****

**SDA ASSIGNMENT**

**DEPARTMENT: SOFTWARE ENGINEERING**

**NAME: KAMRAN FIAZ**

**REG NO: SP23-BSE-143**

**SUBJECT: SOFTWARE DESIGN & ARCHITECTURE**

**SEMESTER: 5TH**

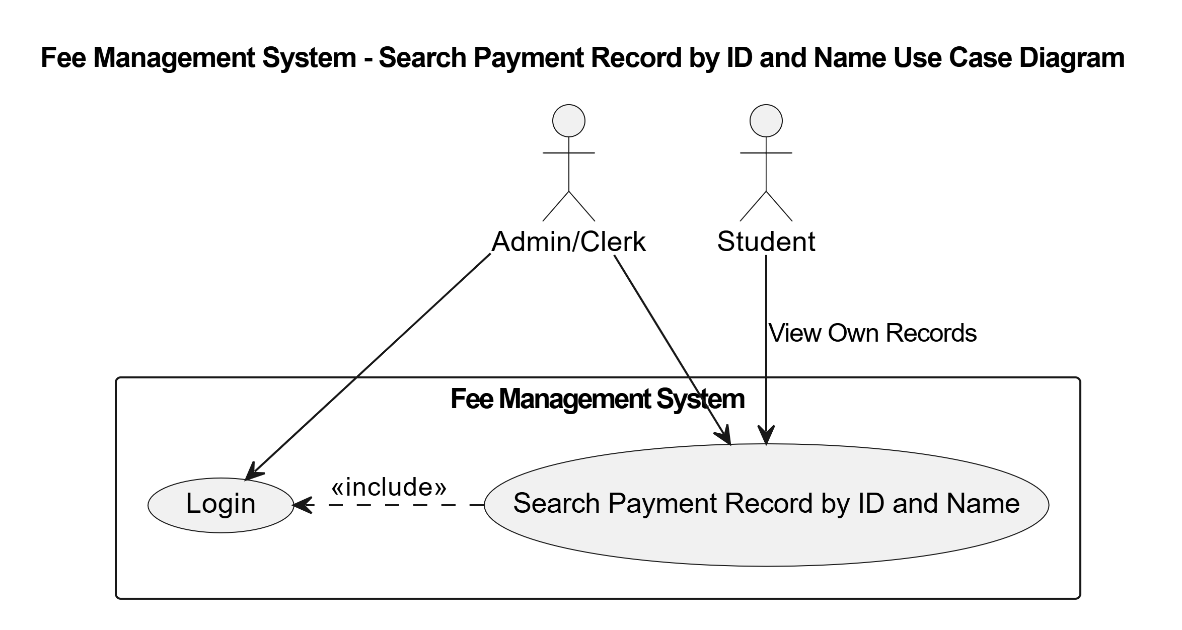
**SUBMITTED TO: SIR MUKHTIAR ZAMIN**

**SUBMISSION DATE : JUNE, 27, 2025**

**PROJECT NAME: FEE MANAGEMENT SYSTEM.**

**USECASE: SEARCH PAYMENT BY ID AND NAM.**

1. **UseCase Diagram:**



1. **Fully Dressed UseCases:**

**Use Case: Search Payment Record by ID and Name**

**Scope:** Fee Management System.  
**Level:** User Goal.  
**Primary Actor:** Admin.

**Stakeholders and Interests:**

* Admin: Wants to quickly search and verify student payment records for fee confirmation or reporting.
* Student: Expects their payment records to be retrievable and accurate.
* System: Ensures data integrity and supports efficient search functionality.

