

软著鉴别材料整理器软件 V1.0

```
1 const fs = require('fs');
2 const path = require('path');
3 const {
4     Document, Packer, Paragraph, TextRun, ImageRun, Header, Footer
5     ,
6     AlignmentType, PageOrientation, HeadingLevel, TableOfContents,
7     LevelFormat, PageNumber
8 } = require('docx');
9 const outPath = path.join(__dirname, 'softreg-doc-v1.0-cn.docx')
10 ;
11 const uiImagePath = path.join(__dirname, 'screenshot-app-ui.png'
12 );
13 const exportImagePath = path.join(__dirname, 'screenshot-export-
14 doc.png');
15 const makeCenter = (text) => new Paragraph({
16     alignment: AlignmentType.CENTER,
17     spacing: { after: 120 },
18     children: [new TextRun({ text, size: 28 })]
19 });
20 const makeParagraph = (text) => new Paragraph({
21     spacing: { after: 120 },
22     children: [new TextRun({ text, size: 24 })]
23 });
24 const makeBullet = (text) => new Paragraph({
25     numbering: { reference: 'bullet-list', level: 0 },
26     children: [new TextRun({ text, size: 24 })]
27 });
28 const makeStep = (text) => new Paragraph({
29     numbering: { reference: 'step-list', level: 0 },
30     children: [new TextRun({ text, size: 24 })]
31 });
32 const scaleToWidth = (w, h, targetW) => {
33     const ratio = targetW / w;
34     return { width: Math.round(w * ratio), height: Math.round(h *
35 ratio) };
36 };
37 const addImageBlock = (title, imagePath, size, altText) => {
38     const data = fs.readFileSync(imagePath);
39     return [
40         new Paragraph({
41             spacing: { after: 120 },
42             children: [new TextRun({ text: title, bold: true, size: 24
43 })]
44     }),
45         new Paragraph({
46             alignment: AlignmentType.CENTER,
47             spacing: { after: 120 },
48             children: [new ImageRun({
49                 type: 'png',
50                 data,
```

```
51         transformation: { width: size.width, height: size.height
52 , rotation: 0 },
53         altText: { title: altText, description: altText, name: a
54 ltText }
55     })
56   })
57 ];
58 };
59 const uiSize = scaleToWidth(939, 1547, 420);
60 const exportSize = scaleToWidth(927, 1291, 420);
61 const doc = new Document({
62   styles: {
63     default: { document: { run: { font: 'SimSun', size: 24 } } }
64 ,
65     paragraphStyles: [
66       { id: 'Title', name: 'Title', basedOn: 'Normal',
67         run: { size: 56, bold: true, color: '000000', font: 'Mic
68 rosoft YaHei' },
69         paragraph: { spacing: { before: 240, after: 120 }, align
70 ment: AlignmentType.CENTER } },
71       { id: 'Heading1', name: 'Heading 1', basedOn: 'Normal', ne
72 xt: 'Normal', quickFormat: true,
73         run: { size: 32, bold: true, color: '000000', font: 'Mic
74 rosoft YaHei' },
75         paragraph: { spacing: { before: 240, after: 180 }, outli
76 neLevel: 0 } },
77       { id: 'Heading2', name: 'Heading 2', basedOn: 'Normal', ne
78 xt: 'Normal', quickFormat: true,
79         run: { size: 28, bold: true, color: '000000', font: 'Mic
80 rosoft YaHei' },
81         paragraph: { spacing: { before: 180, after: 120 }, outli
82 neLevel: 1 } }
83     ]
84   },
85   numbering: {
86     config: [
87       {
88         reference: 'bullet-list',
89         levels: [{{
90           level: 0,
91           format: LevelFormat.BULLET,
92           text: '\u2022',
93           alignment: AlignmentType.LEFT,
94           style: { paragraph: { indent: { left: 720, hanging: 36
95 0 } } }
96         }]
97       },
98       {
99         reference: 'step-list',
100        levels: [{
```

软著鉴别材料整理器软件 V1.0

```
101         level: 0,
102         format: LevelFormat.DECIMAL,
103         text: '%1.',
104         alignment: AlignmentType.LEFT,
105         style: { paragraph: { indent: { left: 720, hanging: 36
106 0 } } }
107     }]
108 }
109 ]
110 },
111 sections: [{  
112     properties: {  
113         page: {  
114             margin: { top: 1440, right: 1440, bottom: 1440, left: 14
115 40 },  
116             size: { orientation: PageOrientation.PORTRAIT },  
117             pageNumbers: { start: 1, formatType: 'decimal' }  
118         }  
119     },  
120     headers: {  
121         default: new Header({  
122             children: [new Paragraph({  
123                 alignment: AlignmentType.RIGHT,  
124                 children: [new TextRun({ text: '软著鉴别材料整理器软件 v1.0 说明
书',  
125 size: 20 })]
126             })]
127         })
128     },
129     footers: {  
130         default: new Footer({  
131             children: [new Paragraph({  
132                 alignment: AlignmentType.CENTER,  
133                 children: [new TextRun({ children: [PageNumber.CURRENT
134 ] })]
135             })]
136         })
137     },
138     children: [  
139         new Paragraph({ heading: HeadingLevel.TITLE, children: [ne
140 w TextRun('软件说明书')] }),  
141         makeCenter('软件名称: 软著鉴别材料整理器软件 v1.0'),  
142         makeCenter(`编制日期: ${new Date().toISOString().slice(0, 10)}`)
143     ),
144         makeCenter('适用范围: 软著鉴别材料（软件说明与鉴别材料整理）'),
145         new Paragraph({ children: [new TextRun({ text: '', size: 2
146 4 })], pageBreakBefore: true }),  
147         new TableOfContents('目录', { hyperlink: true, headingStyleR
148 ange: '1-3' }),  
149         new Paragraph({ children: [new TextRun({ text: '', size: 2
150 4 })], pageBreakBefore: true }),
```

软著鉴别材料整理器软件 V1.0

```
151     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
152   [new TextRun('1 软件概述')] }),
153   makeParagraph('软著鉴别材料整理器软件 v1.0 用于自动化整理软著登记所需的说明与鉴别材料，支持对软件信息、功能模块、界面截图、生成文档等内容进行统一管理与输出。'),
154   makeParagraph('软件定位：面向需要提交软著材料的开发者、代理机构或企业，通过规范化模板提升材料完整性与一致性。'),
155   makeParagraph('主要特点：'),
156   makeBullet('统一模板：内置说明书模板与结构化字段，确保格式规范。'),
157   makeBullet('材料集中管理：截图、功能描述、版本信息集中归档。'),
158   makeBullet('一键生成：自动输出 Word 文档，减少人工整理成本。'),
159   makeBullet('可追溯：生成记录与版本信息可回溯，便于复用。'),
160   new Paragraph({ heading: HeadingLevel.HEADING_1, children:
161     [new TextRun('2 运行环境与部署')] }),
162   makeParagraph('支持操作系统：Windows 10/11。'),
163   makeParagraph('运行依赖：本地安装 Microsoft Word 或兼容的文档查看器（用于查看生成的 docx）'),
164   makeParagraph('部署方式：应用安装包本地安装，无需联网。'),
165   makeParagraph('启动方式：双击应用图标启动，进入主界面。'),
166   new Paragraph({ heading: HeadingLevel.HEADING_1, children:
167     [new TextRun('3 系统架构与模块')] }),
168   makeParagraph('系统采用模块化结构，核心模块如下：'),
169   makeBullet('资料管理模块：录入软件名称、版本、用途、开发环境等基础信息。'),
170   makeBullet('截图管理模块：导入界面截图、输出示例图，支持排序与说明。'),
171   makeBullet('说明书生成模块：按模板生成包含封面、目录、正文的说明文档。'),
172   makeBullet('导出与校验模块：导出 Word 文档并校验完整性。'),
173   makeBullet('设置与日志模块：保存用户配置与生成记录。'),
174   new Paragraph({ heading: HeadingLevel.HEADING_1, children:
175     [new TextRun('4 功能说明')] }),
176   new Paragraph({ heading: HeadingLevel.HEADING_2, children:
177     [new TextRun('4.1 资料采集与整理')] }),
178   makeParagraph('提供字段化信息录入，包括软件名称、版本号、运行环境、功能描述、适用范围等。'),
179   new Paragraph({ heading: HeadingLevel.HEADING_2, children:
180     [new TextRun('4.2 界面截图管理')] }),
181   makeParagraph('支持导入界面截图与输出示例图，自动归类并生成插图说明。'),
182   new Paragraph({ heading: HeadingLevel.HEADING_2, children:
183     [new TextRun('4.3 说明书自动生成')] }),
184   makeParagraph('按软著登记要求生成包含封面、目录、正文与截图的 Word 说明文档。'),
185   new Paragraph({ heading: HeadingLevel.HEADING_2, children:
186     [new TextRun('4.4 导出与打包')] }),
187   makeParagraph('支持一键导出 docx 文件，便于后续打印或提交。'),
188   new Paragraph({ heading: HeadingLevel.HEADING_2, children:
189     [new TextRun('4.5 日志与版本信息')] }),
```

软著鉴别材料整理器软件 V1.0

```
193     makeParagraph('记录每次生成的时间、版本和素材变更，便于追溯。'),
194     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
195       [new TextRun('5 操作流程')] }),
196     makeParagraph('标准操作流程如下：'),
197     makeStep('启动软件并进入主界面。'),
198     makeStep('录入软件基础信息与功能说明。'),
199     makeStep('导入界面截图与输出示例截图。'),
200     makeStep('选择模板并生成说明书。'),
```

软著鉴别材料整理器软件 V1.0

```
201     makeStep('导出 docx 文档并进行检查。'),
202     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
203       [new TextRun('6 数据结构与存储')] }),
204       makeParagraph('软件在本地保存项目数据，包含基础信息、截图素材与生成结
果。数据不上传网络，默认保存于用
205 户工作目录。'),
206       makeParagraph('数据结构包括：项目信息、截图列表、生成记录、导出文档。
'),
207       new Paragraph({ heading: HeadingLevel.HEADING_1, children:
208         [new TextRun('7 安全与权限')] }),
209         makeParagraph('软件为本地离线工具，默认不进行网络通信。'),
210         makeParagraph('数据仅保存在本地，遵循操作系统文件权限控制。'),
211         new Paragraph({ heading: HeadingLevel.HEADING_1, children:
212           [new TextRun('8 兼容性与限制')] }),
213           makeParagraph('输出文档格式为 docx，需在 Microsoft Word 或兼容软件中
打开。'),
214           makeParagraph('部分排版效果在不同版本的 Word 中可能存在细微差异。'),
215           new Paragraph({ heading: HeadingLevel.HEADING_1, children:
216             [new TextRun('9 运行与维护')] }),
217             makeParagraph('建议定期备份项目数据与导出文档。'),
218             makeParagraph('版本升级时可保留旧版本说明书以便比对。'),
219             new Paragraph({ heading: HeadingLevel.HEADING_1, children:
220               [new TextRun('10 附录：截图')] }),
221               new Paragraph({ heading: HeadingLevel.HEADING_2, children:
222                 [new TextRun('10.1 软件界面截图')] }),
223                 ...addImageBlock('图 1 软件主界面（UI 截图）', uiImagePath, uiSize,
'软
224 件主界面截图'),
225   new Paragraph({ heading: HeadingLevel.HEADING_2, children:
226     [new TextRun('10.2 导出文档示例')] }),
227     ...addImageBlock('图 2 导出文档示例', exportImagePath, exportSize,
228   '导出文档示例截图')
229   ]
230   }]
231 });
232 Packer.toBuffer(doc).then((buffer) => {
233   fs.writeFileSync(outPath, buffer);
234   console.log(`Generated: ${outPath}`);
235 }).catch((err) => {
236   console.error(err);
237   process.exit(1);
238 });
239 const { app, BrowserWindow, dialog, ipcMain, shell } = require(
240   'electron');
241 const path = require('path');
242 const fs = require('fs');
243 const { scanProject } = require('./shared/scanner');
244 const { scanProjectForAI } = require('./shared/aiScanner');
245 const os = require('os');
246 const { writePdfFromDocx } = require('./shared/exportPdf');
247 const { writeDocx } = require('./shared/exportDocx');
```

软著鉴别材料整理器软件 V1.0

```
248 const { writeManualDocx } = require('./shared/exportManualDocx')
249 ;
250 const Store = require('electron-store');
```

软著鉴别材料整理器软件 V1.0

```
251 const { pathToFileURL } = require('url');
252 const { autoUpdater } = require('electron-updater');
253 const store = new Store();
254 let mainWindow;
255 let lastScan = null;
256 function getWindowIcon() {
257     const baseDir = path.join(__dirname, '../', 'build');
258     if (process.platform === 'win32') {
259         return path.join(baseDir, 'icon.ico');
260     }
261     return path.join(baseDir, 'icon.png');
262 }
263 function setupAutoUpdate() {
264     if (!app.isPackaged) return;
265     autoUpdater.autoDownload = true;
266     autoUpdater.autoInstallOnAppQuit = true;
267     autoUpdater.on('error', async (error) => {
268         try {
269             await dialog.showMessageBox(mainWindow, {
270                 type: 'error',
271                 title: '更新失败',
272                 message: '检查更新时发生错误',
273                 detail: String(error?.message || error || '')
274             });
275         } catch {}
276     });
277     autoUpdater.on('update-available', async () => {
278         try {
279             await dialog.showMessageBox(mainWindow, {
280                 type: 'info',
281                 title: '发现新版本',
282                 message: '发现新版本，正在后台下载...',
283                 buttons: ['知道了'],
284                 defaultId: 0
285             });
286         } catch {}
287     });
288     autoUpdater.on('update-downloaded', async () => {
289         try {
290             const result = await dialog.showMessageBox(mainWindow, {
291                 type: 'info',
292                 title: '更新已下载',
293                 message: '新版本已下载完成，是否立即重启并安装？',
294                 buttons: ['立即重启安装', '稍后'],
295                 defaultId: 0,
296                 cancelId: 1
297             });
298             if (result.response === 0) {
299                 autoUpdater.quitAndInstall();
300             }
301         } catch {}
302     });
303 }
```

软著鉴别材料整理器软件 V1.0

```
301     } catch {}  
302   );  
303   autoUpdater.checkForUpdates().catch(() => {});  
304 }  
305 function createWindow() {  
306   mainWindow = new BrowserWindow({  
307     width: 1100,  
308     height: 780,  
309     title: '软著代码资料整理器',  
310     icon: getWindowIcon(),  
311     webPreferences: {  
312       preload: path.join(__dirname, 'preload.js'),  
313       contextIsolation: true,  
314       nodeIntegration: false  
315     }  
316   });  
317   mainWindow.loadFile(path.join(__dirname, 'renderer', 'index.ht  
m1'));  
318 }  
319 app.whenReady().then(() => {  
320   createWindow();  
321   setupAutoUpdate();  
322   app.on('activate', () => {  
323     if (BrowserWindow.getAllWindows().length === 0) createWindow  
324   );  
325   });  
326 });  
327 });  
328 app.on('window-all-closed', () => {  
329   if (process.platform !== 'darwin') app.quit();  
330 });  
331 ipcMain.handle('select-project', async () => {  
332   const result = await dialog.showOpenDialog(mainWindow, {  
333     properties: ['openDirectory']  
334   });  
335   if (result.canceled || result.filePaths.length === 0) return n  
ull;  
336   return result.filePaths[0];  
337 });  
338 ipcMain.handle('select-output-dir', async () => {  
339   const result = await dialog.showOpenDialog(mainWindow, {  
340     properties: ['openDirectory']  
341   });  
342   if (result.canceled || result.filePaths.length === 0) return n  
ull;  
343   return result.filePaths[0];  
344 });  
345 ipcMain.handle('select-ai-screenshots', async () => {  
346   const result = await dialog.showOpenDialog(mainWindow, {  
347     properties: ['openFile', 'multiSelections'],  
348     filters: [  
349   ]  
350 })
```

