# SARD India: SharePoint Framework (SPFx) Project Starter
## Complete Development Guide for React-Based Marketing Website

---

## Document Overview

Project: SARD India Marketing Website
Framework: SharePoint Framework (SPFx) with React
Language: TypeScript
Target Environment: SharePoint Online / SharePoint 2019+
Version: 1.0.0
Last Updated: November 2025

## Table of Contents

## 1. Project Overview {#project-overview}

### What is SPFx?

SharePoint Framework (SPFx) is Microsoft's modern development model for building client-side web parts and extensions in SharePoint. For the SARD marketing website, we'll use SPFx with React to create reusable, component-based web parts that connect to SharePoint Lists for data storage and retrieval.

### Technology Stack

- **Frontend Framework:** React 17.x with TypeScript
- **UI Components:** Fluent UI React (Microsoft's design system)
- **Data Access:** PnPjs (abstraction layer for SharePoint REST API)
- **Build Tool:** Webpack (bundled with SPFx)
- **State Management:** React Hooks (useState, useEffect, useContext)
- **Styling:** CSS Modules + Fluent UI theming
- **Backend Data:** SharePoint Lists (CMS-style content management)

### Architecture Philosophy

Each section of the SARD website (Journey, Devices, Technical Excellence, etc.) will be:
- A **separate SPFx web part** (independent, reusable component)
- **Data-driven** from SharePoint Lists
- **Responsive** across desktop, tablet, mobile
- **Branded** with Sony's blue color scheme (#0070D2, #003087)
- **Performant** with lazy loading and caching

### Benefits of This Approach

✅ **Non-developer friendly:** SharePoint list admins can update content without code
✅ **Reusable components:** Use same components across multiple pages
✅ **Scalable:** Easy to add new sections or features
✅ **Secure:** Leverages SharePoint's native security and permissions
✅ **Maintainable:** Centralized data in Lists, decoupled UI logic
✅ **Version controlled:** All code in Git, easy rollback

## 2. SPFx Project Setup Instructions {#spfx-setup}

### Prerequisites

Before starting, ensure you have:

- **Node.js 14.x or higher** (LTS recommended)
  Download: https://nodejs.org/
- **npm 6.x or higher**
  Verify: `npm --version`
- **Visual Studio Code** (recommended)
  Download: https://code.visualstudio.com/
- **SharePoint Online tenant** or SharePoint 2019+
  Access to Site Contents and App Catalog
- **Global SPFx Yeoman generator**
  Install: `npm install -g @microsoft/generator-sharepoint`

### Step-by-Step Setup

#### Step 1: Create Project Directory

```bash
mkdir SARD-Marketing-SPFx
cd SARD-Marketing-SPFx
```

#### Step 2: Generate SPFx Project Using Yeoman

```bash
yo @microsoft/sharepoint
```

When prompted, provide these values:

```
? Do you want to allow the tenant admin to deploy the solution to all sites
  immediately without running in isolated mode?
→ Y (Yes)

? Which type of client-side component to create?
→ WebPart (We'll create web parts)

? Do you want to use the latest version of @microsoft/sp-core-library?
→ Y (Yes)

? Which template would you like to use?
```

→ Minimal (Start with minimal template, add our own React logic)

? What is your Web part name?
→ SARDJourneyWebPart (or SARDTechnicalExcellenceWebPart, etc.)

? What is your Web part description?
→ SARD Journey & Growth Story (or appropriate description)

? Which framework would you like to use?
→ React (We want React components)

? Would you like to add Microsoft Graph API support?
→ N (We'll use SharePoint REST API via PnPjs)

? Which bundling solution would you like to use?
→ Webpack (Default, already configured)

? Which build target would you like to use?
→ ES5 (Better browser compatibility)

? Do you want to allow the caller to pass context to service classes?
→ Y (Yes, for SharePoint context)

? Would you like to clone a new GitHub repository?
→ N (We'll set up Git separately)

#### Step 3: Install Dependencies

```bash
npm install
```

This installs all packages defined in package.json (~2-3 minutes).

#### Step 4: Install Additional Libraries for SARD

```bash
# PnPjs - for SharePoint API access
npm install @pnp/sp @pnp/logging --save

# Fluent UI React - Microsoft's UI component library
npm install @fluentui/react --save

# Optional: Additional utilities
npm install axios --save
```

#### Step 5: Verify Installation

```bash
npm run build
```

Should complete successfully with no errors. This creates optimized bundles.

#### Step 6: Start Development Server

```bash
```

```
npm start
```

Opens browser to `https://localhost:4321`. You can now develop and test locally.

