ASSIGNMENT 6

Software Engineering Lab

Member	Enrollment No
Ramesh Chandra Soren	2022CSB086
Moodu Roopa	2022CSB087
Jitendra Kumar Dodwadiya	2022CSB089
Deep Sutariya	2022CSB090

1. Design Objectives and Considerations

Primary Goals:

- Automation: The schema must enable automated fee assignment, payment tracking, and reporting.

- Modularity: Different modules (for students, finance, hostel management) need to be supported and segregated logically.

- Auditability: Every financial transaction (payment or fee charge) must be traceable.

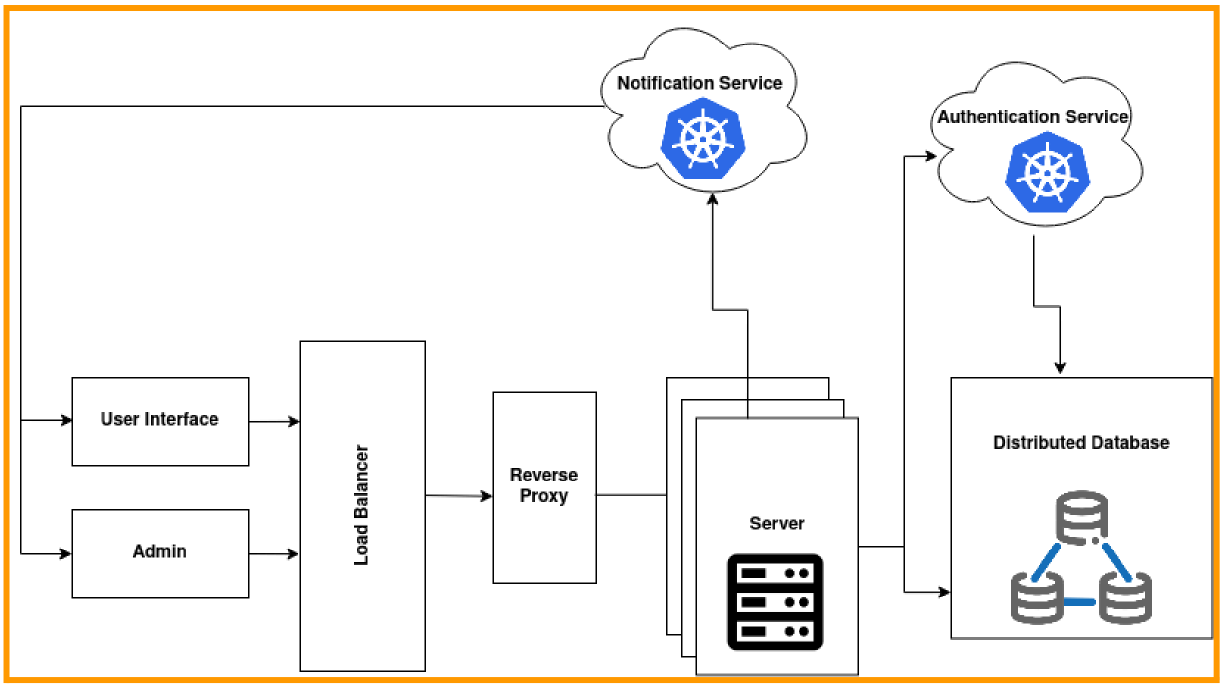
- Flexibility: The design should be extensible to accommodate additional fee types (e.g., library, lab fees) or user roles (such as scholarship administrators).

Key Considerations:

- User Roles and Transactions: Although various users interact with the system (students make payments; finance monitors and processes transactions; hostel wardens oversee hostel fee compliance), the underlying data must be unified so that reports can be generated at both module and global levels.

- Normalization vs. Performance: Start with a fully normalized design (typically 3NF or BCNF) to avoid redundancy and anomalies, and later consider selective denormalization for frequently queried reports.

- Data Integrity & Audit Trails: Use constraints (primary keys, foreign keys, and check constraints) to enforce business rules and maintain accuracy.


Notification Service
Authentication Service
User Interface
Admin
Load Balancer
Reverse Proxy
Server
Distributed Database

2. Identify Major Entities and Their Attributes

Based on the requirements, you can derive the following key entities:

a. Student

- Attributes:
  - `student_id` (Primary Key)
  - `first_name`
  - `last_name`
  - `date_of_birth`
  - `email`
  - `phone_number`
  - `address`
  - `program` or `course`
  - `enrollment_status`
  - `hostel_resident` (Boolean or a foreign key if resident in hostel)

b. Fee_Type (or Fee_Category)

- Attributes:
  - `fee_type_id` (Primary Key)
  - `fee_name` (e.g., Tuition, Hostel, Library, Lab)
  - `frequency` (e.g., one-time, monthly, semester)

c. Fee_Structure

This entity defines what fee amount applies to which program, grade, or student group.

