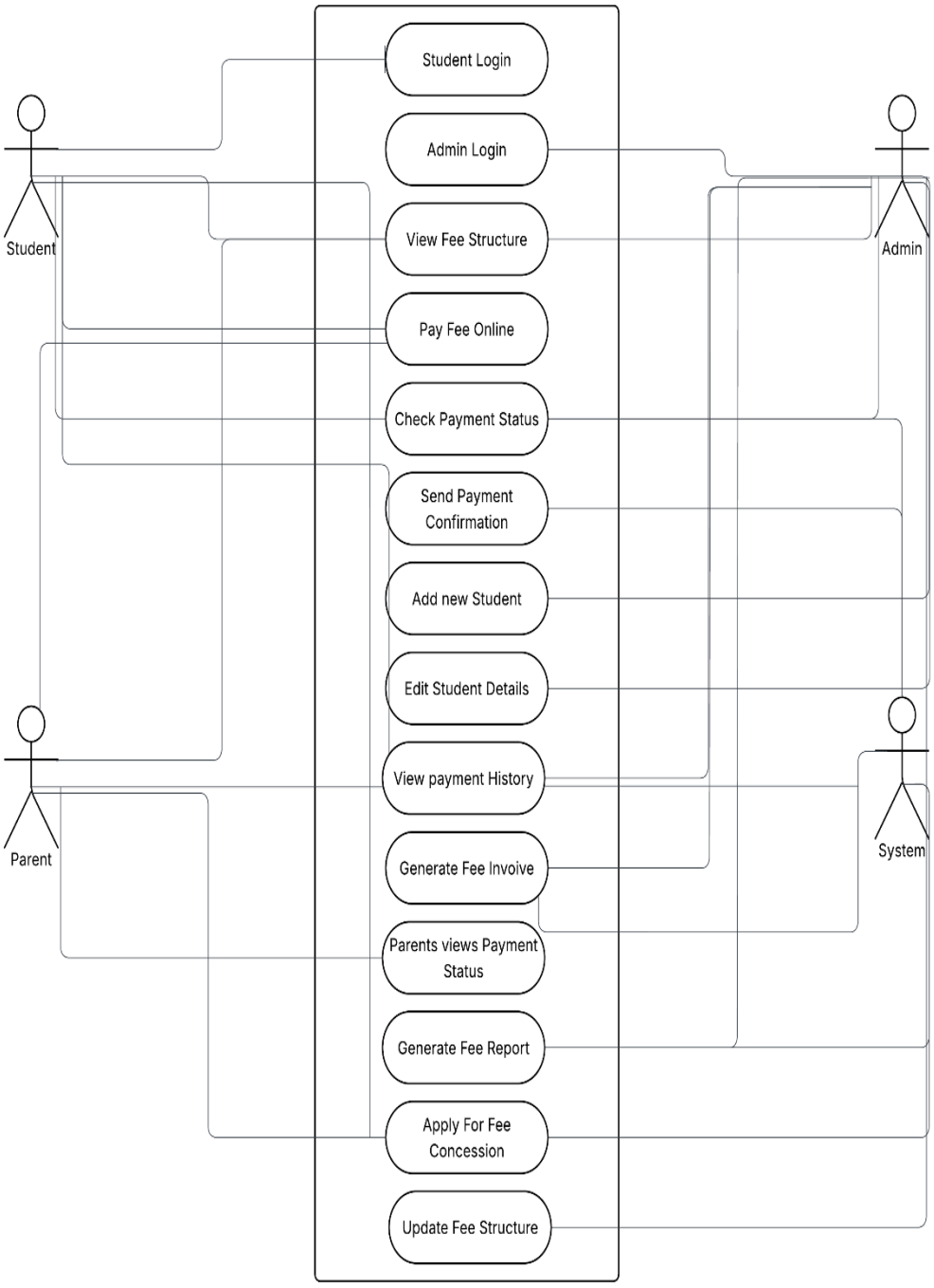
Name:Rafiullah
Sp23-bse-033

Chapter1:Use case for Fee account system



Name:Rafiullah
Sp23-bse-033

Use Case	Admin	Student	Parent	System
1. Admin Login	✓	✗	✗	✗
2. Student Login	✗	✓	✗	✗
3. View Fee Structure	✓	✓	✓	✗
4. Generate Fee Invoice	✓	✗	✗	✓
5. Pay Fee Online	✗	✓	✗	✓
6. Check Payment Status	✓	✓	✓	✓
7. Send Payment Confirmation	✗	✗	✗	✓
8. Add New Student	✓	✗	✗	✗
9. Edit Student Details	✓	✗	✗	✗
10. View Payment History	✓	✓	✓	✓
11. Notify Parent of Due Fee	✓	✗	✓	✓
12. Parent Views Payment Status	✗	✗	✓	✗
13. Generate Fee Report	✓	✗	✗	✓
14. Apply for Fee Concession	✗	✓	✗	✓
15. Update Fee Structure	✓	✗	✗	✗

Name:Rafiullah