## 3. Project Directory Structure {#directory-structure}

Organize your SPFx project for scalability and maintainability:

```
SARD-Marketing-SPFx/
│
├── src/
│   ├── webparts/
│   │   ├── journeyWebPart/
│   │   │   ├── JourneyWebPart.ts          (Web part entry point)
│   │   │   ├── JourneyWebPart.manifest.json
│   │   │   └── components/
│   │   │       ├── Journey.tsx            (Main component)
│   │   │       ├── Journey.module.scss
│   │   │       └── IJourneyProps.ts
│   │   │
│   │   ├── technicalExcellenceWebPart/
│   │   │   ├── TechnicalExcellenceWebPart.ts
│   │   │   ├── TechnicalExcellenceWebPart.manifest.json
│   │   │   └── components/
│   │   │       ├── TechnicalExcellence.tsx
│   │   │       └── TechnicalExcellence.module.scss
│   │   │
│   │   ├── domainExcellenceWebPart/
│   │   ├── deviceExcellenceWebPart/
│   │   ├── verticalDetailsWebPart/
│   │   ├── caseStudiesWebPart/
│   │   ├── capabilityOverviewWebPart/
│   │   ├── whyChooseSardWebPart/
│   │   └── contactOnboardingWebPart/
│   │
│   ├── components/
│   │   ├── Timeline/
│   │   │   ├── Timeline.tsx              (Reusable timeline component)
│   │   │   ├── Timeline.module.scss
│   │   │   └── ITimelineProps.ts
│   │   │
│   │   ├── MetricsCard/
│   │   │   ├── MetricsCard.tsx
│   │   │   ├── MetricsCard.module.scss
│   │   │   └── IMetricsCardProps.ts
│   │   │
│   │   ├── DomainCard/
│   │   ├── DeviceCard/
│   │   ├── CaseStudyCard/
│   │   └── StatsDisplay/
│   │
│   ├── services/
│   │   ├── SharePointService.ts          (Fetch data from lists)
│   │   ├── DataTransformService.ts       (Format/transform data)
│   │   ├── MockDataService.ts            (For dev/testing)
│   │   │
│   │   └── index.ts                      (Export all services)
│   │
│   ├── models/
│   │   ├── IMilestone.ts                 (Journey data interface)
│   │   ├── IDevice.ts                    (Device data interface)
│   │   ├── IDomain.ts                    (Domain data interface)
│   │   ├── ICaseStudy.ts                 (Case study interface)
│   │   ├── IMetrics.ts                   (Metrics interface)
│   │   └── index.ts
│   │
│   ├── hooks/
│   │   ├── useSharePointData.ts          (Custom hook for data fetching)
│   │   └── useTheme.ts                   (Theme hook)
│   │
│   ├── styles/
│   │   ├── theme.scss                    (Sony brand colors)
│   │   ├── responsive.scss               (Mobile breakpoints)
│   │   └── common.module.scss            (Shared styles)
│   │
│   └── index.ts
│
├── config/
│   ├── package-solution.json             (Solution configuration)
│   ├── write-manifests.json
│   └── deploy-azure-storage.json
│
├── sharepoint/
│   ├── solution-debug.json               (Debug configuration)
│   └── solution.json                     (Production solution)
│
├── public/
│   └── images/                           (Static assets)
│
├── package.json                          (Dependencies and scripts)
├── tsconfig.json                         (TypeScript configuration)
├── .gitignore                            (Git ignore file)
├── README.md                             (Project documentation)
└── .env                                  (Environment variables - NOT in git)
```

### Key Directories Explained

webparts/: Contains 8+ web parts (one per SARD section)
components/: Reusable React components used across web parts
services/: SharePoint data access and business logic
models/: TypeScript interfaces for type safety
hooks/: Custom React hooks for common patterns
styles/: Global styles and theme definitions

## 4. SharePoint List Schema (CMS Configuration) {#sharepoint-lists}

Create these SharePoint Lists to store SARD marketing content. Non-developers can then update content via SharePoint UI.

### List 1: SARDMilestones (Journey Timeline)

Purpose: Store milestone data for the Journey & Growth Story page

Columns:

Column Name	Type	Required	Notes
Title	Text	Yes	Milestone title, e.g., "SARD Established"
Year	Number	Yes	Year, e.g., 2015
Description	Multiline Text	Yes	Full description of milestone
KeyMetrics	Text	Yes	Summary metrics, e.g., "25 Engineers • 1 Project"
IconName	Text	No	Fluent UI icon name, e.g., "RocketShape"
MilestoneColor	Text	No	Hex color, e.g., "#0070D2"
OrderIndex	Number	Yes	Sort order (1, 2, 3, ...)

