④ ドキュメント作成：Webアプリ 成果物レポート（Sample）

1. 基本情報

- 受講者氏名：田中　太郎
- 作成日：2025年10月1日
- 対象工程：④ ドキュメント作成（各種ドキュメント概要まとめ）
- 成果物ファイル名：プロジェクト計画書.md, 要件定義書.md, リファクタリング報告書.md, スタイルガイド.md, リリースノート.md, 使用手順書.md
- 成果物バージョン：1.0

2. 概要

本報告書は、経費管理システムWebアプリケーション開発プロジェクトにおいて作成された各種ドキュメントの概要をまとめたものである。本プロジェクトでは、HTML5, CSS3, JavaScript (ES6+) のみで構築された、サーバーレスで動作するシングルページアプリケーション（SPA）として高機能な経費管理システムを開発した。

3. プロジェクト基本情報

項目	内容
プロジェクト名	経費管理システムWebアプリケーション開発
実施期間	2025年10月1日 ～ 2025年11月30日
技術スタック	HTML5, CSS3, JavaScript ES6+, Chart.js 4.x
アーキテクチャ	SPA (Single Page Application)
データ管理	LocalStorage（クライアントサイド完結）

---

4. ドキュメント作成実績

4.1. プロジェクト計画書

主要内容

- **目的・背景**: スプレッドシートベースの経費管理から、モダンなWebアプリケーションへの移行による業務効率化とデータ管理の改善
- **技術的特徴**: フルスタックWebアプリケーション（HTML5、CSS3、JavaScript ES6+）、セキュリティファースト、パフォーマンス最適化
- **アーキテクチャ**: Frontend/Security/Data層の3層構成、ES6モジュール版（modular）によるプレゼンテーション層

技術選定理由

カテゴリ	技術	バージョン	選定理由
フロントエンド	HTML5/CSS3/JavaScript	ES6+	モダンブラウザ対応、保守性重視
外部ライブラリ	Chart.js/ChartDataLabels	4.x/2.x	高品質なデータ可視化
データ管理	LocalStorage	-	クライアントサイド完結、高速アクセス
開発ツール	VS Code/Live Server	最新版	効率的な開発環境

4.2. 要件定義書

主要内容

- **ビジネス要件**: 経費登録時間50%削減、データ入力エラー30%削減のKPI設定
- **機能要件**: 経費管理機能、データ管理機能、分析機能、セキュリティ機能の4つの主要機能群
- **非機能要件**: レスポンシブデザイン、アクセシビリティ準拠（WCAG 2.1 AA）、1秒以内の高速レスポンス

システム要件

- **ブラウザ要件**: Chrome 90+, Firefox 88+, Edge 90+, Safari 14+（ES6モジュール対応必須）
- **画面解像度**: 1024×768以上（レスポンシブ対応）
- **ストレージ**: LocalStorage 10MB利用可能（警告5MB、制限8MB）

4.3. リファクタリング報告書

主要内容

- **改修目的**: 初期実装からES6モジュール構造化アプリケーションへのリファクタリング
- **段階的アプローチ**: セキュリティ強化 → エラーハンドリング強化 → パフォーマンス最適化 → コーディング規約準拠

定量的改善結果

測定項目	リファクタ前	リファクタ後	改善率
初期読み込み時間	3.2秒	0.8秒	75%改善
経費登録処理時間	1.5秒	0.3秒	80%改善
1,000件データ表示	5.0秒	0.5秒	90%改善
メモリ使用量	256MB	128MB	50%削減

脆弱性対応

- XSS攻撃、クリックジャッキング等の脆弱性0件達成
- Content Security Policy（CSP）実装による完全防御
- 入力サニタイズ・検証による安全性確保

4.4. スタイルガイド

主要内容

- **デザインコンセプト**: "Cyber Expense Management" - サイバーパンク風未来的UI
- **統一アーキテクチャ**: ES6モジュール構造、BEM準拠CSS命名規則
- **カラーパレット**: ダークベース（#0a0a0a）、ネオンアクセント（#00f0ff）による統一感

UI/UX仕様

```css
/* プライマリカラー */
--bg-primary: #0a0a0a;        /* メイン背景 */
--accent-cyan: #00f0ff;       /* プライマリアクセント */
--text-primary: #ffffff;      /* メインテキスト */

/* フォントファミリー */
font-family: 'Rajdhani', 'Orbitron'  /* Google Fonts */
```