**Preconditions:**

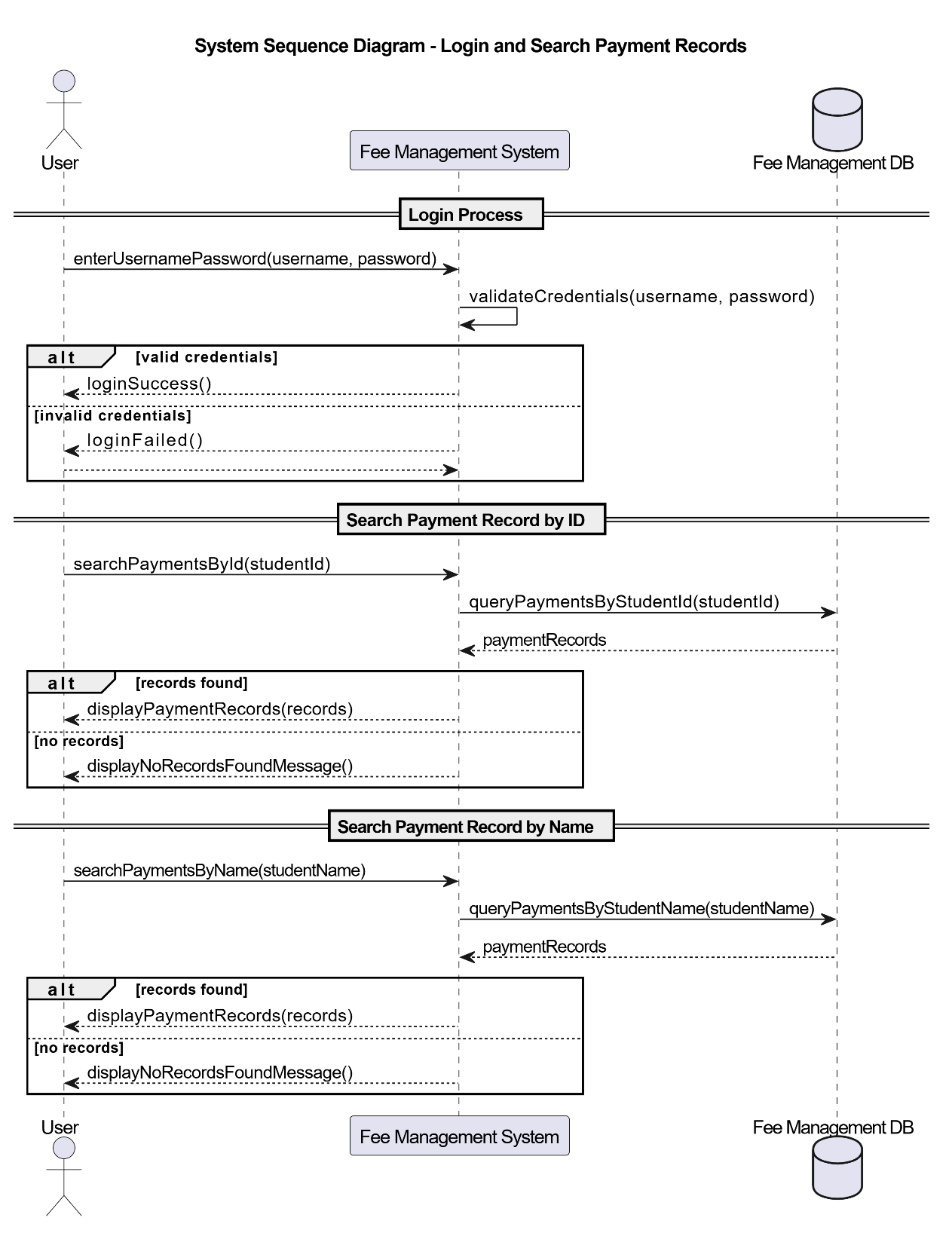
* Admin must be logged into the system.
* Payment records must exist in the system database.

