THOMAS MORE

CampusBridge

Team: [SMO-B]

Analysis and design report

RAVIN SHALMASHI

ALEXANDROS GKIORGKINIS

ESMA AKBAS

MARIN JANUSHAJ

PIERINA LOPEZ MONSERRATE

2024 - 2025

THOMAS MORE

Table of contents

2024 - 2025

1. Introduction

This document provides a clear overview of the functional and non-functional requirements for the International Student Portal at Thomas More University. This online platform is designed to help international students who are studying for a full degree or taking part in the Erasmus exchange program. The main goal of the system is to simplify administrative tasks and improve communication between students, home institutions, program coordinators, landlords, and university staff.

The portal helps international students with important tasks such as finding a student buddy, applying for university admission or housing, and managing academic and daily life matters. Key features include task management, submitting applications, accessing useful guides, and communicating easily with others through a user-friendly interface.

This document uses a structured method to gather requirements, including Use Case Diagrams and detailed Use Case Descriptions that explain how users interact with the system. The MoSCoW method (Must have, Should have, Could have, and Won't have) is used to make sure the most important features are developed first.

Additionally, the document describes key non-functional requirements such as performance, security, ease of use, and compliance with rules like GDPR (General Data Protection Regulation) and university policies. Important security measures include multi-factor authentication, role-based access control, and the ability to grow as more users join the system.

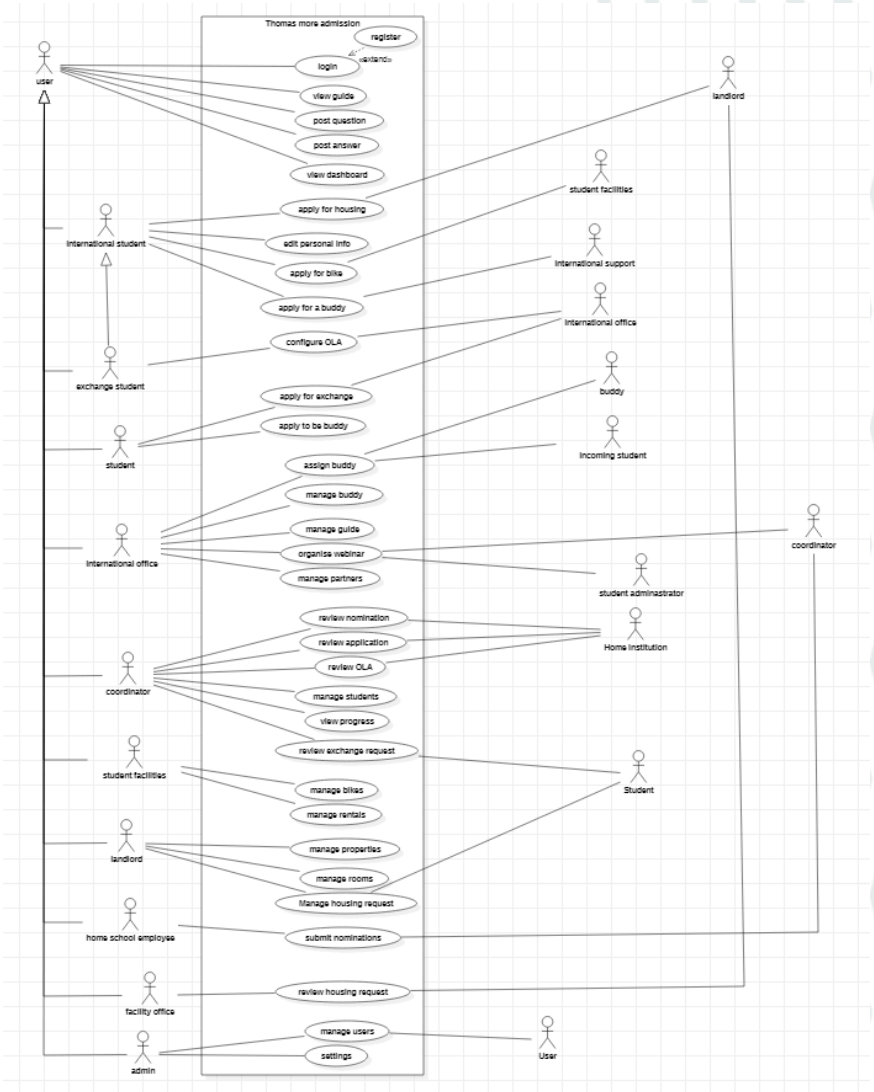
This document is the foundation for developing and launching the International Student Portal, ensuring that it meets user needs while being reliable, secure, and easy to use.