- Attributes:
  - `fee_structure_id` (Primary Key)
  - `fee_type_id` (Foreign Key referencing Fee_Type)
  - `program/grade` (or another grouping attribute)
  - `amount`
  - `effective_date`
  - `status` (active/inactive)

d. Payment (or Transaction)

Records every fee payment or charge event.

- Attributes:
  - `payment_id` (Primary Key)
  - `student_id` (Foreign Key referencing Student)

- `fee_structure_id` (Foreign Key referencing Fee_Structure or directly fee_type if fee amounts are general)
- `transaction_date`
- `amount`
- `payment_method` (e.g., cash, card, online transfer)
- `transaction_type` (credit for payments; debit for fee charges)
- `remarks` (for manual adjustments or notes)

e. Hostel_Details (For Hostel Fee Collection and Management)

If hostel fee management has special data and is overseen by hostel wardens:

- Attributes:
  - `hostel_id` (Primary Key)
  - `hostel_name`
  - `warden_id` (Foreign Key referencing Hostel_Warden)
  - `location`
  - `contact_info`

f. Hostel_Warden

Represents the user responsible for managing hostel fees.

- Attributes:
    - `warden_id` (Primary Key)
    - `first_name`
    - `last_name`
    - `email`
    - `phone_number`

g. Finance_User (Optional)

For the finance department staff that process fee collections and generate reports.

- Attributes:
    - `finance_user_id` (Primary Key)
    - `name`
    - `role`
    - `email`
    - `contact_number`

Note: In some designs, finance users may be managed separately from the core financial transactions; however, it's helpful to include them if you want role-based access within the application.

3. Defining Relationships Among Entities

a. Student – Fee_Structure / Fee_Type Relationship

- Association:
  - Each student is assigned one or more fees.
  - A fee might be applicable based on the student's program or grade.
- Implementation:
  - You can record fee assignments in the Payment (Transaction) table rather than duplicating fee information for each student.
  - Use a junction table or simply link the Payment to both `student_id` and `fee_structure_id` so that you know which fee is being charged or paid.

b. Payment (Transaction) as the Central Process

- Association:
  - Each transaction/ payment record is linked to a specific student and a defined fee.
  - Transactions can be either a fee charge (debit) or a received payment (credit), which helps in tracking balances.
- Implementation:
  - Include an indicator (or use sign convention in the `amount` field) to differentiate debits from credits.

    - Ensure referential integrity through foreign keys.

c. Student – Hostel_Details Relationship

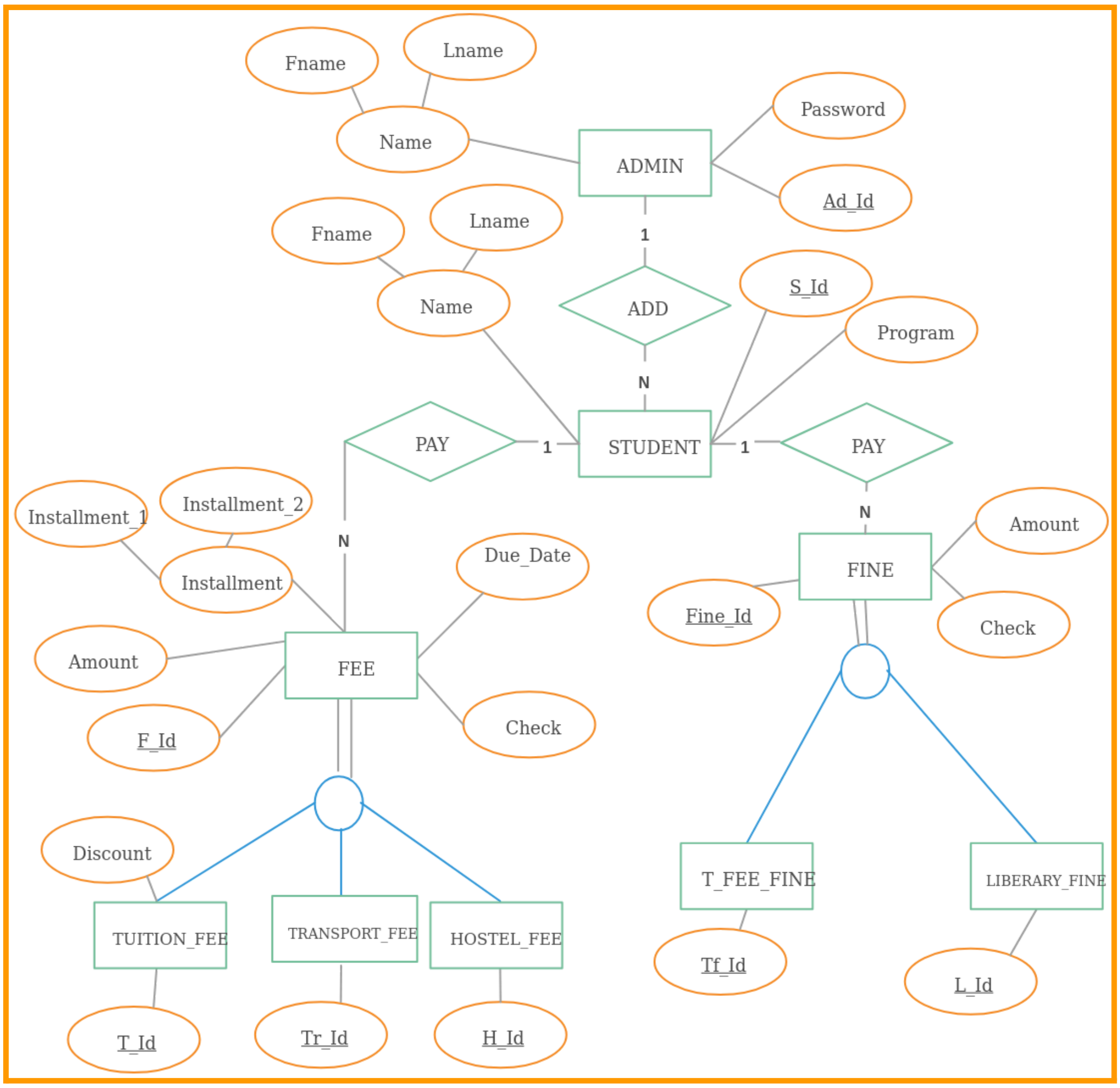
- Association:
    - Not all students may reside in a hostel; those who do should be linked to a hostel record.
- Implementation:
    - This can be a one-to-one (or one-to-many if a hostel can contain many students) relationship.
    - The `Student` table could include a nullable foreign key such as `hostel_id`.