Example Rows:

```
Row 1:
- Title: SARD Established
- Year: 2015
- Description: Started with vision to deliver world-class engineering solutions.
Initial team: 25 engineers, 1 project
- KeyMetrics: 25 Engineers • 1 Project
- IconName: RocketShape
- OrderIndex: 1

Row 2:
- Title: PlayStation Excellence
- Year: 2017
- Description: Expanded focus on gaming division. First major PS4 project delivery.
Team growth to 50+ engineers
- KeyMetrics: 50+ Engineers • 3 Projects • PlayStation focus
- IconName: GamepadShape
- OrderIndex: 2
```

### List 2: SARDDevices (Device Portfolio)

Purpose: Store device information for Device Excellence page

Columns:

Column Name	Type	Required	Notes
Title	Text	Yes	Device name, e.g., "PlayStation Consoles"
DevicesSupported	Text	Yes	Comma-separated, e.g., "PS4, PS5, PS VR"
TeamSize	Text	Yes	Engineer count, e.g., "120+ engineers"
ActiveProjects	Number	Yes	Number of active projects
YearsOfExperience	Number	Yes	Years working on device type
KeyAchievements	Multiline Text	Yes	Bullet-point achievements
TechnicalHighlights	Multiline Text	Yes	Key technical capabilities
TeamBreakdown	Text	No	JSON format: {"SystemProgrammers": 40, "GraphicsEngineers": 30, ...}
IconName	Text	No	Fluent UI icon, e.g., "GamepadShape"

Example Row:

```
- Title: PlayStation Consoles
- DevicesSupported: PS4, PS5, PS VR
- TeamSize: 120+ engineers
- ActiveProjects: 20
- YearsOfExperience: 10
- KeyAchievements: Supported 120M+ PS4 units sold; 40%+ performance improvement;
Zero-day patch system with <24hr deployment
- TechnicalHighlights: Custom OS; GPU Optimization; Low-latency networking;
Hardware encryption
- TeamBreakdown: {"SystemProgrammers": 40, "GraphicsEngineers": 30,
"NetworkEngineers": 20, "SecurityEngineers": 15}
```

### List 3: SARDCaseStudies (Success Stories)

Purpose: Store case study information for Case Studies page

Columns:

Column Name	Type	Required	Notes
Title	Text	Yes	Case study title, e.g., "PS5 Launch Success"
ClientVertical	Choice	Yes	Options: SIE, SSS, Sony Home Entertainment, Sony Imaging, Other
Challenge	Multiline Text	Yes	Problem description
Solution	Multiline Text	Yes	How SARD solved it
Results	Multiline Text	Yes	Quantified outcomes
Duration	Text	Yes	Project duration, e.g., "18 months"
TeamSize	Text	Yes	Engineer count, e.g., "80 engineers"
ImageUrl	Hyperlink	No	Case study image

### List 4: SARDMetrics (KPIs & Statistics)

Purpose: Store quick metrics for stats display on home page and throughout site

Columns:

Column Name	Type	Required	Notes
MetricName	Text	Yes	Name, e.g., "Total Engineers"
MetricValue	Text	Yes	Value, e.g., "450+"
MetricLabel	Text	Yes	Description, e.g., "Skilled workforce"
Category	Choice	Yes	Options: Personnel, Projects, Domains, Verticals, Technologies, Success
LastUpdated	Date	Yes	Auto-updated

Example Rows:

```
- MetricName: Total Engineers, MetricValue: 450+, MetricLabel: Skilled workforce,
Category: Personnel
- MetricName: Active Projects, MetricValue: 15+, MetricLabel: Ongoing initiatives,
Category: Projects
- MetricName: Sony Verticals, MetricValue: 5, MetricLabel: Business units served,
Category: Verticals
- MetricName: Tech Domains, MetricValue: 8+, MetricLabel: Technology expertise
areas, Category: Domains
```

### List 5: SARDDomains (Domain Expertise)

Purpose: Store domain information for Domain Excellence page

Columns:

Column Name	Type	Required	Notes
Title	Text	Yes	Domain name, e.g., "Gaming & Interactive Entertainment"
YearsOfExperience	Number	Yes	Years, e.g., 10
TotalProjects	Number	Yes	Project count
TeamSize	Text	Yes	Engineer count, e.g., "120+"
KeyAchievements	Multiline Text	Yes	Bullet list
Technologies	Text	Yes	Comma-separated: Java, C++, Python, ...
IconName	Text	No	Fluent UI icon

