AI for Actuaries

Understanding what AI is, where it helps, and how to use it effectively

Jason Schultz

Felix Vaks

Skylar Lowe

November 10, 2026

What is available at SSA now?

Currently Available to All

- ASC (Agency Support Companion)
- Microsoft Copilot Chat (https://m365.cloud.microsoft/)
- PAT (Policy Assistant Tool)
- Adobe Acrobat

Available to Some

- Github Copilot
- Devin

AI Is Already Everywhere

- Search engines
- GPS navigation
- Email spam filtering
- Fraud detection
- Recommendation systems
- Medical diagnostics

- AI didn't suddenly appear with ChatGPT—ChatGPT made AI accessible through conversation.

What Is AI?

- AI is a prediction engine.

- It learns patterns and relationships from enormous amounts of information and predicts likely responses.

- Its strength is finding and applying patterns and connections.

Good Uses of AI for Actuaries

- Coding assistance (for example, SQL, Fortran, SAS)
- Documentation
- Summarizing regulations
- Explaining code
- Learning unfamiliar topics
- Drafting emails
- Creating presentations
- Brainstorming approaches

AI is not always logical

Example: Can a man receive social security benefits from his widow’s account?

Logical answer: No (he is dead).

SSA’s chatbot answer: “Yes, under Social Security rules, a widower (a man whose spouse has died) may be eligible to receive survivor benefits based on his deceased spouse’s Social Security record. These are called "widower's benefits" and…”

Lesson: AI answers based on language patterns rather than logic. **Always verify its feedback!**

Common Types of AI Chatbot Errors

1. **Incorrect Facts** – AI may sometimes provide inaccurate or outdated information. As AI technology advances, these errors are becoming less frequent, and accuracy continues to improve.

2. **Lost Context**– AI chatbots can forget details from earlier in the conversation due to memory constraints. This is especially noticeable when requesting complex tasks, such as writing intricate code. Improvements in AI memory are reducing this issue over time.

3. **Misunderstanding the user** – AI cannot read minds, so misinterpretations of user intent remain common—especially in specialized fields like actuarial work. Because AI appears intelligent, users may assume it understands their intent perfectly, which can lead to errors.

Ask AI to Critique its Own Work

Asking AI to review and explain its output is a practical way to catch errors and ensure the solution matches your request.

- **Show AI Its Own Work**: Present the AI-generated program back to the AI and ask, “Does this program do task X?” or “What does this program do?” *This helps the AI review its own output for accuracy.*

- **Request Explanation**: Ask the AI to explain how the program meets the requirements or to walk through its logic. *This encourages the AI to clarify its reasoning and approach*.

- **Identify Gaps**: Prompt the AI to uncover errors, omissions, or misunderstandings, and ask it to correct or clarify its output. *This step helps ensure the solution is complete and accurate*.

AI at Work

Real-Time Tools and Examples for the Actuarial Department

Key Concepts

- Treat AI like an assistant: provide context and clear instructions
- Use AI as a smart new team member: teach and explain, and it figures out how to get things done
- You set the goals; AI helps create the path to achieve them
- AI gets you 80% of the way—you get it to 100%

How AI Supports Our Work

- Leverages collective knowledge from similar projects
- Makes things more efficient
- Saves time and refines tasks
- Delegate repetitive tasks and routine work

Practical Use Cases

- Drafting and refining e-mails
- Making e-mails thoughtful and flow
- Making e-mails friendly

Hands-On Example 1 : Creating AI Poll

- Creating Questions
- Creating Form
- Summarizing results
- Documenting code
- Brainstorming ideas
- Easily modifying the poll as needed

Hands-On Example 2 : Data Extraction

- Pulling data lists from complex documents
- Organizing information for analysis

Using Devin Desktop

- AI having access to your desktop
- Understands the context of the files
- Makes updates fast and efficient

Updating FY 2028 budget mainframe programs

- Traditional Process:
  - Compare prior years' versions manually
  - Update dates and fiscal years across hundreds of lines
  - 2-3 hours of work