Sp23-bse-033

Chapter 2.: fully dress Usecase for Student login.

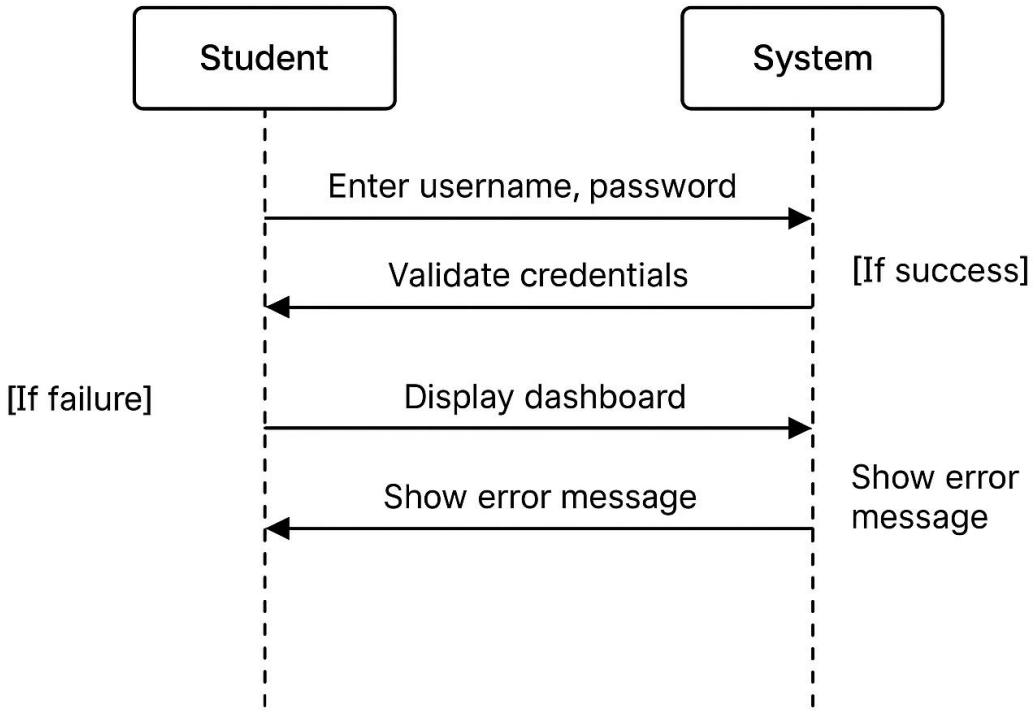
Field	Description
Use Case Name	Student Login
Actor	Student
Description	Student enters username and password to access the system.
Precondition	Student must be registered in the system.
Postconditions	student is logged into the system successfully.
Main Flow	1. Student opens login page. 2. Student enters username and password. 3. System verifies credentials. 4. System grants access to student dashboard.
Alternate Flow	3a. Invalid credentials: - System shows error message. - Admin re-enters credentials.
Exceptions	System maintenance downtime — shows maintenance message.