## 5. React Component Sample Code {#react-components}

### Example 1: Timeline Component (Journey Section)

```typescript
// src/components/Timeline/Timeline.tsx
import React, { useState } from 'react';
import { IMilestone } from '../../models/IMilestone';
import styles from './Timeline.module.scss';

export interface ITimelineProps {
  milestones: IMilestone[];
  loading: boolean;
  error?: string | null;
}

export const Timeline: React.FC<ITimelineProps> = ({ milestones, loading, error }) => {
  const [expandedId, setExpandedId] = useState<number | null>(null);

  if (loading) {
    return <div className={styles.loading}>Loading timeline...</div>;
  }

  if (error) {
    return <div className={styles.error}>Error: {error}</div>;
  }

  return (
    <div className={styles.timeline}>
      <div className={styles.timelineContainer}>
        {milestones && milestones.length > 0 ? (
          milestones.map((milestone, index) => (
            <div
              key={milestone.id}
              className={styles.timelineItem}
            >
              {/* Timeline dot and connector */}
              <div
                className={styles.timelineMarker}
                style={{ backgroundColor: milestone.color || '#0070D2' }}
                onClick={() => setExpandedId(expandedId === milestone.id ? null : milestone.id)}
              >
                <span className={styles.year}>{milestone.year}</span>
              </div>

              {/* Timeline content card */}
              <div className={styles.timelineContent}>
                <h3 className={styles.title}>{milestone.title}</h3>
                <p className={styles.metrics}>{milestone.keyMetrics}</p>

                {/* Expandable description */}
                {expandedId === milestone.id && (
                  <div className={styles.expandedContent}>
                    <p>{milestone.description}</p>
                  </div>
                )}
              </div>
            </div>
          ))
        ) : (
          <p>No milestones available</p>
        )}
      </div>
    </div>
  );
};
```

CSS Module (Timeline.module.scss):

```scss
.timeline {
  padding: 2rem;
  background: linear-gradient(135deg, #f5f5f5 0%, #ffffff 100%);
}

.timelineContainer {
  position: relative;
  padding: 2rem 0;

  &::before {
    content: '';
    position: absolute;
    left: 30px;
    top: 0;
    bottom: 0;
    width: 3px;
    background: #0070D2;
  }
}

.timelineItem {
  margin-bottom: 3rem;
  position: relative;
  padding-left: 100px;

  &:last-child {
    margin-bottom: 0;
  }
}
```

```
.timelineMarker {
  position: absolute;
  left: 0;
  top: 0;
  width: 60px;
  height: 60px;
  border-radius: 50%;
  background-color: #0070D2;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: pointer;
  transition: all 0.3s ease;
  box-shadow: 0 2px 8px rgba(0, 112, 210, 0.3);

  &:hover {
    transform: scale(1.1);
    box-shadow: 0 4px 12px rgba(0, 112, 210, 0.5);
  }
}

.year {
  color: white;
  font-weight: bold;
  font-size: 16px;
}

.timelineContent {
  background: white;
  padding: 1.5rem;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  transition: all 0.3s ease;

  &:hover {
    box-shadow: 0 4px 16px rgba(0, 0, 0, 0.15);
  }
}

.title {
  margin: 0 0 0.5rem 0;
  color: #003087;
  font-size: 18px;
  font-weight: 600;
}

.metrics {
  margin: 0;
  color: #666;
  font-size: 14px;
  font-weight: 500;
}

.expandedContent {
  margin-top: 1rem;
  padding-top: 1rem;
  border-top: 1px solid #e0e0e0;
  color: #333;
  font-size: 14px;
  line-height: 1.6;

  p {
    margin: 0;
  }
}

.loading,
.error {
  padding: 2rem;
  text-align: center;
  font-size: 16px;
}

.error {
  color: #d32f2f;
}

// Responsive
@media (max-width: 768px) {
  .timelineContainer::before {
    left: 15px;
  }

  .timelineItem {
    padding-left: 60px;
  }

  .timelineMarker {
    width: 40px;
    height: 40px;
  }

  .year {
    font-size: 12px;
  }

  .timelineContent {
    padding: 1rem;
  }

  .title {
    font-size: 16px;
  }
}
```

