Toronto Metropolitan University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	**Software Design Architecture**
Course Number:	**COE692**
Semester/Year	**W2025**

Instructor:	**Dr. Faezeh Ensan**

Assignment/Lab Number:	**Lab 2**
Assignment/Lab Title:	**Employing n-Tier Architecture for creating a Web application for the specified project**

Submission Date:	**February 21, 2024 10:00 PM**
Due Date:	**February 21, 2025 11:59 PM**

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Kuganendra	**Aananth**	**501182297**	**2**	**AK**
Sethi	**Ayaan**	**501179118**	**2**	**AS**

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:
https://www.torontomu.ca/content/dam/senate/policies/pol60.pdf

Part 2: Design Based on N-Layer Architecture

In this part of the lab sequence diagrams and a component diagram were created as shown in Figure 2.1, 2.2 and 2.3 respectively for the use-cases implemented by the prototype in lab 2.

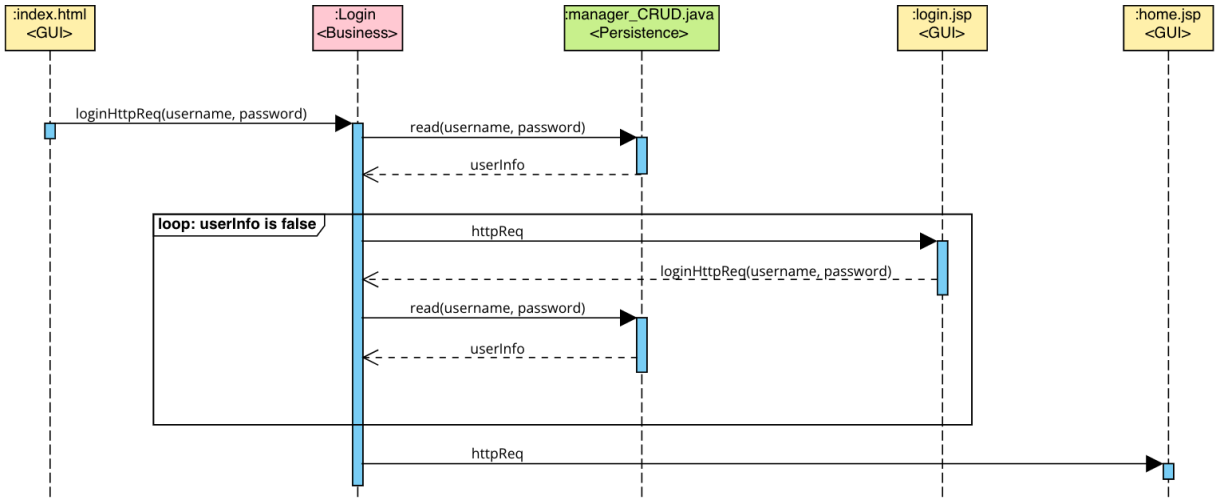

Figure 2.1: Sequence Diagram for Login

1. A user types in their username and password as part of an http request
2. The http request is handled by the Login object
3. The Login object uses the read() method in manager_CRUD to find a manager with the provided credentials. If the credentials are correct for an existing manager, read() returns a boolean, userInfo
    a. If userinfo is false, a loop asks for login credentials again from login.jsp
    b. Else, the user logs-in and is redirected by Login,java to home,jsp

