Fix Plan & Implementation Approach

CA Portal — PartyWorkspace Conversion Engine

> **Scope:** What I will fix, exactly how, file by file, function by function.
> **Ground truth:** `index.html` (standalone GSTR-1 processor, 778 lines)
> **Target:** `pages/PartyWorkspace.tsx` + supporting service files

Approach Philosophy

Rule 1 — Port, don't rewrite.
Every working function in `index.html` gets ported 1-to-1 into TypeScript. No logic changes, no "improvements." If `index.html` does it, the portal does it exactly the same way.

Rule 2 — One source of truth for mappings.
All 22 column-mapping objects (11 sheets × 2 formats) live in a dedicated file: `services/gstrMappings.ts`. Nothing is inlined into the component. If a new sheet or format is needed later, only that file changes.

Rule 3 — One source of truth for transformations.
All data-transformation functions (`processDocsIssued`, `processHSNData`, `updateExemptSheet`, `appendDataToSheetWithMapping`) live in `services/gstrEngine.ts`. The component only calls them — it contains zero CSV/Excel logic itself.

Rule 4 — Surgical edits only.
`App.tsx`, `services/db.ts`, all pages except `PartyWorkspace.tsx`, all auth/registration flows — **untouched**. Only broken things get fixed.

Files to Create / Modify

Action	File	Why
CREATE	`services/gstrMappings.ts`	All 22 mapping objects, extracted from `index.html`
CREATE	`services/gstrEngine.ts`	All transformation + sheet-write functions
MODIFY	`pages/PartyWorkspace.tsx`	Replace broken conversion logic; call engine functions
MODIFY	`services/db.ts`	Add localStorage size guard
MODIFY	`types.ts`	Fix `ProcessingLog` field naming
NO CHANGE	Everything else	Already correct

Step 1 — Create `services/gstrMappings.ts`

What goes in here

A single exported object `MAPPINGS` with two top-level keys: `standard` and `tally`. Each contains all 11 sheet mappings. Tally mappings are identical to Standard for now (as verified in `index.html`) but live in a separate object so they can diverge later without touching engine code.