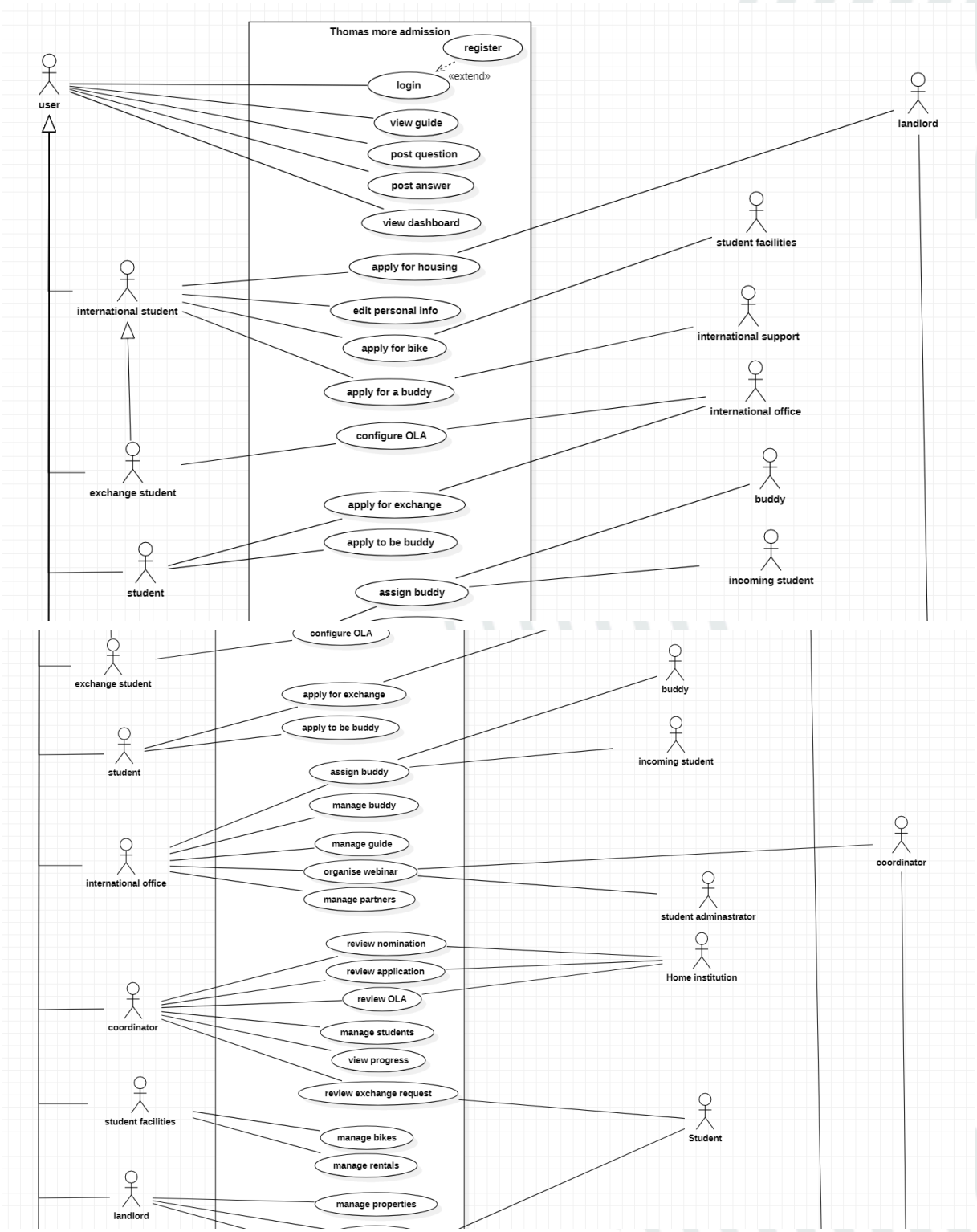
2. Requirements analysis

This section outlines the functional requirements of the International Student Portal, explaining the features needed to support different users. Each requirement is shown with use case diagrams and descriptions that explain how users will use the system.

