Snippets

- Temporary Download URL
- Download with auth

Temporary Download URL

```
import crypto from 'crypto';

// Generate a signed URL valid for 5 minutes
function generateSignedUrl(fileName: string, secret: string) {
  const expiresIn = 5 * 60 * 1000; // 5 minutes
  const expiresAt = Date.now() + expiresIn;
  const signature = crypto.createHmac('sha256', secret)
                          .update(`${fileName}:${expiresAt}`)
                          .digest('hex');

  return `/download/${fileName}?expiresAt=${expiresAt}&signature=${signature}`;
}

// Verify the signed URL on request
app.get('/download/:fileName', (req: Request, res: Response) => {
  const { fileName } = req.params;
  const { expiresAt, signature } = req.query;

  if (!expiresAt || !signature) {
    return res.status(401).json({ message: 'Invalid download link' });
  }

  const secret = 'your_secret_key';
  const expectedSignature = crypto.createHmac('sha256', secret)
                                  .update(`${fileName}:${expiresAt}`)
                                  .digest('hex');

  if (signature !== expectedSignature || Date.now() > Number(expiresAt)) {
    return res.status(403).json({ message: 'Link expired or invalid' });
  }

  // Proceed with file download if the signature is valid and not expired
  const filePath = path.join(__dirname, 'uploads', fileName);
  res.download(filePath);
});
```

Download with auth

Middleware

```
import express, { Request, Response, NextFunction } from 'express';
import jwt from 'jsonwebtoken';

const authenticate = (req: Request, res: Response, next: NextFunction) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'Unauthorized: No token provided' });
  }

  try {
    // Replace 'your_secret_key' with your actual JWT secret key
    const user = jwt.verify(token, 'your_secret_key');
    req.user = user;
    next();
  } catch (error) {
    res.status(403).json({ message: 'Forbidden: Invalid token' });
  }
};
```

Route with protection

```
app.get('/download/:fileName', authenticate, (req: Request, res: Response) => {
  const { fileName } = req.params;
  const filePath = path.join(__dirname, 'uploads', fileName);

  // Check if file exists
  if (!fs.existsSync(filePath)) {
    return res.status(404).send('File not found');
  }

  // Send the file as a response
```

```
  res.download(filePath, (err) => {
    if (err) {
      console.error('Error sending file:', err);
      res.status(500).send('Error downloading file');
    }
  });
});
```

curl command with auth header

```
wget --header="Authorization: Bearer YOUR_TOKEN" http://localhost:3000/download/filename.pdf
```