### Example 2: DeviceCard Component

```typescript
// src/components/DeviceCard/DeviceCard.tsx
import React from 'react';
import { Icon } from '@fluentui/react/lib/Icon';
import { IDevice } from '../../models/IDevice';
import styles from './DeviceCard.module.scss';

export interface IDeviceCardProps {
  device: IDevice;
}

export const DeviceCard: React.FC<IDeviceCardProps> = ({ device }) => {
  const achievements = device.keyAchievements?.split(';') || [];
  const technologies = device.technologies?.split(',') || [];

  return (
    <div className={styles.deviceCard}>
      <div className={styles.cardHeader}>
        {device.iconName && (
          <Icon
            iconName={device.iconName}
            className={styles.icon}
          />
        )}
        <h2 className={styles.title}>{device.title}</h2>
      </div>

      <div className={styles.stats}>
        <div className={styles.stat}>
          <span className={styles.statValue}>{device.teamSize}</span>
          <span className={styles.statLabel}>Engineers</span>
        </div>
        <div className={styles.stat}>
          <span className={styles.statValue}>{device.activeProjects}+</span>
          <span className={styles.statLabel}>Projects</span>
        </div>
        <div className={styles.stat}>
          <span className={styles.statValue}>{device.yearsExperience}</span>
          <span className={styles.statLabel}>Years</span>
        </div>
      </div>

      <div className={styles.section}>
        <h4>Key Achievements:</h4>
        <ul className={styles.list}>
          {achievements.map((ach, idx) => (
            <li key={idx}>{ach.trim()}</li>
          ))}
        </ul>
      </div>

      <div className={styles.section}>
        <h4>Technologies:</h4>
        <div className={styles.techTags}>
          {technologies.map((tech, idx) => (
            <span key={idx} className={styles.tag}>{tech.trim()}</span>
          ))}
        </div>
      </div>
    </div>
  );
};
```

### Example 3: Data Fetching Service

```typescript
// src/services/SharePointService.ts
```

```
import { sp } from '@pnp/sp/presets/all';
import { IMilestone } from '../models/IMilestone';
import { IDevice } from '../models/IDevice';

export class SharePointService {
  /**
   * Fetch milestones from SARDMilestones list
   */
  public static async getMilestones(listName: string = 'SARDMilestones'):
Promise<IMilestone[]> {
    try {
      const items = await sp.web.lists.getByTitle(listName)
        .items
        .select('ID', 'Title', 'Year', 'Description', 'KeyMetrics',
'MilestoneColor', 'OrderIndex')
        .orderBy('OrderIndex', true)
        .get();

      return items.map(item => ({
        id: item.ID,
        title: item.Title,
        year: item.Year,
        description: item.Description,
        keyMetrics: item.KeyMetrics,
        color: item.MilestoneColor || '#0070D2',
      }));
    } catch (error) {
      console.error('Error fetching milestones:', error);
      throw new Error(`Failed to fetch milestones: ${error instanceof Error ?
error.message : 'Unknown error'}`);
    }
  }

  /**
   * Fetch devices from SARDDevices list
   */
  public static async getDevices(listName: string = 'SARDDevices'):
Promise<IDevice[]> {
    try {
      const items = await sp.web.lists.getByTitle(listName)
        .items
        .select('*')
        .orderBy('Title', true)
        .get();

      return items.map(item => ({
        id: item.ID,
        title: item.Title,
        devicesSupported: item.DevicesSupported,
        teamSize: item.TeamSize,
        activeProjects: item.ActiveProjects,
        yearsExperience: item.YearsOfExperience,
        keyAchievements: item.KeyAchievements,
        technologies: item.TechnicalHighlights,
        iconName: item.IconName,
      }));
    } catch (error) {
      console.error('Error fetching devices:', error);
      throw new Error(`Failed to fetch devices: ${error instanceof Error ?
error.message : 'Unknown error'}`);
    }
  }

  /**
   * Create or update item in list
   */
  public static async updateItem(
    listName: string,
    itemId: number,
    updates: Record<string, any>
  ): Promise<void> {
    try {
      await sp.web.lists.getByTitle(listName)
        .items.getById(itemId)
        .update(updates);
    } catch (error) {
      console.error('Error updating item:', error);
      throw error;
    }
  }
}
```

### Example 4: Custom Hook for Data Fetching

```typescript
// src/hooks/useSharePointData.ts
import { useState, useEffect } from 'react';

export function useSharePointData<T>(
  fetchFunction: () => Promise<T[]>,
  dependencies: any[] = []
) {
  const [data, setData] = useState<T[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const loadData = async () => {
      try {
        setLoading(true);
        setError(null);
        const result = await fetchFunction();
        setData(result);
      } catch (err) {
        const message = err instanceof Error ? err.message : 'Error loading data';
        setError(message);
        console.error('Data fetch error:', message);
      } finally {
        setLoading(false);
      }
    };

    loadData();
  }, dependencies);

  return { data, loading, error, refetch: () => loadData() };
}
```