d. Hostel_Details – Hostel_Warden Relationship

- Association:
    - Each hostel is managed by one warden.
    - A hostel warden may oversee one or more hostels.
- Implementation:
    - Use a foreign key in the `Hostel_Details` table referencing the `Hostel_Warden` table.

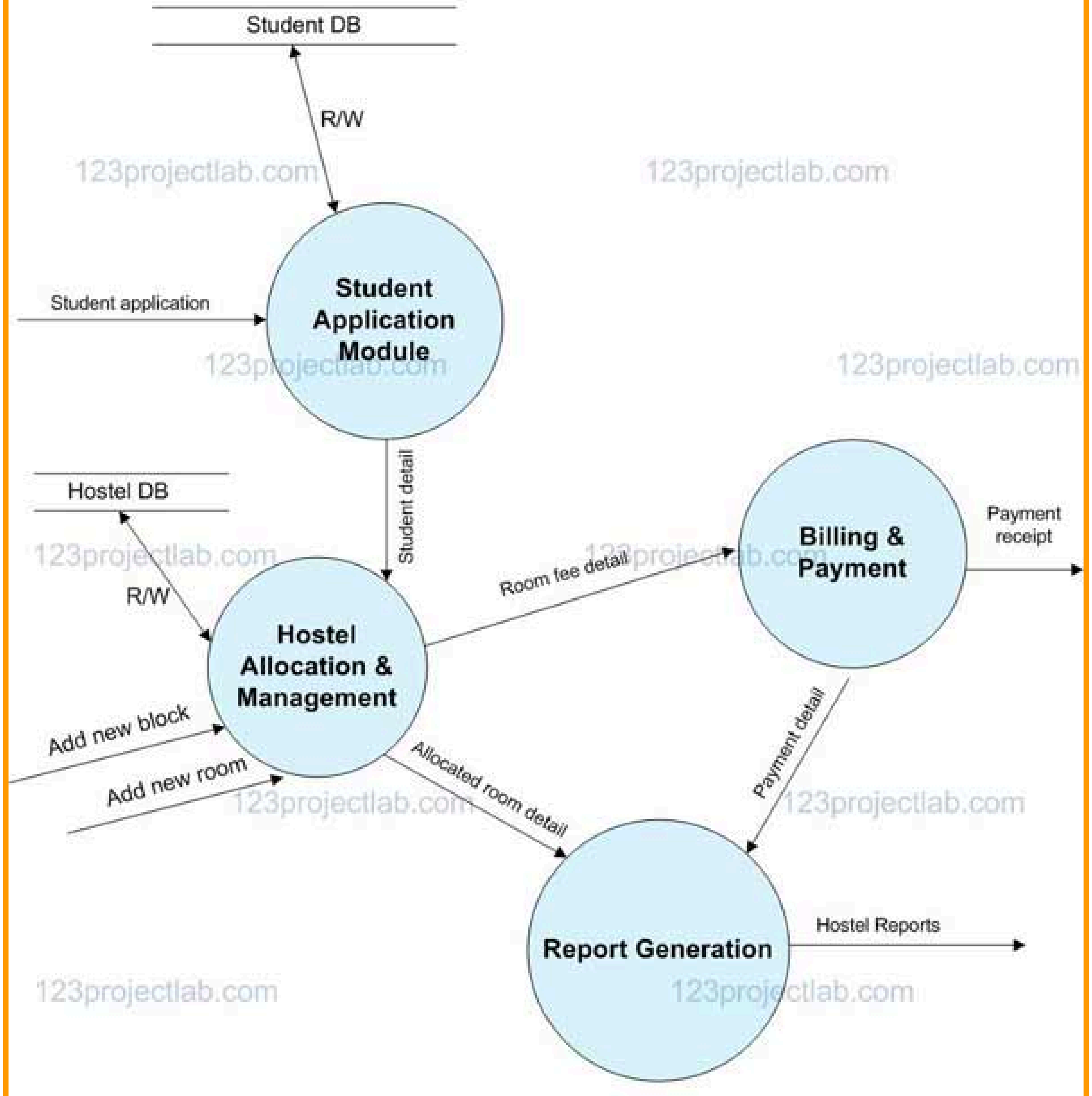
e. Finance and Reporting Views

- Although finance users are not directly part of payment processing, they require interfaces (queries and views) that join the Payment, Student, and Fee-related tables.
- You can create SQL views that aggregate payment histories, fee assignment statuses, overdue amounts, etc., for the finance department.