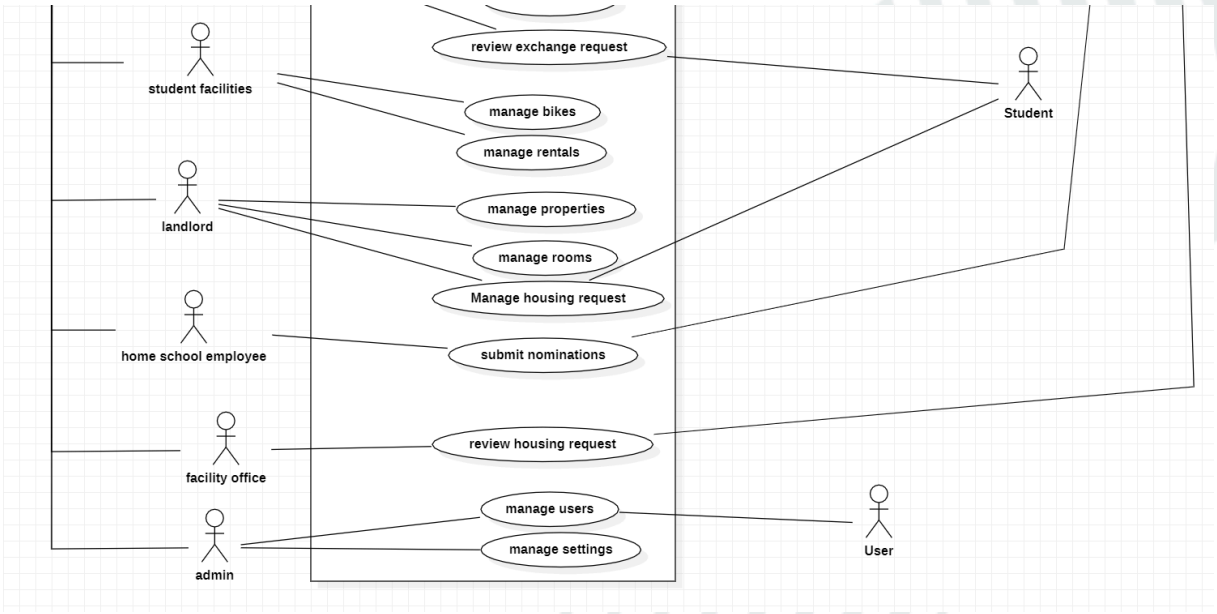
2.1. Functional Requirements

1.1.1. Use Case Diagram




Thomas more admission
register
«extend»
login
view guide
post question
post answer
view dashboard
apply for housing
edit personal info
apply for bike
apply for a buddy
configure OLA
apply for exchange
apply to be buddy
assign buddy
manage buddy
manage guide
organise webinar
manage partners
review nomination
review application
review OLA
manage students
view progress
review exchange request
manage bikes
manage rentals
manage properties
user
international student
exchange student
student
international office
coordinator
student facilities
landlord
landlord
student facilities
international support
international office
buddy
incoming student
coordinator
student adminastrator
Home institution
Student
buddy
incoming student
exchange student
student

2024 - 2025


review exchange request
student facilities
manage bikes
manage rentals
Student
landlord
manage properties
manage rooms
Manage housing request
home school employee
submit nominations
review housing request
facility office
manage users
manage settings
User
admin

1.1.2. Use Cases

1.1.2.1. Login

Functionality: As a User, I can login.

Precondition: The user must have an existing account with valid credentials (email and password).

Extension:
If the user does not have an account, they can register by:

1. Clicking the "Register" button on the login page.
2. Filling out required details such as name, email, and password.
3. Submitting the form to create a new account.