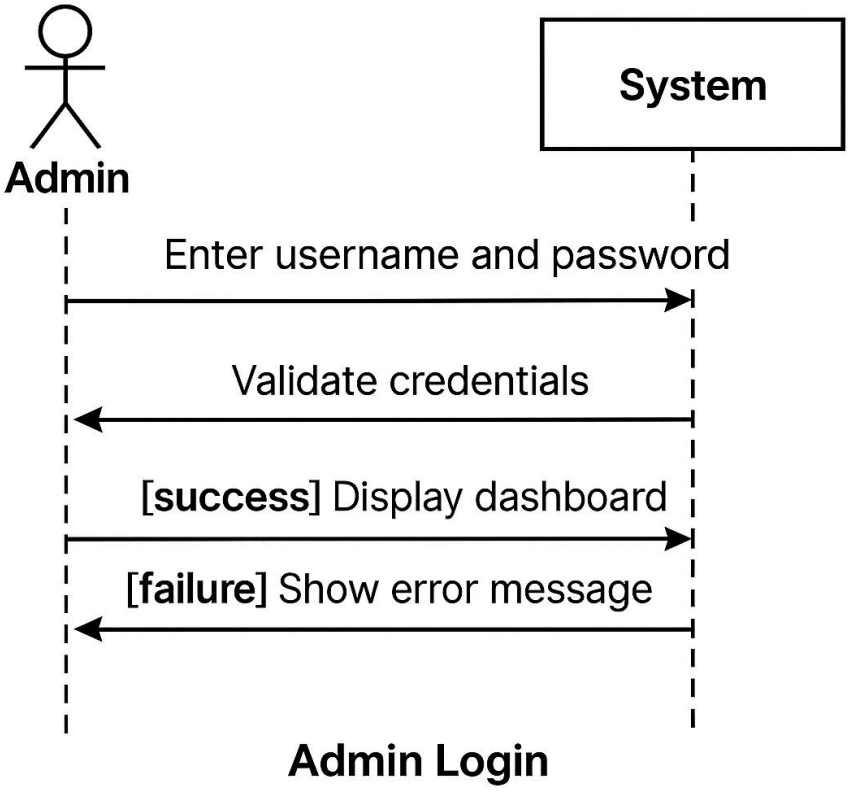
Chapter3:SSD

1.Student login.

2. admin login.

