# Pastebin Lite - Presentation Script
> [!NOTE]
> This document is a script for a video/audio presentation of the Pastebin Lite project. It is divided into "Visual" (what to show on screen) and "Audio" (what to say).
## Preparation
Before recording:
1. **Open VS Code** with the `pastebin-lite` project loaded.
- Open these files in tabs: `src/app/page.js`, `src/app/api/pastes/route.js`, `src/app/p/[id]/page.js`, `src/lib/redis.js`.
2. **Open your Browser** to `http://localhost:3000` (make sure `npm run dev` is running).
3. Have a second browser window (incognito) ready for pasting the link.
---
## Part 1: Project Demo (The Hook)
| Time | Visual Action | Audio Script |
| :--- | :--- | :--- |
| **0:00** | **[Screen: Browser showing Pastebin Lite Homepage]**<br>Slowly scroll/hover over the card to show the glassmorphism effect. | "Hi everyone. Today I want to show you **Pastebin Lite**—a secure, fast, and ephemeral pastebin service I built using Next.js and Upstash Redis." |
| **0:15** | **[Screen: Typing in the text area]**<br>Type: `console.log('Hello World');` | "The goal was to create something that looks premium but remains incredibly simple to use. As you can see, we moved away from the standard dark mode to a 'Light & Elegant' glassmorphism design." |
| **0:25** | **[Screen: Setting Options]**<br>Set 'Expires After' to `30` seconds.<br>Set 'Max Views' to `5`. | "It's not just about looks, though. It's built for security and privacy. You can set time-based expiration—let's say 30 seconds—or limit the number of times a link can be viewed." |
| **0:35** | **[Screen: Click 'Create Paste']**<br>Show the loading spinner, then the success message. | "When I click create, the app instantly generates a unique, collision-resistant ID and saves it to Redis." |
| **0:45** | **[Screen: Click 'Copy' button]**<br>Open Incognito window.<br>Paste the URL and hit Enter. | "Reference links are generated instantly. I can copy this, share it, and the recipient sees the code content perfectly preserved." |
| **0:55** | **[Screen: Refresh the page a few times]**<br>(Optional) Wait 30s and refresh to show 404/Expired page. | "And because we set a view limit (or time limit), once those conditions are met, the data is gone forever—handled automatically by the persistence layer. No database cleanup scripts required." |
---
## Part 2: Code Walkthrough (Under the Hood)
| Time | Visual Action | Audio Script |
| :--- | :--- | :--- |
| **1:10** | **[Screen: VS Code - Open `src/lib/redis.js`]** | "Let's dive into the code. This project is powered by **Next.js 14** (App Router) and **Upstash Redis**." |
| **1:20** | **[Screen: Highlight `new Redis(...)`]** | "Here in `redis.js`, we initialize our serverless Redis client. We chose Redis for its speed and native TTL (Time-To-Live) support, which is perfect for ephemeral data." |
| **1:30** | **[Screen: Open `src/app/api/pastes/route.js`]** | "This is the backend API route for creating pastes. It runs on the Edge or Node.js runtime." |
| **1:40** | **[Screen: Highlight `crypto.randomUUID()` and Validation]** | "First, we validate the input and generate a unique ID using `crypto.randomUUID()`. This ensures we don't have ID collisions." |
| **1:50** | **[Screen: Highlight `redis.set` with `paste` object]** | "The magic happens here. We create a paste object with `expires_at` and `remaining_views`, then store it in Redis. Notice how clean this is—no complex SQL schemas, just a key-value store." |
| **2:10** | **[Screen: Open `src/app/p/[id]/page.js`]** | "Now, for retrieving the paste. This is a **Server Component**. We fetch the data directly from Redis before sending anything to the client." |
| **2:25** | **[Screen: Highlight the Expiry Logic]** | "We check two things server-side: Has the time expired? And have the max views been reached? If either is true, we immediately show an error and don't even render the content." |

| **2:35** | **[Screen: Highlight `remaining_views -= 1`]** | "If it's valid, we decrement the view count atomically and serve the content." |
---
## Part 3: Design & Deployment
| Time | Visual Action | Audio Script |
| :--- | :--- | :--- |
| **2:50** | **[Screen: Open `src/app/page.js`]**<br>Scroll to `styles` object. | "On the frontend, I used a custom CSS-in-JS approach to achieve this Glassmorphism look. We're using `backdrop-filter: blur(20px)` and semi-transparent backgrounds to give it that frosted glass effect." |
| **3:10** | **[Screen: Browser - Show the colorful background blobs]** | "These floating background orbs are positioned absolutely with a radial gradient to create depth, making the interface feel alive." |
| **3:25** | **[Screen: Browser - Show Deployment (e.g., Vercel Dashboard or just the live link)]** | "Finally, the app is deployed on [Vercel/Netlify]. Because it uses Serverless Redis, it scales infinitely without me managing any servers." |
| **3:40** | **[Screen: Face Camera or Project Logo]** | "That's Pastebin Lite. A modern, secure way to share code. Thanks for watching!" |
---
## Key Tech Stack Mentions
Make sure to emphasize these keywords during the presentation:
- **Next.js 14 (App Router)**
- **Server Components** for security.
- **Upstash Redis** for serverless state.
- **Glassmorphism** for UI design.
- **Atomic Operations** for view counting.