Exact mapping contents (ported from `index.html` lines ~242–460)

```
typescript
```

```
// services/gstrMappings.ts

export type SheetName =
  | 'b2b' | 'b2cl' | 'b2cs' | 'export'
  | 'Nil_exempt_NonGST' | 'cdnr' | 'cdnur'
  | 'adv_tax' | 'adv_tax_adjusted'
  | 'Docs_issued' | 'hsn';

export type FormatType = 'standard' | 'tally';

type MappingValue = string | string[];
type SheetMapping = Record<string, MappingValue>;
type FormatMappings = Partial<Record<SheetName, SheetMapping>>;

const standardMappings: FormatMappings = {
  b2b: {
    'GSTIN/UIN of Recipient': 'GSTIN/UIN',
    'Invoice Number':         'Invoice No',
    'Invoice date':           'Date of Invoice',
    'Invoice Value':          'Invoice Value',
    'Rate':                   'GST%',
    'Taxable Value':          'Taxable Value',
    'Cess Amount':            'CESS',
    'Place Of Supply':        'Place Of Supply',
    'Reverse Charge':         'RCM Applicable',
    'Invoice Type':           'Invoice Type',
    'E-Commerce GSTIN':       'E-Commerce GSTIN',
  },
  b2cl: {
    'Invoice Number':   'Invoice No',
    'Invoice date':     'Date of Invoice',
    'Invoice Value':    'Invoice Value',
    'Place Of Supply':  'Place Of Supply',
    'Rate':             'GST%',
    'Taxable Value':    'Taxable Value',
    'Cess Amount':      'CESS',
    'E-Commerce GSTIN': 'E-Commerce GSTIN',
  },
  b2cs: {
    'Type':             'Type',
    'Place Of Supply':  'Place Of Supply',
    'Rate':             'GST%',
    'Taxable Value':    'Taxable Value',
    'Cess Amount':      'CESS',
    'E-Commerce GSTIN': 'E-Commerce GSTIN',
  },
  export: {
    'Export Type':          'Export Type',
    'Invoice Number':       'Invoice No',
    'Invoice date':         'Date of Invoice',
    'Invoice Value':        'Invoice Value',
    'Port Code':            'Port Code',
    'Shipping Bill Number': 'Shipping Bill No',
    'Shipping Bill Date':   'Shipping Bill Date',
    'Rate':                 'GST%',
    'Taxable Value':        'Taxable Value',
  },
  Nil_exempt_NonGST: {
    'Description':                                   'Description',
    'Nil Rated Supplies':                            'Nil Rated Supplies',
    'Exempted (other than nil rated/non GST supply)':         'Exempted(other than nil rated/non GST supply)',
    'Non-GST supplies':                              'Non-GST Supplies',
  },
  cdnr: {
    'GSTIN/UIN of Recipient': 'GSTIN/UIN',
    'Note Number':            'Dr./ Cr. No.',
    'Note Date':              'Dr./Cr. Date',
    'Note Type':              'Type of note               (Dr/ Cr)',
    'Place Of Supply':        'Place of supply',
    'Reverse Charge':         'RCM',
    'Note Supply Type':       'Invoice Type',
    'Note Value':             'Dr./Cr. Value',
    'Rate':                   'GST%',
    'Taxable Value':          'Taxable Value',
    'Cess Amount':            'CESS',
  },
  cdnur: {
    'UR Type':                        'Supply Type',
    'Note/Refund Voucher Number':     'Dr./ Cr. Note No.',
    'Note/Refund Voucher date':       'Dr./ Cr. Note Date',
    'Document Type':                  'Type of note (Dr./ Cr.)',
    'Place Of Supply':                'Place of supply',
    'Note/Refund Voucher Value':      'Dr./Cr. Note Value',
    'Rate':                           'GST%',
    'Taxable Value':                  'Taxable Value',
    'Cess Amount':                    'CESS',
  },
  adv_tax: {
    'Place Of Supply':         'Place Of Supply',
    'Rate':                    'GST%',
    'Gross Advance Received':  'Gross Advance Received',
    'Cess Amount':             'CESS',
  },
  adv_tax_adjusted: {
    'Place Of Supply':        'Place Of Supply',
    'Rate':                   'GST%',
    'Gross Advance Adjusted':  'Gross Advance Adjusted',
    'Cess Amount':            'CESS',
  },
  Docs_issued: {
    'Nature of Document': ['Nature of Document', 'Type of Document'],
    'Sr.No.From':       ['Sr.No.From', 'Sr. No. From', 'Series From'],  // ← 3 aliases
    'Sr.No.To':         ['Sr.No.To',   'Sr. No. To',   'Series To'],
    'Total Number':     ['Total Number'],
    'Cancelled':        ['Cancelled'],
    'Net Issued':       ['Net Issued'],
  },
  hsn: {
    'Type':                   'Type',
    'HSN':                    'HSN',
    'Description':            'Description',
    'UQC':                    'UQC',
    'Total Quantity':         'Total Quantity',
    'Total Value':            'Total Value',
    'Rate':                   'Rate',
    'Taxable Value':          'Total Taxable Value',
    'Integrated Tax Amount':  'IGST',
    'Central Tax Amount':     'CGST',
    'State/UT Tax Amount':    'SGST',
    'Cess Amount':            'CESS',
  },
};

// Tally uses same column names as Standard for now.
// Kept as separate object so Tally-specific variants can be added independently.
const tallyMappings: FormatMappings = { ...standardMappings };

export const MAPPINGS: Record<FormatType, FormatMappings> = {
  standard: standardMappings,
  tally:    tallyMappings,
};
```

Why this matters

The old `PartyWorkspace.tsx` had only 3 of 11 mappings. Adding the other 8 here means CDNR, CDNUR, B2CL, B2CS, EXP, Nil/Exempt, AT, ATADJ files will now correctly fill their respective Excel sheets.

Step 2 — Create `services/gstrEngine.ts`

This file contains **all** data-transform and Excel-write logic, ported exactly from `index.html`.

