 mins)

Future improvements and limitations

- Feature enhancements
    - Caching form answers in local storage.
    - “Finalize Order” button submits form data and informs the user of the successful submission
    - Replace the mock data by loading all options from a live API. The “Finalize Order” button would submit the form data to a database, and the application would track order status."
- UI/UX Refinements
    - Clickable links between states in the breadcrumbs
    - History (back and forward) working as expected with the current stage stored in the site URL