软著鉴别材料整理器软件 V1.0

```
351     { name: 'Images', extensions: ['png', 'jpg', 'jpeg', 'webp',
352   ', 'bmp' ] }
353   ]
354 );
355 if (result.canceled || result.filePaths.length === 0) return [
356 ];
357 return result.filePaths.map(filePath => ({
358   path: filePath,
359   url: pathToFileURL(filePath).href,
360   name: path.basename(filePath)
361 }));
362 });
363 ipcMain.handle('scan-project', async (_event, options) => {
364   const scan = scanProject(options.projectPath);
365   if (scan.totalPages === 0) {
366     throw new Error('未找到可用源码文件。');
367   }
368   lastScan = {
369     ...scan,
370     headerText: options.headerText,
371     pagesToExport: scan.pagesToExport
372   };
373   return {
374     projectPath: scan.projectPath,
375     fileCount: scan.fileCount,
376     effectiveLines: scan.effectiveLines,
377     totalPages: scan.totalPages,
378     selectedPages: scan.pagesToExport.length
379   };
380 });
381 ipcMain.handle('generate-docx', async (_event, options) => {
382   if (!lastScan) throw new Error('请先扫描项目。');
383   const outputPath = path.join(options.outputDir, options.fileName);
384   await writeDocx(outputPath, lastScan.pagesToExport, lastScan.h
385 eaderText);
386   return outputPath;
387 });
388 ipcMain.handle('generate-pdf', async (_event, options) => {
389   if (!lastScan) throw new Error('请先扫描项目。');
390   const outputPath = path.join(options.outputDir, options.fileName);
391   const tempDir = fs.mkdtempSync(path.join(os.tmpdir(), 'softreg
392 -'));
393   const tempDocx = path.join(tempDir, 'program-material.docx');
394   await writeDocx(tempDocx, lastScan.pagesToExport, lastScan.he
395 aderText);
396   writePdfFromDocx(tempDocx, outputPath);
397   return outputPath;
398 });
399 });
400 });
```

软著鉴别材料整理器软件 V1.0

```
401 ipcMain.handle('set-export-pages', async (_event, mode) => {
402   if (!lastScan) throw new Error('请先扫描项目。');
403   if (mode === 'all') {
404     lastScan.pagesToExport = lastScan.pages;
405   } else {
406     lastScan.pagesToExport = lastScan.selectedPages;
407   }
408   return {
409     pagesToExport: lastScan.pagesToExport.length
410   };
411 });
412 ipcMain.handle('init-scan-state', async () => {
413   lastScan = null;
414 });
415 ipcMain.handle('open-external', async (_event, url) => {
416   await shell.openExternal(url);
417 });
418 ipcMain.handle('open-output-dir', async (_event, outputDir) => {
419   const dir = String(outputDir || '').trim();
420   if (!dir) return false;
421   const result = await shell.openPath(dir);
422   if (result) throw new Error(result);
423   return true;
424 });
425 ipcMain.handle('ensure-output-dir', async (_event, outputDir) =>
426 {
427   if (!fs.existsSync(outputDir)) {
428     fs.mkdirSync(outputDir, { recursive: true });
429   }
430   return true;
431 });
432 ipcMain.handle('ai:save-config', async (_event, config) => {
433   store.set('aiConfig', config);
434   return true;
435 });
436 ipcMain.handle('ai:get-config', async () => {
437   return store.get('aiConfig') || {
438     baseUrl: 'https://api.openai.com/v1',
439     apiKey: '',
440     model: 'gpt-3.5-turbo'
441   };
442 });
443 ipcMain.handle('ai:generate-manual', async (_event, { outputPath,
444 config, screenshots }) => {
445   try {
446     const context = scanProjectForAI(projectPath);
447     const shotList = Array.isArray(screenshots) ? screenshots :
448     [];
449     const shotText = shotList.length
450       ? shotList
```

软著鉴别材料整理器软件 V1.0

```
451         .map((s, i) => {
452             const note = String(s?.note || '').trim();
453             const name = String(s?.name || '').trim();
454             const label = note || name || String(s?.path || '').
455             trim();
456             return `${i + 1}. ${label}`;
457         })
458         .join('\n')
459         : '(无)';
460     const prompt = `

461 你是一个专业的软件文档撰写专家。请根据以下项目上下文，撰写一份详细的《软件说明书》。
462 请严格使用 Markdown 标题 (# / ## / ###) 组织结构，标题不要只用纯编号列表代替。
463 ## 截图列表（可用于插图）
464 ${shotText}
465 当正文需要插入某张截图时，请在合适位置“单独输出一行”占位符，格式必须严格为：
466 [[SCREENSHOT:截图说明]]
467 其中“截图说明”必须与上面的截图列表某一项完全一致（优先使用用户填写的备注）。
468 ## 1. 项目结构
469 \```
470 ${context.structure}
471 \```
472 ## 2. README 内容
473 ${context.readme}
474 ## 3. 依赖/配置信息
475 ${context.packageJson}
476 ## 4. 核心代码片段
477 ${context.sourceSnippets.map(s => `--- ${s.path} ---\n${s.content}`).join('\n\n')}

478 请输出 Markdown 格式，包含以下章节：
479 1. # 软件概述（根据 README 和代码推断）
480 2. # 功能列表（详细列出功能点）
481 3. # 技术特点（架构、语言、关键技术）
482 4. # 运行环境要求
483 5. # 使用说明（如有）
484 `;
485
486     const baseUrl = config.baseUrl.replace(/\//, '');
487     const response = await fetch(` ${baseUrl}/chat/completions`,
488     {
489         method: 'POST',
490         headers: {
491             'Content-Type': 'application/json',
492             'Authorization': `Bearer ${config.apiKey}`
493         },
494         body: JSON.stringify({
495             model: config.model,
496             messages: [
497                 { role: 'system', content: '你是一个专业的软件技术文档专家，擅长
编写软件说明书。
498             ' },
```

软著鉴别材料整理器软件 V1.0

```
499      { role: 'user', content: prompt }
500    ],

```

软著鉴别材料整理器软件 V1.0

```
501     temperature: 0.7
502   })
503 });
504 if (!response.ok) {
505   const errText = await response.text();
506   throw new Error(`AI 请求失败 (${response.status}): ${errText}`)
507 );
508 }
509 const data = await response.json();
510 if (!data.choices || data.choices.length === 0) {
511   throw new Error('AI 返回结果为空');
512 }
513 return data.choices[0].message.content;
514 } catch (error) {
515   console.error('AI Generation Error:', error);
516   throw error;
517 }
518 });
519 ipcMain.handle('ai:export-manual-docx', async (_event, options)
520 => {
521   const outputPath = path.join(options.outputDir, options.fileName);
522   await writeManualDocx(outputPath, options);
523   return outputPath;
524 });
525 ipcMain.handle('ai:export-manual-pdf', async (_event, options) =
526 => {
527   const outputPath = path.join(options.outputDir, options.fileName);
528   const tempDir = fs.mkdtempSync(path.join(os.tmpdir(), 'softreg-
529 -manual-'));
530   const tempDocx = path.join(tempDir, 'manual.docx');
531   await writeManualDocx(tempDocx, options);
532   writePdfFromDocx(tempDocx, outputPath);
533   return outputPath;
534 });
535 ipcMain.handle('save-file', async (_event, { content, defaultName
536 }) => {
537   const result = await dialog.showSaveDialog(mainWindow, {
538     defaultPath: defaultName,
539     filters: [{ name: 'Markdown', extensions: ['md'] }]
540   });
541   if (!result.canceled && result.filePath) {
542     fs.writeFileSync(result.filePath, content, 'utf8');
543     return result.filePath;
544   }
545   return null;
546 });
547 });
548 const { contextBridge, ipcRenderer } = require('electron');
549 contextBridge.exposeInMainWorld('softreg', {
```

软著鉴别材料整理器软件 V1.0

```
551     selectProject: () => ipcRenderer.invoke('select-project'),
552     selectOutputDir: () => ipcRenderer.invoke('select-output-dir')
553   ,
554     scanProject: (options) => ipcRenderer.invoke('scan-project', o
555   ptions),
556     generateDocx: (options) => ipcRenderer.invoke('generate-docx', o
557   ptions),
558     generatePdf: (options) => ipcRenderer.invoke('generate-pdf', o
559   ptions),
560     setExportPages: (mode) => ipcRenderer.invoke('set-export-pages
561   ', mode),
562     initScanState: () => ipcRenderer.invoke('init-scan-state'),
563     ensureOutputDir: (outputDir) => ipcRenderer.invoke('ensure-out
564   put-dir', outputDir),
565     openExternal: (url) => ipcRenderer.invoke('open-external', url
566   ),
567     openOutputDir: (outputDir) => ipcRenderer.invoke('open-output-
568   dir', outputDir),
569     ai: {
570       saveConfig: (config) => ipcRenderer.invoke('ai:save-config',
571   config),
572       getConfig: () => ipcRenderer.invoke('ai:get-config'),
573       generateManual: (projectPath, config, screenshots) => ipcRen
574   derer.invoke('ai:generate-manual', { projectPath, config, screen
575   shots }),
576       exportManualDocx: (options) => ipcRenderer.invoke('ai:export-
577   -manual-docx', options),
578       exportManualPdf: (options) => ipcRenderer.invoke('ai:export-
579   -manual-pdf', options)
580     },
581     selectAiScreenshots: () => ipcRenderer.invoke('select-ai-scree
582   nshots'),
583     saveFile: (content, defaultName) => ipcRenderer.invoke('save-f
584   ile', { content, defaultName })
585   });
586 const projectPathInput = document.getElementById('projectPath');
587 const outputDirInput = document.getElementById('outputDir');
588 const softwareNameInput = document.getElementById('softwareName'
589 );
590 const softwareVersionInput = document.getElementById('softwareVe
591   rsion');
592 const statFiles = document.getElementById('statFiles');
593 const statLines = document.getElementById('statLines');
594 const statPages = document.getElementById('statPages');
595 const statExport = document.getElementById('statExport');
596 const statExportLines = document.getElementById('statExportLines
597   ');
598 const statusEl = document.getElementById('status');
599 const btnSelectProject = document.getElementById('btnSelectProje
600   ct');
```

软著鉴别材料整理器软件 V1.0

```
601 const btnSelectOutput = document.getElementById('btnSelectOutput');
602
603 const btnScan = document.getElementById('btnScan');
604 const btnExportDocx = document.getElementById('btnExportDocx');
605 const btnExportPdf = document.getElementById('btnExportPdf');
606 const btnExportBoth = document.getElementById('btnExportBoth');
607 const lightRed = document.querySelector('.light.red');
608 const lightAmber = document.querySelector('.light.amber');
609 const lightGreen = document.querySelector('.light.green');
610 function updateLights(state) {
611     lightRed.classList.remove('active');
612     lightAmber.classList.remove('active');
613     lightGreen.classList.remove('active');
614     switch (state) {
615         case 'idle':
616             lightRed.classList.add('active');
617             break;
618         case 'busy':
619             lightAmber.classList.add('active');
620             break;
621         case 'success':
622             lightGreen.classList.add('active');
623             break;
624         case 'error':
625             lightRed.classList.add('active');
626             break;
627     }
628 }
629 function formatNum(num, length) {
630     return String(num).padStart(length, '0');
631 }
632 function getSoftwareType() {
633     const selected = document.querySelector('input[name="softwareType"]:checked');
634     return selected ? selected.value : 'software';
635 }
636 function buildHeaderText() {
637     const name = softwareNameInput.value.trim();
638     const version = softwareVersionInput.value.trim();
639     const type = getSoftwareType() === 'game' ? '游戏软件' : '软件';
640     if (!name || !version) return '';
641     return `${name}${type}V${version}`;
642 }
643
644 function setStatus(message) {
645     statusEl.textContent = message.toUpperCase();
646     console.log(`[系统] ${message}`);
647 }
648 function validateInputs() {
649     if (!projectIdInput.value) {
650         setStatus('错误：缺失项目路径');
```

软著鉴别材料整理器软件 V1.0

```
651     updateLights('error');
652     return false;
653 }
654 if (!softwareNameInput.value.trim()) {
655     setStatus('错误：需填写软件名称');
656     updateLights('error');
657     return false;
658 }
659 if (!softwareVersionInput.value.trim()) {
660     setStatus('错误：需填写版本号');
661     updateLights('error');
662     return false;
663 }
664 if (!outputDirInput.value) {
665     setStatus('错误：缺失输出路径');
666     updateLights('error');
667     return false;
668 }
669 return true;
670 }
671 async function handleSelectProject() {
672     const selected = await window.softreg.selectProject();
673     if (selected) projectPathInput.value = selected;
674 }
675 async function handleSelectOutput() {
676     const selected = await window.softreg.selectOutputDir();
677     if (selected) outputDirInput.value = selected;
678 }
679 const scanOverlay = document.getElementById('scanOverlay');
680 let audioCtx = null;
681 function playScanSound() {
682     if (!audioCtx) audioCtx = new (window.AudioContext || window.webkitAudioContext)();
683     if (audioCtx.state === 'suspended') audioCtx.resume();
684     const now = audioCtx.currentTime;
685     const duration = 2.5; // 与填充动画时长一致
686     const osc1 = audioCtx.createOscillator();
687     const osc2 = audioCtx.createOscillator(); // 副振荡器（正弦波 - 提供底
蕴）
688     const filter = audioCtx.createBiquadFilter();
689     const gainNode = audioCtx.createGain();
690     osc1.type = 'sawtooth';
691     osc1.frequency.setValueAtTime(80, now);
692     osc1.frequency.exponentialRampToValueAtTime(450, now + duratio
n);
693     osc2.type = 'sine';
694     osc2.frequency.setValueAtTime(40, now);
695     osc2.frequency.exponentialRampToValueAtTime(120, now + duratio
n);
696     filter.type = 'lowpass';
697     filter.Q.value = 15; // 高共振
```

软著鉴别材料整理器软件 V1.0

```
701     filter.frequency.setValueAtTime(200, now);
702     filter.frequency.exponentialRampToValueAtTime(3000, now + duration);
703     gainNode.gain.setValueAtTime(0, now);
704     gainNode.gain.linearRampToValueAtTime(0.08, now + 0.1); // 快速开启
705     gainNode.gain.exponentialRampToValueAtTime(0.001, now + duration); // 逐渐消失
706
707     osc1.connect(filter);
708     osc2.connect(filter);
709     filter.connect(gainNode);
710     gainNode.connect(audioCtx.destination);
711     osc1.start(now);
712     osc2.start(now);
713     osc1.stop(now + duration);
714     osc2.stop(now + duration);
715 }
716
717 function startButtonEffect(btn) {
718     btn.classList.add('processing');
719     playScanSound();
720 }
721
722 function stopButtonEffect(btn) {
723     btn.classList.remove('processing');
724 }
725
726 async function handleScan() {
727     if (!projectPathInput.value) {
728         setStatus('错误：请先选择项目目录');
729         updateLights('error');
730         return;
731     }
732     const headerText = buildHeaderText();
733     if (!headerText) {
734         setStatus('错误：名称和版本号缺失');
735         updateLights('error');
736         return;
737     }
738     try {
739         setStatus('状态：正在扫描项目文件...');
740         updateLights('busy');
741         startButtonEffect(btnScan);
742         await window.softreg.initScanState();
743         const result = await window.softreg.scanProject({
744             projectPath: projectPathInput.value,
745             headerText
746         });
747         await new Promise(resolve => setTimeout(resolve, 2500));
748         statFiles.textContent = formatNum(result.fileCount, 4);
749         statLines.textContent = formatNum(result.effectiveLines, 6);
750         statPages.textContent = formatNum(result.totalPages, 3);
751         statExport.textContent = formatNum(result.selectedPages, 3);
752     } catch (err) {
753         console.error(err);
754     }
755 }
```