Using Devin to update FY 2028 budget mainframe programs : 1st Attempt

- What was provided:
  - FY 2025 and FY 2026 program versions
  - Network locations

Using Devin to update FY 2028 budget mainframe programs : 1st Attempt

- What was provided:
  - FY 2025 and FY 2026 program versions
  - Network locations

- Results:
  - Compared only 2 years
  - Used an abnormal shutdown year as the baseline
  - Wrong naming conventions (months are wrong)

Using Devin to update FY 2028 budget mainframe programs : 2nd Attempt

- Additional context provided:
  - FY 2026 was abnormal due to shutdown
  - Git repository and commit history
  - Organizational naming standards
  - Recent format changes (DASD to TAPE)

Using Devin to update FY 2028 budget mainframe programs : 2nd Attempt

- Additional context provided:
  - FY FY 2026 was abnormal due to shutdown
  - Git repository and commit history
  - Organizational naming standards
  - Recent format changes (DASD to TAPE)

- Results:
  - Reviewed git history
  - Compared three years of data
  - Identified normal vs abnormal patterns
  - Applied proper naming standards
  - Programs ready in about 15 minutes

Conclusion

- AI is a tool, not a replacement for actuarial judgment
- You are the ultimate decision maker
- You direct the AI and check the results
- AI saves time, reduces frustration, and lets you focus on analysis and improvement
- Know which tool works for what

Using Devin Desktop to Implement an Excel-Based Input Data Path in the Actuarial Services Revenue Model

Skylar Lowe | November 10, 2026

SOCIAL SECURITY ADMINISTRATION
ACTUARIAL SERVICES

CONTEXT

The problem: fragmented Form 30 inputs

The revenue model needs a more easily maintained data source without changing established downstream logic.

Before

Multiple hand-maintained Form 30 text files feed model runs.

Design constraint

Preserve the proven Revearn input path. Do not add direct Excel parsing or rewrite downstream Fortran routines.

After

One workbook, generated automatically from the published Form 30 PDFs, becomes the maintainable source. No manual entry.

Scope

Two separate model runs; outputs expected to match baselines except for one documented data correction.

Example: hand-maintained Form 30 text file

C:\GitRepos\econ-taxpay\dat\et8425.dat - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

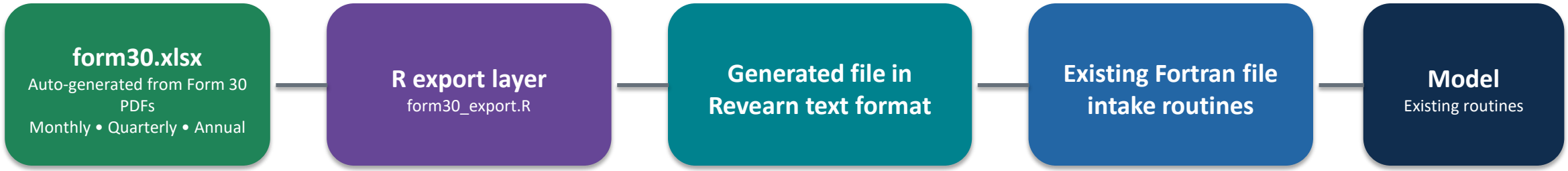
et8425.dat

```
REVENUES
Actual taxes transferred in 1/1/83 to 12/31/25
MXCFICATEAO                    M 1983 1 2025 12
      9313      8181      9346      9485      9649     10775      9816      9850
     10157      9056      9061      9855     11876     10131     11479     11988
     11223     12468     11420     11629     11487     10547     10981     11411
     13338     11428     12919     12172     12556     12949     12575     12744
     12883     11662     11940     12393     14351     11715     14070     13507
     13359     13811     14057     13319     13570     12523     12615     13699
     16087     14444     16221     16577     15672     16607     15945     15565
     16129     15186     15410     16623     19091     16327     18636     18814
     17905     19067     18313     18232     18884     16528     16479     17742
     21505     19039     21045     20516     20133     21475     19092     19234
     19958     17023     17899     18658     22410     19980     22853     22258
     21825     23326     20195     20460     21398     18381     22939     18246
     24019     19309     22664     22203     21767     23244     20535     20866
     21892     19203     20505     21587     24445     20901     23313     23225
     22324     24248     21325     21038     22446     20080     21550     22167
     24701     22119     23480     24124     23224     25257     22402     22787
     23966     21022     22470     23674     26315     22806     25029     25675
     24808     26889     23815     24311     25771     20725     22104     23257
     25898     23220     25183     25096     24172     26242     23200     23517
     24599     21267     22706     23903     27087     24401     27123     26460
     25381     27614     24207     24293     25612     22347     23954     25297
     28892     26631     29383     28127     27108     29570     25692     28248
     29688     24293     26066     27548     30892     28762     31277     29986
     28917     33071     27712     28280     29822     26163     28157     29829
     33106     29934     32574     32231     31117     33977     29565     30180
     31835     28019     30199     32030     35893     32456     35319     34410
     33036     35967     31338     31892     33525     29424     31606     33431
     37936     34470     37273     36456     35020     38138     33023     33143
     34839     30469     32856     34861     39248     35331     38580     37769
```