## 6. Web Part Entry Point {#web-part-entry}

Every web part needs an entry point class that extends BaseClientSideWebPart:

```typescript
// src/webparts/journeyWebPart/JourneyWebPart.ts
import * as React from 'react';
import * as ReactDom from 'react-dom';
import { Version } from '@microsoft/sp-core-library';
import { BaseClientSideWebPart } from '@microsoft/sp-webpart-base';
import {
  IPropertyPaneConfiguration,
  PropertyPaneTextField,
  PropertyPaneToggle,
} from '@microsoft/sp-property-pane';

import Journey from './components/Journey';
import { IJourneyProps } from './components/IJourneyProps';

export interface IJourneyWebPartProps {
  title: string;
  listName: string;
  showDescription: boolean;
}

export default class JourneyWebPart extends
BaseClientSideWebPart<IJourneyWebPartProps> {

  protected onInit(): Promise<void> {
    // Initialize SPFx and PnPjs
    return super.onInit();
  }

  public render(): void {
    const element: React.ReactElement<IJourneyProps> = React.createElement(
      Journey,
      {
        title: this.properties.title,
        spListName: this.properties.listName || 'SARDMilestones',
        context: this.context,
        showDescription: this.properties.showDescription,
      }
    );

    ReactDom.render(element, this.domElement);
  }

  protected onDispose(): void {
    ReactDom.unmountComponentAtNode(this.domElement);
  }

  protected get dataVersion(): Version {
    return Version.parse('1.0.0');
  }
  protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
    return {
      pages: [
        {
          header: {
            description: 'Journey Web Part - Configure settings',
          },
          groups: [
            {
              groupName: 'Settings',
              groupFields: [
                PropertyPaneTextField('title', {
                  label: 'Web Part Title',
                  value: this.properties.title || 'SARD Journey',
                  onGetErrorMessage: (value: string) => {
                    return value && value.length > 0 ? '' : 'Title is required';
                  },
                }),
                PropertyPaneTextField('listName', {
                  label: 'SharePoint List Name',
                  value: this.properties.listName || 'SARDMilestones',
                  description: 'Name of the SharePoint list to fetch data from',
                }),
                PropertyPaneToggle('showDescription', {
                  label: 'Show Descriptions',
                  checked: this.properties.showDescription !== false,
                }),
              ],
            },
          ],
        },
      ],
    };
  }
}
```

## 7. Architecture Diagram {#architecture}

```
┌───────────────────────────────────────────────────────────────────────┐
│                        SHAREPOINT ONLINE SITE                         │
│  ┌─────────────────────────────────────────────────────────────────┐  │
│  │          Modern Site Pages (SARD Marketing Website)            │  │
│  │  ├── Home Page                                                 │  │
│  │  ├── Journey Page    (SARDJourneyWebPart embedded)             │  │
│  │  ├── Tech Excellence (SARDTechExcellenceWebPart embedded)      │  │
│  │  ├── Domains        (SARDDomainExcellenceWebPart embedded)     │  │
│  │  ├── Devices        (SARDDeviceExcellenceWebPart embedded)     │  │
│  │  ├── Verticals      (SARDVerticalDetailsWebPart embedded)      │  │
│  │  ├── Case Studies   (SARDCaseStudiesWebPart embedded)          │  │
│  │  └── Contact        (SARDContactWebPart embedded)              │  │
│  └─────────────────────────────────────────────────────────────────┘  │
└───────────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌───────────────────────────────────────────────────────────────────────┐
```

~~~
│            SPFx Web Parts (React Components - Client-Side)          │
│  ┌──────────────────────────────────────────────────────────────┐   │
│  │  JourneyWebPart      ┌─── Timeline (React component)         │   │
│  │  DeviceWebPart       ├─── DeviceCard (React component)       │   │
│  │  DomainWebPart       ├─── DomainCard (React component)       │   │
│  │  CaseStudyWebPart    └─── CaseStudyCard (React component)    │   │
│  │  ... (8+ web parts)                                          │   │
│  └──────────────────────────────────────────────────────────────┘   │
└─────────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────────────┐
│          Data Services Layer (TypeScript Services)                  │
│  ┌──────────────────────────────────────────────────────────────┐   │
│  │  SharePointService.ts    (PnPjs REST API calls)              │   │
│  │  DataTransformService.ts (Data formatting & transformation)  │   │
│  │  MockDataService.ts      (Development data)                  │   │
│  └──────────────────────────────────────────────────────────────┘   │
└─────────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────────────┐
│           SharePoint Lists (CMS Data Storage)                        │
│  ├── SARDMilestones      (Journey timeline data)                    │
│  ├── SARDDevices         (Device portfolio data)                    │
│  ├── SARDDomains         (Domain expertise data)                    │
│  ├── SARDCaseStudies     (Success story data)                       │
│  ├── SARDMetrics         (KPIs and statistics)                      │
│  ├── SARDVerticals       (Business unit data)                       │
│  └── ... (other data lists)                                         │
└─────────────────────────────────────────────────────────────────────┘
```

