PropertySummary Refactoring Specification (Final)

Objective

Refactor the PropertySummary component with a direct mapping to the existing step metadata, ensuring consistency with the rest of the application.

Leveraging Existing STEP_METADATA

We'll use the existing STEP_METADATA from flows.ts as the source of truth for section names and icons:

```
// registry/sectionComponentRegistry.ts
import { ComponentType } from 'react';
import { STEP_METADATA } from '../../../../../constants/flows';
// Import section components

// Map section IDs to section components, using the same IDs as in STEP_METADATA
export const SECTION_COMPONENT_REGISTRY: Record<string, ComponentType<any>>
= {
  // Residential rent
  'res_rent_basic_details': PropertyDetailsSection,
  'res_rent_location': LocationSection,
  'res_rent_rental': RentalDetailsSection,
  'res_rent_features': AmenitiesSection,

  // Residential sale
  'res_sale_basic_details': PropertyDetailsSection,
  'res_sale_location': LocationSection,
  'res_sale_sale_details': SaleDetailsSection,
  'res_sale_features': AmenitiesSection,

  // Residential flatmates
  'res_flat_basic_details': RoomDetailsSection,
  'res_flat_location': LocationSection,
  'res_flat_flatmate_details': FlatmateDetailsSection,
  'res_flat_features': AmenitiesSection,

  // Residential PG/Hostel
  'res_pg_basic_details': RoomDetailsSection,
  'res_pg_location': LocationSection,
  'res_pg_pg_details': PGDetailsSection,
  'res_pg_features': AmenitiesSection,

  // Commercial rent
  'com_rent_basic_details': CommercialBasicDetailsSection,
  'com_rent_location': LocationSection,
  'com_rent_rental': RentalDetailsSection,
  'com_rent_features': CommercialFeaturesSection,

  // Commercial sale
  'com_sale_basic_details': CommercialBasicDetailsSection,
  'com_sale_location': LocationSection,
  'com_sale_sale_details': CommercialSaleDetailsSection,
  'com_sale_features': CommercialFeaturesSection,

  // Commercial coworking
  'com_cow_basic_details': CoworkingBasicDetailsSection,
  'com_cow_location': LocationSection,
  'com_cow_coworking_details': CoworkingDetailsSection,
  'com_cow_features': AmenitiesSection,

  // Land sale
  'land_sale_basic_details': LandDetailsSection,
  'land_sale_location': LocationSection,
  'land_sale_land_features': LandFeaturesDetailsSection
};

// Function to get a section component with metadata
export function getSectionWithMetadata(sectionId: string) {
  const Component = SECTION_COMPONENT_REGISTRY[sectionId];
  const metadata = STEP_METADATA[sectionId];

  if (!Component) {
    console.warn(`No component found for section ID: ${sectionId}`);
    return {
      Component: FallbackSection,
      metadata: {
        name: sectionId.replace(/_/g, ' '),
        icon: null
      }
    };
  }

  return {
    Component,
    metadata
  };
}
```

Revised Folder Structure (Based on STEP_METADATA Component Names)

```
src/modules/owner/components/property/wizard/sections/PropertySummary/
├── index.tsx                    # Main export point
├── PropertySummaryPage.tsx         # Entry point component
├── components/
│   ├── SummaryHeader.tsx           # Property type + listing type header
│   ├── SummaryContent.tsx          # Flow-specific content container
│   └── fields/                  # Reusable field renderers
│       ├── FieldText.tsx
│       ├── FieldBoolean.tsx
│       ├── FieldCurrency.tsx
│       ├── FieldArea.tsx
│       └── FieldList.tsx
├── flows/
│   ├── base/
│   │   └── BaseSummaryFlow.tsx     # Universal base flow component
│   ├── ResidentialSaleSummary.tsx
│   ├── ResidentialRentSummary.tsx
│   ├── ResidentialPGHostelSummary.tsx
│   ├── ResidentialFlatmatesSummary.tsx
│   ├── CommercialSaleSummary.tsx
│   ├── CommercialRentSummary.tsx
│   ├── CommercialCoworkingSummary.tsx
│   ├── LandSaleSummary.tsx
│   └── FlowFactory.tsx             # Factory to create the right flow component
├── sections/                    # Using exact component names from STEP_METADATA
│   ├── PropertyDetailsSection.tsx  # For "Property Details" (residential)
│   ├── LocationSection.tsx         # For "Location" (all flows)
│   ├── RentalDetailsSection.tsx    # For "Rental Details"
│   ├── SaleDetailsSection.tsx      # For "Sale Details"
│   ├── RoomDetailsSection.tsx      # For "Room Details" (flatmates/PG)
│   ├── PGDetailsSection.tsx        # For "PG Details"
│   ├── FlatmateDetailsSection.tsx  # For "Flatmate Details"
│   ├── CommercialBasicDetailsSection.tsx  # For "Comm. Prop. Details"
│   ├── CommercialSaleDetailsSection.tsx   # For "Sale Details" (commercial)
│   ├── CommercialFeaturesSection.tsx      # For "Features" (commercial)
│   ├── CoworkingBasicDetailsSection.tsx   # For "Basic Details" (coworking)
│   ├── CoworkingDetailsSection.tsx        # For "Coworking Details"
│   ├── LandDetailsSection.tsx             # For "Land/Plot Details"
│   ├── LandFeaturesDetailsSection.tsx     # For "Land Features"
│   └── AmenitiesSection.tsx               # For "Features" (residential/coworking)
├── registry/
│   └── sectionComponentRegistry.ts  # Maps section IDs to components
├── hooks/
│   ├── useFlowDetection.ts         # Existing hook for detecting flow type
│   ├── usePropertyData.ts          # Existing hook for data extraction
│   ├── useSummaryData.ts           # New hook for summary data processing
│   └── usePropertyTitle.ts         # Existing hook for title management
├── services/
│   ├── dataExtractor.ts            # Existing service for data extraction
│   ├── dataFormatter.ts            # Existing service for data formatting
│   ├── flowDetector.ts             # Existing service for flow detection
│   └── titleGenerator.ts           # Existing service for title generation
└── types.ts                        # Extended type definitions
```