Data model view:

User
userId: int PK coordinatorId: int FK1 NA landlordId: int FK22 NA studentID: int FK3 NA roleId: int FK23 NA formerEducationId: int FK6 NA exchangeId: int FK7 NA homeInstitutionId: int FK14 NA languageId: int FK11 NA lastName: string NA firstName: string NA phone: string NA email: string NA image: string NA department: string NA studyProgram: string NA studentType: NA birthPlace: string NA birthDate: date NA languageLevel: int NA nominationStatus: string applicationDate: date NA visaRequired: bool NA scholarshipeligitibility: bool NA gender: string NA nationality: string NA studyCycle: string NA etrCode: string NA iscedCode: string NA

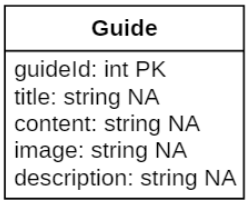
Link to prototype:
Log in Prototype
Register Prototype (extension)

1.1.2.2. View Guide

Functionality: As a User, I can view guides.

Precondition: The user must be logged in to access the guides.

Data model view:



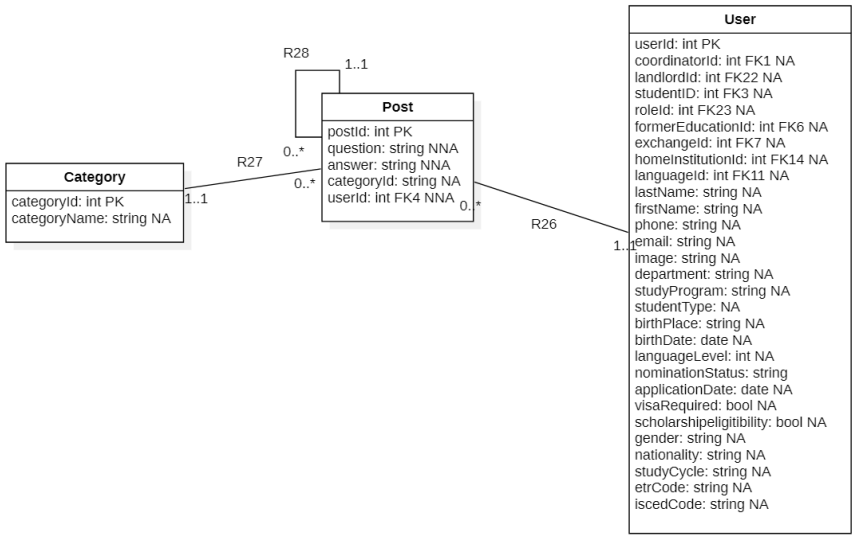
Link to prototype:
Guide Prototype

1.1.2.3. Post Question

Functionality: As a User, I can post questions.

Precondition: The user must be logged in to post a question.

Data model view:



Link to prototype: Contact & Ask Question List Prototype

Data model actions:

- **Category Button:** READ from Category Table. Retrieve the list of available categories from the Category table to allow the user to select one.

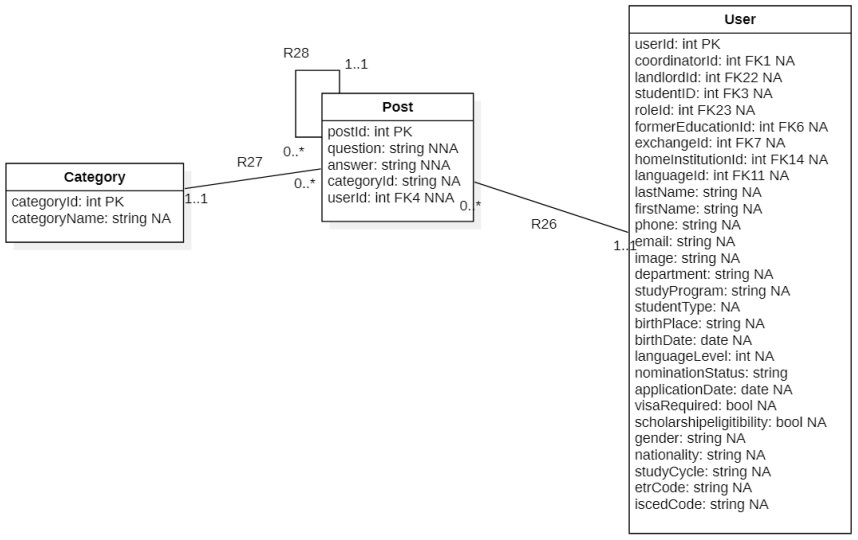
- **Post Button:** CREATE Post Entry in the Post table the the user provides a text for the question attribute, the category is created from the user selection, userId from the user that posted the question, and postTime from current timestamp.

