**Epic: User Authentication System**

As a user of the LMS, I want to securely log in and access my dashboard so that I can use the features according to my role (e.g., student, instructor, admin).

**Sprint: *Authentication Feature – Sprint 1 (1 week)***

**User Stories (Use Cases)**

**1. User Story: Login with valid credentials**

* **As a** registered user
* **I want** to log in using my email/username and password
* **So that** I can access my personal dashboard

**Acceptance Criteria:**

* User is redirected to their dashboard after a successful login
* Incorrect credentials show an error message

**2. User Story: Password encryption**

* **As a** developer
* **I want** passwords to be securely encrypted and stored
* **So that** user data is protected

**Acceptance Criteria:**

* Passwords are hashed using a secure algorithm (e.g., bcrypt)
* Plain-text passwords are never stored

**3. User Story: Forgot password**

* **As a** user
* **I want** to reset my password if I forget it
* **So that** I can regain access to my account

**Acceptance Criteria:**

* "Forgot Password" link sends a reset email
* Link expires after a set duration (e.g., 1 hour)
* New password must be different from the old one

**4. User Story: Login with role-based access**

* **As a** user with a specific role
* **I want** my login to redirect me to a role-specific dashboard
* **So that** I access relevant features only

**Acceptance Criteria:**

* Students go to /student/dashboard
* Instructors go to /instructor/dashboard
* Admins go to /admin/dashboard

**5. User Story: Login rate limiting / brute-force protection**

* **As a** security-conscious system
* **I want** to limit login attempts
* **So that** I can prevent brute-force attacks

**Acceptance Criteria:**

* Max 5 login attempts in 15 minutes
* After that, temporarily block the IP or prompt with CAPTCHA

**Tasks Breakdown**

| **Task** | **Assigned To** | **Est. Time** |
| --- | --- | --- |
| Design login form UI | Frontend Dev | 4 hrs |
| Setup authentication backend (JWT/Session) | Backend Dev | 6 hrs |
| Implement password hashing | Backend Dev | 2 hrs |
| Create password reset flow | Backend Dev | 6 hrs |
| Setup role-based routing | Fullstack | 5 hrs |
| Add rate-limiting middleware | Backend Dev | 2 hrs |
| Test all auth flows (unit + integration) | QA | 4 hrs |
| Deploy authentication system to staging | DevOps | 2 hrs |

**QA & Testing**

* Unit tests for login, forgot password, role-routing
* Integration test for login flow
* Security tests for password hashing and session expiration

**Sprint Goal**

Implement a fully functional, secure login feature for the LMS, with password protection and role-based redirection.