PropertySummary Implementation Document

Version: 2.0.0
Last Modified: 05-12-2024
Purpose: Dynamic Flow-Based Property Summary Implementation

Overview

This document outlines the implementation of a dynamic PropertySummary component that renders different sections based on the detected flow type, with an emphasis on performance and maintainability through a single configuration file approach.

Architecture Design

Core Principle

- **Single Source of Truth**: All configurations in one file (`PropertySummaryConfig.ts`)
- **Performance Optimized**: Minimal imports and fast object lookups
- **Flow-Agnostic**: Works with all 8 property flows
- **Maintainable**: Easy to add new flows or modify existing ones

---

Flow Types

The system supports 8 distinct property flows:

1. **residential_rent**
2. **residential_sale**
3. **residential_flatmates**
4. **residential_pghostel**
5. **commercial_rent**
6. **commercial_sale**
7. **commercial_coworking**
8. **land_sale**

Step Identifiers Mapping

Each flow has unique step identifiers that map to specific sections:

Residential Rent:
- res_rent_basic_details
- res_rent_location
- res_rent_rental
- res_rent_features

Residential Sale:
- res_sale_basic_details
- res_sale_location
- res_sale_sale_details
- res_sale_features

Commercial Rent:
- com_rent_basic_details
- com_rent_location
- com_rent_rental
- com_rent_features

Commercial Sale:
- com_sale_basic_details
- com_sale_location
- com_sale_sale_details
- com_sale_features

[... and so on for all 8 flows]

Implementation Structure

1. PropertySummaryConfig.ts

This single configuration file contains all mappings and configurations:

```
export const SUMMARY_CONFIG = {
  // Which sections to show for each flow
  flowSections: {
    residential_rent: ['basicDetails', 'location', 'rentalDetails', 'features'],
    residential_sale: ['basicDetails', 'location', 'saleDetails', 'features'],
    residential_flatmates: ['basicDetails', 'location', 'flatmateDetails', 'features'],
    residential_pghostel: ['basicDetails', 'location', 'pgDetails', 'features'],
    commercial_rent: ['basicDetails', 'location', 'rentalDetails', 'commercialFeatures'],
    commercial_sale: ['basicDetails', 'location', 'saleDetails', 'commercialFeatures'],
```

```
  commercial_coworking: ['basicDetails', 'location', 'coworkingDetails', 'features'],
  land_sale: ['basicDetails', 'location', 'saleDetails', 'landFeatures']
},

// Map sections to step IDs for each flow
stepMappings: {
  residential_rent: {
    basicDetails: 'res_rent_basic_details',
    location: 'res_rent_location',
    rentalDetails: 'res_rent_rental',
    features: 'res_rent_features'
  },
  // ... mappings for all flows
},

// Section titles for UI display
sectionTitles: {
  basicDetails: 'Basic Details',
  location: 'Location Details',
  rentalDetails: 'Rental Details',
  saleDetails: 'Sale Details',
  flatmateDetails: 'Flatmate Details',
  pgDetails: 'PG Details',
  coworkingDetails: 'Coworking Details',
  features: 'Features & Amenities',
  commercialFeatures: 'Commercial Features',
  landFeatures: 'Land Features'
},

// Field labels for UI display
fieldLabels: {
  // Basic Details Fields
  bhkType: 'BHK Type',
  builtUpArea: 'Built-up Area',
  floor: 'Floor',
  totalFloors: 'Total Floors',
  bathrooms: 'Bathrooms',
  balconies: 'Balconies',
  facing: 'Facing',
  propertyAge: 'Property Age',
  propertyCondition: 'Property Condition',

  // Location Fields
  address: 'Address',
  flatPlotNo: 'Flat/Plot No.',
  landmark: 'Landmark',
  locality: 'Locality',
  city: 'City',
```

```
  state: 'State',
  pinCode: 'PIN Code',

  // Rental Fields
  rentAmount: 'Monthly Rent',
  securityDeposit: 'Security Deposit',
  maintenanceCharges: 'Maintenance Charges',
  availableFrom: 'Available From',

  // Sale Fields
  expectedPrice: 'Expected Price',
  possessionDate: 'Possession Date',
  priceNegotiable: 'Price Negotiable',

  // Features Fields
  amenities: 'Amenities',
  parking: 'Parking',
  petFriendly: 'Pet Friendly',
  nonVegAllowed: 'Non-Veg Allowed',
  waterSupply: 'Water Supply',
  powerBackup: 'Power Backup',
  gatedSecurity: 'Gated Security',

  // Commercial Fields
  commercialType: 'Commercial Type',
  totalArea: 'Total Area',
  carpetArea: 'Carpet Area',
  builtupArea: 'Built-up Area',

  // Land Fields
  landType: 'Land Type',
  plotLength: 'Plot Length',
  plotWidth: 'Plot Width',
  plotArea: 'Plot Area',
  plotFacing: 'Plot Facing',
  cornerPlot: 'Corner Plot',
  boundaryType: 'Boundary Type'
},

// Value formatters
fieldFormatters: {
  rentAmount: 'currency',
  expectedPrice: 'currency',
  securityDeposit: 'currency',
  maintenanceCharges: 'currency',
  builtUpArea: 'area',
  carpetArea: 'area',
  totalArea: 'area',
```

```
    plotArea: 'area',
    plotLength: 'dimension',
    plotWidth: 'dimension',
    petFriendly: 'boolean',
    nonVegAllowed: 'boolean',
    priceNegotiable: 'boolean',
    cornerPlot: 'boolean',
    amenities: 'array',
    availableFrom: 'date',
    possessionDate: 'date'
  },

  // Field lists for each section type
  sectionFields: {
    basicDetails: {
      residential: ['bhkType', 'builtUpArea', 'floor', 'totalFloors', 'bathrooms', 'balconies', 'facing',
'propertyAge', 'propertyCondition'],
      commercial: ['commercialType', 'totalArea', 'carpetArea', 'builtupArea', 'floor',
'totalFloors'],
      land: ['landType', 'plotArea', 'plotLength', 'plotWidth', 'plotFacing', 'boundaryType']
    },
    location: ['address', 'flatPlotNo', 'landmark', 'locality', 'city', 'state', 'pinCode'],
    rentalDetails: ['rentAmount', 'securityDeposit', 'maintenanceCharges', 'availableFrom',
'furnishingStatus'],
    saleDetails: ['expectedPrice', 'priceNegotiable', 'possessionDate'],
    features: ['amenities', 'parking', 'petFriendly', 'nonVegAllowed', 'waterSupply',
'powerBackup', 'gatedSecurity'],
    flatmateDetails: ['preferredGender', 'preferredVegetarian', 'ageRange', 'smokingAllowed',
'drinkingAllowed'],
    pgDetails: ['pgType', 'pgFor', 'mealIncluded', 'laundryIncluded', 'wifiIncluded', 'acIncluded'],
    coworkingDetails: ['spaceType', 'capacity', 'operatingHours', 'deskTypes', 'meetingRooms'],
    landFeatures: ['cornerPlot', 'parkFacing', 'waterConnection', 'electricityConnection',
'roadWidth'],
    commercialFeatures: ['centralAC', 'powerBackup', 'fireExtinguishers', 'cctvSurveillance',
'conferenceRoom']
  }
};

// Helper function to get flow category
export const getFlowCategory = (flowType: string): 'residential' | 'commercial' | 'land' => {
  if (flowType.startsWith('residential_')) return 'residential';
  if (flowType.startsWith('commercial_')) return 'commercial';
  if (flowType.startsWith('land_')) return 'land';
  return 'residential';
};
```

