# CS 305 Module Five Coding Assignment Checksum Verification Template

## Instructions

Using the instructions from theModule Five Coding Assignment Checksum Verification Guidelines and Rubric, replace the bracketed text with the relevant information in your own words.

## Algorithm Cipher

I have chosen the SHA-256 (Secure Hash Algorithm 256-bit) as the algorithm cipher for generating checksums.

## Justification

The selection of the SHA-256 algorithm is grounded in its well-established reputation for security and collision resistance. SHA-256 belongs to the SHA-2 family of cryptographic hash functions, specifically designed to offer a high level of security. Collision resistance is particularly crucial when it comes to checksum verification. This property guarantees that it is computationally infeasible to discover two distinct inputs that produce the same hash value. This is of paramount importance because it ensures the accuracy and trustworthiness of the data being verified. In the absence of collision resistance, different data sets could generate identical checksums, leading to potential data integrity issues and security vulnerabilities. SHA-256's robust cryptographic features make it a highly suitable choice for this purpose.

## Generate Checksum

To generate the checksum, I have meticulously refactored the provided code to adhere to a specific set of steps. Initially, I created an instance of the **MessageDigest** class from the **java.security.MessageDigest** library, which is essential for cryptographic hashing. The **MessageDigest** object was initialized with the SHA-256 algorithm cipher using the following code:

MessageDigest digest = MessageDigest.getInstance("SHA-256");

Subsequently, the code calculated the hash value for the unique data string "Bradly Van Hoorebeke" by converting it to bytes and employing the **digest()** method of the **MessageDigest** class:

byte[] hashBytes = digest.digest(data.getBytes());

The hash value was then transformed into a hexadecimal representation using a custom **StringBuilder**. This process entailed iterating through the bytes of the hash value, converting each byte to its hexadecimal counterpart, and appending it to the **StringBuilder**. The final hexadecimal string was stored as the checksum:

StringBuilder hexString = new StringBuilder();

for (byte hashByte : hashBytes) {

String hex = Integer.toHexString(0xff & hashByte);

if (hex.length() == 1) {

hexString.append('0');

}

hexString.append(hex);

}

String checksum = hexString.toString();

To provide the information to the web browser, a RESTful route was established using the **@RequestMapping** method. This route generated and conveyed all the necessary information, including the hash value, the data string "Bradly Van Hoorebeke," and the algorithm cipher used (SHA-256) in the response. This comprehensive code modification empowers the server to compute the checksum of the unique data string and present all the requisite details in the web browser.

## Verification

Insert a screenshot below of the web browser with your unique information.

