To confirm that the entire project, "Student Registration System," works as expected, let's validate each major component's expected behavior and flow. Below is a checklist-like guide to testing the functionality step by step:

**1. Signup Functionality**

**Expected Behavior**

* **Scenario 1**: Admin enters a valid email and password.
  + Password hashing occurs through PasswordUtil.hashPassword().
  + Account is created successfully in the database (AdminDAO writes to AdminEntity table).
  + Admin sees a **"Signup successful! Please login"** message.
  + Admin is redirected to the login page.
* **Scenario 2**: Admin enters an already registered email or leaves fields empty.
  + Error occurs: Redirected back to the signup page with an appropriate error message:
    - "Email already exists."
    - "Email and password cannot be empty."

**How to Test**

1. Launch the signup.jsp page.
2. Fill out the form with a valid **email** and **password**:
   * Confirm the database stores the hashed password.
   * Confirm redirection to login.jsp on success.
3. Test edge cases:
   * Use an already registered email.
   * Leave fields blank.
4. Ensure secure cookie creation (auth\_token stored).

**2. Login Functionality**

**Expected Behavior**

* **Scenario 1**: Admin provides valid credentials.
  + The credentials are validated using AdminDAO.validateLogin() (which uses PasswordUtil.verifyPassword()).
  + Admin is authenticated and redirected to dashboard.jsp.
* **Scenario 2**: Admin provides invalid credentials.
  + Redirected back to the login page with an error message: "Invalid login credentials. Please try again."
* **Scenario 3**: Admin clicks "Forgot Password."
  + Redirected to the password reset page (optional feature, if implemented).

**How to Test**

1. Launch the login.jsp page.
2. Enter:
   * Correct email and matching **hashed** password in the database.
   * Incorrect combinations (invalid email/password).
3. Verify:
   * A valid session is created with a secure cookie (auth\_token).
   * Redirection to dashboard.jsp or back to login.jsp on failure.
4. Check error messaging for failed attempts.

**3. Dashboard Access and AuthFilter Functionality**

**Expected Behavior**

* Authenticated users can access dashboard.jsp.
* Unauthenticated users:
  + Are redirected to login.jsp with: "Please login first."
* Persistent authentication through **session and cookies**:
  + If auth\_token in the session matches the auth\_token cookie value (AuthServlet.isAuthenticated()).

**How to Test**

1. Log in successfully and access dashboard.jsp.
2. Delete authentication cookies:
   * Attempt to access dashboard.jsp directly.
   * Confirm redirection to login.jsp.
3. Log out (if functionality exists) or manually invalidate the session:
   * Test access control to protected routes.

**4. Add, Edit, and Delete Student Records**

**Expected Behavior**

* **Add**:
  + Admin can create new student records.
  + Form validates required fields and submits to StudentServlet with action=add.
  + New record appears in the database.
* **Edit**:
  + Admin can update student data.
  + Changes are reflected in the database and updated on the frontend dynamically.
* **Delete**:
  + Admin can delete student records.
  + The corresponding entry is removed from the database and table.

**How to Test**

1. Verify UI behavior in dashboard.jsp:
   * Add a new student.
   * Edit an existing student’s details.
   * Delete a record.
2. Check the database for the corresponding changes.
3. Test edge cases:
   * Invalid/missing data (e.g., blank fields for name/email).
   * Duplications (e.g., same email for multiple students).

**5. Cookie and Session Management**

**Expected Behavior**

* Valid login sessions:
  + auth\_token stored as a secure, HTTP-only cookie.
  + Sessions expire after the configured timeout (e.g., 30 minutes).
* Invalidated sessions:
  + Users accessing a protected route after session expiry or cookie deletion are redirected to login.jsp.

**How to Test**

1. Verify cookies:
   * Log in and inspect the browser’s cookies.
   * Confirm:
     + Cookie auth\_token exists.
     + HttpOnly and Secure flags are set.
   * Clear cookies and attempt restricted access (dashboard.jsp).
2. Simulate session timeout:
   * Invalidate the session on the server or wait for expiry.
   * Test persistent login attempts.
3. Test logout functionality (if available):
   * Confirm session termination and cookies clearing.

**6. Password Security**

**Expected Behavior**

* Passwords are stored as hashes using PasswordUtil.hashPassword().
* Admin login verifies raw passwords using PasswordUtil.verifyPassword().

**How to Test**

1. Ensure no raw passwords are stored in the database.
2. Use existing hashed passwords:
   * Attempt logins with the correct raw password (success).
   * Attempt logins with incorrect passwords (failure).

**7. Error Handling**

**Expected Behavior**

* Clear and user-friendly error messages for invalid inputs, authentication failure, and unexpected issues.

**How to Test**

1. Trigger various errors:
   * Invalid login credentials.
   * Password mismatch in the signup form.
   * Missing required fields.
2. Perform server-side validation:
   * Use malicious/invalid data inputs (e.g., SQL injection attempts).
   * Observe backend protection and appropriate responses.

**8. Frontend User Feedback**

**Expected Behavior**

* Success and error notices are displayed appropriately:
  + E.g., signup.jsp, login.jsp, dashboard.jsp.
* User-friendly and intuitive UI:
  + Forms provide feedback for invalid inputs.
  + AJAX-based updates dynamically refresh content.

**How to Test**

1. Submit forms with incomplete data and inspect error messages.
2. Ensure success messages are displayed after successful operations.

**Overall Testing Summary:**

By completing these steps, you can validate that the system works as expected. Cookies, sessions, password hashing, role-based access control, and CRUD operations should function properly. Let me know if you'd like assistance automating these tests or anything else!