Resolving a Security and Debugging Disagreement in a Sprint Retrospective Meeting

Scenario for Practical Communication & Problem-Solving Practice Between Trainees

Title: Resolving a Security and Debugging Disagreement in a Sprint Retrospective Meeting

Roles:

- **Team Lead (Scrum Master/Tech Lead)**
- **Backend Developer**
- **QA Engineer**

Scenario Setup:

The development team recently completed a sprint where they implemented a **new authentication system** for user logins. However, during testing, the QA Engineer discovered **two critical issues:**

1. **Security Concern:** Passwords are stored in **plain text** instead of being **hashed**, creating a security vulnerability.
2. **Technical Bug:** Some users are **unable to log in** due to an **expired authentication token issue**, causing session failures.

The **QA Engineer** strongly believes these issues **must be fixed before deployment**, while the **Tech Lead** is under pressure to release the sprint on time. The **Backend Developer** is caught in the middle, evaluating **how much time** it would take to fix the issues versus meeting the deadline.

This discussion will be a **five-minute practical communication exercise** where trainees **debate, analyze problems, and find solutions together.**

Live Discussion Structure (5 Minutes of Practical Debate & Resolution)

1. Identifying the Problems (Active Listening & Clarification)

💡 **QA Engineer (Trainee 1):**
*"I ran security and functional tests on the authentication system. I noticed that user passwords are stored in **plain text** instead of being **hashed**. This is a major security flaw. Also, some users*

can’t log in because the authentication token expires too quickly. Both issues should be fixed before deployment."

💡 **Team Lead (Trainee 2):**
*"I understand the concerns, but we’re on a tight deadline. Is the password storage issue an **immediate security risk**, or is it just a best-practice improvement?"*

💡 **QA Engineer:**
*"It’s **a critical vulnerability**. If there's a data breach, user passwords will be exposed. We should be hashing passwords using **bcrypt** or another secure method."*

💡 **Backend Developer (Trainee 3):**
*"Regarding the **authentication token issue**, I checked the logs and found that tokens are **expiring too soon** because the refresh logic isn’t working correctly. I can patch this quickly, but I need to check how long fixing the hashing will take."*

---

2. Assessing the Impact & Debugging Discussion (Analyzing the Cause & Solution Options)

💡 **Team Lead:**
"Let’s break this down. First, how serious is the authentication token issue? Is it happening to all users or only some?"

💡 **Backend Developer:**
*"From the logs, about **30% of users** are experiencing it. It happens when they stay idle for too long, and the system doesn’t correctly refresh their tokens. This is an **urgent** fix."*

💡 **QA Engineer:**
*"And for the password storage issue? This is a **security risk**. If we deploy as is, and a hacker gains access to the database, all user credentials are exposed."*

💡 **Team Lead:**
*"So, we have one **technical bug** that is **affecting login functionality** and one **security issue** that **needs refactoring**. How much time do we need to fix both?"*

💡 **Backend Developer:**
*"Fixing the authentication token bug will take about **2-3 hours**. Implementing password hashing and testing it properly will take at least **a full day**."*

💡 **QA Engineer:**
*"Skipping password hashing is **not an option**—it's a **security compliance** issue."*

---

3. Proposing Solutions (Collaborative Decision-Making & Prioritization)

💡 **Team Lead:**
"Alright. If we fix both, the release will be delayed by a day. What are our alternatives?"

💡 **Backend Developer:**
*"For the authentication token bug, I can push a **quick patch** within a few hours."*

💡 **QA Engineer:**
*"For password hashing, we **must** do it before deploying. Could we **focus on hashing passwords for new users first**, then hash the old ones in a background process?"*

💡 **Backend Developer:**
*"That's a good idea. We can **apply hashing for new accounts** immediately and write a **database migration script** to hash existing passwords gradually."*

💡 **Team Lead:**
*"If we follow this plan, we can **fix the login issue today** and complete the password hashing over the next sprint. Does this work?"*

💡 **QA Engineer:**
*"Yes. But I want to ensure the database migration plan is properly **documented** and that all existing passwords get hashed within **the next sprint**."*

💡 **Backend Developer:**
"Agreed. I'll start working on the login issue fix immediately."

💡 **Team Lead:**
*"Alright. I'll update the Product Owner on our decision and **prioritize security in the next sprint**."*

---

4. Communicating with Stakeholders & Making a Decision

💡 **Team Lead (speaking as if reporting to the Product Owner):**
*"After discussing with the team, we've decided to immediately fix the **authentication token issue** so users can log in properly. For security, we will implement **password hashing for all new users now** and migrate existing passwords over the next sprint."*

💡 **QA Engineer:**
"We'll add this to the sprint retrospective notes and ensure it's tracked."

💡 **Backend Developer:**
*"I'll deploy the **login fix first**, then start hashing passwords."*

💡 **Team Lead:**
"Great work, team. Let's execute this plan!"

Key Learning Outcomes from This Exercise:

✅ **Handling Team Disagreements** – Practicing logical reasoning instead of authority-based arguments.
✅ **Debugging a Complex Issue** – Walking through **logs, identifying causes, and structuring a fix**.
✅ **Making a Decision Under Pressure** – Weighing risks and **prioritizing security vs. functionality**.
✅ **Effective Stakeholder Communication** – Presenting the **decision and trade-offs clearly**.