SOLUTION

Target architecture: convert at the boundary

Excel is translated into the same text-file interface Revearn already uses.



Cutoff-driven export

R reads Form 30 data cutoff dates from run configuration file.

Validation before intake

The export validates workbook structure, dates, columns, cutoffs, and output formatting before Revearn reads data.

Unchanged downstream path

Established Fortran data input routines continue to operate.

How Devin Desktop fits in

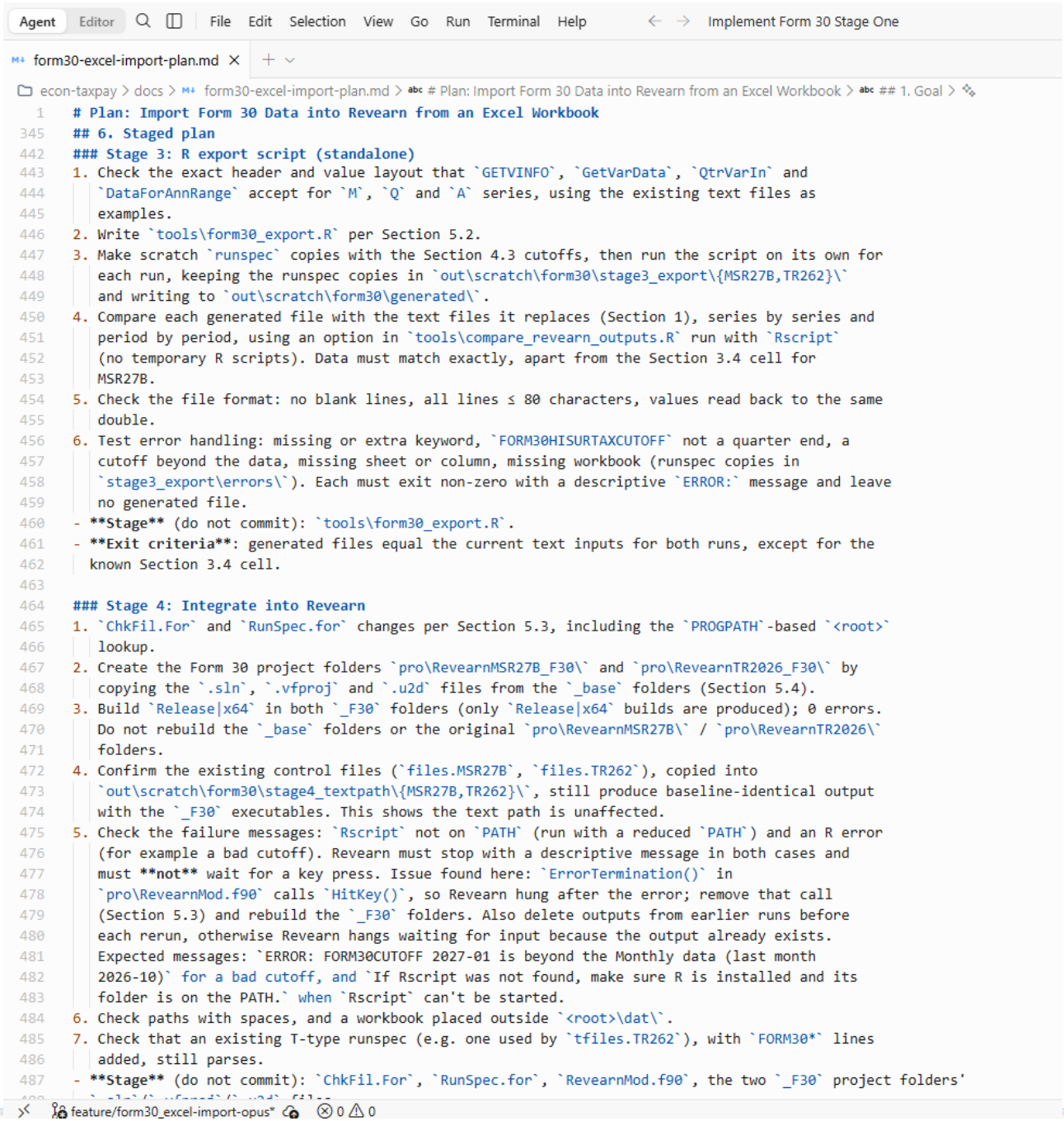
1. **Create the plan**
   Background and instructions for each stage plus verification steps

