E-Commerce Web App — AI-Assisted Development (Report-4)

Tools Used

- **ChatGPT** – Implemented password reset and forgot password flows; introduced token-based reset logic with hashed tokens; added email utility for password reset; assisted with debugging async issues and secure token storage.

- **Claude AI** – Recommended consistent token hashing, modular password utilities, and global async error handling; suggested improving schema methods and centralizing authentication utilities.

- **GitHub Copilot** – Auto-completed Mongoose schema methods, email sending boilerplate, route scaffolding, and controller logic for reset/update password endpoints.

Project Overview

This phase focused on strengthening **user account security and password recovery**, including:

1. **Forgot/Reset Password Flow**
    - Secure token generation for password reset links.
    - Hashing tokens in the database for security.
    - Expiration handling for reset tokens.
2. **Email Integration for Password Reset**
    - Utility function to send reset emails using SMTP.
    - Fallback logs for testing when SMTP fails.

3. **Schema & Controller Enhancements**
    - Added `getResetPasswordToken()` method in the User schema.
    - Updated userController.js with `forgotPassword` and `resetPassword` methods.
    - Integrated `catchAsyncErrors` and global error handling for consistency.
4. **Debugging & Testing**
    - Fixed async issues where token was returned before DB update.
    - Verified token hashing and expiry logic.
    - Added logging for token lifecycle and DB state.

AI Interaction & Prompts

Forgot Password

ChatGPT suggested:

- Generate a random token.
- Hash the token before saving in DB.
- Return plain token to send via email.
- Set expiry (15 min) and save user.

Schema Method Example:

```
signupSchema.methods.getResetPasswordToken = function () {
    const resetToken = crypto.randomBytes(20).toString("hex");
    this.resetPasswordToken =
crypto.createHash("sha256").update(resetToken).digest("hex");
    this.resetPasswordExpire = Date.now() + 15 * 60 * 1000;
    return resetToken; // plain token for email
```

```
};
```

Controller Example:

```
exports.forgotPassword = catchAsyncErrors(async (req, res, next) => {
    const user = await signupModel.findOne({ email: req.body.email });
    if (!user) return next(new ErrorHandler("User not found", 404));

    const resetToken = user.getResetPasswordToken();
    await user.save({ validateBeforeSave: false });

    const resetUrl =
`${req.protocol}://${req.get("host")}/api/v1/signup/password/reset/${r
esetToken}`;
    try {
        await sendEmail({
            email: user.email,
            subject: `Password Recovery`,
            message: `Reset your password: ${resetUrl}`
        });
        res.status(200).json({ success: true, message: `Email sent to
${user.email}` });
    } catch (err) {
        user.resetPasswordToken = undefined;
        user.resetPasswordExpire = undefined;
        await user.save({ validateBeforeSave: false });
        return next(new ErrorHandler(err.message, 500));
    }
});
```

Reset Password

ChatGPT suggested:

- Hash token from URL and match with DB.
- Verify expiry.

- Update password and clear token fields.

Controller Example:

```
exports.resetPassword = catchAsyncErrors(async (req, res, next) => {
    const resetPasswordToken =
crypto.createHash("sha256").update(req.params.token).digest("hex");
    const user = await signupModel.findOne({
        resetPasswordToken,
        resetPasswordExpire: { $gt: Date.now() }
    });
    if (!user) return next(new ErrorHandler("Token invalid or
expired", 404));
    if (req.body.password !== req.body.confirmPassword)
        return next(new ErrorHandler("Passwords do not match", 400));

    user.password = req.body.password;
    user.resetPasswordToken = undefined;
    user.resetPasswordExpire = undefined;
    await user.save();

    sendToken(user, 200, res);
});
```

Impact & Code Highlights

1. **Security Improvements**
    - Reset tokens hashed before saving → prevents token leaks if DB compromised.
    - Tokens expire after 15 minutes → reduces risk window.
    - Password reset endpoints wrapped in `catchAsyncErrors` → consistent error handling.
2. **User Experience**

    - Password reset emails (or console fallback) allow smooth recovery.
    - Clear error messages for expired tokens, mismatched passwords, or missing email.

3. **Code Reusability**
    - `sendEmail` utility can handle other notifications.
    - `catchAsyncErrors` used consistently across controllers.

4. **Debugging & Testing Logs**

```
REQ BODY: { email: 'user@example.com' }
USER FOUND: { _id: '...', email: 'user@example.com' }
RESET TOKEN: c3ee1884fab6b444e6421fba30ad5404b74d7ab8
HASHED TOKEN SAVED:
fe93d961cb00785c9dfb7d43a47d20f1f6ac2e95e4993fe18180203c363bc07d
EMAIL SENT: success / fallback to console
```

Advantages

- Modular and secure password recovery.
- Token lifecycle fully logged and validated.
- Reusable utilities for authentication and email.
- Consistent with prior phase architecture: thin controllers, global error handling, async wrappers.

Limitations

- Email sending depends on SMTP configuration (requires credentials or app password).

- No rate-limiting on forgot password → could be abused.
- Frontend integration with password reset forms ongoing.

Reflection

- ChatGPT: Scaffolded token-based password reset, hash logic, error handling.
- Claude AI: Recommended modular schema methods, centralizing logic, and async error handling.
- Copilot: Auto-completed boilerplate code for schema methods and controller logic.
- Hybrid AI + manual coding workflow reduced debugging time and ensured **secure user account handling**.