The language used in the provided code is JavaScript, specifically for a Node.js environment using an HTTP handling library/framework (potentially Express.js) and the bcrypt library for password hashing.

There are a few potential security and logic issues with the code:

1. **Hashing the Password on Login**:

   - Typically, password hashing is done during user registration or when updating a password, not during login. During login, the usual process is to hash the provided password with the same salt used during registration and compare it to the stored hash.

   - The code appears to be hashing the password upon login, which doesn't make sense unless it's part of a larger login process not shown here.

2. **No Username or Email Handling**:

   - The code only retrieves the password from the request body. For a login process, you'd typically also need a username, email, or some other identifier to determine which user is trying to log in.

3. **No Comparison with Stored Hash**:

   - To authenticate a user, you'd need to compare the hashed version of the input password with the stored hash in the database. This step is missing from the provided code.

4. **No Response to the Client**:

   - After processing the request, there's no response being sent back to the client. This would leave the client hanging and waiting indefinitely.

5. **Lack of Error Handling**:

   - There's no error handling in the code. Operations like `bcrypt.genSalt()` and `bcrypt.hash()` could fail for various reasons, and it's important to handle such failures gracefully, sending appropriate error messages or responses to the client.

To enhance security and functionality:

- For the login process, retrieve the stored hash and salt for the given user (identified by username, email, etc.) from the database.

- Hash the provided password using the retrieved salt and compare the result with the stored hash to authenticate the user.

- Always send a response to the client, whether it's a success message, error message, or other relevant information.

- Implement error handling to manage potential failures and provide feedback to the user.

- If this is indeed the login route, it should also handle user identification (like a username or email) and provide some kind of session or token upon successful login.

Remember to always consult best practices and guidelines specific to the frameworks and libraries you're using.