

Rapid Reporter Session

- **Tester:** Del Dewar
- **Charter:** Demonstrate how Rapid Reporter structures exploratory session notes into a clear, skim-readable Markdown report with summary indicators and embedded evidence.
- **Started:** 23 February 2026 10:53 GMT
- **Duration:** 30 minutes

Summary



1 Bug



1 Idea



1 Observation



1 Question



1 Warning

Notes

Rapid Reporter is designed to reduce friction during exploratory testing by keeping note capture immediate and contextual.

This session demonstrates how the majority of captured notes are simple Test entries without visual noise or icon indicators.

The goal is to preserve flow state while still producing a structured, exportable report.

This is a Test comment.

Notes are recorded chronologically and exported in the same order to preserve session narrative.

And this input box also shows the elapsed time in the current session (in blue)

And if you forget where you are, the recap drawer (below) helps recover that context

The exported Markdown begins with metadata including tester name, charter, start time, and duration.



Observation entries help capture behaviour without implying defect or risk.

Icon-labelled entries appear in the summary section to aid quick scanning by stakeholders.



If screenshots are not captured with sufficient resolution, evidence may lack clarity in exported reports.

The majority of session notes typically remain neutral Test entries.



Consider allowing custom note categories for team-specific workflows.

Exported reports are fully portable and can be shared without requiring Rapid Reporter to view them.



Session data is stored in memory until export; unexpected system shutdown would lose unsaved notes.

The recap drawer allows the tester to maintain context without interrupting note entry.



Should Rapid Reporter support automatic cloud backup of exported sessions?

```
{
  "tool": "Rapid Reporter",
  "version": "1.0.0",
  "purpose": "Low-friction exploratory session capture"
}
```

Snippet entries are exported as fenced Markdown code blocks to preserve formatting.

```
const [appVersion, setAppVersion] = useState<string | null>(<null>);

useEffect(() => {
  invokeString("app_version")
    .then(setAppVersion)
    .catch(() => setAppVersion(null));
}, []);

// Restore window sizing logic (start vs capture; recap collapsed vs expanded)
useEffect(() => {
  const win = getCurrentWebviewWindow();
  const startSessionSize = new LogicalSize(600, 520);
  // Size for the Start Session modal (fits comfortably)
  const startSessionSize = new LogicalSize(600, 520);
  // Size for the main capture UI (collapsed vs expanded recap)
  const captureSizeCollapsed = new LogicalSize(1200, 200);
  const captureSizeExpanded = new LogicalSize(1200, 300);
  const captureSize = recapOpen ? captureSizeExpanded : captureSizeCollapsed;
  win.setSize(session ? captureSize : startSessionSize).catch(console.error);
}, [session, recapOpen]);

const sessionRef = useRef<Session | null>(<null>);
useEffect(() => {
  sessionRef.current = session;
}, [session]);

useEffect(() => {
  // Only the main window should listen/respond
```

Generated by Rapid Reporter v1.0.0