软著鉴别材料整理器软件 V1.0

```
751     statExportLines.textContent = formatNum(result.selectedPages
752     * 50, 4);
753     const truncationNote = result.totalPages > 60 ? ` (已截断至 3000
754 行)` : '';
755     setStatus(`状态: 扫描完成${truncationNote}. 准备就绪.`);
756     updateLights('success');
757   } catch (err) {
758     setStatus(`错误: 扫描失败 // ${String(err.message || err).toUpperCase()}`);
759     updateLights('error');
760   } finally {
761     stopButtonEffect(btnScan);
762   }
763 }
764 }
765 function buildFileName(suffix) {
766   const headerText = buildHeaderText();
767   return `${headerText}_${suffix}`;
768 }
769 async function exportDocx() {
770   if (!validateInputs()) return;
771   try {
772     setStatus('状态: 正在生成 DOCX 数据流...');
773     updateLights('busy');
774     startButtonEffect(btnExportDocx);
775     await window.softreg.ensureOutputDir(outputDirInput.value);
776     const fileName = buildFileName('程序鉴别材料.docx');
777     const outputPath = await window.softreg.generateDocx({
778       outputDir: outputDirInput.value,
779       fileName
780     });
781     await new Promise(resolve => setTimeout(resolve, 2500));
782     setStatus(`成功: DOCX 已保存至磁盘`);
783     updateLights('success');
784   } catch (err) {
785     setStatus(`错误: DOCX 生成失败 // ${String(err.message || err).toUpperCase()}`);
786     updateLights('error');
787   } finally {
788     stopButtonEffect(btnExportDocx);
789   }
790 }
791 }
792 async function exportPdf() {
793   if (!validateInputs()) return;
794   try {
795     setStatus('状态: 正在导出 PDF 缓冲区...');
796     const fs = require('fs');
797     const path = require('path');
798     const {
799       Document, Packer, Paragraph, TextRun, ImageRun, Header, Footer
800     ,
```

软著鉴别材料整理器软件 V1.0

```
801     AlignmentType, PageOrientation, HeadingLevel, TableOfContents,
802     LevelFormat, PageNumber
803 } = require('docx');
804 const outPath = path.join(__dirname, 'softreg-doc-v1.0-cn.docx')
805 ;
806 const uiImagePath = path.join(__dirname, 'screenshot-app-ui.png')
807 );
808 const exportImagePath = path.join(__dirname, 'screenshot-export-
809 doc.png');
810 const makeCenter = (text) => new Paragraph({
811     alignment: AlignmentType.CENTER,
812     spacing: { after: 120 },
813     children: [new TextRun({ text, size: 28 })]
814 });
815 const makeParagraph = (text) => new Paragraph({
816     spacing: { after: 120 },
817     children: [new TextRun({ text, size: 24 })]
818 });
819 const makeBullet = (text) => new Paragraph({
820     numbering: { reference: 'bullet-list', level: 0 },
821     children: [new TextRun({ text, size: 24 })]
822 });
823 const makeStep = (text) => new Paragraph({
824     numbering: { reference: 'step-list', level: 0 },
825     children: [new TextRun({ text, size: 24 })]
826 });
827 const scaleToWidth = (w, h, targetW) => {
828     const ratio = targetW / w;
829     return { width: Math.round(w * ratio), height: Math.round(h *
830 ratio) };
831 };
832 const addImageBlock = (title, imagePath, size, altText) => {
833     const data = fs.readFileSync(imagePath);
834     return [
835         new Paragraph({
836             spacing: { after: 120 },
837             children: [new TextRun({ text: title, bold: true, size: 24
838 })]
839     }),
840         new Paragraph({
841             alignment: AlignmentType.CENTER,
842             spacing: { after: 120 },
843             children: [new ImageRun({
844                 type: 'png',
845                 data,
846                 transformation: { width: size.width, height: size.height
847 , rotation: 0 },
848                 altText: { title: altText, description: altText, name: a
849 ltText }
850             })]
```

```
851     })
852   ];
853 };
854 const uiSize = scaleToWidth(939, 1547, 420);
855 const exportSize = scaleToWidth(927, 1291, 420);
856 const doc = new Document({
857   styles: {
858     default: { document: { run: { font: 'SimSun', size: 24 } } }
859   ,
860   paragraphStyles: [
861     { id: 'Title', name: 'Title', basedOn: 'Normal',
862       run: { size: 56, bold: true, color: '000000', font: 'Mic
863 rosoft YaHei' },
864       paragraph: { spacing: { before: 240, after: 120 }, align
865 ment: AlignmentType.CENTER } },
866     { id: 'Heading1', name: 'Heading 1', basedOn: 'Normal', ne
867 xt: 'Normal', quickFormat: true,
868       run: { size: 32, bold: true, color: '000000', font: 'Mic
869 rosoft YaHei' },
870       paragraph: { spacing: { before: 240, after: 180 }, outli
871 neLevel: 0 } },
872     { id: 'Heading2', name: 'Heading 2', basedOn: 'Normal', ne
873 xt: 'Normal', quickFormat: true,
874       run: { size: 28, bold: true, color: '000000', font: 'Mic
875 rosoft YaHei' },
876       paragraph: { spacing: { before: 180, after: 120 }, outli
877 neLevel: 1 } }
877   ]
878 },
879   numbering: {
880     config: [
881       {
882         reference: 'bullet-list',
883         levels: [{{
884           level: 0,
885           format: LevelFormat.BULLET,
886           text: '\u2022',
887           alignment: AlignmentType.LEFT,
888           style: { paragraph: { indent: { left: 720, hanging: 36
889             0 } } }
890         }]
891       }],
892     },
893     {
894       reference: 'step-list',
895       levels: [{{
896         level: 0,
897         format: LevelFormat.DECIMAL,
898         text: '%1.',
899         alignment: AlignmentType.LEFT,
900         style: { paragraph: { indent: { left: 720, hanging: 36
901           0 } } }
902       }]
903     }
904   }
905 }
```

软著鉴别材料整理器软件 V1.0

```
901 0 } } }
902     }]
903   }
904   ]
905 },
906 sections: [
907   properties: {
908     page: {
909       margin: { top: 1440, right: 1440, bottom: 1440, left: 14
910 40 },
911       size: { orientation: PageOrientation.PORTRAIT },
912       pageNumbers: { start: 1, formatType: 'decimal' }
913     }
914   },
915   headers: {
916     default: new Header({
917       children: [new Paragraph({
918         alignment: AlignmentType.RIGHT,
919         children: [new TextRun({ text: '软著鉴别材料整理器软件 v1.0 说明
书',
920         size: 20 })]
921       })
922     })
923   },
924   footers: {
925     default: new Footer({
926       children: [new Paragraph({
927         alignment: AlignmentType.CENTER,
928         children: [new TextRun({ children: [PageNumber.CURRENT
929 ] })]
930       })
931     })
932   },
933   children: [
934     new Paragraph({ heading: HeadingLevel.TITLE, children: [ne
935 w TextRun('软件说明书')] }),
936     makeCenter('软件名称: 软著鉴别材料整理器软件 v1.0'),
937     makeCenter(`编制日期: ${new Date().toISOString().slice(0, 10)}`),
938   ],
939   makeCenter('适用范围: 软著鉴别材料（软件说明与鉴别材料整理）'),
940   new Paragraph({ children: [new TextRun({ text: '', size: 2
941 4 })], pageBreakBefore: true }),
942   new TableOfContents('目录', { hyperlink: true, headingStyleR
943 ange: '1-3' }),
944   new Paragraph({ children: [new TextRun({ text: '', size: 2
945 4 })], pageBreakBefore: true }),
946   new Paragraph({ heading: HeadingLevel.HEADING_1, children:
947   [new TextRun('1 软件概述')] }),
948   makeParagraph('软著鉴别材料整理器软件 v1.0 用于自动化整理软著登记所需
的说明与鉴别材料，支持对软件信
949 息、功能模块、界面截图、生成文档等内容进行统一管理与输出。'),
```

软著鉴别材料整理器软件 V1.0

950 makeParagraph('软件定位：面向需要提交软著材料的开发者、代理机构或企业，通过规范化模板提升材料完整性与

软著鉴别材料整理器软件 V1.0

```
951 一致性。'),
952     makeParagraph('主要特点: '),
953     makeBullet('统一模板: 内置说明书模板与结构化字段, 确保格式规范。'),
954     makeBullet('材料集中管理: 截图、功能描述、版本信息集中归档。'),
955     makeBullet('一键生成: 自动输出 Word 文档, 减少人工整理成本。'),
956     makeBullet('可追溯: 生成记录与版本信息可回溯, 便于复用。'),
957     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
958       [new TextRun('2 运行环境与部署')] }),
959     makeParagraph('支持操作系统: Windows 10/11。'),
960     makeParagraph('运行依赖: 本地安装 Microsoft Word 或兼容的文档查看器
(用于查看生成的 doc
961   x)。'),
962     makeParagraph('部署方式: 应用安装包本地安装, 无需联网。'),
963     makeParagraph('启动方式: 双击应用图标启动, 进入主界面。'),
964     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
965       [new TextRun('3 系统架构与模块')] }),
966     makeParagraph('系统采用模块化结构, 核心模块如下: '),
967     makeBullet('资料管理模块: 录入软件名称、版本、用途、开发环境等基础信
息。'),
968     makeBullet('截图管理模块: 导入界面截图、输出示例图, 支持排序与说明。
'),
969     makeBullet('说明书生成模块: 按模板生成包含封面、目录、正文的说明文档。
'),
970     makeBullet('导出与校验模块: 导出 Word 文档并校验完整性。'),
971     makeBullet('设置与日志模块: 保存用户配置与生成记录。'),
972     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
973       [new TextRun('4 功能说明')] }),
974     new Paragraph({ heading: HeadingLevel.HEADING_2, children:
975       [new TextRun('4.1 资料采集与整理')] }),
976     makeParagraph('提供字段化信息录入, 包括软件名称、版本号、运行环境、功
能描述、适用范围等。'),
977     new Paragraph({ heading: HeadingLevel.HEADING_2, children:
978       [new TextRun('4.2 界面截图管理')] }),
979     makeParagraph('支持导入界面截图与输出示例图, 自动归类并生成插图说明。
'),
980     new Paragraph({ heading: HeadingLevel.HEADING_2, children:
981       [new TextRun('4.3 说明书自动生成')] }),
982     makeParagraph('按软著登记要求生成包含封面、目录、正文与截图的 Word 说
明文档。'),
983     new Paragraph({ heading: HeadingLevel.HEADING_2, children:
984       [new TextRun('4.4 导出与打包')] }),
985     makeParagraph('支持一键导出 docx 文件, 便于后续打印或提交。'),
986     new Paragraph({ heading: HeadingLevel.HEADING_2, children:
987       [new TextRun('4.5 日志与版本信息')] }),
988     makeParagraph('记录每次生成的时间、版本和素材变更, 便于追溯。'),
989     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
990       [new TextRun('5 操作流程')] }),
991     makeParagraph('标准操作流程如下: '),
992     makeStep('启动软件并进入主界面。'),
993     makeStep('录入软件基础信息与功能说明。'),
994     makeStep('导入界面截图与输出示例截图。'),
```

软著鉴别材料整理器软件 V1.0

```
995     makeStep('选择模板并生成说明书。'),
996     makeStep('导出 docx 文档并进行检查。'),
997     new Paragraph({ heading: HeadingLevel.HEADING_1, children:
998       [new TextRun('6 数据结构与存储')]}),
999     makeParagraph('软件在本地保存项目数据，包含基础信息、截图素材与生成结
果。数据不上传网络，默认保存于用
1000 户工作目录。'),
```

软著鉴别材料整理器软件 V1.0

```
1001     makeParagraph('数据结构包括：项目信息、截图列表、生成记录、导出文档。  
' ),  
1002     new Paragraph({ heading: HeadingLevel.HEADING_1, children:  
1003       [new TextRun('7 安全与权限')] }),  
1004       makeParagraph('软件为本地离线工具，默认不进行网络通信。'),  
1005       makeParagraph('数据仅保存在本地，遵循操作系统文件权限控制。'),  
1006       new Paragraph({ heading: HeadingLevel.HEADING_1, children:  
1007         [new TextRun('8 兼容性与限制')] }),  
1008         makeParagraph('输出文档格式为 docx，需在 Microsoft Word 或兼容软件中  
打开。'),  
1009         makeParagraph('部分排版效果在不同版本的 Word 中可能存在细微差异。'),  
1010         new Paragraph({ heading: HeadingLevel.HEADING_1, children:  
1011           [new TextRun('9 运行与维护')] }),  
1012           makeParagraph('建议定期备份项目数据与导出文档。'),  
1013           makeParagraph('版本升级时可保留旧版本说明书以便比对。'),  
1014           new Paragraph({ heading: HeadingLevel.HEADING_1, children:  
1015             [new TextRun('10 附录：截图')] }),  
1016             new Paragraph({ heading: HeadingLevel.HEADING_2, children:  
1017               [new TextRun('10.1 软件界面截图')] }),  
1018                 ...addImageBlock('图 1 软件主界面（UI 截图）', uiImagePath, uiSize,  
'软  
件主界面截图'),  
1019                 new Paragraph({ heading: HeadingLevel.HEADING_2, children:  
1020                   [new TextRun('10.2 导出文档示例')] }),  
1021                     ...addImageBlock('图 2 导出文档示例', exportImagePath, exportSize,  
1022 '导出文档示例截图')  
1023           ]  
1024         }]  
1025     });  
1026   );  
1027   Packer.toBuffer(doc).then((buffer) => {  
1028     fs.writeFileSync(outPath, buffer);  
1029     console.log(`Generated: ${outPath}`);  
1030   }).catch((err) => {  
1031     console.error(err);  
1032     process.exit(1);  
1033   });  
1034   const { app, BrowserWindow, dialog, ipcMain, shell } = require('electron');  
1035   const path = require('path');  
1036   const fs = require('fs');  
1038   const { scanProject } = require('./shared/scanner');  
1039   const { scanProjectForAI } = require('./shared/aiScanner');  
1040   const os = require('os');  
1041   const { writePdfFromDocx } = require('./shared/exportPdf');  
1042   const { writeDocx } = require('./shared/exportDocx');  
1043   const { writeManualDocx } = require('./shared/exportManualDocx')  
1044   ;  
1045   const Store = require('electron-store');  
1046   const { pathToFileURL } = require('url');  
1047   const { autoUpdater } = require('electron-updater');  
1048   const store = new Store();
```