2. useSummarySections.ts (Updated)

```
import { useMemo } from 'react';
import { SUMMARY_CONFIG, getFlowCategory } from '../PropertySummaryConfig';
import { getFieldValue } from '../services/dataExtractor';
import { formatValue } from '../services/dataFormatter';

export const useSummarySections = (formData, flowType, coordinates) => {
  return useMemo(() => {
    const sections = SUMMARY_CONFIG.flowSections[flowType] || [];
    const stepMappings = SUMMARY_CONFIG.stepMappings[flowType] || {};
    const flowCategory = getFlowCategory(flowType);

    const sectionData = {};

    sections.forEach(section => {
      const stepId = stepMappings[section];
      if (!stepId) return;

      // Get field list for this section
      const fieldList = SUMMARY_CONFIG.sectionFields[section];
      const fields = Array.isArray(fieldList) ? fieldList : fieldList[flowCategory] || [];

      // Build section items
      const items = fields.map(field => {
        const value = getFieldValue(formData, stepId, field);
        if (value === undefined || value === null || value === '') return null;

        const label = SUMMARY_CONFIG.fieldLabels[field] || field;
        const formatter = SUMMARY_CONFIG.fieldFormatters[field];
        const formattedValue = formatter ? formatValue(value, formatter) : value;

        return { label, value: formattedValue };
      }).filter(Boolean);

      // Add section title and items
      if (items.length > 0) {
        sectionData[section] = {
          title: SUMMARY_CONFIG.sectionTitles[section],
          items
        };
      }
    });

    // Add coordinates to location section if available
    if (sectionData.location && coordinates) {
      sectionData.location.items.push({
        label: 'Coordinates',
```

```
      value: coordinates
    });
  }

  return sectionData;
}, [formData, flowType, coordinates]);
};
```