2.1 `getSheetNameFromFile(fileName: string): SheetName | null`

Port from `index.html` lines ~462–474. Already mostly correct in portal — just move it here.

```typescript
// Exact port of index.html getSheetNameFromFile()
export function getSheetNameFromFile(fileName: string): SheetName | null {
  const upper = fileName.toUpperCase();
  if (upper.includes('HSN'))                          return 'hsn';
  if (upper.includes('B2B'))                          return 'b2b';
  if (upper.includes('B2CL'))                         return 'b2cl';
  if (upper.includes('B2CS'))                         return 'b2cs';
  if (upper.includes('EXP'))                          return 'export';
  if (upper.includes('EXEMP'))                        return 'Nil_exempt_NonGST';
  if (upper.includes('CDNR') && !upper.includes('CDNUR')) return 'cdnr';
  if (upper.includes('CDNUR'))                        return 'cdnur';
  if (upper.includes('ATADJ'))                        return 'adv_tax_adjusted';
  if (upper.includes('AT') && !upper.includes('ATADJ'))  return 'adv_tax';
  if (upper.includes('DOC'))                          return 'Docs_issued';
  return null;
}
```

2.2 `cleanPlaceOfSupply(value: any): string`

Port from `index.html`. Strips numeric state code prefix (e.g. `"27-Maharashtra"` → `"Maharashtra"`).

```typescript
export function cleanPlaceOfSupply(value: any): string {
  if (!value) return '';
  return value.toString().trim().replace(/^\d+-\s*/, '').trim();
}
```

2.3 `trimAllFields(obj: any): any`

Port from `index.html`. Trims whitespace from all string values in a row object.

```typescript
```

```
export function trimAllFields(obj: any): any {
  const cleaned: any = {};
  for (const [key, value] of Object.entries(obj)) {
    cleaned[key] = typeof value === 'string' ? value.trim() : value;
  }
  return cleaned;
}
```

2.4 `parseCSV(text: string): any[]` — NEW: quote-aware

The current portal has this function inline. Move it to the engine. The implementation is already correct (quote-aware parser). No logic change.

2.5 `processDocsIssued(data: any[]): any[]` — MISSING → ADD

Port from `index.html` lines ~286–300.

```typescript
export function processDocsIssued(data: any[]): any[] {
  return data.map(row => {
    const cleaned = trimAllFields(row);
    const totalNumber = parseFloat(cleaned['Total Number']) || 0;
    const cancelled   = parseFloat(cleaned['Cancelled'])    || 0;
    return {
      'Nature of Document': cleaned['Nature of Document'] || cleaned['Type of Document'] || '',
      'Sr.No.From':  cleaned['Sr.No.From']  || cleaned['Sr. No. From']  || cleaned['Series From']  || '',
      'Sr.No.To':    cleaned['Sr.No.To']    || cleaned['Sr. No. To']    || cleaned['Series To']    || '',
      'Total Number': totalNumber,
      'Cancelled':   cancelled,
      'Net Issued':  totalNumber - cancelled,  // ← computed field
    };
  });
}
```

Why critical: Without this, DOC files produce empty rows in the Excel output and `Net Issued` is never calculated.

2.6 `processHSNData(data: any[], fileName: string): any[]` — MISSING → ADD

Port from `index.html` lines ~301–310.

```typescript
```

```
export function processHSNData(data: any[], fileName: string): any[] {
  const upper = fileName.toUpperCase();
  return data.map(row => {
    const cleaned = trimAllFields(row);
    if (upper.includes('B2B'))      cleaned['Type'] = 'B2B';
    else if (upper.includes('B2C'))  cleaned['Type'] = 'B2C';
    if (!cleaned['Rate'] || cleaned['Rate'] === '') cleaned['Rate'] = 0;
    return cleaned;
  });
}
```

Why critical: GSTR-1 portal requires `Type` field on every HSN row. Missing it causes a portal validation error on submission.