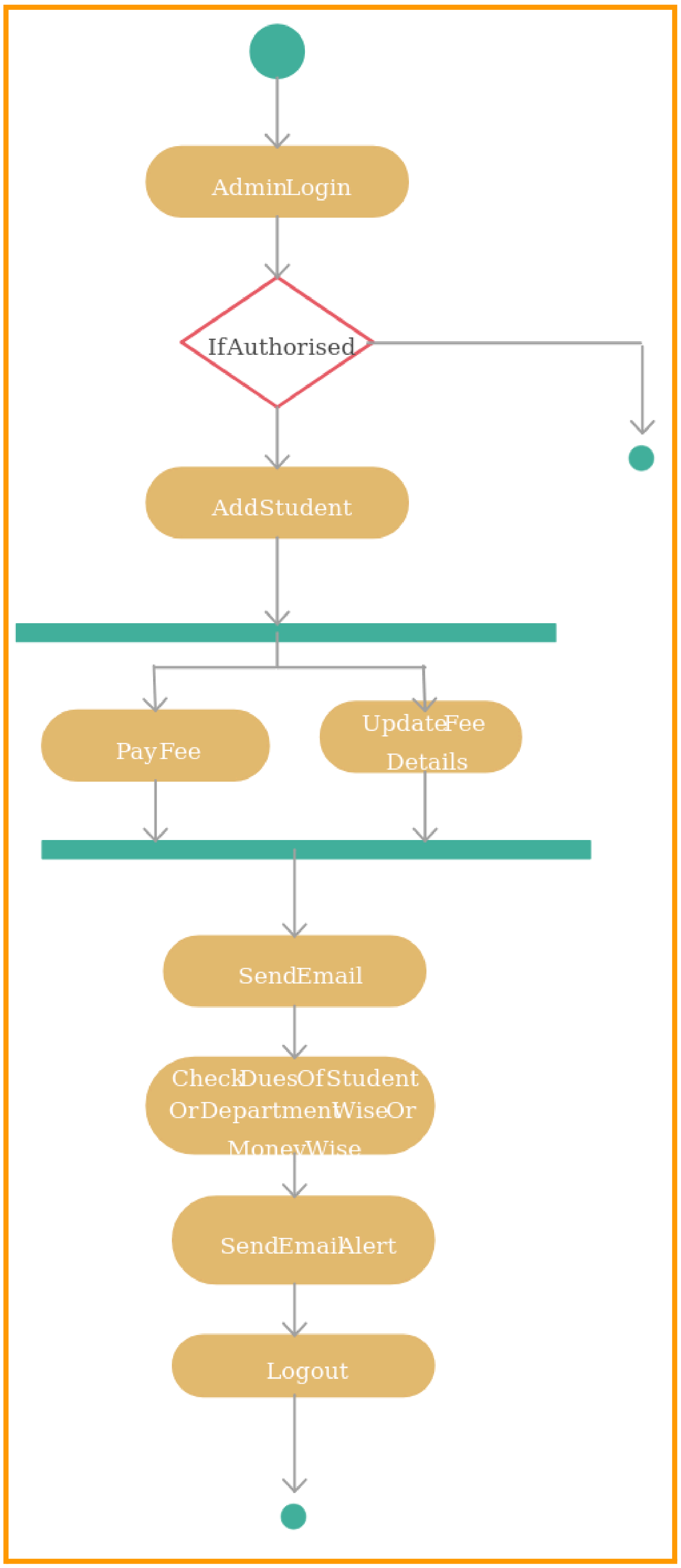


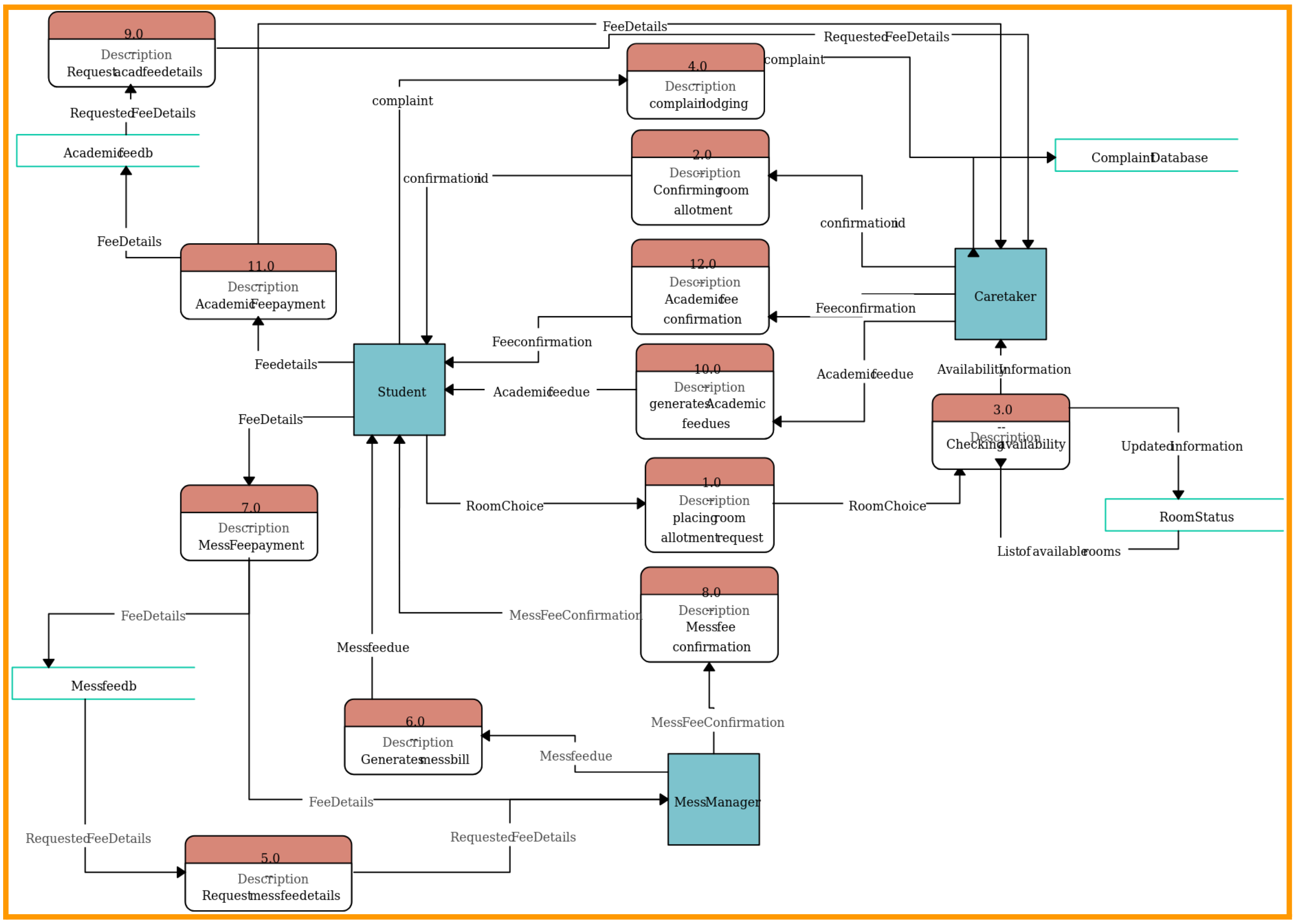
ER-Diagram

Activity Diagram: In **software engineering**, an **activity diagram** is a **UML (Unified Modeling Language)** diagram that is used to model the **workflow of a system**. It visually represents the **sequence of activities, decisions**, and possible **parallel processes** within a software system or a part of it.

