User Stories

As a [user role], I want to [perform an action] so that [I can achieve a goal].

- Your user stories should cover all major functionalities of your system, including user interactions, data storage, retrieval, and updates.

1. **Can register profile:**
   As a student, I want to register my profile in the system so that I can access dormitory services and manage my accommodation needs.

2. **Can view available rooms:**
   As a student, I want to view all available rooms so that I can make informed decisions about my accommodation options.

3. **Filter by type of room:**
   As a student, I want to filter rooms by type so that I can find accommodations that match my preferences.

4. **Filter by type of facilities:**
   As a student, I want to filter rooms by available facilities so that I can find accommodations that meet my specific needs.

5. **Book room:**
   As a student, I want to book a room for a specific period so that I can secure my preferred accommodation.

6. **Submit repair requests:**
   As a student, I want to submit repair requests so that room issues can be fixed.

7. **Track repair status:**
   As a student, I want to track the status of my repair requests so that I I know when issues will be resolved.

8. **Receive reminders for booking deadlines:**
   As a student, I want to receive reminders about booking deadlines so that I don't miss important accommodation reservation periods.

9. **Receive reminders for maintenance updates:**
   As a student, I want to receive notifications about maintenance updates so that I know when repairs in my room will be completed.

10. **Admins can assign rooms:**
    As an administrator, I want to assign rooms so that I can manage dormitory occupancy.

11. **Admins can approve bookings:**
    As an administrator, I want to approve bookings so that I can control room assignments.

12. **Admins can view occupancy reports:**
    As an administrator, I want to view occupancy reports so that I can monitor dormitory capacity and make informed housing decisions.

Identify Key Entity Sets and Relationships

Key Entity Sets

1. **Student**
    - Represents registered users of the dormitory system
    - Contains profile information
2. **Room**
    - Represents individual dormitory rooms
    - Contains room type and facilities information
3. **Booking**
    - Represents room reservations
    - Contains booking dates and status
4. **RepairRequest**
    - Represents maintenance requests
    - Contains issue description and status
5. **Admin**
    - Represents system administrators
    - Contains admin credentials and permissions

Relationships

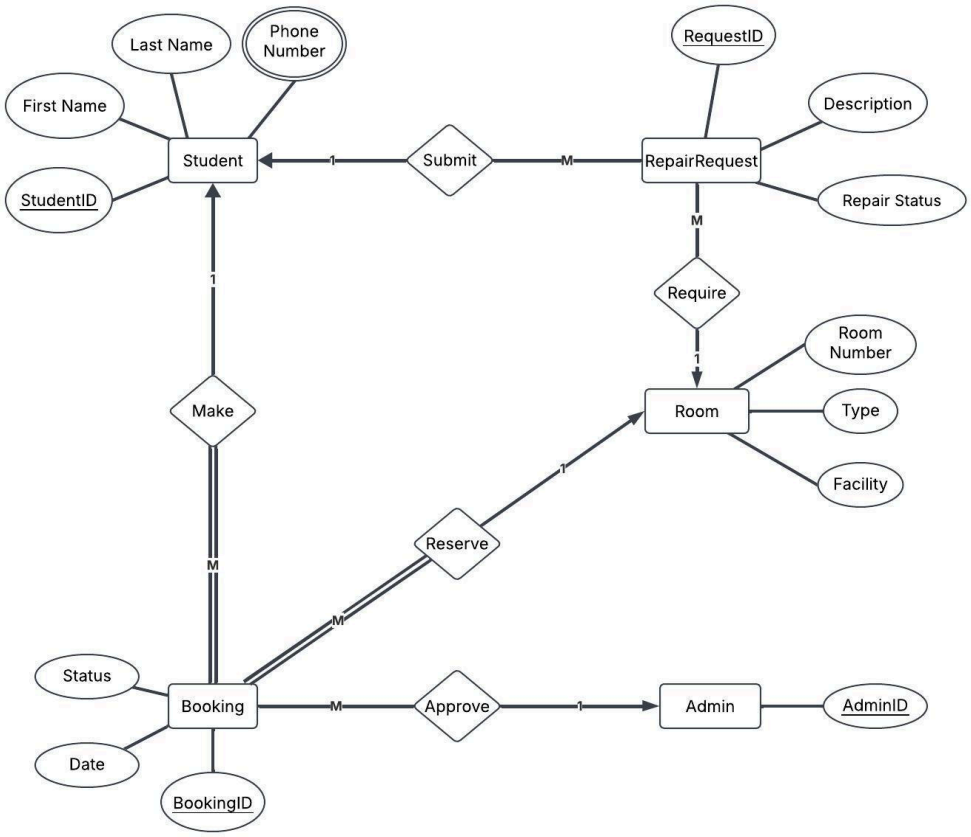
1. **Makes** (Student - Booking)
    - One-to-Many (1:M)
    - A student can make multiple bookings
    - Each booking is made by exactly one student
    - Student: Partial (not all students must make bookings)
    - Booking: Total (every booking must be made by a student)
2. **Reserves** (Booking - Room)
    - Many-to-One (M:1)
    - Multiple bookings can be for one room
    - Each booking is for exactly one room
    - Booking: Total (every booking must be for a room)
    - Room: Partial (not all rooms must be booked)
3. **Submits** (Student - RepairRequest)
    - One-to-Many (1:M)