3. PropertySummary.tsx (Updated)

```
import React from 'react';
import { FormSection } from '@/components/FormSection';
import { SUMMARY_CONFIG } from './PropertySummaryConfig';
import { useFlowDetection } from './hooks/useFlowDetection';
import { usePropertyTitle } from './hooks/usePropertyTitle';
import { usePropertyData } from './hooks/usePropertyData';
import { useSummarySections } from './hooks/useSummarySections';
import { PropertyTitleEditor } from './components/PropertyTitleEditor';
import { SummarySection } from './components/SummarySection';
import { DescriptionSection } from './components/DescriptionSection';

export const PropertySummary: React.FC<PropertySummaryProps> = (props) => {
  // ... existing logic for hooks ...

  // Get all section data
  const sectionData = useSummarySections(formData, flowType, coordinates);

  // Get sections to render for this flow
  const sectionsToRender = SUMMARY_CONFIG.flowSections[flowType] || [];

  return (
    <FormSection
      title="Review Property Details"
      description="Review all details before saving or publishing"
    >
      <div className="space-y-6">
        {/* Property title editor */}
        <PropertyTitleEditor {...titleProps} />

        {/* Flow Information */}
        <SummarySection
          title="Listing Information"
          icon={<Info className="h-4 w-4" />}
          items={[{ label: 'Flow Type', value: flowInfo }]}
        />

        {/* Dynamic sections based on flow */}
```

```
        <div className="grid gap-6 md:grid-cols-2">
          {sectionsToRender.map(section => {
            const data = sectionData[section];
            if (!data || !data.items || data.items.length === 0) return null;

            return (
              <SummarySection
                key={section}
                title={data.title}
                items={data.items}
              />
            );
          })}
        </div>

        {/* Description Section */}
        <DescriptionSection description={description} />
      </div>
    </FormSection>
  );
};
```

4. dataFormatter.ts

```
// Value formatting functions
export const formatters = {
  currency: (value: number) => `₹${value.toLocaleString('en-IN')}`,
  area: (value: number, unit = 'sq.ft.') => `${value} ${unit}`,
  dimension: (value: number, unit = 'ft') => `${value} ${unit}`,
  boolean: (value: boolean) => value ? 'Yes' : 'No',
  array: (value: string[]) => value.join(', '),
  date: (value: string) => new Date(value).toLocaleDateString('en-IN'),
  percentage: (value: number) => `${value}%`,
  default: (value: any) => String(value)
};

export const formatValue = (value: any, formatterType: string): string => {
  const formatter = formatters[formatterType] || formatters.default;
  return formatter(value);
};
```

Performance Optimizations

1. **Single Import**: Only `PropertySummaryConfig.ts` is imported

2. **Object Lookups**: All data access is O(1) complexity
3. **Memoization**: `useMemo` prevents unnecessary recalculations
4. **Conditional Rendering**: Empty sections are not rendered
5. **Lazy Loading**: Sections are built only when needed

Maintenance Guide

Adding a New Flow

1. Add the flow type to `flowSections` in `PropertySummaryConfig.ts`
2. Add step mappings to `stepMappings`
3. If new sections are needed, add to `sectionTitles` and `sectionFields`
4. No changes needed in component files!

Adding New Fields

1. Add field label to `fieldLabels`
2. Add formatter to `fieldFormatters` if needed
3. Add field to appropriate section in `sectionFields`
4. No component changes required!

Debugging

- Check `SUMMARY_CONFIG.flowSections` for flow-specific sections
- Verify step IDs in `stepMappings`
- Ensure field labels exist in `fieldLabels`
- Test with different flow types to ensure correct sections appear

---

Testing Checklist

- [ ] All 8 flows render correct sections
- [ ] Field labels display properly
- [ ] Values are formatted correctly
- [ ] Empty sections are hidden
- [ ] Performance is acceptable on slow devices
- [ ] Configuration changes reflect immediately
- [ ] No JavaScript errors in console
- [ ] Responsive design works on mobile

Migration Notes

When migrating from the previous implementation:

1. Update imports in `PropertySummary.tsx`
2. Replace `useSummarySections` with new version
3. Remove old individual section imports
4. Test all flows thoroughly
5. Verify backward compatibility

Conclusion

This implementation provides a maintainable, performant, and flexible solution for dynamic property summary rendering. All configurations are centralized in a single file, making maintenance and updates straightforward while ensuring optimal performance.