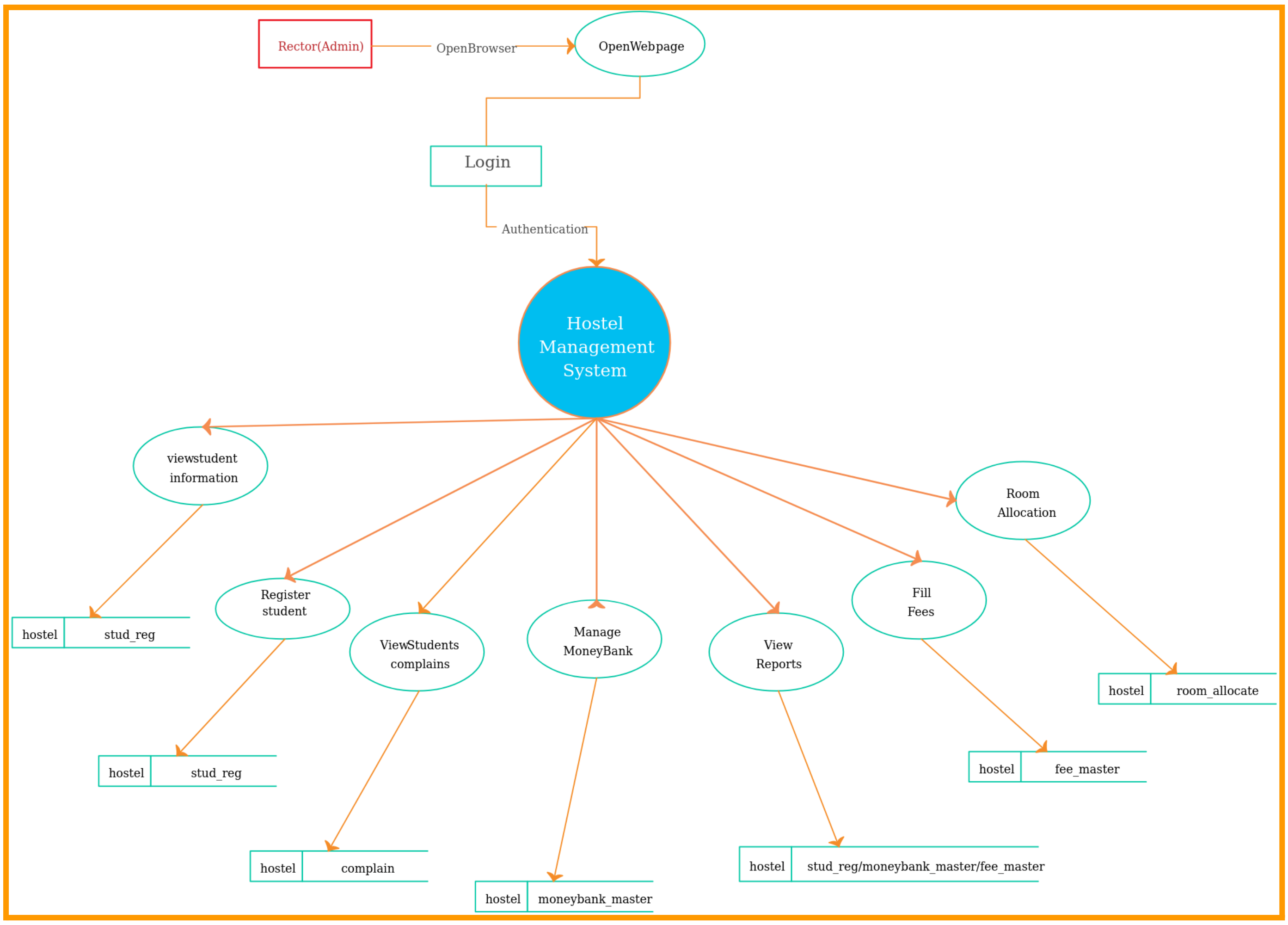


Level 1 DFD for Hostel Management System

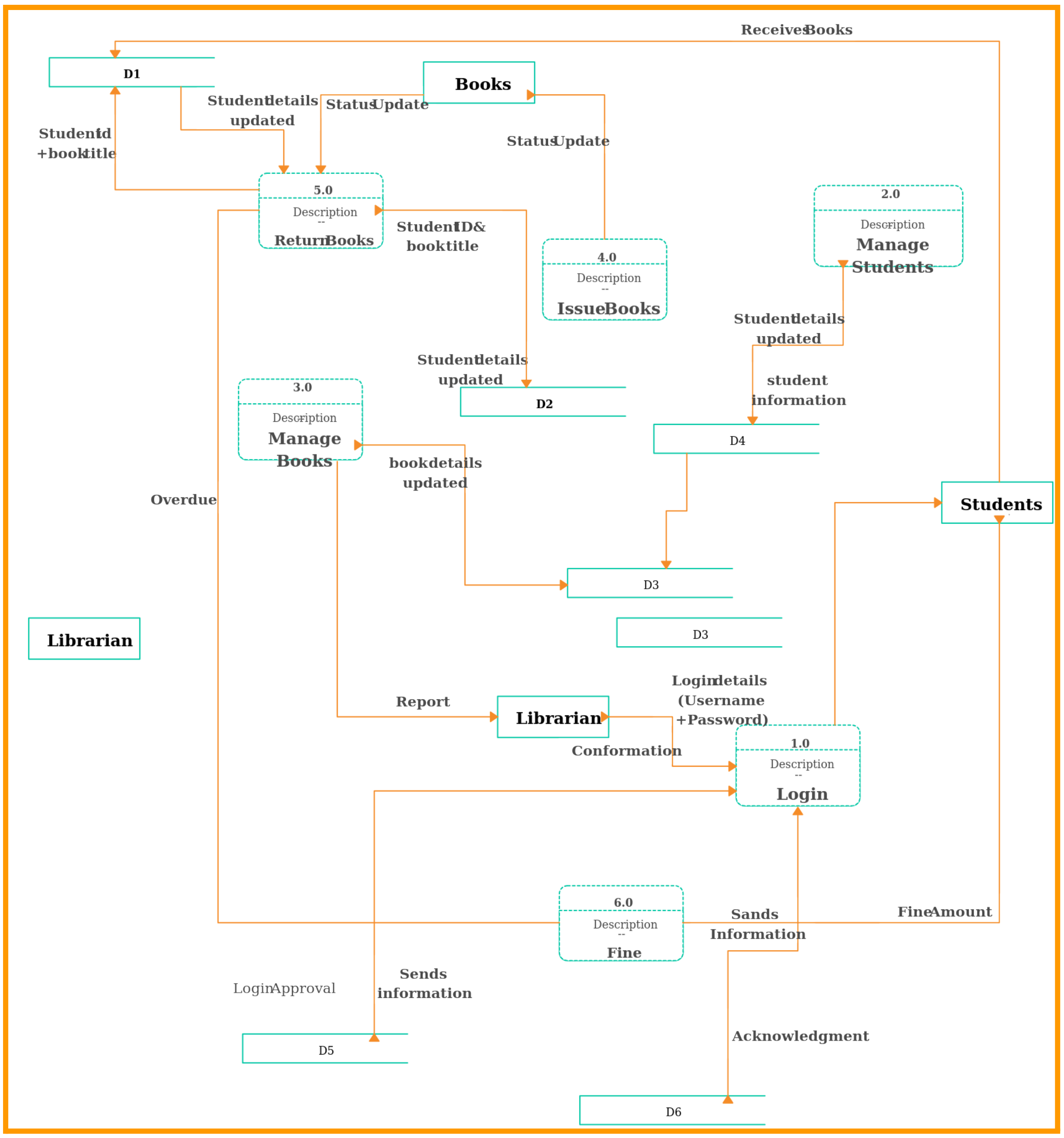

AdminLogin
IfAuthorised
AddStudent
PayFee
UpdateFee Details