2.7 `updateExemptSheet(sheet, data, columnMapping)` — MISSING → ADD

Port from `index.html` lines ~648–693. This is the most unique function — it does **in-place row matching**, not appending.

```typescript
```

```
export function updateExemptSheet(sheet: any, data: any[], columnMapping: any): void {
  if (data.length === 0) return;
  const range = XLSX.utils.decode_range(sheet['!ref'] || 'A1');

  // Read template headers from row 0
  const templateHeaders: string[] = [];
  for (let col = range.s.c; col <= range.e.c; col++) {
    const cell = sheet[XLSX.utils.encode_cell({ r: 0, c: col })];
    templateHeaders.push(cell?.v ? cell.v.toString().trim() : '');
  }

  // For each CSV row, find matching template row by Description
  data.forEach(csvRow => {
    const csvDesc = (csvRow['Description'] || '').toString().trim();
    for (let row = 1; row <= range.e.r; row++) {
      const descCell = sheet[XLSX.utils.encode_cell({ r: row, c: 0 })];
      const templateDesc = descCell?.v ? descCell.v.toString().trim() : '';
      if (templateDesc === csvDesc) {
        // Fill this matched row
        templateHeaders.forEach((templateHeader, colIndex) => {
          if (!templateHeader || colIndex === 0) return;  // skip Description column
          let csvValue: any = '';
          if (csvRow[templateHeader] !== undefined) {
            csvValue = csvRow[templateHeader];
          } else {
            for (const [csvCol, excelCol] of Object.entries(columnMapping)) {
              const excelCols = Array.isArray(excelCol) ? excelCol : [excelCol];
              if (excelCols.includes(templateHeader) && csvRow[csvCol] !== undefined) {
                csvValue = csvRow[csvCol];
                break;
              }
            }
          }
          const cellAddr = XLSX.utils.encode_cell({ r: row, c: colIndex });
          if (!csvValue && csvValue !== 0) {
            sheet[cellAddr] = { t: 'n', v: 0 };  // default to 0, not blank
          } else {
            const numValue = parseFloat(csvValue.toString().replace(/,/g, ''));
            sheet[cellAddr] = isNaN(numValue)
              ? { t: 's', v: csvValue.toString().trim() }
              : { t: 'n', v: numValue };
          }
        });
        break;  // stop after first match
      }
    }
  });
}
```

Why critical: The Nil/Exempt sheet has pre-existing fixed rows (Nil Rated, Exempted, Non-GST). Appending new rows after them breaks the template structure. Only in-place updates work here.

2.8 `appendDataToSheetWithMapping(sheet, data, columnMapping)` — FIX 3 MISSING TRANSFORMS

The function itself exists in the portal but is missing three field-level transformations. Add them inside the per-column loop:

```typescript
// TRANSFORM 1 — Place of Supply: strip state code prefix
if (templateHeader.toLowerCase().includes('place of supply')) {
  finalValue = cleanPlaceOfSupply(csvValue);
}

// TRANSFORM 2 — Invoice Type: strip trailing " B2B" or " B2C" suffix
if (templateHeader === 'Invoice Type') {
  finalValue = csvValue.toString().replace(' B2B', '').replace(' B2C', '').trim();
}

// TRANSFORM 3 — RCM: Y → Yes, N → No
if (templateHeader === 'RCM Applicable' || templateHeader === 'RCM') {
  finalValue = csvValue === 'Y' ? 'Yes' : csvValue === 'N' ? 'No' : csvValue;
}

// TRANSFORM 4 — Rate: empty → 0
if (templateHeader === 'Rate' || templateHeader === 'GST%') {
  if (!csvValue || csvValue === '') finalValue = 0;
}

// TRANSFORM 5 — Date: DD-Mon-YY → DD-MM-YYYY
if (templateHeader.toLowerCase().includes('date')) {
  const dateMatch = csvValue.toString().match(/(\d{1,2})-([A-Za-z]{3})-(\d{2})/);
  if (dateMatch) {
    const months: Record<string,number> = {
      Jan:0, Feb:1, Mar:2, Apr:3, May:4, Jun:5,
      Jul:6, Aug:7, Sep:8, Oct:9, Nov:10, Dec:11
    };
    const day   = parseInt(dateMatch[1]);
    const month = months[dateMatch[2]];
    const year  = 2000 + parseInt(dateMatch[3]);
    finalValue = `${String(day).padStart(2,'0')}-${String(month+1).padStart(2,'0')}-${year}`;
  }
}
```

Transforms 1 and 5 exist in the portal. Transforms 2, 3, 4 are **new additions**.