### Data Flow Sequence

1. **User navigates to SharePoint page** → Page renders SPFx web part
2. **Web part mounts** → `useEffect` hook triggers
3. **Component calls service** → `SharePointService.getMilestones()`
4. **Service uses PnPjs** → REST API call to SharePoint List
5. **SharePoint returns JSON** → Data fetched from SARDMilestones list
6. **Service transforms data** → DataTransformService formats it
7. **State updated** → React re-renders component with new data
8. **Timeline component renders** → User sees beautiful milestone timeline

---

## 8. Deployment & Configuration {#deployment}

### Local Development

```bash
# Start development server
npm start

# Opens https://localhost:4321
# Bundles in watch mode (auto-rebuild on changes)
```

### Build for Production
~~~

```bash
# Build optimized bundles
npm run build

# Bundle and create solution package
npm run bundle --ship

# Create .sppkg file for deployment
npm run package-solution --ship
```

This creates: `sharepoint/solution/sard-marketing-spfx.sppkg`

### Deploy to SharePoint

#### Step 1: Upload to App Catalog

1. Navigate to **SharePoint App Catalog** (usually https://your-
tenant.sharepoint.com/sites/appcatalog)
2. Click **Distribute apps for Office** or **Apps for SharePoint**
3. Click **Upload** and select the `.sppkg` file
4. Fill in app details
5. Click **Deploy**

#### Step 2: Make App Available on Your Site

1. Go to your SARD marketing site
2. Go to **Site Contents** → **Apps**
3. Click **From Your Organization**
4. Find your app (e.g., "SARD Journey Web Part")
5. Click it to install on the site

#### Step 3: Add Web Part to Pages

1. Edit a Modern Site Page
2. Click **+** button (Add web part)
3. Search for your web part (e.g., "SARD Journey")
4. Click to add it
5. Configure properties in the web part pane
6. Publish page

#### Step 4: Verify SharePoint Lists Exist

Before web parts can load data:

1. Ensure all required lists exist (SARDMilestones, SARDDevices, etc.)
2. Ensure columns are properly configured
3. Add sample data to lists
4. Test web part on page

### Troubleshooting Deployment

Issue: Web part doesn't appear in search
Solution: Ensure app was deployed to app catalog and installed on site

Issue: Web part shows error "List not found"
Solution: Create SharePoint list with exact name, verify permissions

Issue: Data not loading
Solution: Check browser console for PnPjs errors, verify API permissions

---

## 9. Common SPFx Patterns {#spfx-patterns}

### Pattern 1: Error Handling with Fallback UI

```typescript
export const JourneyComponent: React.FC<IJourneyProps> = ({
  spListName,
  context,
}) => {
  const { data, loading, error } = useSharePointData(
    () => SharePointService.getMilestones(spListName)
  );

  // Handle error state
  if (error) {
    return (
      <div style={{ padding: '20px', color: 'red' }}>
        <h3>Unable to load milestones</h3>
        <p>{error}</p>
        <button onClick={() => window.location.reload()}>
          Retry
        </button>
      </div>
    );
  }

  // Handle loading state
  if (loading) {
    return <div>Loading...</div>;
  }

  // Render data
  return <Timeline milestones={data} loading={false} />;
};
```

### Pattern 2: Data Caching with Local Storage

```typescript
const getCachedData = async <T,>(
  key: string,
  fetchFn: () => Promise<T[]>,
  ttl: number = 3600000 // 1 hour
): Promise<T[]> => {
  const cached = localStorage.getItem(key);
  if (cached) {
    const parsed = JSON.parse(cached);
    if (new Date().getTime() - parsed.timestamp < ttl) {
      return parsed.data;
    }
  }

  const fresh = await fetchFn();
  localStorage.setItem(key, JSON.stringify({
    data: fresh,
    timestamp: new Date().getTime(),
  }));
  return fresh;
};
```

### Pattern 3: Responsive Grid Layout

```scss
.gridContainer {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
  gap: 2rem;
  padding: 2rem;

  @media (max-width: 768px) {
    grid-template-columns: 1fr;
    gap: 1rem;
    padding: 1rem;
  }

  @media (max-width: 480px) {
    grid-template-columns: 1fr;
    gap: 0.5rem;
    padding: 0.5rem;
  }
}
```

### Pattern 4: Theming with CSS Variables

```scss
// theme.scss
:root {
  --color-primary: #0070D2;        // Sony blue
  --color-dark: #003087;            // Dark blue
  --color-accent: #FF6600;          // Orange
  --color-text: #1A1A1A;            // Dark text
  --color-bg-light: #F5F5F5;        // Light background