软著鉴别材料整理器软件 V1.0

```
1049 let mainWindow;  
1050 let lastScan = null;
```

软著鉴别材料整理器软件 V1.0

```
1051 function getWindowIcon() {
1052     const baseDir = path.join(__dirname, '..', 'build');
1053     if (process.platform === 'win32') {
1054         return path.join(baseDir, 'icon.ico');
1055     }
1056     return path.join(baseDir, 'icon.png');
1057 }
1058 function setupAutoUpdate() {
1059     if (!app.isPackaged) return;
1060     autoUpdater.autoDownload = true;
1061     autoUpdater.autoInstallOnAppQuit = true;
1062     autoUpdater.on('error', async (error) => {
1063         try {
1064             await dialog.showMessageBox(mainWindow, {
1065                 type: 'error',
1066                 title: '更新失败',
1067                 message: '检查更新时发生错误',
1068                 detail: String(error?.message || error || '')
1069             });
1070         } catch {}
1071     });
1072     autoUpdater.on('update-available', async () => {
1073         try {
1074             await dialog.showMessageBox(mainWindow, {
1075                 type: 'info',
1076                 title: '发现新版本',
1077                 message: '发现新版本，正在后台下载...',
1078                 buttons: ['知道了'],
1079                 defaultId: 0
1080             });
1081         } catch {}
1082     });
1083     autoUpdater.on('update-downloaded', async () => {
1084         try {
1085             const result = await dialog.showMessageBox(mainWindow, {
1086                 type: 'info',
1087                 title: '更新已下载',
1088                 message: '新版本已下载完成，是否立即重启并安装？',
1089                 buttons: ['立即重启安装', '稍后'],
1090                 defaultId: 0,
1091                 cancelId: 1
1092             });
1093             if (result.response === 0) {
1094                 autoUpdater.quitAndInstall();
1095             }
1096         } catch {}
1097     });
1098     autoUpdater.checkForUpdates().catch(() => {});
1099 }
1100 function createWindow() {
```

软著鉴别材料整理器软件 V1.0

```
1101     mainWindow = new BrowserWindow({
1102         width: 1100,
1103         height: 780,
1104         title: '软著代码资料整理器',
1105         icon: getWindowIcon(),
1106         webPreferences: {
1107             preload: path.join(__dirname, 'preload.js'),
1108             contextIsolation: true,
1109             nodeIntegration: false
1110         }
1111     });
1112     mainWindow.loadFile(path.join(__dirname, 'renderer', 'index.ht
1113 ml'));
1114 }
1115 app.whenReady().then(() => {
1116     createWindow();
1117     setupAutoUpdate();
1118     app.on('activate', () => {
1119         if (BrowserWindow.getAllWindows().length === 0) createWindow
1120     ());
1121     });
1122 });
1123 app.on('window-all-closed', () => {
1124     if (process.platform !== 'darwin') app.quit();
1125 });
1126 ipcMain.handle('select-project', async () => {
1127     const result = await dialog.showOpenDialog(mainWindow, {
1128         properties: ['openDirectory']
1129     });
1130     if (result.canceled || result.filePaths.length === 0) return n
1131 ull;
1132     return result.filePaths[0];
1133 });
1134 ipcMain.handle('select-output-dir', async () => {
1135     const result = await dialog.showOpenDialog(mainWindow, {
1136         properties: ['openDirectory']
1137     });
1138     if (result.canceled || result.filePaths.length === 0) return n
1139 ull;
1140     return result.filePaths[0];
1141 });
1142 ipcMain.handle('select-ai-screenshots', async () => {
1143     const result = await dialog.showOpenDialog(mainWindow, {
1144         properties: ['openFile', 'multiSelections'],
1145         filters: [
1146             { name: 'Images', extensions: ['png', 'jpg', 'jpeg', 'webp
1147 ', 'bmp'] }
1148         ]
1149     });
1150     if (result.canceled || result.filePaths.length === 0) return [
```

软著鉴别材料整理器软件 V1.0

```
1151 ];
1152     return result.filePaths.map(filePath => ({
1153         path: filePath,
1154         url: pathToFileURL(filePath).href,
1155         name: path.basename(filePath)
1156     }));
1157 });
1158 ipcMain.handle('scan-project', async (_event, options) => {
1159     const scan = scanProject(options.projectPath);
1160     if (scan.totalPages === 0) {
1161         throw new Error('未找到可用源码文件。');
1162     }
1163     lastScan = {
1164         ...scan,
1165         headerText: options.headerText,
1166         pagesToExport: scan.pagesToExport
1167     };
1168     return {
1169         projectPath: scan.projectPath,
1170         fileCount: scan.fileCount,
1171         effectiveLines: scan.effectiveLines,
1172         totalPages: scan.totalPages,
1173         selectedPages: scan.pagesToExport.length
1174     };
1175 });
1176 ipcMain.handle('generate-docx', async (_event, options) => {
1177     if (!lastScan) throw new Error('请先扫描项目。');
1178     const outputPath = path.join(options.outputDir, options.fileName);
1179     await writeDocx(outputPath, lastScan.pagesToExport, lastScan.headerText);
1180     return outputPath;
1181 });
1182 ipcMain.handle('generate-pdf', async (_event, options) => {
1183     if (!lastScan) throw new Error('请先扫描项目。');
1184     const outputPath = path.join(options.outputDir, options.fileName);
1185     const tempDir = fs.mkdtempSync(path.join(os.tmpdir(), 'softreg-'));
1186     const tempDocx = path.join(tempDir, 'program-material.docx');
1187     await writeDocx(tempDocx, lastScan.pagesToExport, lastScan.headerText);
1188     writePdfFromDocx(tempDocx, outputPath);
1189     return outputPath;
1190 });
1191 ipcMain.handle('set-export-pages', async (_event, mode) => {
1192     if (!lastScan) throw new Error('请先扫描项目。');
1193     if (mode === 'all') {
1194         lastScan.pagesToExport = lastScan.pages;
1195     } else {
```

软著鉴别材料整理器软件 V1.0

```
1201     lastScan.pagesToExport = lastScan.selectedPages;
1202   }
1203   return {
1204     pagesToExport: lastScan.pagesToExport.length
1205   };
1206 });
1207 ipcMain.handle('init-scan-state', async () => {
1208   lastScan = null;
1209 });
1210 ipcMain.handle('open-external', async (_event, url) => {
1211   await shell.openExternal(url);
1212 });
1213 ipcMain.handle('open-output-dir', async (_event, outputDir) => {
1214   const dir = String(outputDir || '').trim();
1215   if (!dir) return false;
1216   const result = await shell.openPath(dir);
1217   if (result) throw new Error(result);
1218   return true;
1219 });
1220 ipcMain.handle('ensure-output-dir', async (_event, outputDir) =>
1221 {
1222   if (!fs.existsSync(outputDir)) {
1223     fs.mkdirSync(outputDir, { recursive: true });
1224   }
1225   return true;
1226 });
1227 ipcMain.handle('ai:save-config', async (_event, config) => {
1228   store.set('aiConfig', config);
1229   return true;
1230 });
1231 ipcMain.handle('ai:get-config', async () => {
1232   return store.get('aiConfig') || {
1233     baseUrl: 'https://api.openai.com/v1',
1234     apiKey: '',
1235     model: 'gpt-3.5-turbo'
1236   };
1237 });
1238 ipcMain.handle('ai:generate-manual', async (_event, { projectPat
1239 h, config, screenshots }) => {
1240   try {
1241     const context = scanProjectForAI(projectPath);
1242     const shotList = Array.isArray(screenshots) ? screenshots :
1243   [];
1244     const shotText = shotList.length
1245       ? shotList
1246         .map((s, i) => {
1247           const note = String(s?.note || '').trim();
1248           const name = String(s?.name || '').trim();
1249           const label = note || name || String(s?.path || '').
1250 trim();
```

软著鉴别材料整理器软件 V1.0

```
1251         return `${i + 1}. ${label}`;
1252     })
1253     .join('\n')
1254     : '(无)';
1255     const prompt =
1256 你是一个专业的软件文档撰写专家。请根据以下项目上下文，撰写一份详细的《软件说明书》。
1257 请严格使用 Markdown 标题 (# / ## / ###) 组织结构，标题不要只用纯编号列表代替。
1258 ## 截图列表（可用于插图）
1259 ${shotText}
1260 当正文需要插入某张截图时，请在合适位置“单独输出一行”占位符，格式必须严格为：
1261 [[SCREENSHOT:截图说明]]
1262 其中“截图说明”必须与上面的截图列表某一项完全一致（优先使用用户填写的备注）。
1263 ## 1. 项目结构
1264 `````
1265 ${context.structure}
1266 ``````
1267 ## 2. README 内容
1268 ${context.readme}
1269 ## 3. 依赖/配置信息
1270 ${context.packageJson}
1271 ## 4. 核心代码片段
1272 ${context.sourceSnippets.map(s => `--- ${s.path} ---\n${s.content}`).join('\n\n')}`;
1273 请输出 Markdown 格式，包含以下章节：
1274 1. # 软件概述（根据 README 和代码推断）
1275 2. # 功能列表（详细列出功能点）
1276 3. # 技术特点（架构、语言、关键技术）
1277 4. # 运行环境要求
1278 5. # 使用说明（如有）
1279 `;
1280     const baseUrl = config.baseUrl.replace(/\//, '');
1281     const response = await fetch(`$baseUrl}/chat/completions`,
1282 {
1283     method: 'POST',
1284     headers: {
1285         'Content-Type': 'application/json',
1286         'Authorization': `Bearer ${config.apiKey}`
1287     },
1288     body: JSON.stringify({
1289         model: config.model,
1290         messages: [
1291             { role: 'system', content: '你是一个专业的软件技术文档专家，擅长
编写软件说明书。'
1292         },
1293         { role: 'user', content: prompt }
1294     ],
1295     temperature: 0.7
1296 });
1297 });
1298 });
```

软著鉴别材料整理器软件 V1.0

```
1299     if (!response.ok) {  
1300         const errText = await response.text();
```

软著鉴别材料整理器软件 V1.0

```
1301      throw new Error(`AI 请求失败 (${response.status}): ${errText}`);
1302  );
1303  }
1304  const data = await response.json();
1305  if (!data.choices || data.choices.length === 0) {
1306    throw new Error('AI 返回结果为空');
1307  }
1308  return data.choices[0].message.content;
1309} catch (error) {
1310  console.error('AI Generation Error:', error);
1311  throw error;
1312}
1313);
1314 ipcMain.handle('ai:export-manual-docx', async (_event, options)
1315 => {
1316  const outputPath = path.join(options.outputDir, options.fileName);
1317  await writeManualDocx(outputPath, options);
1318  return outputPath;
1319});
1320 ipcMain.handle('ai:export-manual-pdf', async (_event, options) =
1321 > {
1322  const outputPath = path.join(options.outputDir, options.fileName);
1323  const tempDir = fs.mkdtempSync(path.join(os.tmpdir(), 'softreg-
1324 -manual-'));
1325  const tempDocx = path.join(tempDir, 'manual.docx');
1326  await writeManualDocx(tempDocx, options);
1327  writePdfFromDocx(tempDocx, outputPath);
1328  return outputPath;
1329});
1330 ipcMain.handle('save-file', async (_event, { content, defaultName
1331 e }) => {
1332  const result = await dialog.showSaveDialog(mainWindow, {
1333    defaultPath: defaultName,
1334    filters: [{ name: 'Markdown', extensions: ['md'] }]
1335  });
1336  if (!result.canceled && result.filePath) {
1337    fs.writeFileSync(result.filePath, content, 'utf8');
1338    return result.filePath;
1339  }
1340  return null;
1341});
1342);
1343 const { contextBridge, ipcRenderer } = require('electron');
1344 contextBridge.exposeInMainWorld('softreg', {
1345   selectProject: () => ipcRenderer.invoke('select-project'),
1346   selectOutputDir: () => ipcRenderer.invoke('select-output-dir')
1347 ,
1348   scanProject: (options) => ipcRenderer.invoke('scan-project', o
1349 ptions),
```

软著鉴别材料整理器软件 V1.0

```
1351 generateDocx: (options) => ipcRenderer.invoke('generate-docx',  
1352   options),  
1353   generatePdf: (options) => ipcRenderer.invoke('generate-pdf', o  
1354   ptions),  
1355   setExportPages: (mode) => ipcRenderer.invoke('set-export-pages  
1356   ', mode),  
1357   initScanState: () => ipcRenderer.invoke('init-scan-state'),  
1358   ensureOutputDir: (outputDir) => ipcRenderer.invoke('ensure-out  
1359   put-dir', outputDir),  
1360   openExternal: (url) => ipcRenderer.invoke('open-external', url  
1361   ),  
1362   openOutputDir: (outputDir) => ipcRenderer.invoke('open-output-  
1363   dir', outputDir),  
1364   ai: {  
1365     saveConfig: (config) => ipcRenderer.invoke('ai:save-config',  
1366       config),  
1367     getConfig: () => ipcRenderer.invoke('ai:get-config'),  
1368     generateManual: (projectPath, config, screenshots) => ipcRen  
1369     derer.invoke('ai:generate-manual', { projectPath, config, screen  
1370       shots }),  
1371     exportManualDocx: (options) => ipcRenderer.invoke('ai:export-  
1372       -manual-docx', options),  
1373     exportManualPdf: (options) => ipcRenderer.invoke('ai:export-  
1374       -manual-pdf', options)  
1375   },  
1376   selectAiScreenshots: () => ipcRenderer.invoke('select-ai-scree  
1377   nshots'),  
1378   saveFile: (content, defaultName) => ipcRenderer.invoke('save-f  
1379   ile', { content, defaultName })  
1380 });  
1381 const projectPathInput = document.getElementById('projectPath');  
1382 const outputDirInput = document.getElementById('outputDir');  
1383 const softwareNameInput = document.getElementById('softwareName'  
1384 );  
1385 const softwareVersionInput = document.getElementById('softwareVe  
1386 rsion');  
1387 const statFiles = document.getElementById('statFiles');  
1388 const statLines = document.getElementById('statLines');  
1389 const statPages = document.getElementById('statPages');  
1390 const statExport = document.getElementById('statExport');  
1391 const statExportLines = document.getElementById('statExportLines'  
1392 );  
1393 const statusEl = document.getElementById('status');  
1394 const btnSelectProject = document.getElementById('btnSelectProje  
1395 ct');  
1396 const btnSelectOutput = document.getElementById('btnSelectOutput  
1397 ');  
1398 const btnScan = document.getElementById('btnScan');  
1399 const btnExportDocx = document.getElementById('btnExportDocx');  
1400 const btnExportPdf = document.getElementById('btnExportPdf');
```

软著鉴别材料整理器软件 V1.0

```
1401 const btnExportBoth = document.getElementById('btnExportBoth');
1402 const lightRed = document.querySelector('.light.red');
1403 const lightAmber = document.querySelector('.light.amber');
1404 const lightGreen = document.querySelector('.light.green');
1405 function updateLights(state) {
1406     lightRed.classList.remove('active');
1407     lightAmber.classList.remove('active');
1408     lightGreen.classList.remove('active');
1409     switch (state) {
1410         case 'idle':
1411             lightRed.classList.add('active');
1412             break;
1413         case 'busy':
1414             lightAmber.classList.add('active');
1415             break;
1416         case 'success':
1417             lightGreen.classList.add('active');
1418             break;
1419         case 'error':
1420             lightRed.classList.add('active');
1421             break;
1422     }
1423 }
1424 function formatNum(num, length) {
1425     return String(num).padStart(length, '0');
1426 }
1427 function getSoftwareType() {
1428     const selected = document.querySelector('input[name="softwareT
1429 ype"]':checked');
1430     return selected ? selected.value : 'software';
1431 }
1432 function buildHeaderText() {
1433     const name = softwareNameInput.value.trim();
1434     const version = softwareVersionInput.value.trim();
1435     const type = getSoftwareType() === 'game' ? '游戏软件' : '软件';
1436     if (!name || !version) return '';
1437     return `${name}${type}V${version}`;
1438 }
1439 function setStatus(message) {
1440     statusEl.textContent = message.toUpperCase();
1441     console.log(`[系统] ${message}`);
1442 }
1443 function validateInputs() {
1444     if (!projectIdInput.value) {
1445         setStatus('错误：缺失项目路径');
1446         updateLights('error');
1447         return false;
1448     }
1449     if (!softwareNameInput.value.trim()) {
1450         setStatus('错误：需填写软件名称');
```