2.9 `runBatchConversion(...)` — EXPORT THE ORCHESTRATOR

Move the main conversion orchestrator out of the component into the engine. It accepts the files list, format, workbook, and a log callback. This makes it testable and keeps the component thin.

```
typescript
```

```
export async function runBatchConversion(
  files: UploadedFile[],
  format: FormatType,
  workbook: any,
  onLog: (msg: string) => void
): Promise<{ filesProcessed: number; filesSkipped: number }> {
  const mappings = MAPPINGS[format];
  let filesProcessed = 0;
  let filesSkipped   = 0;

  for (const file of files) {
    const sheetName = getSheetNameFromFile(file.fileName);

    if (!sheetName || !workbook.Sheets[sheetName]) {
      onLog(`⚠ Skipped (unrecognized): ${file.fileName}`);
      filesSkipped++;
      continue;
    }

    onLog(`→ Processing: ${file.fileName}  [${sheetName}]`);

    const csvContent = file.content || '';
    let csvData = parseCSV(csvContent);
    if (csvData.length === 0) {
      onLog(`⚠ Empty CSV, skipped: ${file.fileName}`);
      filesSkipped++;
      continue;
    }

    // Dispatch to correct pre-processor
    const upper = file.fileName.toUpperCase();
    if (upper.includes('DOC')) {
      csvData = processDocsIssued(csvData);
      onLog(`  ✓ Docs Issued: Net Issued computed for ${csvData.length} rows`);
    } else if (upper.includes('HSN')) {
      csvData = processHSNData(csvData, file.fileName);
      onLog(`  ✓ HSN: Type field set for ${csvData.length} rows`);
    } else {
      csvData = csvData.map(row => trimAllFields(row));
    }

    const sheet   = workbook.Sheets[sheetName];
    const mapping = mappings[sheetName] || {};

    // Dispatch to correct sheet-writer
    if (sheetName === 'Nil_exempt_NonGST') {
      updateExemptSheet(sheet, csvData, mapping);
      onLog(`  ✓ Nil/Exempt: ${csvData.length} descriptions matched in-place`);
    } else {
      appendDataToSheetWithMapping(sheet, csvData, mapping);
      onLog(`  ✓ Appended ${csvData.length} rows`);
    }

    filesProcessed++;
  }

  return { filesProcessed, filesSkipped };
}
```

Step 3 — Modify `pages/PartyWorkspace.tsx`

The component becomes **thin**. It handles state, UI, and calls the engine. Zero CSV/Excel logic inside the component.

3.1 Remove inline logic

Delete from the component:

- `cleanPlaceOfSupply()`
- `trimAllFields()`
- `parseCSV()`
- `getSheetNameFromFile()`
- `standardMappings` and `tallyMappings` objects
- `appendDataToSheetWithMapping()`
- `EMBEDDED_TEMPLATE_BASE64` (keep, this is data not logic)

3.2 Import engine functions

```typescript
import {
  runBatchConversion,
  getSheetNameFromFile,
} from '../services/gstrEngine';
import type { FormatType } from '../services/gstrMappings';
```