2. **Prompt in Devin Desktop to implement a stage**
   Code changes, build, test model runs, and compare output

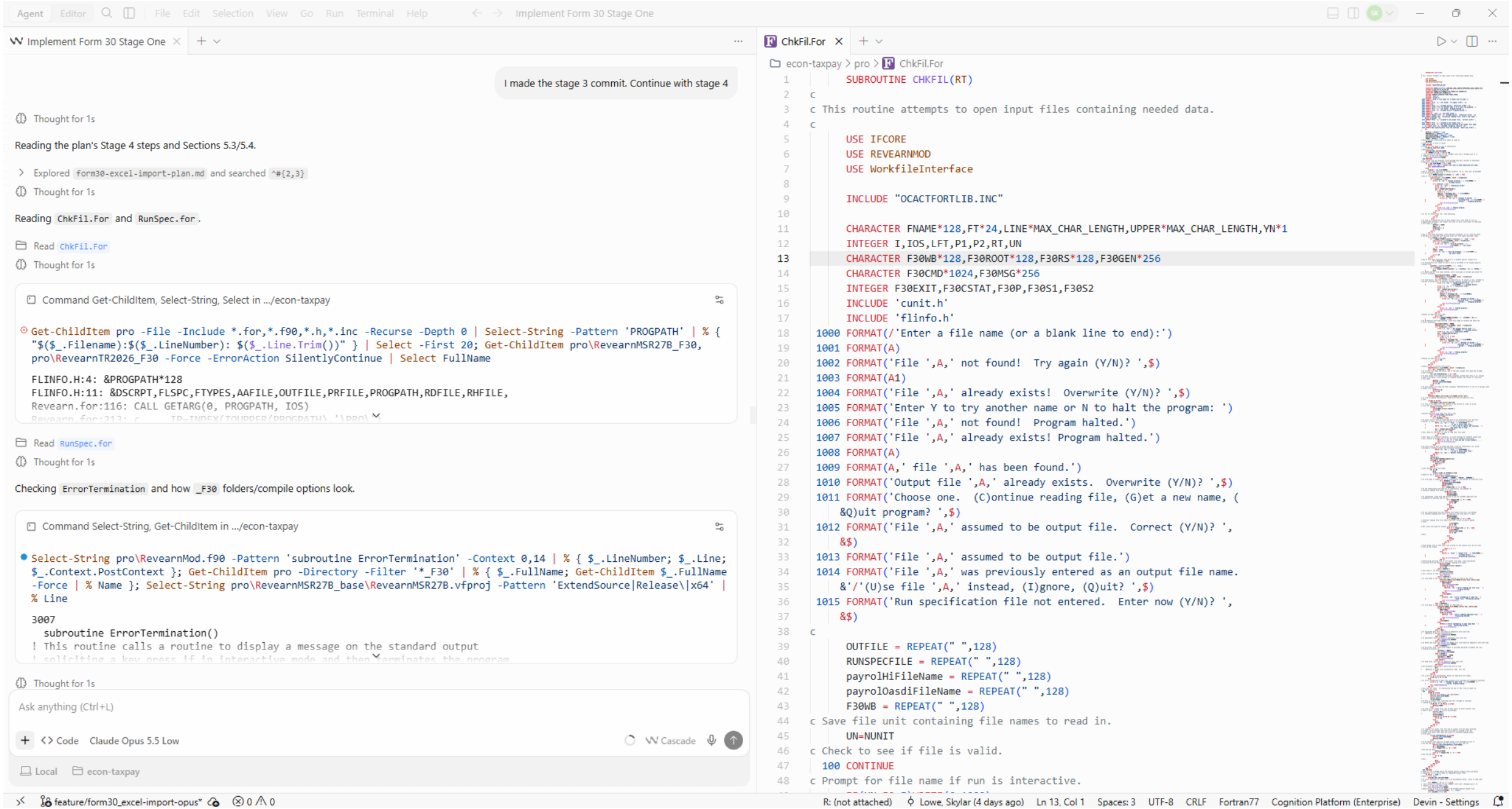
3. **Review and approval**
   Moving forward to the next stage is only permitted after human review