**Postconditions:**

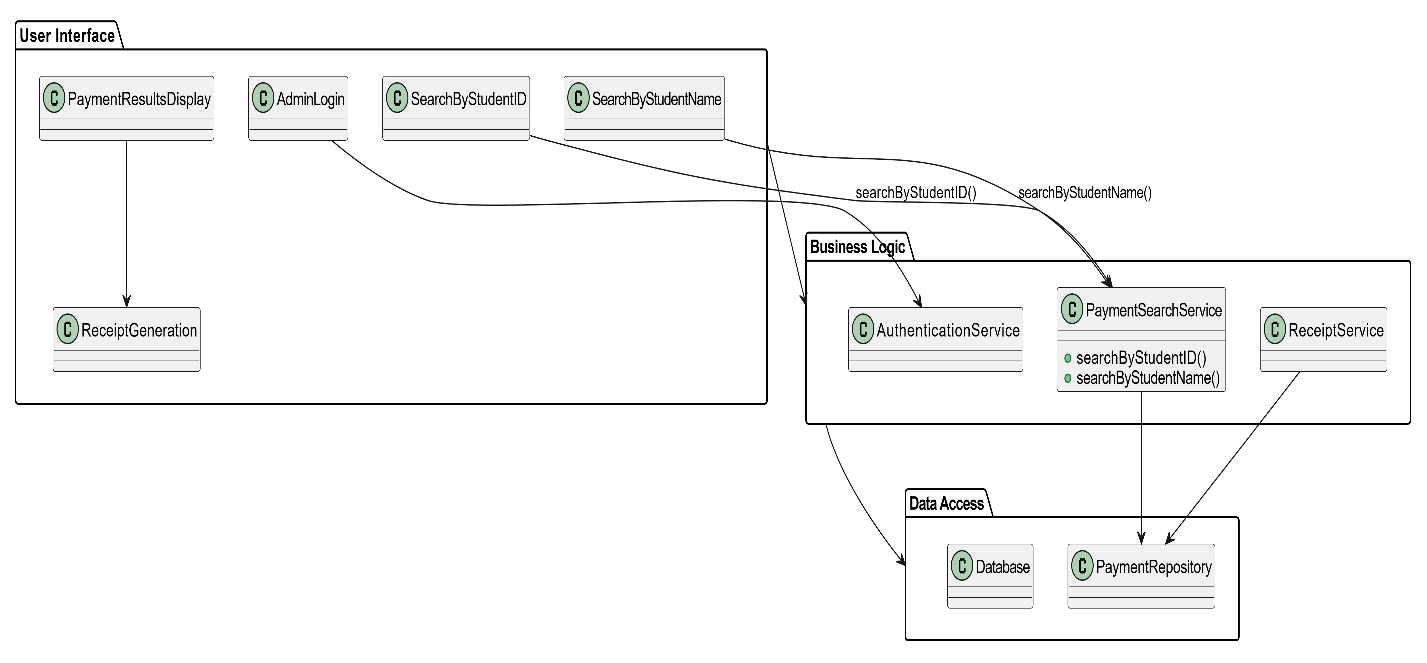
* Payment records matching the entered ID or Name are displayed.
* Optionally, the search results can be downloaded or printed as a receipt.

