**CREATE RESET PASSWORD TOKEN**

Here is the provided code with detailed comments explaining each step:

// Mongoose instance method to create a reset password token

// Define a method on the userSchema to create a reset password token

userSchema.methods.createResetPasswordToken = function () {

// Generate a random token of 32 bytes and convert it to a hexadecimal string

const resetToken = crypto.randomBytes(32).toString('hex');

// Hash the generated token using SHA-256 and store the hash in the user document

this.passwordResetToken = crypto.createHash('sha256').update(resetToken).digest('hex');

// Set the token expiration time to 10 minutes from now

this.passwordResetTokenExp = Date.now() + 10 \* 60 \* 1000;

// Return the original (unhashed) reset token so it can be sent to the user

return resetToken;

};

Let's break down the line of code:

javascript

this.passwordResetToken = crypto.createHash('sha256').update(resetToken).digest('hex');

This line is responsible for hashing the resetToken and storing the hashed value in the passwordResetToken property of the current user object. Here's a detailed explanation of each part:

1. **crypto.createHash('sha256')**:
   * crypto: This refers to Node.js's built-in crypto module, which provides cryptographic functionality.
   * createHash('sha256'): This method creates and returns a hash object that uses the SHA-256 algorithm. SHA-256 is a cryptographic hash function that produces a 256-bit hash value, which is typically rendered as a 64-character hexadecimal string. It is widely used for its security and efficiency.
2. **.update(resetToken)**:
   * The update method is called on the hash object created in the previous step.
   * resetToken: This is the original, randomly generated token that we want to hash.
   * The update method takes the resetToken as input and updates the hash content with the token data. Essentially, it feeds the token into the SHA-256 hash function.
3. **.digest('hex')**:
   * The digest method is called on the hash object after the update method.
   * 'hex': This argument specifies that the output of the digest method should be in hexadecimal format.
   * The digest method computes the hash of the data provided to the update method (in this case, the resetToken) and returns it as a hexadecimal string. This hexadecimal string is the hashed version of the resetToken.
4. **this.passwordResetToken = ...**:
   * this: Refers to the current user object instance, within which the createResetPasswordToken method is defined.
   * passwordResetToken: This is the property of the user object where the hashed reset token will be stored.
   * The hashed token, resulting from the digest('hex') method, is assigned to the passwordResetToken property of the current user object.

**Summary**

This line of code takes the original reset token, hashes it using the SHA-256 algorithm, and stores the resulting hash in the passwordResetToken property of the user object. Hashing the token ensures that even if someone gains access to the database, they cannot use the hashed token to reset the password without knowing the original token. The original token, which is sent to the user, can be compared against this hashed value to verify its validity.