Devin Desktop speeds up the work; a person stays accountable.

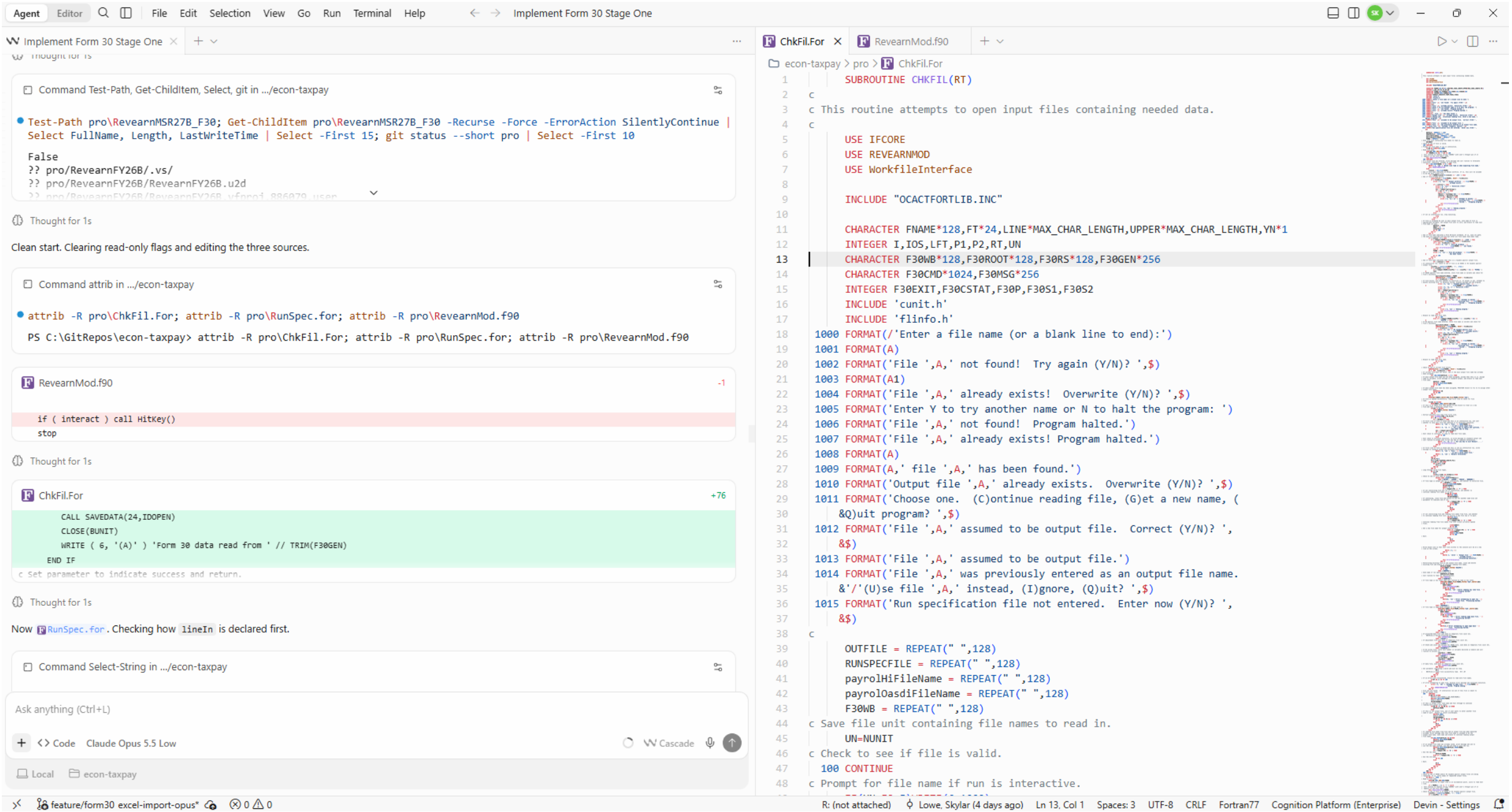
The plan


Agent
Editor
File
Edit
Selection
View
Go
Run
Terminal
Help
Implement Form 30 Stage One
form30-excel-import-plan.md
econ-taxpay > docs > form30-excel-import-plan.md > # Plan: Import Form 30 Data into Revearn from an Excel Workbook > ## 1. Goal
Plan: Import Form 30 Data into Revearn from an Excel Workbook
6. Staged plan
Stage 3: R export script (standalone)
1. Check the exact header and value layout that `GETVINFO`, `GetVarData`, `QtrVarIn` and `DataForAnnRange` accept for `M`, `Q` and `A` series, using the existing text files as examples.
2. Write `tools\form30_export.R` per Section 5.2.
3. Make scratch `runspec` copies with the Section 4.3 cutoffs, then run the script on its own for each run, keeping the runspec copies in `out\scratch\form30\stage3_export\{MSR27B,TR262}\` and writing to `out\scratch\form30\generated\`.
4. Compare each generated file with the text files it replaces (Section 1), series by series and period by period, using an option in `tools\compare_revearn_outputs.R` run with `Rscript` (no temporary R scripts). Data must match exactly, apart from the Section 3.4 cell for MSR27B.
5. Check the file format: no blank lines, all lines ≤ 80 characters, values read back to the same double.
6. Test error handling: missing or extra keyword, `FORM30HISURTAXCUTOFF` not a quarter end, a cutoff beyond the data, missing sheet or column, missing workbook (runspec copies in `stage3_export\errors\`). Each must exit non-zero with a descriptive `ERROR:` message and leave no generated file.
- **Stage** (do not commit): `tools\form30_export.R`.
- **Exit criteria**: generated files equal the current text inputs for both runs, except for the known Section 3.4 cell.
Stage 4: Integrate into Revearn