1.1.2.4. Post Answer

Functionality: As a User, I can post answers.

Precondition: The user must be logged in to provide an answer to an existing question.

Data model view:



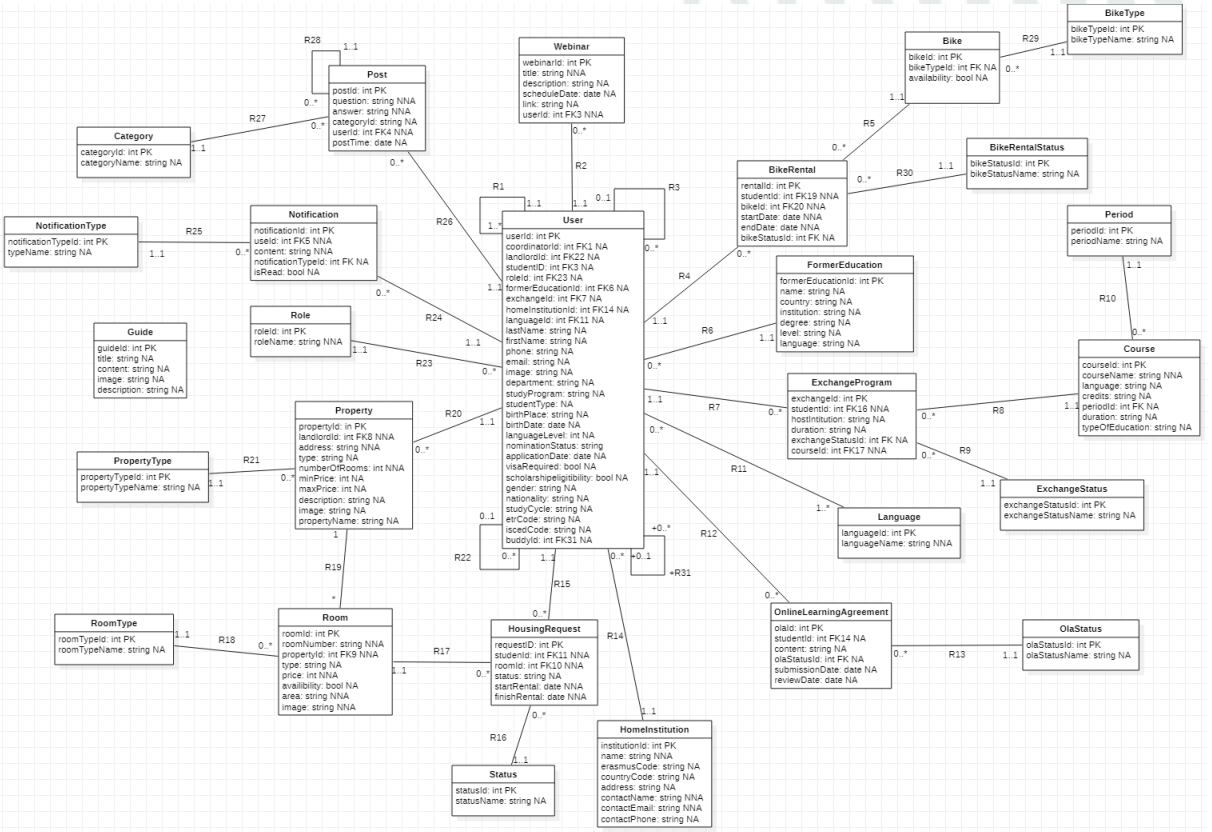
Link to prototype: [Contact & Ask Answer Question Prototype]

1.1.2.5. View Dashboard

Functionality: As a User, I can view the dashboard.

Precondition: The user must be logged in to access the dashboard.