  --font-size-lg: 24px;
  --font-size-md: 16px;
  --font-size-sm: 14px;

  --spacing-lg: 2rem;
  --spacing-md: 1rem;
  --spacing-sm: 0.5rem;
}

// Usage in components
.button {
  background-color: var(--color-primary);
  padding: var(--spacing-md);
  font-size: var(--font-size-md);
}
```

## 10. Development Best Practices {#best-practices}

### 1. **Component Organization**

✅ **DO:**
- Keep components small and focused (single responsibility)
- One component per file
- Extract reusable logic into custom hooks
- Separate presentational from container components

❌ **DON'T:**
- Create mega-components (>300 lines)
- Mix business logic with UI rendering
- Prop drill deeply (use context when needed)

### 2. **Performance Optimization**

✅ **DO:**
- Use React.memo for expensive components
- Implement lazy loading for long lists
- Debounce search/filter operations
- Cache SharePoint data appropriately

❌ **DON'T:**
- Fetch data on every render
- Create new object/function references in render
- Render large lists without virtualization

### 3. **Error Handling**

✅ **DO:**
- Wrap all SharePoint API calls in try-catch
- Show user-friendly error messages
- Log errors for debugging
- Provide fallback UI states

❌ **DON'T:**
- Silently fail
- Show technical error messages to users
- Ignore errors
- Leave broken states

### 4. **Styling & Responsiveness**

✅ **DO:**
- Use CSS Modules for scoped styles
- Follow Sony brand colors (#0070D2, #003087)
- Test on mobile, tablet, desktop
- Use Fluent UI's responsive grid

❌ **DON'T:**
- Use inline styles (except for dynamic values)
- Mix global and local styles
- Ignore mobile users
- Hardcode colors

### 5. **Data Management**

✅ **DO:**
- Use TypeScript interfaces for all data types
- Validate data before rendering
- Implement data transformation layer
- Keep mock data for testing

❌ **DON'T:**
- Use `any` type
- Trust external data without validation
- Mix data and UI logic
- Hardcode data in components

### 6. **SharePoint Integration**

✅ **DO:**
- Use PnPjs for easier API calls
- Test with real SharePoint data
- Handle permission errors gracefully
- Version your web parts

❌ **DON'T:**
- Make raw REST calls
- Only test with mock data
- Ignore permissions
- Skip version management

### 7. **Testing**

✅ **DO:**
- Write unit tests for components (Jest + React Testing Library)
- Test data fetching scenarios
- Mock SharePoint API calls in tests
- Test responsive layouts

❌ **DON'T:**
- Skip testing
- Only test happy paths
- Test against real SharePoint in unit tests
- Ignore accessibility testing

### 8. **Code Quality**

✅ **DO:**
- Use TypeScript strict mode
- Follow consistent naming conventions
- Add JSDoc comments for functions
- Keep files focused and clean

❌ **DON'T:**
- Use `any` types
- Use inconsistent naming
- Leave commented code
- Create files >500 lines

### 9. **Documentation**

✅ **DO:**
- Document component props and interfaces
- Add comments for complex logic

- Keep README updated
- Document SharePoint list schemas

❌ **DON'T:**
- Leave code undocumented
- Assume others understand your code
- Outdated documentation
- Missing setup instructions

### 10. **Git & Version Control**

✅ **DO:**
- Commit frequently with meaningful messages
- Use feature branches
- Keep .gitignore updated
- Tag releases

❌ **DON'T:**
- Commit large changes
- Use vague commit messages
- Commit node_modules or .env files
- Merge without testing

## Quick Reference: Common Commands

```bash
# Development
npm start                     # Start local dev server
npm run build                # Build optimized bundles

# Production
npm run bundle --ship        # Bundle for production
npm run package-solution --ship    # Create .sppkg file

# Testing (add when ready)
npm test                     # Run unit tests
npm test -- --coverage       # With coverage report

# Cleanup
npm run clean                # Remove build artifacts
rm -rf node_modules          # Remove dependencies (reinstall with npm install)
```

## Useful Resources

- **Microsoft SPFx Documentation:**
https://learn.microsoft.com/sharepoint/dev/spfx/
- **PnPjs Documentation:** https://pnp.github.io/pnpjs/
- **Fluent UI React:** https://react.fluentui.dev/
- **React Hooks Guide:** https://react.dev/reference/react/hooks
- **TypeScript Handbook:** https://www.typescriptlang.org/docs/

## Support & Questions

For issues or questions:
- **Email:** contact@sard.sony.com
- **SharePoint Admin:** Your IT department
- **GitHub Issues:** (if using GitHub repo)

Document Version: 1.0.0
Last Updated: November 2025
Status: Ready for Development