Implementation Example with STEP_METADATA

Base Summary Flow Component

```
// flows/base/BaseSummaryFlow.tsx
import React from 'react';
import { getSectionWithMetadata } from '../../registry/sectionComponentRegistry';
import { SummarySection } from '../../sections/SummarySection';
import { FormData } from '../../../../types';

export interface BaseSummaryFlowProps {
  formData: FormData;
}

export abstract class BaseSummaryFlow extends
React.Component<BaseSummaryFlowProps> {
  // Abstract method that must be implemented by all flow components
  abstract getSectionIds(): string[];

  render() {
    const { formData } = this.props;
    const sectionIds = this.getSectionIds();

    return (
      <div className="space-y-6">
        {sectionIds.map(sectionId => {
          // Get both the component and metadata in one call
          const { Component, metadata } = getSectionWithMetadata(sectionId);
          const sectionData = formData.steps?.[sectionId] || {};

          return (
            <SummarySection
              key={sectionId}
              title={metadata.name}
              icon={metadata.icon && <metadata.icon className="h-4 w-4" />}
            >
              <Component
                data={sectionData}
                flowType={formData.flow?.category}
                listingType={formData.flow?.listingType}
              />
            </SummarySection>
          );
        })}
      </div>
    );
  }
}
```

Residential Sale Flow Component (Example)

```
// flows/ResidentialSaleSummary.tsx
import { BaseSummaryFlow } from './base/BaseSummaryFlow';

export class ResidentialSaleSummary extends BaseSummaryFlow {
  // Use the exact section IDs from STEP_METADATA
  getSectionIds(): string[] {
    return [
      'res_sale_basic_details',  // "Property Details"
      'res_sale_location',       // "Location"
      'res_sale_sale_details',   // "Sale Details"
      'res_sale_features'        // "Features"
    ];
  }
}
```

Flow Factory (with FLOW_TYPES)

```
// flows/FlowFactory.tsx
import { FLOW_TYPES } from '../../../../constants/flows';
import { BaseSummaryFlow } from './base/BaseSummaryFlow';
import { ResidentialSaleSummary } from './ResidentialSaleSummary';
import { ResidentialRentSummary } from './ResidentialRentSummary';
// ... other flow component imports

export class FlowFactory {
  static getFlowComponent(flowType: string): typeof BaseSummaryFlow {
    switch (flowType) {
      case FLOW_TYPES.RESIDENTIAL_SALE:
        return ResidentialSaleSummary;
      case FLOW_TYPES.RESIDENTIAL_RENT:
        return ResidentialRentSummary;
      case FLOW_TYPES.RESIDENTIAL_PGHOSTEL:
        return ResidentialPGHostelSummary;
      case FLOW_TYPES.RESIDENTIAL_FLATMATES:
        return ResidentialFlatmatesSummary;
      case FLOW_TYPES.COMMERCIAL_SALE:
        return CommercialSaleSummary;
      case FLOW_TYPES.COMMERCIAL_RENT:
        return CommercialRentSummary;
      case FLOW_TYPES.COMMERCIAL_COWORKING:
        return CommercialCoworkingSummary;
      case FLOW_TYPES.LAND_SALE:
        return LandSaleSummary;
      default:
        console.warn(`Unknown flow type: ${flowType}, using default flow`);
        return ResidentialRentSummary; // Default fallback
    }
  }
}
```

Section Component Example

```
// sections/PropertyDetailsSection.tsx - For "Property Details" in residential flows
import React from 'react';
import { FieldText } from '../components/fields/FieldText';
import { FieldBoolean } from '../components/fields/FieldBoolean';
import { FieldArea } from '../components/fields/FieldArea';

interface PropertyDetailsSectionProps {
  data: any;
  flowType?: string;
  listingType?: string;
}

export const PropertyDetailsSection: React.FC<PropertyDetailsSectionProps> = ({
  data,
  flowType,
  listingType
}) => {
  if (!data) return null;

  return (
    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
      {data.title && (
        <FieldText label="Title" value={data.title} />
      )}
      {data.propertyType && (
        <FieldText label="Property Type" value={data.propertyType} />
      )}
      {data.bhkType && (
        <FieldText label="BHK Type" value={data.bhkType} />
      )}
      {data.bedrooms && (
        <FieldText label="Bedrooms" value={data.bedrooms} />
      )}
      {data.bathrooms && (
        <FieldText label="Bathrooms" value={data.bathrooms} />
      )}
      {data.builtUpArea && (
        <FieldArea
          label="Built-up Area"
          value={data.builtUpArea}
          unit={data.builtUpAreaUnit || 'sqft'}
        />
      )}
      {/* Add additional fields as needed */}
    </div>
  );
};
```

Benefits of This Approach

1. **Consistency with existing code**: Uses the same section IDs and names from STEP_METADATA
2. **Reuse of existing metadata**: Leverages icons and descriptions already defined
3. **Clear mapping**: Explicit mapping between section IDs and component implementations
4. **Type safety**: Strong typing throughout the codebase
5. **Flexibility**: Each flow defines exactly which sections to include and in what order
6. **Maintainability**: Changes to section rendering logic can be made in one place
7. **Extensibility**: Easy to add new flow types or section components

This updated architecture ensures that the PropertySummary component is consistent with the existing STEP_METADATA definitions, making the codebase more maintainable and easier to understand.