软著鉴别材料整理器软件 V1.0

```
1451     updateLights('error');
1452     return false;
1453 }
1454 if (!softwareVersionInput.value.trim()) {
1455     setStatus('错误：需填写版本号');
1456     updateLights('error');
1457     return false;
1458 }
1459 if (!outputDirInput.value) {
1460     setStatus('错误：缺失输出路径');
1461     updateLights('error');
1462     return false;
1463 }
1464 return true;
1465 }
1466 async function handleSelectProject() {
1467     const selected = await window.softreg.selectProject();
1468     if (selected) projectPathInput.value = selected;
1469 }
1470 async function handleSelectOutput() {
1471     const selected = await window.softreg.selectOutputDir();
1472     if (selected) outputDirInput.value = selected;
1473 }
1474 const scanOverlay = document.getElementById('scanOverlay');
1475 let audioCtx = null;
1476 function playScanSound() {
1477     if (!audioCtx) audioCtx = new (window.AudioContext || window.webkitAudioContext)();
1478     if (audioCtx.state === 'suspended') audioCtx.resume();
1479     const now = audioCtx.currentTime;
1480     const duration = 2.5; // 与填充动画时长一致
1481     const osc1 = audioCtx.createOscillator();
1482     const osc2 = audioCtx.createOscillator(); // 副振荡器（正弦波 - 提供底蕴）
1483     const filter = audioCtx.createBiquadFilter();
1484     const gainNode = audioCtx.createGain();
1485     osc1.type = 'sawtooth';
1486     osc1.frequency.setValueAtTime(80, now);
1487     osc1.frequency.exponentialRampToValueAtTime(450, now + duration);
1488     osc2.type = 'sine';
1489     osc2.frequency.setValueAtTime(40, now);
1490     osc2.frequency.exponentialRampToValueAtTime(120, now + duration);
1491     filter.type = 'lowpass';
1492     filter.Q.value = 15; // 高共振
1493     filter.frequency.setValueAtTime(200, now);
1494     filter.frequency.exponentialRampToValueAtTime(3000, now + duration);
1495     gainNode.gain.setValueAtTime(0, now);
1496     gainNode.gain.linearRampToValueAtTime(0.08, now + 0.1); // 快速开
```

软著鉴别材料整理器软件 V1.0

```
1501  启
1502      gainNode.gain.exponentialRampToValueAtTime(0.001, now + duration);
1503  on); // 逐渐消失
1504  osc1.connect(filter);
1505  osc2.connect(filter);
1506  filter.connect(gainNode);
1507  gainNode.connect(audioCtx.destination);
1508  osc1.start(now);
1509  osc2.start(now);
1510  osc1.stop(now + duration);
1511  osc2.stop(now + duration);
1512 }
1513 function startButtonEffect(btn) {
1514     btn.classList.add('processing');
1515     playScanSound();
1516 }
1517 function stopButtonEffect(btn) {
1518     btn.classList.remove('processing');
1519 }
1520 async function handleScan() {
1521     if (!projectPathInput.value) {
1522         setStatus('错误：请先选择项目目录');
1523         updateLights('error');
1524         return;
1525     }
1526     const headerText = buildHeaderText();
1527     if (!headerText) {
1528         setStatus('错误：名称和版本号缺失');
1529         updateLights('error');
1530         return;
1531     }
1532     try {
1533         setStatus('状态：正在扫描项目文件...');
1534         updateLights('busy');
1535         startButtonEffect(btnScan);
1536         await window.softreg.initScanState();
1537         const result = await window.softreg.scanProject({
1538             projectPath: projectPathInput.value,
1539             headerText
1540         });
1541         await new Promise(resolve => setTimeout(resolve, 2500));
1542         statFiles.textContent = formatNum(result.fileCount, 4);
1543         statLines.textContent = formatNum(result.effectiveLines, 6);
1544         statPages.textContent = formatNum(result.totalPages, 3);
1545         statExport.textContent = formatNum(result.selectedPages, 3);
1546         statExportLines.textContent = formatNum(result.selectedPages
1547             * 50, 4);
1548         const truncationNote = result.totalPages > 60 ? ' (已截断至 3000
1549 行)' : '';
1550         setStatus(`状态：扫描完成${truncationNote}. 准备就绪.`);
```

软著鉴别材料整理器软件 V1.0

```
1551     updateLights('success');
1552 } catch (err) {
1553     setStatus(`错误: 扫描失败 // ${String(err.message || err).toUpperCase()}`);
1554 }
1555     updateLights('error');
1556 } finally {
1557     stopButtonEffect(btnScan);
1558 }
1559 }
1560 function buildFileName(suffix) {
1561     const headerText = buildHeaderText();
1562     return `${headerText}_${suffix}`;
1563 }
1564 async function exportDocx() {
1565     if (!validateInputs()) return;
1566     try {
1567         setStatus('状态: 正在生成 DOCX 数据流...');
1568         updateLights('busy');
1569         startButtonEffect(btnExportDocx);
1570         await window.softreg.ensureOutputDir(outputDirInput.value);
1571         const fileName = buildFileName('程序鉴别材料.docx');
1572         const outputPath = await window.softreg.generateDocx({
1573             outputDir: outputDirInput.value,
1574             fileName
1575         });
1576         await new Promise(resolve => setTimeout(resolve, 2500));
1577         setStatus(`成功: DOCX 已保存至磁盘`);
1578         updateLights('success');
1579     } catch (err) {
1580         setStatus(`错误: DOCX 生成失败 // ${String(err.message || err).toUpperCase()}`);
1581     }
1582     updateLights('error');
1583 } finally {
1584     stopButtonEffect(btnExportDocx);
1585 }
1586 }
1587 async function exportPdf() {
1588     if (!validateInputs()) return;
1589     try {
1590         setStatus('状态: 正在导出 PDF 缓冲区...');
1591         updateLights('busy');
1592         startButtonEffect(btnExportPdf);
1593         await window.softreg.ensureOutputDir(outputDirInput.value);
1594         const fileName = buildFileName('程序鉴别材料.pdf');
1595         const outputPath = await window.softreg.generatePdf({
1596             outputDir: outputDirInput.value,
1597             fileName
1598         });
1599         await new Promise(resolve => setTimeout(resolve, 2500));
1600         setStatus(`成功: PDF 导出完成`);
```

软著鉴别材料整理器软件 V1.0

```
1601     updateLights('success');
1602 } catch (err) {
1603     setStatus(`错误: PDF 导出失败 // ${String(err.message || err).toUpperCase()});
1604 }
1605     updateLights('error');
1606 } finally {
1607     stopButtonEffect(btnExportPdf);
1608 }
1609 }
1610 async function exportBoth() {
1611     if (!validateInputs()) return;
1612     await exportDocx();
1613     await exportPdf();
1614 }
1615 updateLights('idle');
1616 btnSelectProject.addEventListener('click', handleSelectProject);
1617 btnSelectOutput.addEventListener('click', handleSelectOutput);
1618 btnScan.addEventListener('click', handleScan);
1619 btnExportDocx.addEventListener('click', exportDocx);
1620 btnExportPdf.addEventListener('click', exportPdf);
1621 btnExportBoth.addEventListener('click', exportBoth);
1622 document.getElementById('linkBilibili').addEventListener('click',
1623 , (e) => {
1624     e.preventDefault();
1625     window.softreg.openExternal('https://space.bilibili.com/309927
1626 2');
1627 });
1628 document.getElementById('linkGithub').addEventListener('click',
1629 (e) => {
1630     e.preventDefault();
1631     window.softreg.openExternal('https://github.com/itxys/softwork
1632 -code-organizer');
1633 });
1634 document.getElementById('linkCopyright').addEventListener('click
1635 ', (e) => {
1636     e.preventDefault();
1637     window.softreg.openExternal('https://www.ccopyright.com.cn/');
1638 });
1639 const btnAiConfig = document.getElementById('btnAiConfig');
1640 const btnAiManual = document.getElementById('btnAiManual');
1641 const btnAiUploadScreenshots = document.getElementById('btnAiUp
1642 loadScreenshots');
1643 const aiScreenshotList = document.getElementById('aiScreenshotLi
1644 st');
1645 const btnExportAiDocx = document.getElementById('btnExportAiDocx
1646 ');
1647 const btnExportAiPdf = document.getElementById('btnExportAiPdf')
1648 ;
1649 const btnExportAiBoth = document.getElementById('btnExportAiBoth
1650 ');
```

软著鉴别材料整理器软件 V1.0

```
1651 const btnOpenOutputDir = document.getElementById('btnOpenOutputD  
1652 ir');  
1653 const btnOpenOutputDirAi = document.getElementById('btnOpenOutpu  
1654 tDirAi');  
1655 const aiConfigModal = document.getElementById('aiConfigModal');  
1656 const aiResultModal = document.getElementById('aiResultModal');  
1657 const aiProviderPreset = document.getElementById('aiProviderPres  
1658 et');  
1659 const aiBaseUrl = document.getElementById('aiBaseUrl');  
1660 const aiApiKey = document.getElementById('aiApiKey');  
1661 const aiModelPreset = document.getElementById('aiModelPreset');  
1662 const aiModel = document.getElementById('aiModel');  
1663 const btnTestAiConnection = document.getElementById('btnTestAiCo  
1664 nnection');  
1665 const aiConnectionStatus = document.getElementById('aiConnection  
1666 Status');  
1667 const aiTestResult = document.getElementById('aiTestResult');  
1668 const aiConfigToast = document.getElementById('aiConfigToast');  
1669 const aiToastTitle = document.getElementById('aiToastTitle');  
1670 const aiToastMsg = document.getElementById('aiToastMsg');  
1671 const btnSaveAiConfig = document.getElementById('btnSaveAiConfig  
1672 ');  
1673 function showModalToast(title, msg, type = 'info') {  
1674     if (!aiConfigToast) return;  
1675     aiToastTitle.textContent = title;  
1676     aiToastMsg.textContent = msg;  
1677     aiConfigToast.className = `modal-toast active ${type}`;  
1678     setTimeout(() => {  
1679         aiConfigToast.classList.remove('active');  
1680     }, 3000);  
1681 }  
1682 const btnCloseAiConfig = document.getElementById('btnCloseAiConf  
1683 ig');  
1684 const btnSaveAiManual = document.getElementById('btnSaveAiManual  
1685 ');  
1686 const btnCloseAiResult = document.getElementById('btnCloseAiResu  
1687 lt');  
1688 const aiResultText = document.getElementById('aiResultText');  
1689 let aiManualMarkdown = '';  
1690 let aiScreenshots = [];  
1691 let aiGeneratingManual = false;  
1692 const AI_PROVIDER_PRESETS = [  
1693     {  
1694         id: 'custom',  
1695         name: '自定义（兼容 OpenAI 接口）',  
1696         baseUrl: '',  
1697         models: []  
1698     },  
1699     {  
1700         id: 'openai',
```

软著鉴别材料整理器软件 V1.0

```
1701     name: 'OpenAI 官方',
1702     baseUrl: 'https://api.openai.com/v1',
1703     models: ['gpt-5.2', 'gpt-5.1', 'o3-mini', 'o3-pro', 'gpt-4.1
1704 ', 'gpt-4o']
1705 },
1706 {
1707     id: 'siliconflow',
1708     name: '硅基流动 (SiliconFlow) ',
1709     baseUrl: 'https://api.siliconflow.cn/v1',
1710     models: [
1711         'deepseek-ai/DeepSeek-R1',
1712         'deepseek-ai/DeepSeek-V3',
1713         'Qwen/Qwen2.5-72B-Instruct',
1714         'Qwen/Qwen2.5-Coder-32B-Instruct',
1715         'TeleAI/TeleChat2'
1716     ]
1717 },
1718 {
1719     id: 'deepseek',
1720     name: 'DeepSeek 官方',
1721     baseUrl: 'https://api.deepseek.com/v1',
1722     models: ['deepseek-reasoner', 'deepseek-chat']
1723 },
1724 {
1725     id: 'openrouter',
1726     name: 'OpenRouter',
1727     baseUrl: 'https://openrouter.ai/api/v1',
1728     models: [
1729         'anthropic/clause-4.5-sonnet',
1730         'anthropic/clause-4.5-opus',
1731         'google/gemini-3-pro',
1732         'deepseek/deepseek-r1',
1733         'meta-llama/llama-4-70b-instruct'
1734     ]
1735 },
1736 {
1737     id: 'qwen',
1738     name: '阿里云 DashScope (Qwen) ',
1739     baseUrl: 'https://dashscope.aliyuncs.com/compatible-mode/v1'
1740 ,
1741     models: ['qwen-max', 'qwen-plus', 'qwen-turbo', 'qwen-long']
1742 },
1743 {
1744     id: 'zhipu',
1745     name: '智谱 ZhipuAI (GLM) ',
1746     baseUrl: 'https://open.bigmodel.cn/api/paas/v4',
1747     models: ['glm-4-plus', 'glm-4-air', 'glm-4-flash', 'glm-4-lo
1748 ng']
1749 },
1750 {
```

软著鉴别材料整理器软件 V1.0

```
1751     id: 'moonshot',
1752     name: 'Moonshot AI (Kimi)',
1753     baseUrl: 'https://api.moonshot.cn/v1',
1754     models: ['moonshot-v1-128k', 'moonshot-v1-32k', 'moonshot-v1
1755 -8k']
1756   },
1757   {
1758     id: 'minimax',
1759     name: 'MiniMax',
1760     baseUrl: 'https://api.minimax.chat/v1',
1761     models: ['abab7-chat-preview', 'abab6.5s-chat', 'abab6.5-cha
1762 t']
1763   },
1764   {
1765     id: 'doubao',
1766     name: '字节跳动 Doubao (火山引擎)',
1767     baseUrl: 'https://ark.cn-beijing.volces.com/api/v3',
1768     models: ['doubao-pro-32k', 'doubao-lite-32k', 'doubao-pro-12
1769 8k']
1770   },
1771   {
1772     id: 'modelscope',
1773     name: 'ModelScope (魔搭)',
1774     baseUrl: 'https://api-inference.modelscope.cn/v1',
1775     models: ['Qwen/Qwen2.5-72B-Instruct', 'Qwen/Qwen2.5-7B-Instr
1776 uct']
1777   },
1778   {
1779     id: 'claude',
1780     name: 'Claude (Official/Proxy)',
1781     baseUrl: 'https://api.anthropic.com/v1',
1782     models: ['claude-opus-4-5', 'claude-3-5-sonnet-20241022', 'c
1783 laude-3-5-haiku-20241022']
1784   },
1785   {
1786     id: 'gemini',
1787     name: 'Google Gemini',
1788     baseUrl: 'https://generativelanguage.googleapis.com/v1beta/o
1789 penai',
1790     models: ['gemini-3-pro-preview', 'gemini-3-flash-preview', 'g
1791 emini-2.0-flash-exp', 'gemini-1.5-pro', 'gemini-1.5-flash']
1792   }
1793 ];
1794 let aiPresetsInitialized = false;
1795 function getProviderPresetById(id) {
1796   return AI_PROVIDER_PRESETS.find(p => p.id === id) || AI_PROVID
1797 ER_PRESETS[0];
1798 }
1799 function findProviderPresetIdByBaseUrl(baseUrl) {
1800   const normalized = String(baseUrl || '').trim().replace(/\//,
```

