Project Title: "Screenshot Capture Pro" - A Smart Chrome Extension

1. Mission & Core Objective

The goal is to create a lightweight, intuitive Chrome extension named "Screenshot Capture Pro". Its primary function is to allow users to capture a specific area of a webpage, annotate it with simple tools, and seamlessly save the result to their Google Drive. The entire experience should be fast, user-centric, and reliable.

2. Core User Flow (Step-by-Step)

1. **Initiate Capture:** The user clicks the "Screenshot Capture Pro" icon in the Chrome toolbar.
2. **Select Area:** The cursor immediately changes to a crosshair. The page may have a semi-transparent dark overlay to indicate capture mode.
   - The user **clicks and holds the left mouse button (mousedown)** to define the starting corner of the capture area.
   - They **drag** the mouse to expand the selection rectangle. The selected area should be clear, while the rest of the page remains under the dark overlay.
   - They **release the left mouse button (mouseup)** to finalize the selection.
3. **Capture & Open Editor:** Upon mouseup, the extension captures the selected rectangular area of the visible static page content.
4. **Annotate:** A new browser tab opens immediately, displaying the captured image within a simple annotation editor.
5. **Save & Share:** The user uses the annotation tools and then chooses to download, copy, or get a link to the image, which is automatically saved in their Google Drive.

3. Detailed Feature Breakdown

3.1. Capture Mechanism

- **Trigger:** Left-click on the extension toolbar icon.
- **Permissions:** The extension should use the activeTab and scripting permissions to inject a content script for the capture UI. This avoids requesting overly broad permissions.
- **UI:**
  - A crosshair cursor (cursor: crosshair;).
  - A <div> overlay with a semi-transparent background (e.g., rgba(0, 0, 0, 0.3)).
  - The selection rectangle should be visually clear and have a distinct border (e.g., a 1px dashed red line).
- **Functionality:** The capture should be of the currently visible part of the page only (no scrolling). It should correctly handle pages with fixed elements and

varying zoom levels. The output must be a high-quality PNG image.

3.2. Annotation Editor (New Tab)

This new tab (editor.html) is the core of the post-capture experience. It will contain the captured image on an HTML <canvas> element to allow for drawing.

- **Toolbar UI:** A clean, simple toolbar should be positioned horizontally at the top of the page.
- **Annotation Tools (Default color: Red #FF0000)**
  - **Rectangle:** Click to activate, then click-and-drag on the canvas to draw a rectangle.
  - **Circle/Ellipse:** Click to activate, then click-and-drag on the canvas to draw an ellipse.
  - **Pen:** Click to activate, then draw freehand on the canvas.
  - **Blur/Obfuscate:** Click to activate. Click-and-drag over an area of the image to apply a pixelation or blur effect to hide sensitive information.
- **Action Buttons (Right side of the toolbar):**
  - **Download:** Downloads the canvas content (with annotations) as a screenshot.png file.
  - **Copy to Clipboard:** Copies the image data from the canvas to the user's clipboard. Provide a visual confirmation (e.g., "Copied!").
  - **Get Drive Link:** This button is initially disabled. Once the image is successfully saved to Google Drive, it becomes active. Clicking it copies the direct Google Drive shareable link to the clipboard.

3.3. Google Drive Integration

- **Authentication:**
  - On first use (or if permissions are revoked), the extension must prompt the user to authorize access to their Google Drive.
  - Use Google's Identity API for Chrome Extensions to manage the OAuth 2.0 flow.
  - Request the minimum necessary permission scope: https://www.googleapis.com/auth/drive.file. This scope allows the extension to create new files but not see or modify existing ones it didn't create.
- **File Storage:**
  - Upon successful annotation and saving, the extension must check for a folder named **"screenshots"** in the root of the user's Google Drive.
  - If the folder doesn't exist, create it.
  - Upload the final annotated image (from the canvas) as a PNG file into this "screenshots" folder. The filename can be a timestamp (e.g.,

    screenshot-2025-06-28-12-30-00.png).

- **User Feedback:**
    - While uploading to Drive, show a subtle loading indicator (e.g., "Saving to Drive...").
    - Upon success, show a confirmation message (e.g., "Saved to Drive!") and enable the "Get Drive Link" button.
    - If the upload fails, display a clear error message (e.g., "Failed to save. Please check permissions.").

4. Technical Specifications

- **Manifest Version:** Manifest V3.
- **Permissions:** "activeTab", "scripting", "identity", "storage".
- **Background Script (background.js):**
    - Listens for the extension icon click.
    - Manages the Google Identity/OAuth flow.
    - Handles the communication with the Google Drive API for folder checking and file uploading.
- **Content Script (content.js):**
    - Injected by the background script to handle the screen capture UI (overlay, crosshairs, selection logic).
    - Once an area is selected, it sends the coordinates and a data URL of the captured image to the background script.
- **Editor Page (editor.html, editor.js, editor.css):**
    - A standalone page for the annotation tools.
    - The editor.js will receive the image data, draw it to a canvas, and handle all annotation and button logic.

5. Key Deliverables

- A complete, installable, and functional Chrome extension folder.
- All source code, including manifest.json, background and content scripts, and all files for the editor page.
- Code should be well-commented, explaining the logic for capture, canvas manipulation, and the Google Drive API interactions.
- A clean, intuitive UI that requires no user manual.