SendEmail
CheckDuesOfStudent OrDepartmentWiseOr MoneyWise
SendEmailAlert
Logout



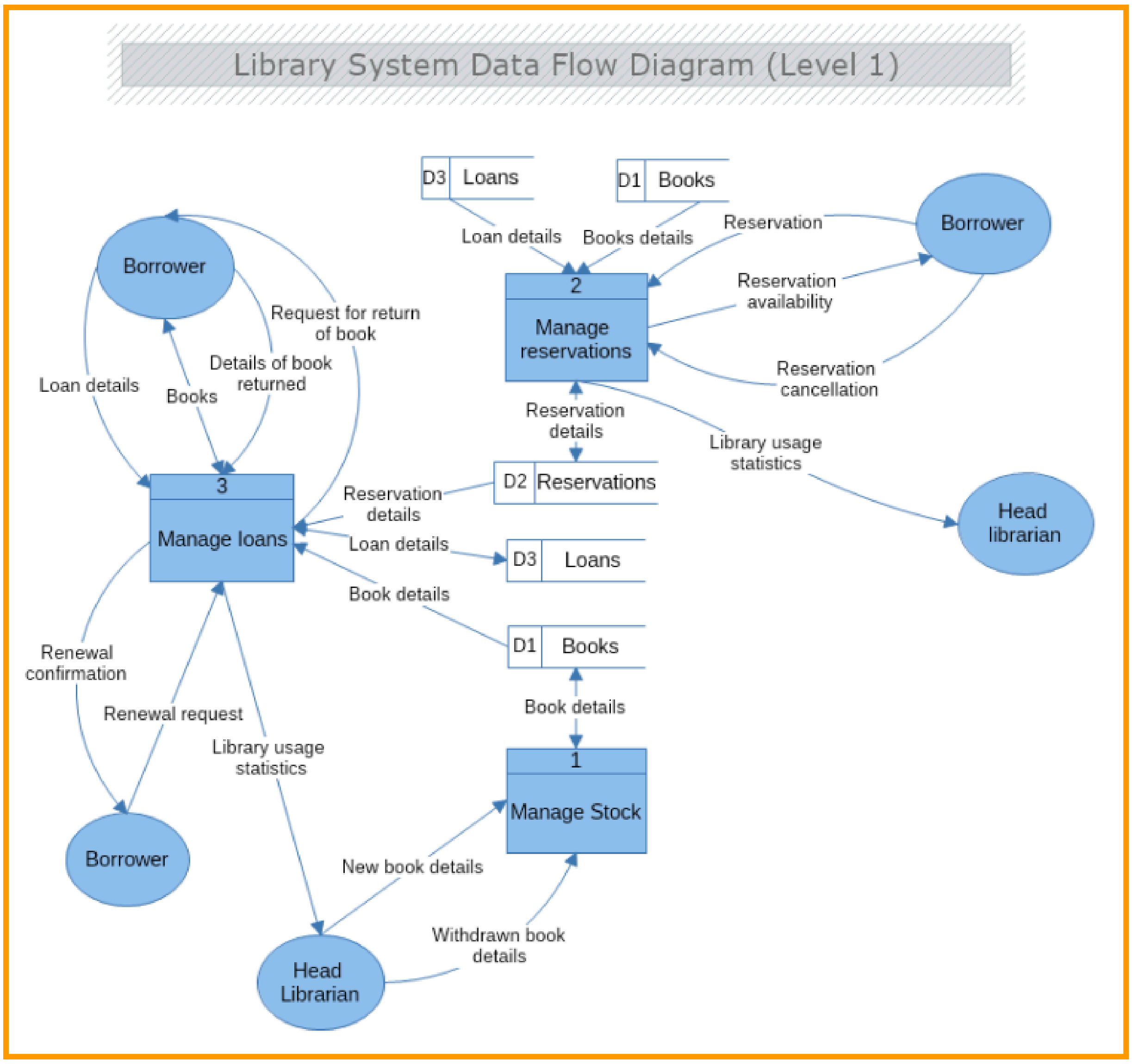
Data Flow Diagram for Hostel management system