软著鉴别材料整理器软件 V1.0

```
1801    '');
1802    const match = AI_PROVIDER_PRESETS.find(p => p.baseUrl && p.bas
1803 eUrl.replace(/\/$/, '') === normalized);
1804    return match ? match.id : 'custom';
1805 }
1806 function setSelectOptions(selectEl, options) {
1807     selectEl.innerHTML = '';
1808     for (const opt of options) {
1809         const optionEl = document.createElement('option');
1810         optionEl.value = opt.value;
1811         optionEl.textContent = opt.label;
1812         selectEl.appendChild(optionEl);
1813     }
1814 }
1815 function populateProviderPresetOptions() {
1816     setSelectOptions(
1817         aiProviderPreset,
1818         AI_PROVIDER_PRESETS.map(preset => ({
1819             value: preset.id,
1820             label: preset.name
1821         }))
1822     );
1823 }
1824 function populateModelPresetOptions(providerId) {
1825     const provider = getProviderPresetById(providerId);
1826     const modelOptions = [{ value: 'custom', label: '自定义' }].conca
1827 t(
1828     provider.models.map(m => ({ value: m, label: m }))
1829 );
1830     setSelectOptions(aiModelPreset, modelOptions);
1831 }
1832 function syncProviderPresetFromBaseUrl() {
1833     const providerId = findProviderPresetIdByBaseUrl(aiBaseUrl.val
1834 ue);
1835     if (aiProviderPreset.value !== providerId) {
1836         aiProviderPreset.value = providerId;
1837         populateModelPresetOptions(providerId);
1838         syncModelPresetFrommodelName();
1839     }
1840 }
1841 function syncModelPresetFrommodelName() {
1842     const modelName = String(aiModel.value || '').trim();
1843     const optionExists = Array.from(aiModelPreset.options).some(o
1844 => o.value === modelName);
1845     aiModelPreset.value = optionExists ? modelName : 'custom';
1846 }
1847 function applyProviderPreset(providerId) {
1848     const provider = getProviderPresetById(providerId);
1849     populateModelPresetOptions(providerId);
1850     if (provider.baseUrl) {
```

软著鉴别材料整理器软件 V1.0

```
1851     aiBaseUrl.value = provider.baseUrl;
1852   }
1853   if (!String(aiModel.value || '').trim() && provider.models.length > 0) {
1854     aiModel.value = provider.models[0];
1855   }
1856 }
1857 syncModelPresetFrommodelName();
1858 }
1859 function initAiPresetControls() {
1860   if (aiPresetsInitialized) return;
1861   aiPresetsInitialized = true;
1862   populateProviderPresetOptions();
1863   populateModelPresetOptions('custom');
1864   aiProviderPreset.addEventListener('change', () => {
1865     applyProviderPreset(aiProviderPreset.value);
1866   });
1867   aiModelPreset.addEventListener('change', () => {
1868     if (aiModelPreset.value === 'custom') return;
1869     aiModel.value = aiModelPreset.value;
1870   });
1871   aiBaseUrl.addEventListener('input', () => {
1872     syncProviderPresetFromBaseUrl();
1873   });
1874   aiModel.addEventListener('input', () => {
1875     syncModelPresetFrommodelName();
1876   });
1877 }
1878 function renderAiScreenshotList() {
1879   if (!aiScreenshotList) return;
1880   aiScreenshotList.innerHTML = '';
1881   if (!aiScreenshots.length) return;
1882   let dragIndex = null;
1883   aiScreenshots.forEach((item, index) => {
1884     const row = document.createElement('div');
1885     row.className = 'screenshot-item';
1886     row.draggable = true;
1887     row.dataset.index = String(index);
1888     const img = document.createElement('img');
1889     img.className = 'screenshot-thumb';
1890     img.alt = item.name || `截图 ${index + 1}`;
1891     img.src = item.url;
1892     const name = document.createElement('div');
1893     name.className = 'screenshot-name';
1894     name.textContent = `${index + 1}. ${item.name || item.path} |`;
1895     | ''}`);
1896     const noteInput = document.createElement('input');
1897     noteInput.type = 'text';
1898     noteInput.className = 'screenshot-note';
1899     noteInput.placeholder = '备注: 例如 首页截图 / 目录截图';
1900     noteInput.value = String(item.note || '');
```

软著鉴别材料整理器软件 V1.0

```
1901     noteInput.draggable = false;
1902     noteInput.addEventListener('input', () => {
1903         const idx = Number(row.dataset.index);
1904         if (Number.isNaN(idx) || !aiScreenshots[idx]) return;
1905         aiScreenshots[idx].note = noteInput.value;
1906     });
1907     row.appendChild(img);
1908     row.appendChild(name);
1909     row.appendChild(noteInput);
1910     row.addEventListener('dragstart', (e) => {
1911         if (e.target && e.target.closest && e.target.closest('input')) {
1912             e.preventDefault();
1913             return;
1914         }
1915         dragIndex = index;
1916         row.classList.add('dragging');
1917         try {
1918             e.dataTransfer.setData('text/plain', String(index));
1919             e.dataTransfer.effectAllowed = 'move';
1920             } catch (_err) {}
1921         });
1922         row.addEventListener('dragend', () => {
1923             row.classList.remove('dragging');
1924         });
1925         row.addEventListener('dragover', (e) => {
1926             e.preventDefault();
1927             try {
1928                 e.dataTransfer.dropEffect = 'move';
1929                 } catch (_err) {}
1930             });
1931             row.addEventListener('drop', (e) => {
1932                 e.preventDefault();
1933                 let fromIndex = dragIndex;
1934                 const toIndex = Number(row.dataset.index);
1935                 try {
1936                     const raw = e.dataTransfer.getData('text/plain');
1937                     if (raw !== '') fromIndex = Number(raw);
1938                     } catch (_err) {}
1939                     if (Number.isNaN(fromIndex) || Number.isNaN(toIndex) || fromIndex === toIndex) return;
1940                     const next = aiScreenshots.slice();
1941                     const [moved] = next.splice(fromIndex, 1);
1942                     next.splice(toIndex, 0, moved);
1943                     aiScreenshots = next;
1944                     renderAiScreenshotList();
1945                 });
1946                 aiScreenshotList.appendChild(row);
1947             });
1948         });
1949     });
1950 }
```

软著鉴别材料整理器软件 V1.0

```
1951  async function loadAiConfig() {
1952      initAiPresetControls();
1953      const config = await window.softreg.ai.getConfig();
1954      aiBaseUrl.value = config.baseUrl;
1955      aiApiKey.value = config.apiKey;
1956      aiModel.value = config.model;
1957      const providerId = config.providerId || findProviderPresetIdBy
1958      baseUrl(config.baseUrl);
1959      aiProviderPreset.value = providerId;
1960      populateModelPresetOptions(providerId);
1961      syncModelPresetFrommodelName();
1962  }
1963  btnAiConfig.addEventListener('click', async () => {
1964      initAiPresetControls();
1965      await loadAiConfig();
1966      if (aiTestResult) aiTestResult.textContent = '';
1967      if (aiConnectionStatus) aiConnectionStatus.className = 'status
1968      -dot-outer';
1969      aiConfigModal.classList.remove('hidden');
1970  });
1971  btnCloseAiConfig.addEventListener('click', () => {
1972      aiConfigModal.classList.add('hidden');
1973      if (aiTestResult) aiTestResult.textContent = '';
1974      if (aiConnectionStatus) aiConnectionStatus.className = 'status
1975      -dot-outer';
1976  });
1977  if (btnTestAiConnection) {
1978      btnTestAiConnection.addEventListener('click', async () => {
1979          const baseUrl = aiBaseUrl.value.trim();
1980          const apiKey = aiApiKey.value.trim();
1981          const model = aiModel.value.trim();
1982          if (!baseUrl || !apiKey || !model) {
1983              if (aiTestResult) {
1984                  aiTestResult.textContent = '请先填写完整的配置信息';
1985                  aiTestResult.className = 'test-result-text error';
1986              }
1987              showModalToast('配置不完整', '请填写 API 地址、密钥和模型名称',
1988              'error');
1989          }
1990          if (aiConnectionStatus) {
1991              aiConnectionStatus.className = 'status-dot-outer';
1992              aiConnectionStatus.title = '正在测试连接... / Testing Connection
1993              ...';
1994          }
1995          if (aiTestResult) {
1996              aiTestResult.textContent = '正在连接服务器...';
1997              aiTestResult.className = 'test-result-text';
1998          }
1999          const originalText = btnTestAiConnection.textContent;
2000          btnTestAiConnection.disabled = true;
```

软著鉴别材料整理器软件 V1.0

```
2001     btnTestAiConnection.textContent = '测试中...';
2002     try {
2003         let cleanBaseUrl = baseUrl.replace(/\/+$/, '');
2004         const url = `${cleanBaseUrl}/chat/completions`;
2005         const controller = new AbortController();
2006         const timeoutId = setTimeout(() => controller.abort(), 150
2007 00); // 15s timeout
2008         const response = await fetch(url, {
2009             method: 'POST',
2010             headers: {
2011                 'Content-Type': 'application/json',
2012                 'Authorization': `Bearer ${apiKey}`
2013             },
2014             body: JSON.stringify({
2015                 model: model,
2016                 messages: [{ role: 'user', content: 'Hi' }],
2017                 max_tokens: 1
2018             }),
2019             signal: controller.signal
2020         });
2021         clearTimeout(timeoutId);
2022         if (response.ok) {
2023             if (aiConnectionStatus) {
2024                 aiConnectionStatus.classList.add('success');
2025                 aiConnectionStatus.title = '连接成功 / Connection Successf
2026 ul';
2027             }
2028             if (aiTestResult) {
2029                 aiTestResult.textContent = '连接测试成功！接口响应正常。';
2030                 aiTestResult.className = 'test-result-text success';
2031             }
2032             showModalToast('测试成功', 'AI 接口连接正常，响应成功！',
2033 'success');
2034         } else {
2035             const errText = await response.text();
2036             console.error('AI Test Error:', response.status, errText
2037 );
2038             if (aiConnectionStatus) {
2039                 aiConnectionStatus.classList.add('error');
2040                 aiConnectionStatus.title = `连接失败 / Connection Failed:
2041 ${response.status}`;
2042             }
2043             if (aiTestResult) {
2044                 aiTestResult.textContent = `失败: ${response.status} - $2045 {errText.substring(0, 30)}...`;
2046                 aiTestResult.className = 'test-result-text error';
2047             }
2048             showModalToast('测试失败', `状态码: ${response.status}\n原因:
2049 ${errText.substring(0, 50)}`, 'error');
2050         }
2051     }
```

软著鉴别材料整理器软件 V1.0

2050 } catch (error) {

软著鉴别材料整理器软件 V1.0

```
2051     console.error('AI Test Exception:', error);
2052     if (aiConnectionStatus) {
2053         aiConnectionStatus.classList.add('error');
2054         aiConnectionStatus.title = `连接错误 / Connection Error`;
2055     }
2056     if (aiTestResult) {
2057         aiTestResult.textContent = `错误: ${error.message}`;
2058         aiTestResult.className = 'test-result-text error';
2059     }
2060     showModalToast('连接错误', error.message, 'error');
2061 } finally {
2062     btnTestAiConnection.disabled = false;
2063     btnTestAiConnection.textContent = originalText;
2064 }
2065 });
2066 }
2067 btnSaveAiConfig.addEventListener('click', async () => {
2068     const config = {
2069         baseUrl: aiBaseUrl.value.trim(),
2070         apiKey: aiApiKey.value.trim(),
2071         model: aiModel.value.trim()
2072     };
2073     config.providerId = aiProviderPreset.value;
2074     await window.softreg.ai.saveConfig(config);
2075     aiConfigModal.classList.add('hidden');
2076     if (aiTestResult) aiTestResult.textContent = '';
2077     if (aiConnectionStatus) aiConnectionStatus.className = 'status-dot-outer';
2078     setStatus('状态: AI 配置已保存');
2079 });
2080
2081 async function generateAiManualMarkdown() {
2082     if (aiGeneratingManual) return '';
2083     if (!projectPathInput.value) {
2084         setStatus('错误: 请先选择项目目录');
2085         updateLights('error');
2086         return '';
2087     }
2088     const config = await window.softreg.ai.getConfig();
2089     if (!config.apiKey) {
2090         setStatus('错误: 未配置 AI API KEY');
2091         updateLights('error');
2092         await loadAiConfig();
2093         aiConfigModal.classList.remove('hidden');
2094         return '';
2095     }
2096     try {
2097         aiGeneratingManual = true;
2098         setStatus('状态: AI 正在分析项目并撰写说明书... ');
2099         updateLights('busy');
2100         startButtonEffect(btnAiManual);
```

软著鉴别材料整理器软件 V1.0

```
2101     aiResultText.value = '正在分析项目结构...\n正在读取关键文件...\n正在  
调用 AI 生成（可  
2102 能需要 30-60 秒）...\n请稍候...';  
2103     aiResultModal.classList.remove('hidden');  
2104     const manualMarkdown = await window.softreg.ai.generateManua  
2105 l(  
2106     projectPathInput.value,  
2107     config,  
2108     aiScreenshots.map(s => ({  
2109         path: s.path,  
2110         name: s.name,  
2111         note: String(s.note || '').trim()  
2112     }));  
2113 );  
2114     aiManualMarkdown = manualMarkdown;  
2115     aiResultText.value = manualMarkdown;  
2116     setStatus('状态: 说明书生成完成');  
2117     updateLights('success');  
2118     return manualMarkdown;  
2119 } catch (err) {  
2120     setStatus(`错误: 生成失败 // ${String(err.message || err).toUpperCase()}`);  
2121     updateLights('error');  
2122     aiResultText.value = `生成失败:\n${String(err.message || err)}`;  
2123     return '';  
2124 } finally {  
2125     aiGeneratingManual = false;  
2126     stopButtonEffect(btnAiManual);  
2127 }  
2128 }  
2129 }  
2130 btnAiManual.addEventListener('click', async () => {  
2131     await generateAiManualMarkdown();  
2132 });  
2133 btnSaveAiManual.addEventListener('click', async () => {  
2134     await generateAiManualMarkdown();  
2135 });  
2136 btnCloseAiResult.addEventListener('click', () => {  
2137     aiManualMarkdown = aiResultText.value;  
2138     aiResultModal.classList.add('hidden');  
2139     setStatus('状态: 已保存生成结果');  
2140 });  
2141 async function ensureAiManualReady() {  
2142     if (String(aiManualMarkdown || '').trim()) return aiManualMark  
down;  
2143     return await generateAiManualMarkdown();  
2144 }  
2145 }  
2146 async function exportAiDocx() {  
2147     if (!validateInputs()) return;  
2148     const markdown = await ensureAiManualReady();  
2149     if (!String(markdown || '').trim()) return;  
2150     try {
```