Name:Rafiullah
Sp23-bse-033

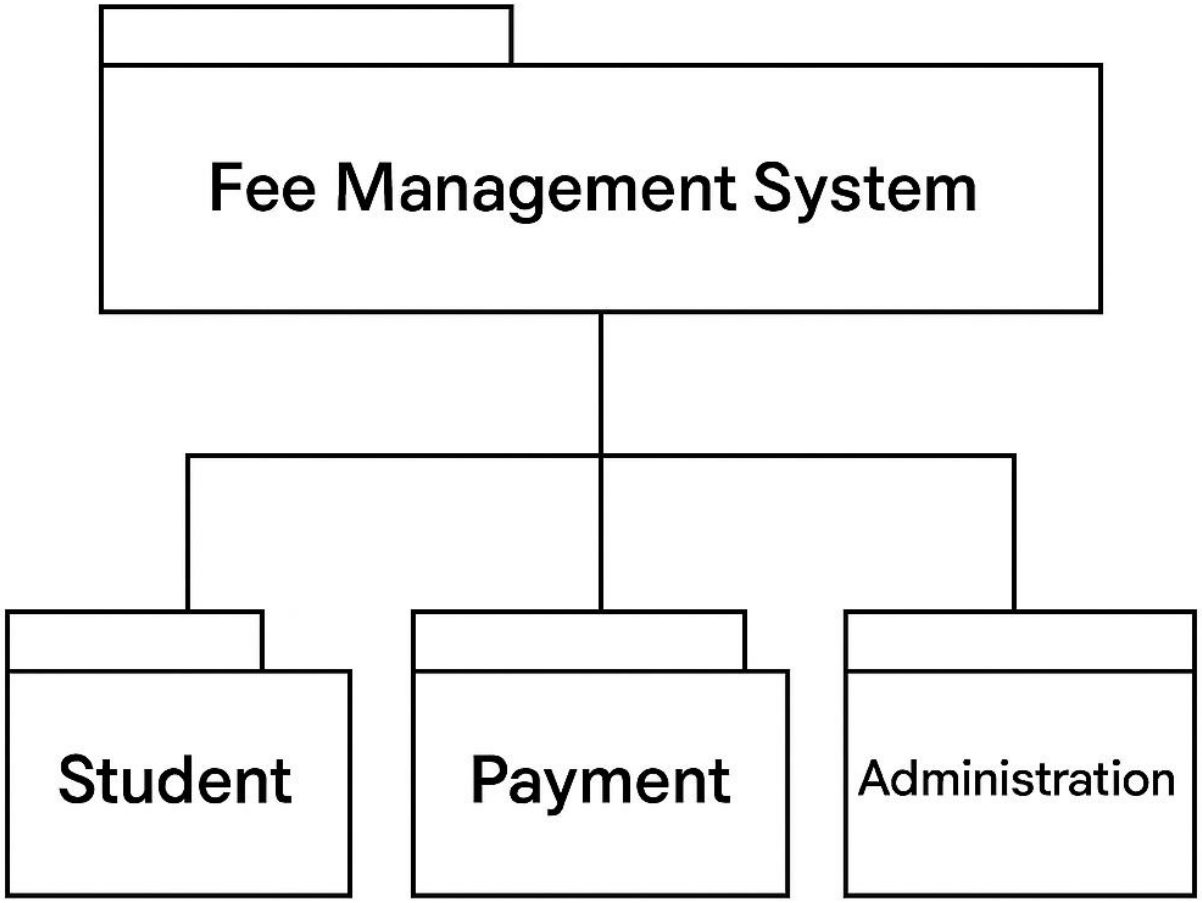




Admin Login

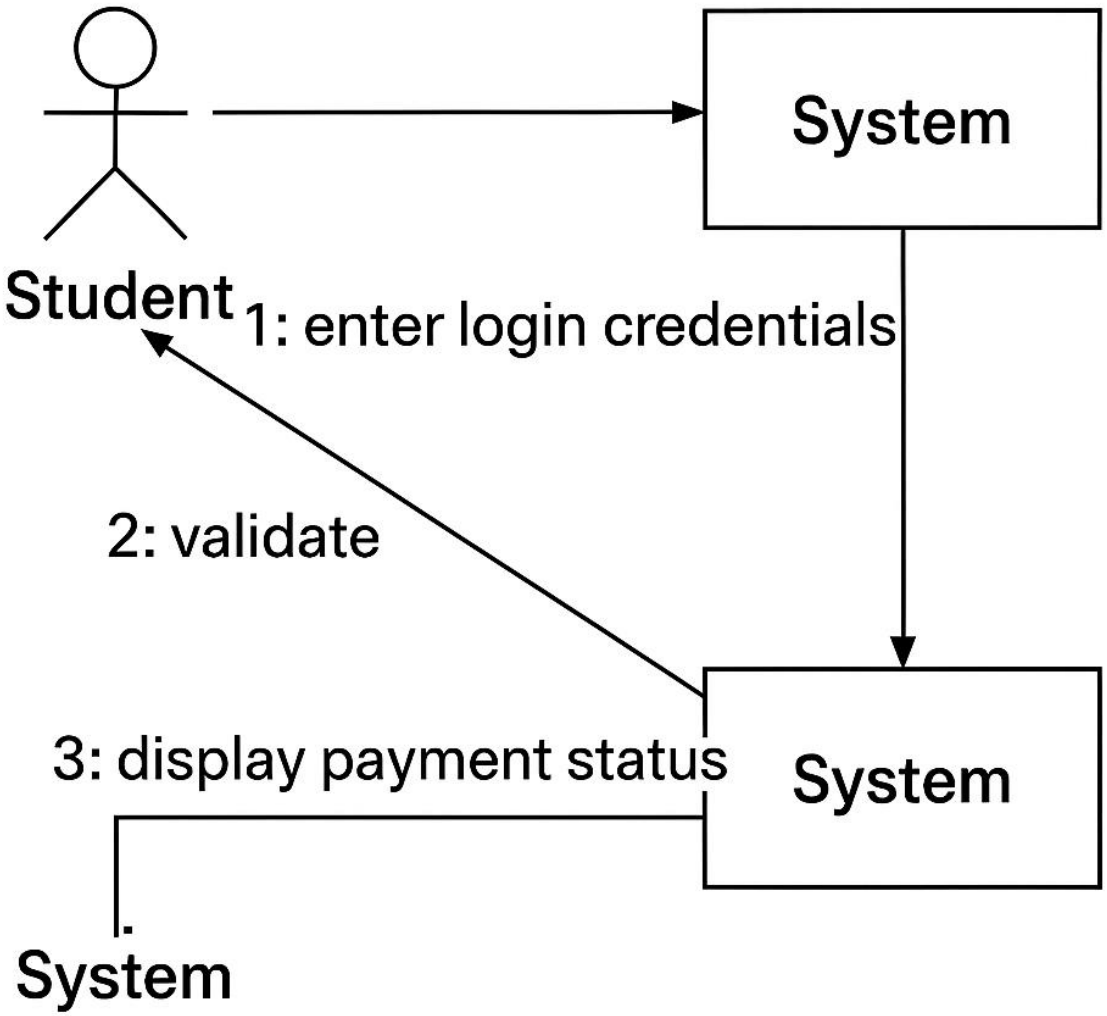
Name:Rafiullah
Sp23-bse-033

Chapter 4. Package Diagram



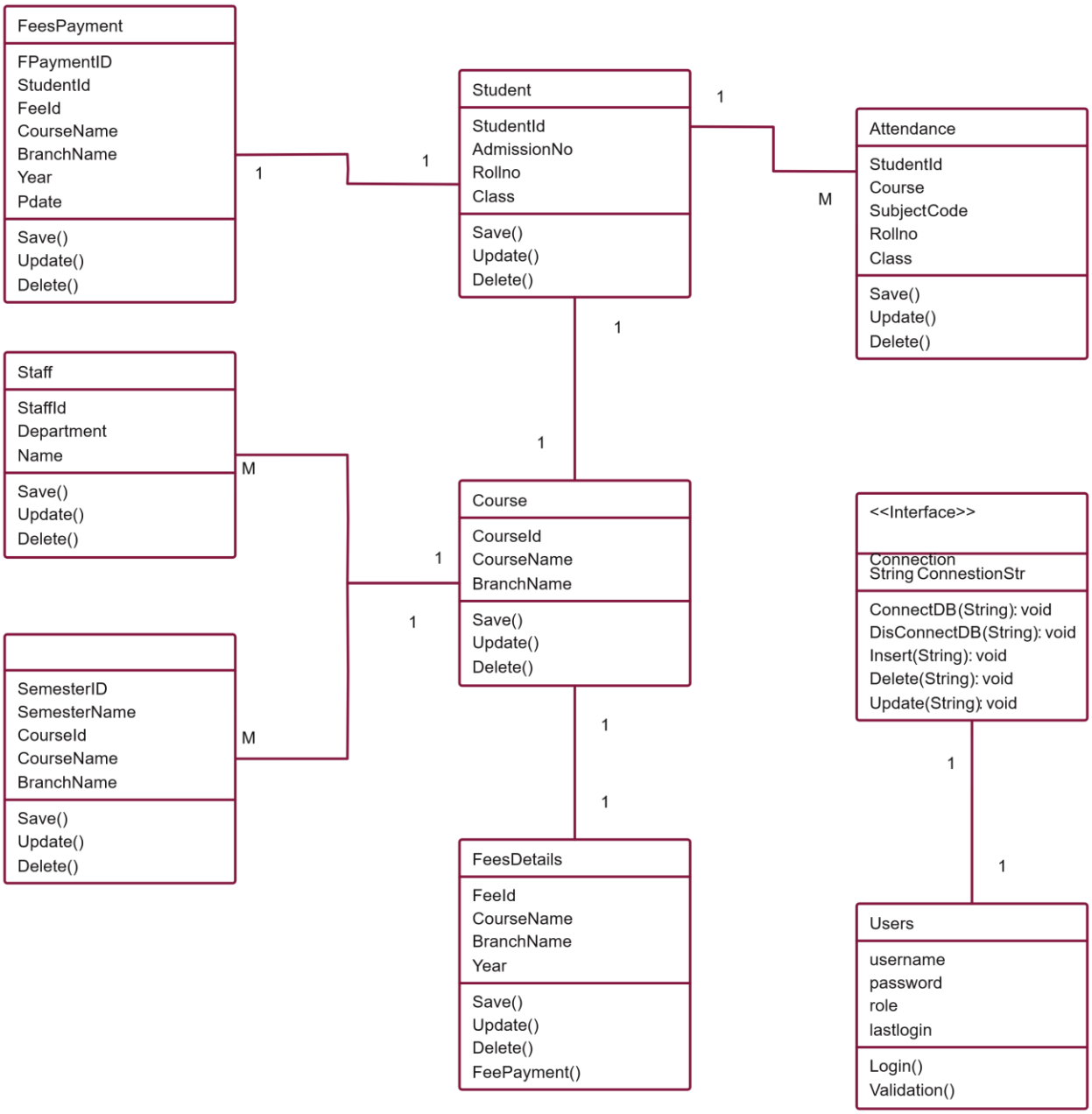
Name:Rafiullah
Sp23-bse-033

Chapter5: Collaboration Diagram



Name:Rafiullah
Sp23-bse-033

Chapter6:Class diagram



Chapter7:coding standards.

Coding Standards Overview

Name:Rafiullah
Sp23-bse-033

1. Naming Conventions

- **Classes: Use PascalCase to denote classes.**
  ***Example*: StudentProfile, FeeTransaction**
- **Methods: Use camelCase for method names.**
  ***Example*: calculateTotalFee(), generateReceipt()**
- **Variables: Use descriptive names in camelCase.**
  ***Example*: studentId, paymentStatus**
- **Constants: Use uppercase letters with underscores.**
  ***Example*: MAX_DISCOUNT, LATE_FEE_PENALTY**

2. File and Folder Structure