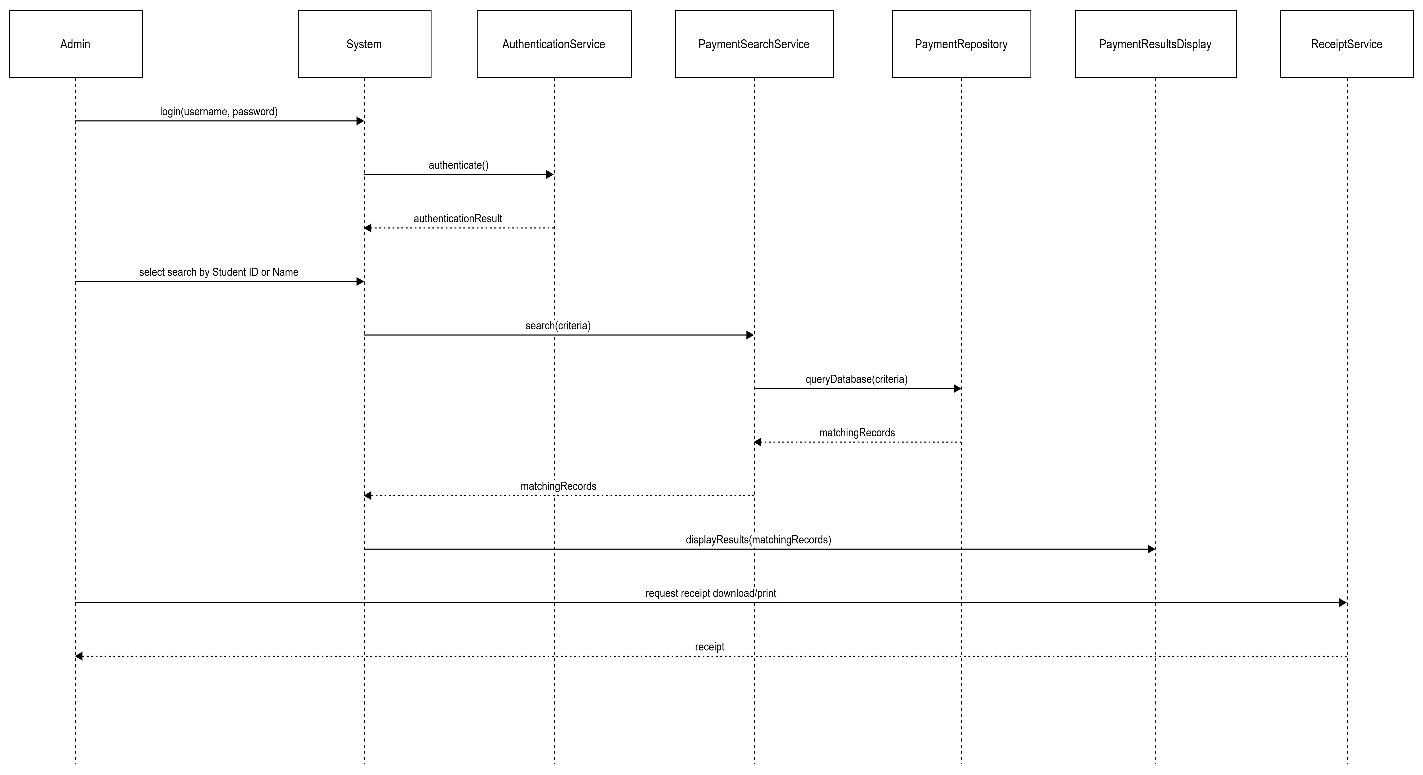
**Main Success Scenario (Basic Flow):**

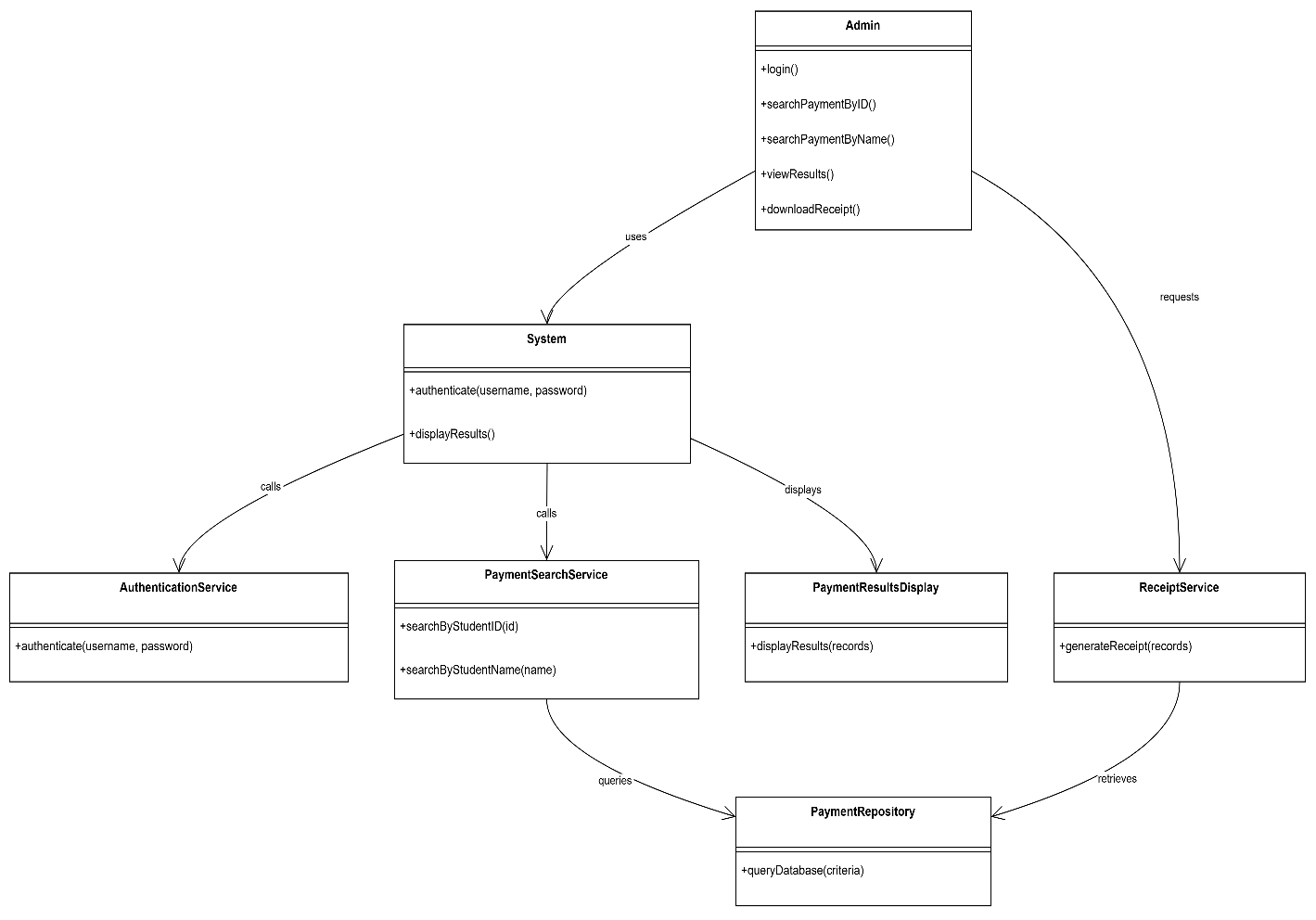
1. Admin logs into the system.
2. Admin navigates to the “Search Payments” section.
3. Admin enters a Student ID or Name in the search field.
4. System queries the database for matching payment records.
5. System displays the list of matching payment transactions (Student ID, Name, Date, Amount, Mode, Status).
6. Admin reviews the displayed records.
7. Admin optionally downloads or prints the search results as a receipt.
8. **System Sequence Diagram:**



1. **Package Diagram:**



1. **Cummunication Diagram:**
2. **Class Diagram:**



1. **Coding Standards:**

**1. Naming Conventions**

* **Class names:** Use **PascalCase**.  
  e.g., PaymentSearchService, StudentRepository.
* **Method names:** Use **camelCase**.  
  e.g., searchByStudentID(), searchByStudentName().
* **Variable names:** Use meaningful **camelCase** names.  
  e.g., studentId, studentName, paymentList.
* **Constants:** Use **UPPERCASE\_WITH\_UNDERSCORES**.  
  e.g., MAX\_SEARCH\_RESULTS.

**2. Method Design**

* Keep each method **single-responsibility**.  
  e.g., searchByStudentID() should only handle ID-based search logic.
* Use **clear input parameter types** (e.g. String id) and **appropriate return types** (e.g. List<Payment>).
* Validate inputs at the start of methods to avoid null or invalid data errors.

**3. Exception Handling**

* Implement **try-catch blocks** for database queries.
* Throw custom exceptions for:
  + **Invalid input** (e.g., empty ID or Name).
  + **Database connection errors**.
  + **No results found** (optional, based on business rules).