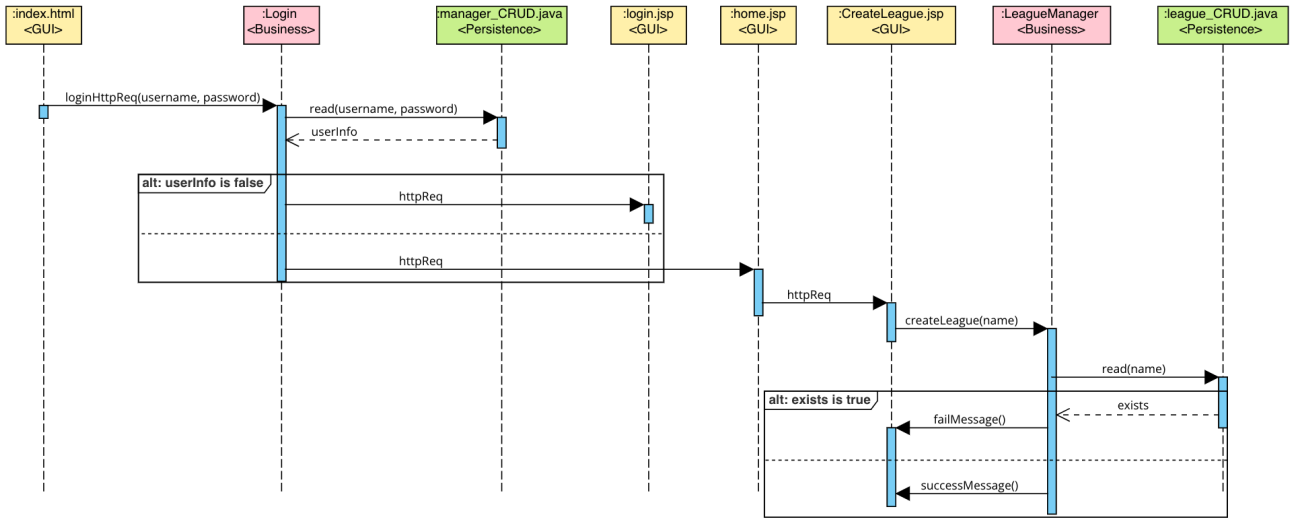

Figure 2.2: Sequence Diagram for Create League

1. A user attempts to login as a manager
    a. If login is unsuccessful they are redirected to login.jsp by Login.java
    b. Else, they are redirected to home.jsp
2. User requests to be redirected to CreateTables.jsp
3. User enters a league to be created as an http request handled by LeagueManager.java
4. LeagueMember.java uses the read() method from League_CRUD, which returns boolean “exists” == true if league exists, to check for an existing league in the database
    a. If exists is true, league name already exists, error message by Login.java
    b. Else, league doesn’t exist, success message is sent, and league is created

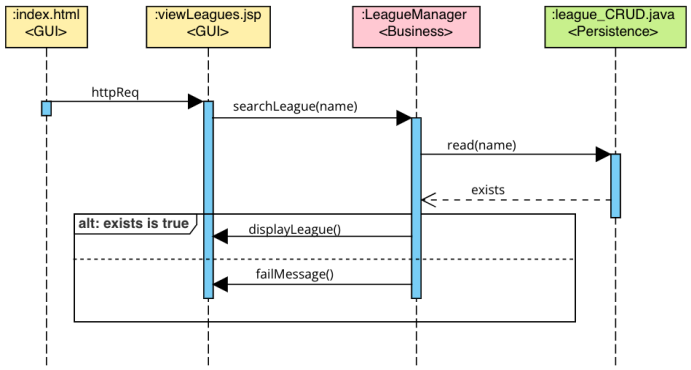

Figure 2.3: Sequence Diagram for Search League

1. A viewer requests to view the leagues via http and is redirected to viewLeagues.jsp
2. The search method in LeagueManager is called to display the league being searched
3. LeagueManager object calls read() method in league_CRUD to check the requested league exists, The method returns boolean exists,
    a. If exists is true, the league exists, displayLeagues() method from LeagueManager is called in viewLeagues,jsp
    b. Else, an error message is displayed

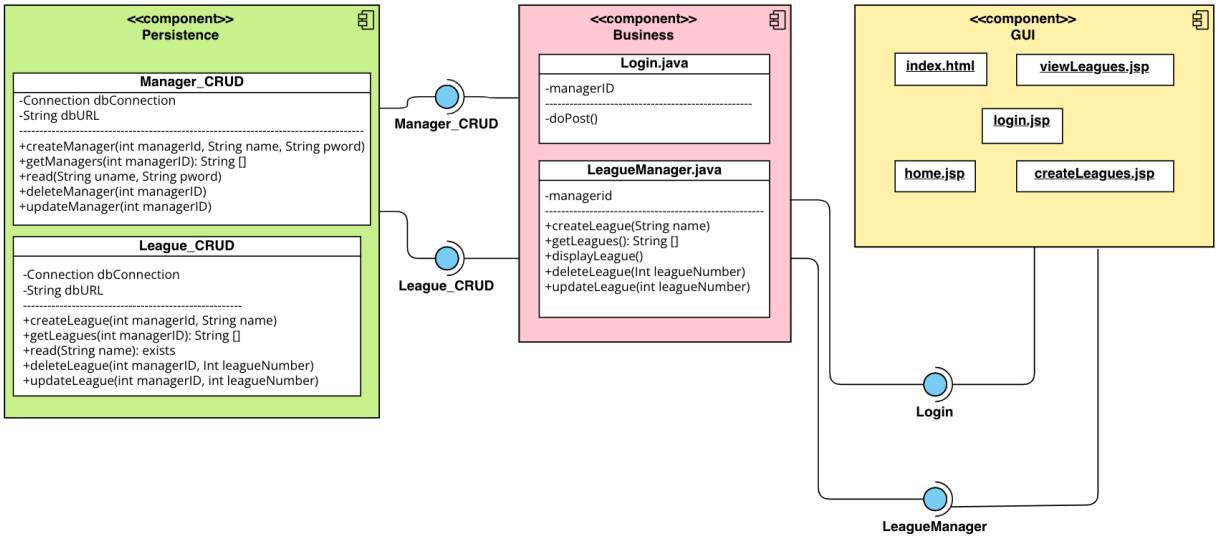

Figure 2.4: Component Diagram

The component diagram consists of three major components for 3 of the 4 layers (database not shown) in this 4-layer software architecture. For this component diagram, only the use cases implemented from the lab 2 prototype are shown.

1. GUI: All html and jsp files that present data to the user. Interfaces from Login and LeagueManager are used from the Business component.
2. Business: Login.java and LeagueManager.java set the logic for logging in as a manager, creating leagues, and viewing them. Interfaces from Login and LeagueManager are provided while Manager_CRUD and League_CRUD are used.
3. Persistence: Manager_CRUD and League_CRUD are responsible for linking the Business component to the database by providing the necessary interfaces. All classes in this component can create, read, update and delete records in the entities, Manager and League.