④ 必須：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 必須（Webアプリケーション実装）
- 成果物ファイル名：index.html, src/ExpenseApp.js, src/styles/main.css
- 成果物バージョン：1.0
- 対応ブラウザ／動作環境：
  - Google Chrome 90+（推奨：最新版）
  - Mozilla Firefox 88+（推奨：最新版）
  - Microsoft Edge 90+（推奨：最新版）
  - Safari 14+（推奨：最新版）
  - 要件：ES6モジュール対応、LocalStorage 10MB利用可能、HTTPサーバー経由実行

---

2. 成果物概要

- 成果物の目的：モダンWeb技術によるクライアントサイド完結型経費管理システムの構築
- 主な機能・画面構成：
  - **ホーム画面**：ダッシュボード・統計表示（経費総額、件数、月次推移）
  - **経費登録画面**：個別登録フォーム・CSV一括登録機能
  - **経費一覧画面**：仮想スクロール対応一覧表示・検索フィルタ・ソート機能
  - **レポート画面**：Chart.js活用の月別推移グラフ・種別別円グラフ
  - **設定・管理機能**：CSVエクスポート・全データ削除・バックアップ機能
- 対象データや入力条件：
  - 入力：経費データ（タイトル、種別、金額、日付、備考）
  - 保存先：ブラウザLocalStorage（最大10MB）
  - ファイル形式：CSV（UTF-8 BOM付き、カンマ区切り）
  - 画面解像度：1024×768以上（レスポンシブ対応）
- 出力内容：
  - 経費データのリアルタイム表示・統計分析
  - CSV形式でのデータエクスポート機能
  - Chart.js による可視化グラフ（PNG/SVG）
- 想定利用者：経費管理業務担当者・個人事業主・小規模事業者

---

3. 作成手順概要

1. 元となる仕様書・要件定義の確認内容：
   - 要件定義書.md（328行の包括的機能仕様）の詳細分析
   - プロジェクト計画書.md（325行のアーキテクチャ設計）の技術仕様確認
   - モジュラー設計・セキュリティ要件・パフォーマンス最適化指針の把握
2. 実装時に行った追加調査や分析：
   - Vanilla JavaScript ES6+モジュール設計パターンの技術検証
   - Chart.js 4.x + ChartDataLabels 2.x統合実装の詳細調査
   - CSP（Content Security Policy）設定とXSS対策の実装検証
   - 仮想スクロール・デバウンス・キャッシュ最適化の性能検証
3. 使用した生成AIモデル：人間による手動設計・実装（段階的リファクタリング）
4. 作成時に参照した資料や仕様書：
   - HTML5・CSS3・JavaScript ES6+公式仕様書
   - Chart.js Official Documentation & Examples
   - Web Content Accessibility Guidelines (WCAG) 2.1
   - LocalStorage API・Web Storage仕様書

---

4. 要件適合性

- 必須要件の充足状況：100%充足
  - **機能要件**：経費管理（登録・一覧・編集・削除）、レポート・分析、データ管理（CSV I/O）
  - **UI/UX要件**：レスポンシブデザイン、直感的操作性、1秒以内高速レスポンス
  - **技術要件**：Vanilla JavaScript ES6+、モジュラー設計、外部フレームワーク非依存
  - **セキュリティ要件**：CSP設定、XSS対策、入力検証、セッション管理
- 完成サンプルとの比較結果：
  - アーキテクチャ設計・モジュール構成が設計書と100%一致
  - パフォーマンス要件（応答時間・メモリ使用量）を全て達成
  - セキュリティ・アクセシビリティ要件の完全実装確認

---

5. 品質評価（自己評価）

- 正確性：98%（全機能の動作検証完了、Chart.js表示・CSV I/O・LocalStorage操作）
- 完全性：100%（要件定義書記載の全機能実装、エラーハンドリング完備）
- 一貫性：100%（統一UI/UXデザイン、命名規則・コーディング標準の適用）
- 可読性：95%（ES6モジュール分割、JSDoc記述、構造化CSS）
- 再現性：100%（HTTPサーバー実行手順明確化、Live Server推奨設定）
- 検証可能性：98%（ブラウザ開発者ツール・コンソール・Network監視対応）
- 保守性：95%（モジュラー設計、設定外部化、拡張性考慮設計）

---

6. 検証結果

- 検証方法：
  - **機能テスト**：全画面・全機能の手動操作テスト（正常系・異常系）
  - **互換性テスト**：4ブラウザ（Chrome/Firefox/Edge/Safari）での動作確認
  - **性能テスト**：1000件データでの仮想スクロール・検索・ソート性能測定
  - **セキュリティテスト**：XSS攻撃・CSP設定・入力検証の検証
- 検証環境：
  - Windows 11・macOS Monterey・Ubuntu 22.04
  - VS Code Live Server（localhost:5500）
  - Chrome DevTools Performance・Network・Security監視
- 検証結果概要：
  - **応答性能**：ページ読み込み0.8秒、経費登録0.3秒、一覧表示0.5秒（全て目標達成）
  - **データ処理**：1000件表示・検索・ソートが1秒以内で完了
  - **メモリ効率**：LocalStorage使用量5MB以下、ブラウザメモリ300MB以下
  - **互換性**：全対応ブラウザで機能・表示の完全動作確認
- 不具合や改善点：
  - Safari 14での一部CSS Grid表示調整（レガシー対応追加）
  - Chart.js初期化時の日本語フォント設定最適化
  - 大量データ（5000件超）での仮想スクロール初期化時間改善

---

7. 改善提案

- 今後の改善案：
  - **PWA対応**：Service Worker・オフライン機能・インストール対応
  - **多言語対応**：i18n実装・動的言語切り替え機能
  - **高度分析**：予算管理・支出予測・異常検知・カテゴリ分析
  - **外部連携**：銀行API・会計ソフト連携・OCR領収書読み取り
- 他工程への展開可能性：
  - **汎用業務アプリテンプレート**：他業務システムの標準アーキテクチャとして活用
  - **モジュール部品化**：Chart.js統合・仮想スクロール・CSV処理の再利用部品
  - **企業内標準化**：セキュリティ設計・パフォーマンス最適化パターンの社内展開
  - **教育・研修用途**：モダンWeb開発・ES6+・セキュリティ対策の学習教材

---

8. 添付・参考資料

- 添付ファイル名：
  - html/index.html（メインHTML・エントリーポイント）
  - html/src/ExpenseApp.js（メインアプリケーション制御）
  - html/src/components/ChartManager.js（Chart.js統合管理）
  - html/src/services/（ExpenseService.js, StorageService.js, ValidationService.js）
  - html/src/utils/（Constants.js, ErrorHandler.js, PerformanceUtils.js）
  - html/src/styles/main.css（統一スタイルシート）
  - html/samples/経費データ.csv（サンプルデータ）
- 参考URLや文献：
  - 要件定義書.md（機能・非機能要件詳細仕様）
  - プロジェクト計画書.md（アーキテクチャ・技術選定指針）
  - Chart.js Documentation（https://www.chartjs.org/docs/）
  - MDN Web Docs（ES6 Modules, LocalStorage, CSP）
  - WCAG 2.1 Guidelines（アクセシビリティ準拠基準）