3.3 Replace `runBatchConversion` implementation

```typescript
```

```
const handleRunBatchConversion = async () => {
  if (filesForSelectedPeriod.length === 0) return;
  setIsProcessing(true);

  const log = (msg: string) => setAutomationLog(prev => [...prev, msg]);

  log(`◉ Session started — ${automationMonth} ${automationYear}`);
  log(`◉ Format: ${sourceFormat.toUpperCase()}`);
  log(`◉ Files in period: ${filesForSelectedPeriod.length}`);

  try {
    const base64Data   = EMBEDDED_TEMPLATE_BASE64.replace(/\s/g, '');
    const templateBinary = Uint8Array.from(atob(base64Data), c => c.charCodeAt(0));
    const workbook     = XLSX.read(templateBinary, { type: 'array', cellStyles: true });

    log('✓ Template loaded');

    const { filesProcessed, filesSkipped } = await runBatchConversion(
      filesForSelectedPeriod,
      sourceFormat,
      workbook,
      log
    );

    const outputData = XLSX.write(workbook, { bookType: 'xlsx', type: 'array', cellStyles: true });
    const blob = new Blob([outputData], {
      type: 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
    });
    saveAs(blob, `GSTR1_${party?.fullName}_${automationMonth}_${automationYear}.xlsx`);

    log(`✅ Done — ${filesProcessed} processed, ${filesSkipped} skipped`);
    log('✅ Output file downloaded.');

    // Mark files as Completed
    filesForSelectedPeriod.forEach(f => db.updateFileStatus(f.id, 'Completed'));

    db.saveLog({
      id:           Math.random().toString(36).substr(2, 9),
      timestamp:     new Date().toLocaleString(),
      action:       `GSTR-1 Batch (${sourceFormat})`,
      fileName:     `${automationMonth} ${automationYear}`,
      result:       'Success',
      status:       'Completed',
      caCode:       user!.caCode,
      customerId:   partyId!,
      filesProcessed,
      filesSkipped,
    });

    refreshData();
  } catch (err: any) {
    log(`❌ Error: ${err.message}`);
    console.error(err);
  } finally {
    setIsProcessing(false);
  }
};
```

3.4 UI changes in the Automation module

Add format active indicator in console header:

```tsx
<span className="ml-4 text-[10px] font-black text-slate-500 tracking-[3px] uppercase">
  Engine Output — {sourceFormat.toUpperCase()} MODE
</span>
```

Improve empty state message:

```tsx
// Instead of just showing "0 Files Ready"
{filesForSelectedPeriod.length === 0 ? (
  <div className="text-xs text-orange-500 font-semibold">
    No files for {automationMonth} {automationYear}.
    Check uploads or change period.
  </div>
) : (
  <div className="text-xl font-black text-green-600">
    {filesForSelectedPeriod.length} Files Ready
  </div>
)}
```

Replace fake Verify module:

```tsx
```

```
{activeModule === 'verify' && (
  <div className="max-w-xl mx-auto bg-amber-50 border border-amber-200 p-10 rounded-xl text-center">
    <p className="text-2xl font-black text-amber-700 mb-2">🔧 Coming Soon</p>
    <p className="text-amber-600 text-sm">
      GSTIN format validator, date-range checker, and HSN code verifier
      will be available in the next release.
    </p>
  </div>
)}
```

Step 4 — Modify `services/db.ts`

Add a size guard before writing file content to `localStorage`:

```
typescript
```

```
const MAX_CONTENT_BYTES = 2 * 1024 * 1024; // 2 MB safe limit

saveFile: (file: UploadedFile) => {
  const files = db.getFiles();

  // Guard: warn if content is too large for localStorage
  if (file.content && file.content.length > MAX_CONTENT_BYTES) {
    console.warn(
      `[db] File "${file.fileName}" content is ${(file.content.length / 1024 / 1024).toFixed(1)}MB — ` +
      `truncated to prevent localStorage overflow. Upload smaller files for full conversion.`
    );
    // Save the record but without content — CA will see Pending but conversion will produce empty output
    // Better: throw so CustomerDashboard can show an error toast
    throw new Error(
      `File too large for browser storage (${(file.content.length / 1024 / 1024).toFixed(1)}MB). ` +
      `Maximum supported size is 2MB per file.`
    );
  }

  files.push(file);
  try {
    localStorage.setItem(FILES_KEY, JSON.stringify(files));
  } catch (e: any) {
    if (e.name === 'QuotaExceededError') {
      throw new Error('Browser storage is full. Please ask your CA to clear old records.');
    }
    throw e;
  }
},
```

Also update `CustomerDashboard.tsx` to catch this error and show it to the user:

```typescript
} catch (err: any) {
  setUploadError(err.message || 'Upload failed. Please try again.');
  setIsUploading(false);
}
```

Step 5 — Modify `types.ts`

Update `ProcessingLog` to match what the engine now saves:

```typescript
```

```
export interface ProcessingLog {
  id:           string;
  timestamp:    string;
  action:       string;
  fileName:     string;
  result:       string;
  status:       string;
  caCode:       string;
  customerId:   string;
  filesProcessed?: number;  // renamed from errorCount (misleading name)
  filesSkipped?:  number;   // new field
  // Keep legacy fields for backward compat with old log records:
  errorCount?:    number;
  migratedAmount?: number;
}
```