1. `ChkFil.For` and `RunSpec.for` changes per Section 5.3, including the `PROGPATH`-based `<root>` lookup.
2. Create the Form 30 project folders `pro\RevearnMSR27B_F30\` and `pro\RevearnTR2026_F30\` by copying the `.sln`, `.vfproj` and `.u2d` files from the `_base` folders (Section 5.4).
3. Build `Release|x64` in both `_F30` folders (only `Release|x64` builds are produced); 0 errors. Do not rebuild the `_base` folders or the original `pro\RevearnMSR27B\` / `pro\RevearnTR2026\` folders.
4. Confirm the existing control files (`files.MSR27B`, `files.TR262`), copied into `out\scratch\form30\stage4_textpath\{MSR27B,TR262}\`, still produce baseline-identical output with the `_F30` executables. This shows the text path is unaffected.
5. Check the failure messages: `Rscript` not on `PATH` (run with a reduced `PATH`) and an R error (for example a bad cutoff). Revearn must stop with a descriptive message in both cases and must **not** wait for a key press. Issue found here: `ErrorTermination()` in `pro\RevearnMod.f90` calls `HitKey()`, so Revearn hung after the error; remove that call (Section 5.3) and rebuild the `_F30` folders. Also delete outputs from earlier runs before each rerun, otherwise Revearn hangs waiting for input because the output already exists. Expected messages: `ERROR: FORM30CUTOFF 2027-01 is beyond the Monthly data (last month 2026-10)` for a bad cutoff, and `If Rscript was not found, make sure R is installed and its folder is on the PATH.` when `Rscript` can't be started.
6. Check paths with spaces, and a workbook placed outside `<root>\dat\`.
7. Check that an existing T-type runspec (e.g. one used by `tfiles.TR262`), with `FORM30*` lines added, still parses.
- **Stage** (do not commit): `ChkFil.For`, `RunSpec.for`, `RevearnMod.f90`, the two `_F30` project folders'
feature/form30_excel-import-opus*
0 0

