1. **Backend with Express.js**:
   * Set up an Express.js server.
   * Create an endpoint (e.g., **POST /compress-images**) that accepts a set of image files.
   * Use middleware like **multer** to handle file uploads.
   * On receiving the images, run the compression using your script.
   * Instead of reading from the file system, you'd read the image files from the request.
   * After compression, you can zip all the compressed images using a library like **archiver**.
   * Respond with the zipped file as the download.
2. **Frontend with Next.js**:
   * Create a simple UI with a file input field where users can select and upload multiple images.
   * Send the images to your Express.js backend when the form is submitted.
   * Show a progress bar or loader while processing.
   * Once the processing is done, provide the user with a download link for the zipped folder of compressed images.
3. **Enhancements**:
   * **Error Handling**: Ensure that your backend can handle errors gracefully, especially if an uploaded image is corrupt or in an unsupported format.
   * **Rate Limiting**: To prevent misuse, you might want to add some rate limiting on your endpoint.
   * **File Size Limit**: Limit the size and/or number of images a user can upload at once to avoid server overloads.
   * **Authentication/Authorization**: If necessary, add a layer of authentication so only authorized users can access your service.
4. **Deployment**:
   * Once you have your frontend and backend ready, you can deploy them. Platforms like Vercel (ideal for Next.js) and Heroku or DigitalOcean (for the backend) are good choices.
5. **Security Concerns**:
   * Ensure you handle files securely. Avoid keeping uploaded or compressed files for longer than necessary. Delete them after processing.
   * Ensure you're using the latest versions of libraries and regularly check for security vulnerabilities in your stack.