- A student can submit multiple repair requests
- Each repair request is submitted by one student
- Student: Partial (not all students must submit repairs)
- RepairRequest: Total (every repair request must be linked to a student)

4. **Require** (RepairRequest - Room)
   - Many-to-One (M:1)
   - Multiple repair requests can be for one room
   - Each repair request is for exactly one room
   - RepairRequest: Total (every repair request must be for a room)
   - Room: Partial (not all rooms must have repair requests)
5. **Approves** (Admin - Booking)
   - One-to-Many (1:M)
   - An admin can approve multiple bookings
   - Each booking is approved by one admin
   - Admin: Partial (not all admins must approve bookings)
   - Booking: Total (every booking must be approved by an admin)

Conceptual Design – Entity-Relationship (ER) Diagram

https://lucid.app/lucidchart/ef298967-34b1-42ab-9828-a9ae1abfa53c/edit?viewport_loc=-11%2C-11%2C1351%2C1557%2C0_0&invitationId=inv_489bedd3-a2d5-4038-82e7-41c810b9def7



Explanation and Justification of Design

The rationale behind your entity

This design is structured around five core entities: Student, Room, Booking, RepairRequest, and Admin. The Student and Admin entities were created as separate entities to distinguish clearly between user roles and their permissions. The Room entity serves as a fundamental component storing essential information about dormitory spaces. Booking and RepairRequest were implemented as separate entities rather than attributes because they represent complex processes with their lifecycles and associated data.

The rationale behind Relationship Choices and Why you selected specific connectivity types (1:1, 1:M, M:N)

- **Student-Booking (1:M):** Each student can make multiple bookings, but each booking must belong to exactly one student. This relationship tracks booking status and date and allows for sending deadline reminders to relevant students.
- **Booking-Room (M:1):** Each booking is for exactly one room, while a room can have multiple bookings over time. This relationship prevents double bookings, tracks room availability, and enables occupancy reporting.
- **Student-RepairRequest (1:M):** Students can submit multiple repair requests, with each request tied to one student for accountability. This relationship enables tracking the students who request and sending maintenance updates to the correct students.
- **RepairRequest-Room (M:1):** Each repair request is for one specific room, while a room can have multiple repair requests. This relationship supports maintenance tracking and helps identify recurring issues in specific rooms.
- **Admin-Booking (1:M):** Each booking needs one admin's approval, while admins can approve multiple bookings. This relationship ensures proper booking approval requirements.

How your design aligns with the functional requirements

Each entity, relationship, and attribute directly support each functional requirement:

1. **Can register profile** - Supported by Student entity with several attributes with student info.

2. **Can view available rooms** - Enabled through Room entity with availability status derived from Booking relationships.

3. **Filter by type of room** - Implemented via type attribute in Room entity.

4. **Filter by type of facilities** - Implemented via facilities attribute in Room entity.

5. **Book room** - Supported by Booking entity and its relationships with Student and Room.

6. **Submit repair requests** - Handled by RepairRequest entity and Student SUBMITS relationship.

7. **Track repair status** - Enabled by status attribute in RepairRequest entity.

8. **Admins can assign rooms** - Supported through Admin-Booking relationship.

9. **Admins can approve bookings** - Implemented via Admin APPROVES Booking relationship.

10. **Admins can view occupancy reports** - Facilitated by Room-Booking relationships.

11. **Receive reminders for booking deadlines** - Supported by status and date attributes in Booking entity.

12. **Receive reminders for maintenance updates** - Enabled by status tracking in RepairRequest entity.

Assumptions Made

- A booking cannot be split across multiple rooms
- A repair request cannot be transferred between rooms
- Only one admin needs to approve a booking
- Students cannot book multiple rooms for the same period