Devin Desktop chat and code view



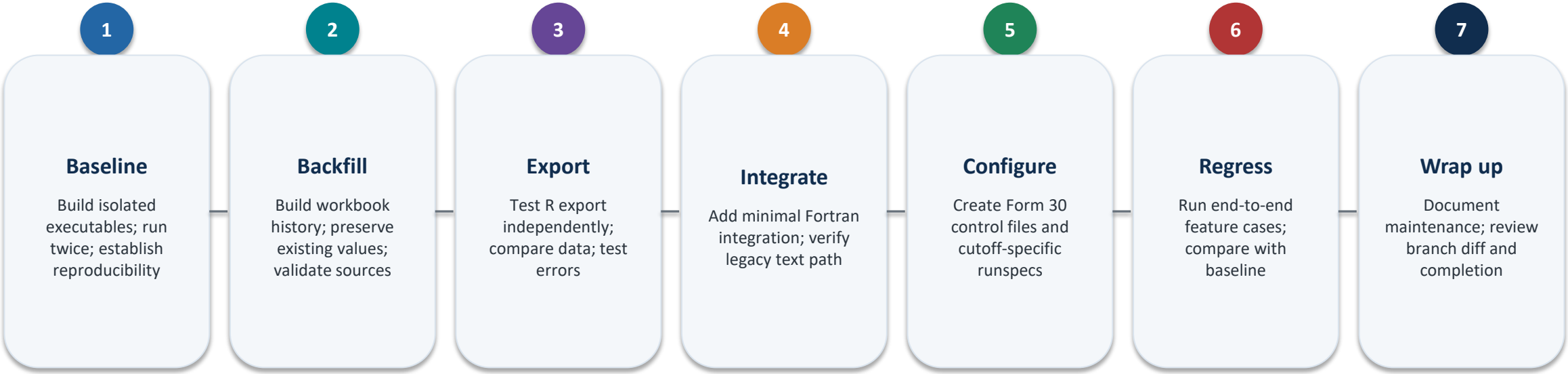
Devin Desktop chat and code view



EXECUTION

Staged implementation allowed for human review

Each stage had a defined objective, evidence, and review point.



Control gate after every stage

The plan required human review and a human-authored commit before work continued.

Conclusion: a reusable pattern for legacy model enhancements

- **Modernize**
  Use a more easily (or automatically) maintained source format while preserving trusted model interfaces.
- **Prove behavior**
  Keep isolated baselines and output comparisons.
- **Govern agent use**
  Use Devin Desktop within an approved plan with review gates and human acceptance.

Questions / discussion