コーディング規約

- **HTML**: セマンティックタグ（header, nav, h1等）の使用
- **CSS**: CSS変数（--bg-primary等）による一元管理
- **JavaScript**: ES6モジュール構文（import/export）、クラスベース実装

4.5. リリースノート

主要内容

- **バージョン**: v1.0.0（2025年10月1日リリース）
- **システム要件**: モダンブラウザ（ES6モジュール対応）、HTTPサーバー経由アクセス必須
- **主要機能**: 経費CRUD機能、データ永続化、CSVインポート/エクスポート、Chart.js統合レポート

ファイル構成

```
html/
├── index.html（メインページ）
├── src/
│   ├── ExpenseApp.js（メインアプリケーション）
│   ├── components/ChartManager.js（UI コンポーネント）
│   ├── services/（ビジネスロジック層）
│   ├── utils/（ユーティリティ層）
│   └── styles/main.css（スタイルシート）
└── samples/経費データ.csv（サンプルデータ）
```

外部依存関係

ライブラリ	バージョン	用途	配信方法
Chart.js	4.x	データ可視化	CDN + フォールバック
ChartDataLabels	2.x	グラフラベル表示	CDN + ローカル
Google Fonts	Latest	Rajdhani・Orbitron	CDN + ローカル

4.6. 使用手順書

主要内容

- **システム概要**: サーバーレスで動作する経費管理Webアプリケーション（file://では動作不可）
- **環境セットアップ**: VS Code + Live Server による開発環境構築
- **基本操作**: 経費登録、一覧表示、ダッシュボード確認の操作手順

操作フロー

1. **経費登録**: タイトル、種別、金額、日付、備考の入力
2. **経費一覧**: 検索・フィルタリング、編集・削除機能
3. **レポート**: Chart.jsによる月次統計、推移グラフ、種別別集計の確認

主要内容

- **バージョン**: v1.0.0（2025年10月1日リリース）
- **システム要件**: Windows 10・11、最新版ブラウザ、HTTPサーバー経由アクセス必須
- **主要機能**: 経費CRUD機能、データ永続化、CSVインポート/エクスポート、Chart.js統合レポート

モジュール構成

```
html/
├── index.html（メインページ）
├── main.css（統合スタイルシート）
├── js/
│   ├── app.js（1,343行 - メインアプリケーション）
│   ├── services/（ビジネスロジック層）
│   ├── utils/（ユーティリティ層）
│   └── components/（UIコンポーネント層）
```

外部依存関係

ライブラリ	バージョン	用途	配信方法
Chart.js	4.x	データ可視化	CDN + フォールバック
ChartDataLabels	2.x	グラフラベル表示	CDN + フォールバック
Google Fonts	Latest	Rajdhani・Orbitron	CDN + ローカル

4.6. 使用手順書

主要内容

- **システム概要**: サーバーレスで動作する経費管理Webアプリケーション
- **環境セットアップ**: VS Code + Live Server による開発環境構築
- **基本操作**: 経費登録、一覧表示、ダッシュボード確認の操作手順

操作フロー

1. **経費登録**: タイトル、種別、金額、日付、備考の入力
2. **経費一覧**: 検索・フィルタリング、編集・削除機能
3. **ダッシュボード**: 月次統計、推移グラフ、種別別集計の確認

---

5. 技術的成果

5.1. アーキテクチャ成果

Frontend Architecture

```
├── HTML5（セマンティック構造）
├── CSS3（CSS Variables・レスポンシブ）
├── JavaScript ES6+（モジュラー設計）
├── Chart.js 4.x（データ可視化）
└── LocalStorage（データ永続化）
```

Security Layer

- CSP（Content Security Policy）
- XSS・SQLインジェクション対策
- セッション管理・タイムアウト（30分）
- 入力検証・サニタイズ

Performance Layer

- 仮想スクロール（VirtualScrollManager）
- デバウンス処理（300ms）
- インメモリキャッシング
- 遅延読み込み

5.2. 機能実装成果

経費管理機能

- **単一経費登録**: フォーム入力によるリアルタイム検証
- **CSV一括登録**: 最大5,000件の一括インポート
- **経費一覧表示**: 仮想スクロール対応、1,000件以上の高速表示
- **検索・フィルタリング**: デバウンス処理による高速検索

