PROJECT ABSTRACT

Screenshot Annotation Tool for Bug Reports

A desktop screenshot annotation tool with arrows, boxes, blur redaction and numbered steps, purpose-built for clear bug reports — delivered built-to-order with full source, report, PPT and viva kit.

01. Title

Screenshot Annotation Tool for Bug Reports.

02. Introduction / Background

Bug reports live or die on clarity. A tester who writes “the button doesn't work” with no visual evidence forces the developer into a long reproduction cycle, while raw screenshots often hide the actual problem in a sea of pixels. Professional teams solve this with annotated screenshots: the failing element circled, reproduction steps numbered over the flow, sensitive data blurred out. This project builds that capability as a focused desktop application — an annotation editor whose entire workflow (capture → annotate → export) is designed around filing better bug reports.

03. Problem Statement

Generic image editors are poor bug-reporting tools: pasting, drawing attention, redacting and exporting takes a dozen manual steps, and nothing about them understands “steps to reproduce”. Testers and developers need a single-purpose tool that turns a raw screenshot into a self-explanatory annotated image in under a minute, without uploading potentially sensitive screens to a web service.

04. Objectives

- Accept screenshots from the OS clipboard and from PNG/JPG/WebP files.
- Provide six annotation tools: arrow, rectangle, ellipse, blur redaction, text callouts and auto-numbered steps.
- Maintain a 50-step undo stack so annotation experiments never destroy the base image.
- Export flattened PNG/JPG locally or copy the result back to the clipboard — fully offline.
- Package as a cross-platform Electron desktop application with a user manual.

05. Proposed Methodology

The application is built on Electron with an HTML5 canvas rendering core. Screenshots are drawn to canvas at native resolution; every annotation is stored as a vector object (type, coordinates, text) in an undoable list, and the canvas re-renders from base image plus annotation stack on every change. The blur tool pixelates regions by downscale/upscale, destroying detail irreversibly. Export serializes the composed canvas to PNG or JPG via canvas APIs, or writes it to the OS clipboard.

06. System Architecture / Working

Three layers: (1) the Electron shell handling windows, menus, file dialogs and clipboard; (2) the canvas engine — image loading, annotation objects, undo stack, hit-free immediate-mode rendering; (3) the export module — PNG/JPG serialization and clipboard write-back. The toolbar selects the active tool; mouse drag creates vector annotations; the side panel documents the three-step workflow.

07. Hardware / Software Requirements

- A desktop/laptop running Windows, macOS or Linux (any machine that runs Electron).
- Node.js 18+ and Electron for building; no runtime dependencies beyond the packaged app.
- No network connection required — the tool is fully offline by design.
- 4 GB RAM recommended for annotating 4K screenshots.

08. Expected Outcomes

- A working desktop app that pastes, annotates and exports screenshots in a single session.
- Annotated bug-report images demonstrably clearer than raw screenshots in the report's before/after examples.
- Complete source code, user manual, project report, PPT and viva Q&A.;

09. Applications

- QA teams attaching evidence to Jira, GitHub Issues and Bugzilla tickets.
- Support teams redacting customer data before sharing screens.
- Tutorial and documentation authors marking up UI screenshots.
- Students demonstrating canvas graphics and desktop-app engineering.

10. Future Scope

- OS-level screen capture hotkey so the app captures as well as annotates.
- A re-editable project file format preserving annotation objects.
- Cloud team library with shared annotation templates.