Data model view:



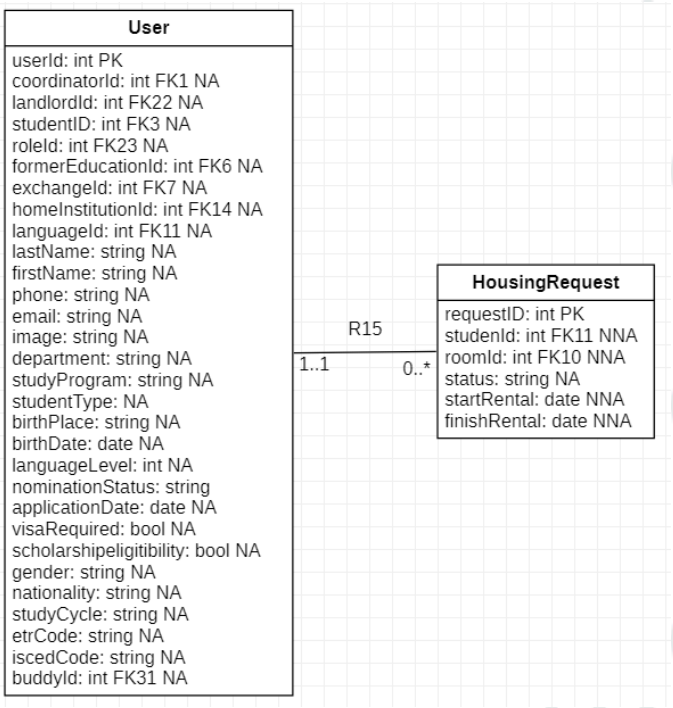
Link to prototype: [Dashboard Prototype]

1.1.2.6. *Apply for Housing*

Functionality: As an international student, I can apply for housing.

Precondition: The student must be accepted into a university program and provide required housing documents.

Data model view:



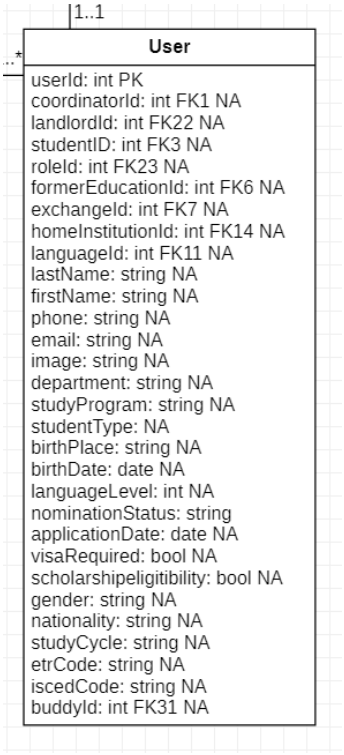
Link to prototype: [Apply for Housing Prototype]

1.1.2.7. Edit personal information

Functionality: As an international student, I can edit personal information.

Precondition: The student must have an active university account.

Data model view:



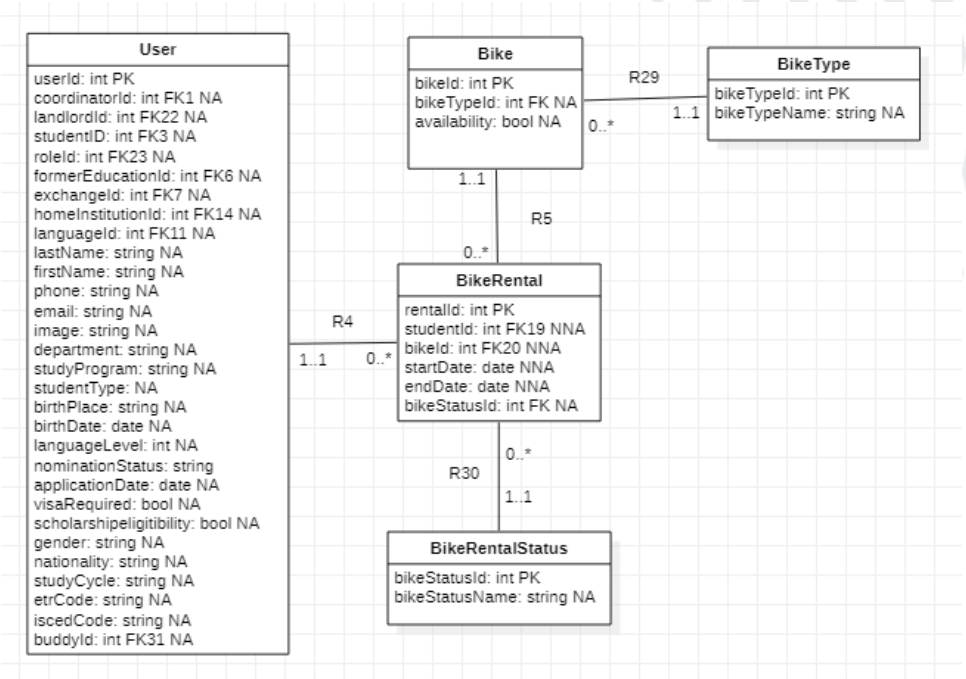
Link to prototype: Edit personal information prototype