Organize your project files logically to enhance maintainability:

bash

CopyEdit

/fee-management-system/

├── /controllers/

│ └── FeeController.java

├── /models/

│ └── Student.java

├── /views/

│ └── paymentForm.jsp

├── /utils/

│ └── DatabaseConnection.java

├── /config/

│ └── application.properties

3. Code Formatting

- **Indentation: Use 4 spaces per indentation level.**
- **Braces: Place opening braces on the same line.**

java

Name:Rafiullah
Sp23-bse-033

CopyEdit

```
public void calculateFee() {
   // code
}
```

- **Line Length: Limit lines to 100 characters.**
- **Spacing: Use spaces around operators and after commas.**

4. Commenting and Documentation

- **Class and Method Comments: Use Javadoc-style comments to describe the purpose and functionality.**

java

CopyEdit

```
/**
 * Calculates the total fee for a student.
 * @param studentId The ID of the student.
 * @return The total fee amount.
 */
public double calculateTotalFee(int studentId) {
   // implementation
}
```

- **Inline Comments: Use sparingly to explain complex logic.**

5. Error Handling

- **Exceptions: Catch specific exceptions rather than generic ones.**

java

CopyEdit

```
try {
   // code
} catch (SQLException e) {
```

// handle exception

}

- **Logging: Use a logging framework (e.g., Log4j) to log errors and important events.**

6. Security Practices

- **Input Validation: Validate all user inputs on both client and server sides.**
- **Password Handling: Store passwords securely using hashing algorithms like bcrypt.**
- **Session Management: Implement proper session handling to prevent unauthorized access.**

7. Version Control

- **Commit Messages: Write clear and concise commit messages.**
  ***Example*: Fix: Corrected fee calculation logic in FeeController.java**
- **Branching: Use feature branches for new features and bug fixes.**