软著鉴别材料整理器软件 V1.0

```
2151     setStatus('状态：正在导出说明书 DOCX...');
2152     updateLights('busy');
2153     startButtonEffect(btnExportAiDocx);
2154     await window.softreg.ensureOutputDir(outputDirInput.value);
2155     const headerText = buildHeaderText();
2156     const fileName = `${headerText} || '项目'}_软件说明书.docx`;
2157     await window.softreg.ai.exportManualDocx({
2158         outputDir: outputDirInput.value,
2159         fileName,
2160         markdown,
2161         screenshots: aiScreenshots.map(s => ({
2162             path: s.path,
2163             name: s.name,
2164             note: String(s.note || '').trim()
2165         })),
2166         softwareName: softwareNameInput.value.trim(),
2167         softwareVersion: softwareVersionInput.value.trim(),
2168         headerText
2169     });
2170     await new Promise(resolve => setTimeout(resolve, 800));
2171     setStatus('成功：说明书 DOCX 已保存至磁盘');
2172     updateLights('success');
2173 } catch (err) {
2174     setStatus(`错误：说明书 DOCX 导出失败 // ${String(err.message || err)}
2175 .toUpperCase()}`);
2176     updateLights('error');
2177 } finally {
2178     stopButtonEffect(btnExportAiDocx);
2179 }
2180 }
2181 async function exportAiPdf() {
2182     if (!validateInputs()) return;
2183     const markdown = await ensureAiManualReady();
2184     if (!String(markdown || '').trim()) return;
2185     try {
2186         setStatus('状态：正在导出说明书 PDF...');
2187         updateLights('busy');
2188         startButtonEffect(btnExportAiPdf);
2189         await window.softreg.ensureOutputDir(outputDirInput.value);
2190         const headerText = buildHeaderText();
2191         const fileName = `${headerText} || '项目'}_软件说明书.pdf`;
2192         await window.softreg.ai.exportManualPdf({
2193             outputDir: outputDirInput.value,
2194             fileName,
2195             markdown,
2196             screenshots: aiScreenshots.map(s => ({
2197                 path: s.path,
2198                 name: s.name,
2199                 note: String(s.note || '').trim()
2200             })),
2201         });
2202     } catch (err) {
2203         setStatus(`错误：说明书 PDF 导出失败 // ${String(err.message || err)}
2204 .toUpperCase()}`);
2205         updateLights('error');
2206     } finally {
2207         stopButtonEffect(btnExportAiPdf);
2208     }
2209 }
```

软著鉴别材料整理器软件 V1.0

```
2201     softwareName: softwareNameInput.value.trim(),
2202     softwareVersion: softwareVersionInput.value.trim(),
2203     headerText
2204   });
2205   await new Promise(resolve => setTimeout(resolve, 800));
2206   setStatus('成功：说明书 PDF 已保存至磁盘');
2207   updateLights('success');
2208 } catch (err) {
2209   setStatus(`错误：说明书 PDF 导出失败 // ${String(err.message || err)}.
2210   toUpperCase()}`);
2211   updateLights('error');
2212 } finally {
2213   stopButtonEffect(btnExportAiPdf);
2214 }
2215 }
2216 async function exportAiBoth() {
2217   if (!validateInputs()) return;
2218   await exportAiDocx();
2219   await exportAiPdf();
2220 }
2221 btnExportAiDocx.addEventListener('click', exportAiDocx);
2222 btnExportAiPdf.addEventListener('click', exportAiPdf);
2223 btnExportAiBoth.addEventListener('click', exportAiBoth);
2224 function openSelectedOutputDir() {
2225   const out = String(outputDirInput.value || '').trim();
2226   if (!out) {
2227     setStatus('错误：缺失输出路径');
2228     updateLights('error');
2229     return;
2230   }
2231   window.softreg.openOutputDir(out);
2232 }
2233 btnOpenOutputDir.addEventListener('click', openSelectedOutputDir);
2234 );
2235 btnOpenOutputDirAi.addEventListener('click', openSelectedOutputDir);
2236
2237 btnAiUploadScreenshots.addEventListener('click', async () => {
2238   const items = await window.softreg.selectAiScreenshots();
2239   if (!Array.isArray(items) || items.length === 0) return;
2240   const existing = new Set(aiScreenshots.map(s => s.path));
2241   const merged = aiScreenshots.slice();
2242   for (const it of items) {
2243     if (!it || !it.path || existing.has(it.path)) continue;
2244     merged.push({ ...it, note: '' });
2245     existing.add(it.path);
2246   }
2247   aiScreenshots = merged;
2248   renderAiScreenshotList();
2249 });
2250 window.addEventListener('DOMContentLoaded', () => {
```

软著鉴别材料整理器软件 V1.0

```
2251     aiConfigModal.classList.add('hidden');
2252     aiResultModal.classList.add('hidden');
2253     renderAiScreenshotList();
2254   });
2255   const fs = require('fs');
2256   const path = require('path');
2257   const EXCLUDED_DIRS = new Set([
2258     'node_modules', 'dist', 'build', 'out', '.git', 'vendor', 'bin',
2259     'obj',
2260     'target', '.idea', '.vscode', '.vs', '.svn', '.hg', '.turbo',
2261     '.next',
2262     '.cache', '.angular', '.gradle'
2263   ]);
2264   const SUPPORTED_EXTS = new Set([
2265     '.js', '.jsx', '.ts', '.tsx',
2266     '.py', '.java', '.cs', '.c', '.cpp', '.h', '.hpp',
2267     '.go', '.rs', '.php', '.swift', '.kt', '.json', '.md', '.txt'
2268   ]);
2269   function isTextFile(filePath) {
2270     const ext = path.extname(filePath).toLowerCase();
2271     return SUPPORTED_EXTS.has(ext) || ext === '';
2272   }
2273   function getProjectStructure(root, depth = 0, maxDepth = 3) {
2274     if (depth > maxDepth) return '';
2275     let output = '';
2276     let entries;
2277     try {
2278       entries = fs.readdirSync(root, { withFileTypes: true });
2279     } catch (e) {
2280       return '';
2281     }
2282     entries.sort((a, b) => {
2283       if (a.isDirectory() && !b.isDirectory()) return -1;
2284       if (!a.isDirectory() && b.isDirectory()) return 1;
2285       return a.name.localeCompare(b.name);
2286     });
2287     const indent = ' '.repeat(depth);
2288     for (const entry of entries) {
2289       if (entry.name.startsWith('.')) continue; // Skip dotfiles
2290       if (entry.isDirectory()) {
2291         if (EXCLUDED_DIRS.has(entry.name)) {
2292           output += `${indent}— ${entry.name}/\n`;
2293         } else {
2294           output += `${indent}— ${entry.name}/\n`;
2295           output += getProjectStructure(path.join(root, entry.name),
2296             depth + 1, maxDepth);
2297         }
2298       } else {
2299         output += `${indent}— ${entry.name}\n`;
2300       }
2301     }
2302   }
2303 }
```

软著鉴别材料整理器软件 V1.0

```
2301     }
2302     return output;
2303 }
2304 function getFileContent(filePath, maxLines = 100) {
2305     try {
2306         const content = fs.readFileSync(filePath, 'utf8');
2307         const lines = content.split('\n');
2308         if (lines.length > maxLines) {
2309             return lines.slice(0, maxLines).join('\n') + '\n... (trunc
2310 ated)';
2311         }
2312         return content;
2313     } catch (e) {
2314         return `Error reading file: ${e.message}`;
2315     }
2316 }
2317 function scanProjectForAI(projectPath) {
2318     const context = {
2319         structure: '',
2320         readme: '',
2321         packageJson: '',
2322         sourceSnippets: []
2323     };
2324     context.structure = getProjectStructure(projectPath);
2325     const readmeNames = ['README.md', 'readme.md', 'README.txt', 'r
eadme.txt'];
2326     for (const name of readmeNames) {
2327         const p = path.join(projectPath, name);
2328         if (fs.existsSync(p)) {
2329             context.readme = getFileContent(p, 200); // Read up to 200
2330             lines of README
2331             break;
2332         }
2333     }
2334     const configFiles = ['package.json', 'requirements.txt', 'pom.
xml', 'go.mod', 'Cargo.toml'];
2335     for (const name of configFiles) {
2336         const p = path.join(projectPath, name);
2337         if (fs.existsSync(p)) {
2338             context.packageJson += `--- ${name} ---\n${getFileContent(
2339             p, 100)}\n\n`;
2340         }
2341     }
2342     const snippets = [];
2343     let fileCount = 0;
2344     const MAX_FILES = 5;
2345     function walkAndPick(current) {
2346         if (fileCount >= MAX_FILES) return;
2347         let entries;
2348         try {
```

软著鉴别材料整理器软件 V1.0

```
2351     entries = fs.readdirSync(current, { withFileTypes: true })
2352 ;
2353     } catch { return; }
2354     for (const entry of entries) {
2355         if (fileCount >= MAX_FILES) return;
2356         const fullPath = path.join(current, entry.name);
2357         if (entry.isDirectory()) {
2358             if (!EXCLUDED_DIRS.has(entry.name) && !entry.name.startsWith('.'))
2359                 walkAndPick(fullPath);
2360             }
2361         } else {
2362             const ext = path.extname(entry.name).toLowerCase();
2363             if(['.json', '.md', '.txt', '.lock'].includes(ext)) continue;
2364             if (SUPPORTED_EXTS.has(ext)) {
2365                 snippets.push({
2366                     path: path.relative(projectPath, fullPath),
2367                     content: getFileContent(fullPath, 50) // First 50 lines
2368                 });
2369                 fileCount++;
2370             }
2371         }
2372     }
2373 }
2374 }
2375 }
2376 }
2377 const srcPath = path.join(projectPath, 'src');
2378 if (fs.existsSync(srcPath)) {
2379     walkAndPick(srcPath);
2380 }
2381 if (fileCount < MAX_FILES) {
2382     walkAndPick(projectPath);
2383 }
2384 context.sourceSnippets = snippets;
2385 return context;
2386 }
2387 module.exports = {
2388     scanProjectForAI
2389 };
2390 const fs = require('fs');
2391 const {
2392     AlignmentType,
2393     Document,
2394     Footer,
2395     Header,
2396     LineRuleType,
2397     PageNumber,
2398     NumberFormat,
2399     Paragraph,
2400     Packer,
```

软著鉴别材料整理器软件 V1.0

```
2401     TextRun,
2402     TabStopType,
2403     TabStopPosition
2404   } = require('docx');
2405   function lineParagraph(line, lineNumber, isFirstLineOfPage) {
2406     const text = line.length === 0 ? ' ' : line;
2407     const lineNumStr = String(lineNumber).padStart(5, ' ') + ' ';
2408     return new Paragraph({
2409       pageBreakBefore: isFirstLineOfPage,
2410       spacing: {
2411         before: 0,
2412         after: 0,
2413         line: 270,
2414         lineRule: LineRuleType.EXACT
2415       },
2416       children: [
2417         new TextRun({
2418           text: lineNumStr,
2419           font: 'Consolas',
2420           size: 21,
2421           color: '888888' // 浅灰色行号
2422         }),
2423         new TextRun({
2424           text,
2425           font: 'Consolas',
2426           size: 21,
2427           preserve: true
2428         })
2429       ]
2430     });
2431   }
2432   async function writeDocx(outputPath, pages, headerText) {
2433     const children = [];
2434     let globalLineNumber = 1;
2435     pages.forEach((pageLines, pageIndex) => {
2436       pageLines.forEach((line, lineIndex) => {
2437         const isFirstLineOfPage = (pageIndex > 0 && lineIndex ===
2438 0);
2439         children.push(lineParagraph(line, globalLineNumber, isFirstLineOfPage));
2440         globalLineNumber++;
2441       });
2442     });
2443   };
2444   const header = new Header({
2445     children: [
2446       new Paragraph({
2447         children: [new TextRun({ text: headerText, font: 'Microso
2448 ft YaHei', size: 24 })],
2449         alignment: AlignmentType.LEFT,
2450         spacing: { before: 0, after: 0 }
```

软著鉴别材料整理器软件 V1.0

```
2451         })
2452     ]
2453   });
2454   const footer = new Footer({
2455     children: [
2456       new Paragraph({
2457         alignment: AlignmentType.CENTER,
2458         children: [
2459           new TextRun({ text: '- 第 ', font: 'Microsoft YaHei', s
2460           ize: 20 }),
2461           new TextRun({
2462             children: [PageNumber.CURRENT]
2463           }),
2464           new TextRun({ text: ' 页 / 共 ', font: 'Microsoft YaHei'
2465           , size: 20 }),
2466           new TextRun({
2467             children: [PageNumber.TOTAL_PAGES]
2468           }),
2469           new TextRun({ text: ' 页 -', font: 'Microsoft YaHei', s
2470           ize: 20 })
2471         ],
2472         spacing: { before: 0, after: 0 }
2473       })
2474     ]
2475   });
2476   const doc = new Document({
2477     sections: [
2478       {
2479         properties: {
2480           page: {
2481             margin: {
2482               top: 1440,
2483               bottom: 1440,
2484               left: 1440, // 不再需要为 Word 行号预留空间，因为使用手动行号
2485               right: 1440,
2486               header: 720,
2487               footer: 720
2488             },
2489             pageNumbers: {
2490               start: 1,
2491               formatType: NumberFormat.DECIMAL
2492             }
2493           }
2494         },
2495         headers: { default: header, first: header },
2496         footers: { default: footer, first: footer },
2497         children
2498       }
2499     ]
2500   });

```

软著鉴别材料整理器软件 V1.0

```
2501     const buffer = await Packer.toBuffer(doc);
2502     fs.writeFileSync(outputPath, buffer);
2503 }
2504 module.exports = {
2505     writeDocx
2506 };
2507 const fs = require('fs');
2508 const { nativeImage } = require('electron');
2509 const path = require('path');
2510 const {
2511     AlignmentType,
2512     Document,
2513     Footer,
2514     Header,
2515     HeadingLevel,
2516     ImageRun,
2517     NumberFormat,
2518     PageNumber,
2519     Paragraph,
2520     Packer,
2521     TableOfContents,
2522     TextRun
2523 } = require('docx');
2524 function normalizeScreenshotKey(value) {
2525     return String(value || '').trim().toLowerCase();
2526 }
2527 function buildScreenshotEntry(raw) {
2528     if (typeof raw === 'string') {
2529         return { path: raw, name: path.basename(raw), note: '' };
2530     }
2531     const p = String(raw?.path || '').trim();
2532     return {
2533         path: p,
2534         name: String(raw?.name || (p ? path.basename(p) : '')).trim(
2535     ),
2536         note: String(raw?.note || '').trim()
2537     };
2538 }
2539 function buildScreenshotKey(entry) {
2540     return normalizeScreenshotKey(entry.note || entry.name || entr
2541     y.path);
2542 }
2543 function scaleImageSize(original, maxWidth, maxHeight) {
2544     const width = Math.max(1, Number(original?.width || 1));
2545     const height = Math.max(1, Number(original?.height || 1));
2546     const widthRatio = maxWidth / width;
2547     const heightRatio = maxHeight / height;
2548     const ratio = Math.min(widthRatio, heightRatio, 1);
2549     return {
2550         width: Math.max(1, Math.floor(width * ratio)),
```

软著鉴别材料整理器软件 V1.0

```
2551     height: Math.max(1, Math.floor(height * ratio))
2552   };
2553 }
2554 function imageParagraphForScreenshot(entry) {
2555   const img = nativeImage.createFromPath(entry.path);
2556   const size = img.getSize();
2557   const scaled = scaleImageSize(size, 520, 700);
2558   const data = fs.readFileSync(entry.path);
2559   const children = [
2560     new ImageRun({
2561       data,
2562       transformation: scaled
2563     })
2564   ];
2565   return new Paragraph({
2566     alignment: AlignmentType.CENTER,
2567     children,
2568     spacing: { after: 200 }
2569   });
2570 }
2571 function markdownToDocChildren(markdown, screenshots) {
2572   const lines = String(markdown || '').replace(/\r\n/g, '\n').split('\n');
2573   const children = [];
2574   let inCodeBlock = false;
2575   const screenshotEntries = (Array.isArray(screenshots) ? screenshots : []).map(buildScreenshotEntry);
2576   const screenshotByKey = new Map(screenshotEntries.map(e => [buildScreenshotKey(e), e]));
2577   const usedScreenshotKeys = new Set();
2578   for (const rawLine of lines) {
2579     const line = rawLine.trimEnd();
2580     if (line.startsWith('```')) {
2581       inCodeBlock = !inCodeBlock;
2582       continue;
2583     }
2584     if (inCodeBlock) {
2585       children.push(
2586         new Paragraph({
2587           children: [
2588             new TextRun({
2589               text: line.length === 0 ? ' ' : line,
2590               font: 'Consolas',
2591               size: 21,
2592               preserve: true
2593             })
2594           ]
2595         })
2596       );
2597     }
2598   }
2599   continue;
```