1.1.2.8. Apply for bike

Functionality: As an international student, I can apply for a bike.

Precondition: The student must have a valid university ID and a local address.

Data model view:



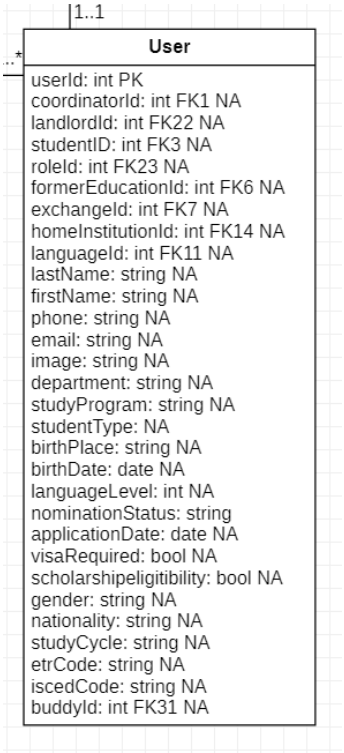
Link to prototype: [Apply for bike prototype]

1.1.2.9. Apply for a buddy

Functionality: As an international student, I can apply for a buddy.

Precondition: The student must be registered as an international student.

Data model view:

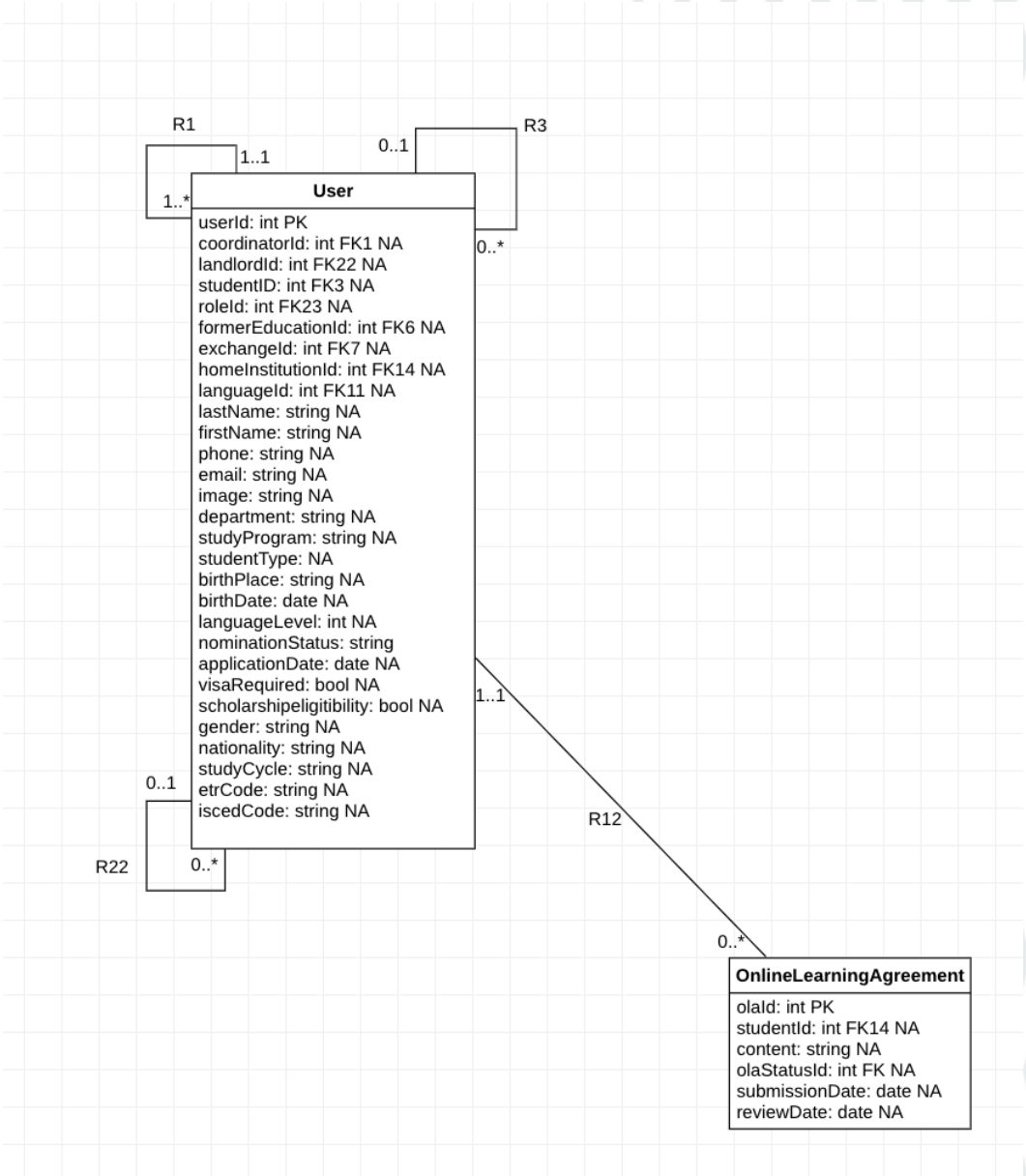