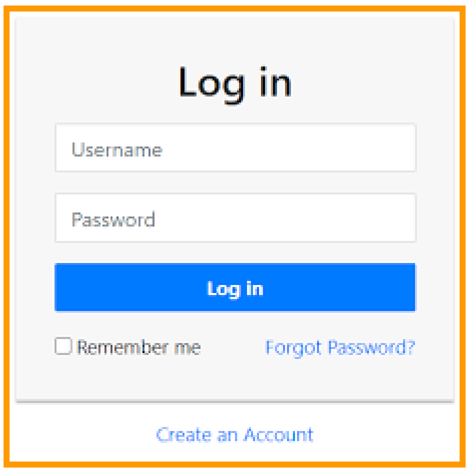


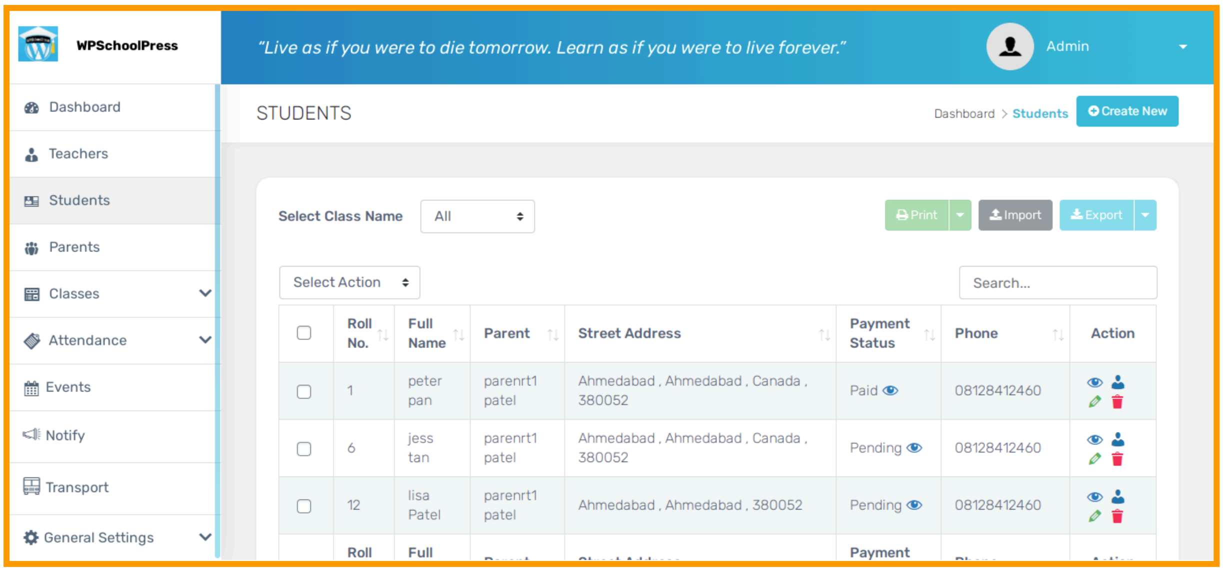
Functions of Hostel Management System



Data Flow Diagram for library management system


Library System Data Flow Diagram (Level 1)
D3 Loans
D1 Books
Loan details
Books details
Reservation
Borrower
Reservation availability
2 Manage reservations
Reservation cancellation
Reservation details
Library usage statistics
Head librarian
Borrower
Request for return of book
Loan details
Books
Details of book returned
3 Manage loans
D2 Reservations
Reservation details
Loan details
D3 Loans
Book details
D1 Books
Book details
1 Manage Stock
Renewal confirmation
Renewal request
Library usage statistics
Borrower
New book details
Withdrawn book details
Head Librarian





"Live as if you were to die tomorrow. Learn as if you were to live forever."

STUDENTS

	Roll No.	Full Name	Parent	Street Address	Payment Status	Phone	Action
☐	1	peter pan	parenrt1 patel	Ahmedabad , Ahmedabad , Canada , 380052	Paid	08128412460	
☐	6	jess tan	parenrt1 patel	Ahmedabad , Ahmedabad , Canada , 380052	Pending	08128412460	
☐	12	lisa Patel	parenrt1 patel	Ahmedabad , Ahmedabad , 380052	Pending	08128412460	