软著鉴别材料整理器软件 V1.0

```
2601     }
2602     const shot = line.match(/^\\[\\[SCREENSHOT:(.+?)\\]\\]\\s*/);
2603     if (shot) {
2604         const key = normalizeScreenshotKey(shot[1]);
2605         const entry = screenshotByKey.get(key);
2606         if (entry) {
2607             try {
2608                 children.push(imageParagraphForScreenshot(entry));
2609                 usedScreenshotKeys.add(key);
2610             } catch (_err) {
2611                 children.push(
2612                     new Paragraph({
2613                         children: [
2614                             new TextRun({
2615                                 text: `截图加载失败: ${entry.path}`,
2616                                 font: 'Microsoft YaHei',
2617                                 size: 22
2618                             })
2619                         ]
2620                     })
2621                 );
2622             }
2623         } else {
2624             children.push(
2625                 new Paragraph({
2626                     children: [
2627                         new TextRun({
2628                             text: `未找到匹配截图: ${shot[1]}`,
2629                             font: 'Microsoft YaHei',
2630                             size: 22
2631                         })
2632                         ]
2633                     })
2634                 );
2635             }
2636             continue;
2637         }
2638     if (line.trim().length === 0) {
2639         children.push(new Paragraph({ text: '' }));
2640         continue;
2641     }
2642     const h3 = line.match(/^###\\s+(.*)$/);
2643     if (h3) {
2644         children.push(new Paragraph({ text: h3[1].trim(), heading:
2645 HeadingLevel.Heading_3 }));
2646         continue;
2647     }
2648     const h2 = line.match(/^##\\s+(.*)$/);
2649     if (h2) {
2650         children.push(new Paragraph({ text: h2[1].trim(), heading:
```

软著鉴别材料整理器软件 V1.0

```
2651     HeadingLevel.HEADING_2 })));
2652         continue;
2653     }
2654     const h1 = line.match(/^#\s+(.*)$/);
2655     if (h1) {
2656         children.push(new Paragraph({ text: h1[1].trim(), heading:
2657             HeadingLevel.HEADING_1 }));
2658         continue;
2659     }
2660     children.push(
2661         new Paragraph({
2662             children: [new TextRun({ text: line, font: 'Microsoft Ya
2663             Hei', size: 24 })]
2664         })
2665     );
2666 }
2667 const unused = screenshotEntries.filter(e => !usedScreenshotKe
2668 ys.has(buildScreenshotKey(e)));
2669 return { children, unused };
2670 }
2671 function buildHeaderFooter(headerText) {
2672     const header = new Header({
2673         children: [
2674             new Paragraph({
2675                 children: [new TextRun({ text: headerText || '', font: 'Microsoft YaHei', size: 24 })],
2676                 alignment: AlignmentType.LEFT,
2677                 spacing: { before: 0, after: 0 }
2678             })
2679         ]
2680     ]);
2681     const footer = new Footer({
2682         children: [
2683             new Paragraph({
2684                 alignment: AlignmentType.CENTER,
2685                 children: [
2686                     new TextRun({ text: '- 第 ', font: 'Microsoft YaHei', size: 20 }),
2687                     new TextRun({ children: [PageNumber.CURRENT] }),
2688                     new TextRun({ text: ' 页 / 共 ', font: 'Microsoft YaHei' ,
2689                     size: 20 }),
2690                     new TextRun({ children: [PageNumber.TOTAL_PAGES] }),
2691                     new TextRun({ text: ' 页 -', font: 'Microsoft YaHei', size: 20 })
2692                 ],
2693                 spacing: { before: 0, after: 0 }
2694             })
2695         ],
2696         spacing: { before: 0, after: 0 }
2697     });
2698 }
2699 return { header, footer };
2700 }
```

软著鉴别材料整理器软件 V1.0

```
2701 }
2702 async function writeManualDocx(outputPath, options) {
2703   const softwareName = String(options?.softwareName || '').trim(
2704 );
2705   const softwareVersion = String(options?.softwareVersion || '')
2706 .trim();
2707   const headerText = String(options?.headerText || '').trim();
2708   const markdown = String(options?.markdown || '');
2709   const screenshots = Array.isArray(options?.screenshots) ? opti
2710 ons.screenshots : [];
2711   const { header, footer } = buildHeaderFooter(headerText);
2712   const coverTitle = softwareName ? `${softwareName} 软件说明书` : '软
2713 件说明书';
2714   const coverSubtitle = softwareVersion ? `版本: ${softwareVersion}`
2715 ` `;
2716   const coverChildren = [
2717     new Paragraph({ text: '', spacing: { after: 400 } }),
2718     new Paragraph({
2719       alignment: AlignmentType.CENTER,
2720       children: [new TextRun({ text: coverTitle, font: 'Microsof
2721 t YaHei', size: 56, bold: true })],
2722       spacing: { before: 2400, after: 500 }
2723     })
2724   ];
2725   if (coverSubtitle) {
2726     coverChildren.push(
2727       new Paragraph({
2728         alignment: AlignmentType.CENTER,
2729         children: [new TextRun({ text: coverSubtitle, font: 'Mic
2730 rosoft YaHei', size: 28 })],
2731         spacing: { before: 200, after: 2000 }
2732       })
2733     );
2734   }
2735   const bodyChildren = [];
2736   bodyChildren.push(new Paragraph({ text: '目录', heading: Heading
2737 Level.HEADING_1 }));
2738   bodyChildren.push(
2739     new TableOfContents('目录', {
2740       hyperlink: true,
2741       headingStyleRange: '1-3'
2742     })
2743   );
2744   bodyChildren.push(new Paragraph({ pageBreakBefore: true, text:
2745 '正文', heading: HeadingLevel.HEADING_1 }));
2746   const parsed = markdownToDocChildren(markdown, screenshots);
2747   bodyChildren.push(...parsed.children);
2748   if (parsed.unused.length > 0) {
2749     bodyChildren.push(new Paragraph({ pageBreakBefore: true, tex
2750 t: '截图（未引用）', heading: HeadingLevel.HEADING_1 }));

```

软著鉴别材料整理器软件 V1.0

```
2751     for (const entry of parsed.unused) {
2752         try {
2753             bodyChildren.push(imageParagraphForScreenshot(entry));
2754         } catch (_err) {
2755             bodyChildren.push(
2756                 new Paragraph({
2757                     children: [new TextRun({ text: `截图加载失败: ${entry.path}`}],
2758                     font: 'Microsoft YaHei', size: 22 })
2759                 })
2760             );
2761         }
2762     }
2763 }
2764 const doc = new Document({
2765     features: { updateFields: true },
2766     sections: [
2767         {
2768             properties: {
2769                 page: {
2770                     margin: { top: 1440, bottom: 1440, left: 1440, right
2771 : 1440 }
2772                 }
2773             },
2774             children: coverChildren
2775         },
2776         {
2777             properties: {
2778                 page: {
2779                     margin: {
2780                         top: 1440,
2781                         bottom: 1440,
2782                         left: 1440,
2783                         right: 1440,
2784                         header: 720,
2785                         footer: 720
2786                     },
2787                     pageNumbers: { start: 1, formatType: NumberFormat.DE
2788 CIMAL }
2789                 }
2790             },
2791             headers: { default: header, first: header },
2792             footers: { default: footer, first: footer },
2793             children: bodyChildren
2794         }
2795     ]
2796 });
2797 const buffer = await Packer.toBuffer(doc);
2798 fs.writeFileSync(outputPath, buffer);
2799 }
2800 module.exports = {
```

软著鉴别材料整理器软件 V1.0

```
2801     writeManualDocx
2802   };
2803   const childProcess = require('child_process');
2804   function toEncodedCommand(script) {
2805     return Buffer.from(script, 'utf16le').toString('base64');
2806   }
2807   function escapePsPath(value) {
2808     return value.replace(/'/g, "''");
2809   }
2810   function writePdfFromDocx(docxPath, pdfPath) {
2811     const docx = escapePsPath(docxPath);
2812     const pdf = escapePsPath(pdfPath);
2813     const script = `
2814     $word = New-Object -ComObject Word.Application
2815     $word.Visible = $false
2816     $doc = $word.Documents.Open('${docx}')
2817     $doc.ExportAsFixedFormat('${pdf}', 17)
2818     $doc.Close()
2819     $word.Quit()
2820   `;
2821   const encoded = toEncodedCommand(script);
2822   childProcess.execFileSync('powershell', [
2823     '-NoProfile',
2824     '-NonInteractive',
2825     '-ExecutionPolicy',
2826     'Bypass',
2827     '-EncodedCommand',
2828     encoded
2829   ], { stdio: 'ignore' });
2830 }
2831 module.exports = {
2832   writePdfFromDocx
2833 };
2834 const fs = require('fs');
2835 const path = require('path');
2836 const EXCLUDED_DIRS = new Set([
2837   'node_modules', 'dist', 'build', 'out', '.git', 'vendor', 'bin',
2838   'obj',
2839   'target', '.idea', '.vscode', '.vs', '.svn', '.hg', '.turbo',
2840   '.next',
2841   '.cache', '.angular', '.gradle'
2842 ]);
2843 const EXCLUDED_FILES = new Set([
2844   'package-lock.json', 'yarn.lock', 'pnpm-lock.yaml'
2845 ]);
2846 const SUPPORTED_EXTS = new Set([
2847   '.js', '.jsx', '.ts', '.tsx',
2848   '.py', '.java', '.cs', '.c', '.cpp', '.h', '.hpp',
2849   '.go', '.rs', '.php', '.swift', '.kt'
2850 ]);
```

软著鉴别材料整理器软件 V1.0

```
2851 const MAX_COLUMNS = 64;
2852 const LINES_PER_PAGE = 50;
2853 const TARGET_PAGES = 60;
2854 const TARGET_LINES = LINES_PER_PAGE * TARGET_PAGES;
2855 const END_LINE_REGEX = /{}\s*;?\s*\$|end\s*;?\s*\$|return\s*;?\s*\$/
2856 )/i;
2857 function getCommentSyntax(ext) {
2858     switch (ext) {
2859         case '.py':
2860             return { line: ['#'], block: [''''', "'''"] };
2861         case '.php':
2862             return { line: ['//', '#'], block: ['/*'] };
2863             state.blockEnd = nextBlock.start === '*' ? '*' : nextBlock
2864 .start;
2865             i = nextBlock.idx + nextBlock.start.length;
2866     }
2867     return { code: output, state };
2868 }
2869 function stripLineComment(code, syntax) {
2870     let earliest = -1;
2871     for (const marker of syntax.line) {
2872         const idx = code.indexOf(marker);
2873         if (idx !== -1) {
2874             if (earliest === -1 || idx < earliest) earliest = idx;
2875         }
2876     }
2877     if (earliest === -1) return code;
2878     return code.slice(0, earliest);
2879 }
2880 function isEffectiveLine(line, state, syntax) {
2881     if (line.trim().length === 0) return false;
2882     const blockResult = stripComments(line, state, syntax);
2883     const afterBlock = blockResult.code;
2884     const withoutLine = stripLineComment(afterBlock, syntax);
2885     if (withoutLine.trim().length === 0) return false;
2886     return true;
2887 }
2888 function readLines(filePath) {
2889     try {
2890         const raw = fs.readFileSync(filePath, 'utf8');
2891         return raw.replace(/\r\n/g, '\n').replace(/\r/g, '\n').split
2892 ('\\n');
2893     } catch {
2894         return [];
2895     }
2896 }
2897 function paginate(lines, perPage) {
2898     const pages = [];
2899     for (let i = 0; i < lines.length; i += perPage) {
2900         pages.push(lines.slice(i, i + perPage));
```

软著鉴别材料整理器软件 V1.0

```
2901     }
2902     return pages;
2903   }
2904   function normalizeLine(line) {
2905     return line.replace(/\t/g, '    ');
2906   }
2907   function wrapLine(line, maxCols) {
2908     if (line.length <= maxCols) return [line];
2909     const parts = [];
2910     for (let i = 0; i < line.length; i += maxCols) {
2911       parts.push(line.slice(i, i + maxCols));
2912     }
2913     return parts;
2914   }
2915   function scanProject(projectPath) {
2916     const files = walkDir(projectPath);
2917     const fileBlocks = [];
2918     let effectiveLines = 0;
2919     for (const file of files) {
2920       const ext = path.extname(file).toLowerCase();
2921       const syntax = getCommentSyntax(ext);
2922       const state = { inBlock: false, blockEnd: null };
2923       const fileLines = readLines(file);
2924       const effective = [];
2925       for (const line of fileLines) {
2926         const isEffective = isEffectiveLine(line, state, syntax);
2927         if (isEffective) {
2928           effective.push(normalizeLine(line));
2929         }
2930       }
2931       if (effective.length > 0) {
2932         const wrappedLines = [];
2933         for (const line of effective) {
2934           const chunks = wrapLine(line, MAX_COLUMNS);
2935           for (const chunk of chunks) {
2936             if (chunk.trim().length === 0) continue;
2937             wrappedLines.push(chunk);
2938           }
2939         }
2940         if (wrappedLines.length > 0) {
2941           fileBlocks.push({
2942             path: file,
2943             lines: wrappedLines,
2944             lastLine: wrappedLines[wrappedLines.length - 1]
2945           });
2946         }
2947       }
2948     }
2949     const lastFileIdx = (() => {
2950       for (let i = fileBlocks.length - 1; i >= 0; i -= 1) {
```

软著鉴别材料整理器软件 V1.0

```
2951     if (END_LINE_REGEX.test(fileBlocks[i].lastLine)) return i;
2952   }
2953   return fileBlocks.length - 1;
2954 })();
2955 const eligibleBlocks = lastFileIdx >= 0 ? fileBlocks.slice(0,
2956 lastFileIdx + 1) : [];
2957 const wrappedLines = eligibleBlocks.flatMap((block) => block.l
2958 ines);
2959 effectiveLines = wrappedLines.length;
2960 const pages = paginate(wrappedLines, LINES_PER_PAGE);
2961 const totalPages = pages.length;
2962 let exportLines = [];
2963 if (wrappedLines.length >= TARGET_LINES) {
2964   if (totalPages > TARGET_PAGES) {
2965     exportLines = pages.slice(0, 30).concat(pages.slice(-30)).f
2966 lat();
2967   } else {
2968     exportLines = wrappedLines.slice(0, TARGET_LINES);
2969   }
2970 } else {
2971   exportLines = wrappedLines.slice();
2972 }
2973 if (wrappedLines.length === 0) {
2974   exportLines = [];
2975 } else if (exportLines.length < TARGET_LINES) {
2976   const paddingCount = TARGET_LINES - exportLines.length;
2977   const padLines = [];
2978   let idx = 0;
2979   while (padLines.length < paddingCount) {
2980     padLines.push(wrappedLines[idx % wrappedLines.length]);
2981     idx += 1;
2982   }
2983   exportLines = padLines.concat(exportLines);
2984 } else if (exportLines.length > TARGET_LINES) {
2985   exportLines = exportLines.slice(0, TARGET_LINES);
2986 }
2987 const pagesToExport = paginate(exportLines, LINES_PER_PAGE);
2988 return {
2989   projectPath,
2990   fileCount: files.length,
2991   effectiveLines,
2992   totalPages,
2993   pages,
2994   pagesToExport,
2995   selectedPages: pagesToExport.length
2996 };
2997 }
2998 module.exports = {
2999   scanProject
3000 };
```