Complete Change Summary

`services/gstrMappings.ts` (NEW — ~120 lines)

- All 11 standard sheet mappings
- All 11 tally sheet mappings (currently same, easily divergeable)
- Exported `SheetName`, `FormatType`, `MAPPINGS` types and constants

`services/gstrEngine.ts` (NEW — ~200 lines)

Function	Status	Source
`getSheetNameFromFile`	Moved from component	index.html ~462
`cleanPlaceOfSupply`	Moved from component	index.html ~245
`trimAllFields`	Moved from component	index.html ~250
`parseCSV`	Moved from component	index.html ~614
`processDocsIssued`	**NEW — was missing**	index.html ~286
`processHSNData`	**NEW — was missing**	index.html ~301
`updateExemptSheet`	**NEW — was missing**	index.html ~648
`appendDataToSheetWithMapping`	Moved + **3 transforms added**	index.html ~694

Function	Status	Source
`runBatchConversion`	Moved + **correct dispatch**	index.html ~538

`pages/PartyWorkspace.tsx` (MODIFY — net reduction in lines)

Change	Detail
Remove 8 inline functions	Moved to engine
Remove 2 partial mapping objects	Moved to mappings file
Replace conversion handler	Calls `runBatchConversion()` from engine
Add format label in console	UX improvement
Fix empty state message	Shows why no files match
Replace fake Verify module	"Coming Soon" honest state
Fix log saving	Uses `filesProcessed` / `filesSkipped`

`services/db.ts` (MODIFY — add ~20 lines)

Change	Detail
Add size guard in `saveFile`	Throws before localStorage overflow
Add `QuotaExceededError` catch	Propagates meaningful error to UI

`types.ts` (MODIFY — add 2 fields)

Change	Detail
Add `filesProcessed?: number`	Replaces misleading `errorCount` in new logs
Add `filesSkipped?: number`	New field
Keep old fields	Backward compat with existing log records

What Is NOT Changing

File	Reason
`App.tsx`	Auth/routing is correct
`pages/Login.tsx`	Correct
`pages/RegisterCA.tsx`	Correct
`pages/RegisterCustomer.tsx`	Correct
`pages/CADashboard.tsx`	Correct
`pages/CustomerDashboard.tsx`	Only add error display for upload failure
`services/compliance.ts`	Left as-is; it's a separate optional layer, not touched
`index.html` (portal entry)	Correct
`vite.config.ts`	Correct
`tsconfig.json`	Correct
`package.json`	Correct
Embedded template Base64	Same in both files, confirmed identical

---

Testing Checklist After Implementation

Test	Expected Result
Upload a B2B CSV, run conversion	B2B sheet filled correctly
Upload a B2CL CSV, run conversion	B2CL sheet filled (was broken before)
Upload a B2CS CSV, run conversion	B2CS sheet filled (was broken before)
Upload a DOC CSV, run conversion	Docs_issued sheet filled + Net Issued computed
Upload an HSN CSV (B2B filename), run	HSN sheet filled + Type = 'B2B' on all rows
Upload a CDNR CSV, run conversion	cdnr sheet filled (was broken before)
Upload an EXEMP CSV, run conversion	Nil/Exempt rows updated in-place, not appended

Test	Expected Result
Upload a file with `Y` in Reverse Charge	Output shows `Yes` not `Y`
Upload a file with `Invoice Type: Regular B2B`	Output shows `Regular` not `Regular B2B`
Upload a file with `27-Maharashtra` in POS	Output shows `Maharashtra`
Upload a file > 2MB	Error message shown to customer, not silent failure
Audit log after conversion	Shows `filesProcessed` and `filesSkipped` counts
Switch format to Tally, run	Console shows `TALLY MODE`
Select month with no uploads	Shows descriptive message, not just "0 Files"

This document covers exactly what will be implemented, the precise code for every new/changed function, and the rationale for each decision.