データ管理機能

- **CSVエクスポート**: UTF-8 BOM付きでの出力
- **自動バックアップ**: 1時間毎の自動データバックアップ
- **データ復旧**: 最新5件のバックアップからの復元

分析・レポート機能

- **リアルタイムダッシュボード**: 経費統計の即座更新
- **Chart.js統合**: 円グラフ・折れ線グラフ・棒グラフ可視化
- **月次・年次レポート**: 期間別経費分析

---

6. 品質保証活動

6.1. 静的解析結果

品質指標	リファクタ前	リファクタ後	改善率
ESLint警告	147件	0件	100%改善
ESLintエラー	23件	0件	100%改善
循環的複雑度	平均18.5	平均8.2	56%改善
重複コード率	35%	8%	77%改善

6.2. セキュリティ強化

セキュリティ項目	対策内容	達成状況
XSS攻撃防止	入力サニタイズ・CSP実装	100%防御
クリックジャッキング	X-Frame-Options設定	完全対応
コンテンツタイプスニッフィング	X-Content-Type-Options設定	完全対応

6.3. アクセシビリティ対応

- WCAG 2.1 AA準拠
- キーボードナビゲーション対応
- スクリーンリーダー対応
- 高コントラストモード対応

---

7. システム運用・保守

7.1. 運用監視機能

リアルタイム監視

- **性能監視**: LocalStorageの使用容量監視（警告5MB、制限8MB）
- **エラー監視**: JavaScript例外の自動検知・ログ記録
- **セッション監視**: ユーザーセッション状態・タイムアウト（30分）管理
- **アクセス監視**: リクエスト頻度制限（60回/分）・異常アクセス検知

運用ダッシュボード

```javascript
const OperationMonitor = {
    dataUsage: () => {
        const used = new Blob([localStorage.getItem('expenseData')]).size;
        const limit = 10 * 1024 * 1024; // 10MB
        return { used, limit, percentage: (used / limit) * 100 };
    },
    sessionStatus: () => {
        return {
            active: SecurityManager.sessionValid,
            remaining: SecurityManager.getRemainingTime(),
            lastActivity: SecurityManager.lastActivity
        };
    }
};
```

7.2. 保守性向上

モジュラー設計による保守効率化

- **ES6モジュール構造**: 機能別ファイル分割による独立性確保
- **統一コーディング規約**: BEM命名規則・CSS変数による一元管理
- **型安全性**: JSDoc型ヒント・厳密なエラーハンドリング
- **自動化テスト**: 単体テスト・統合テストの継続実行

設定管理の外部化

```javascript
// config/app-config.js
export const AppConfig = {
    storage: {
        maxSize: 10 * 1024 * 1024,        // 10MB
        warningSize: 5 * 1024 * 1024,     // 5MB
        backupInterval: 3600000           // 1時間
    },
    security: {
        sessionTimeout: 30 * 60 * 1000,   // 30分
        maxInactivity: 15 * 60 * 1000,    // 15分
        maxRequestsPerMinute: 60
    },
    ui: {
        debounceDelay: 300,               // 300ms
        animationDuration: 200,           // 200ms
        itemsPerPage: 50
    }
};
```

拡張性確保

- **プラグイン対応**: 新機能追加時の最小変更
- **API化準備**: 将来のサーバーサイド連携対応
- **多言語対応**: i18n対応の基盤実装
- **PWA対応**: Service Worker・オフライン対応準備

---

8. 総括・成果まとめ

本プロジェクトでは、モダンWeb技術を活用したシングルページアプリケーションとして、高機能かつ高性能な経費管理システムを構築した。ES6モジュール構造による保守性の向上、セキュリティファーストの設計、パフォーマンス最適化により、企業レベルの要求に応える品質を実現した。

特に注目すべきは、サーバーレス構成でありながら、LocalStorageを活用したクライアントサイド完結型のアーキテクチャにより、高速な応答性能と優れたユーザビリティを両立させた点である。

また、統一されたコーディング規約とスタイルガイドの確立により、将来の機能拡張や保守作業において高い効率性を確保し、持続可能な開発基盤を構築した。Chart.jsを活用したデータ可視化機能により、経費データの分析・レポート機能も充実させ、単なる入力システムを超えた価値あるビジネスツールとして完成させた。