Link to prototype: Apply for a buddy prototype

1.1.2.10. Configure OLA

Functionality: As an exchange student, I can configure OLA.
Precondition: The user must be logged in as an exchange student

Data model view:



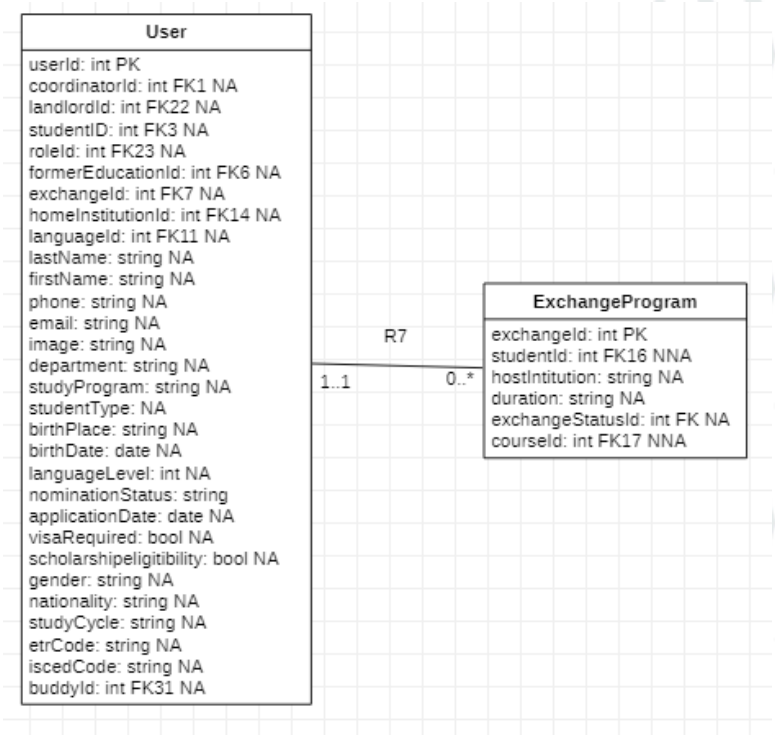
Link to prototype: [Configure ola prototype]

1.1.2.11. Apply for exchange

Functionality: As a student, I can apply for exchange.

Precondition: The student must be enrolled in a university that offers an exchange program.

Data model view:



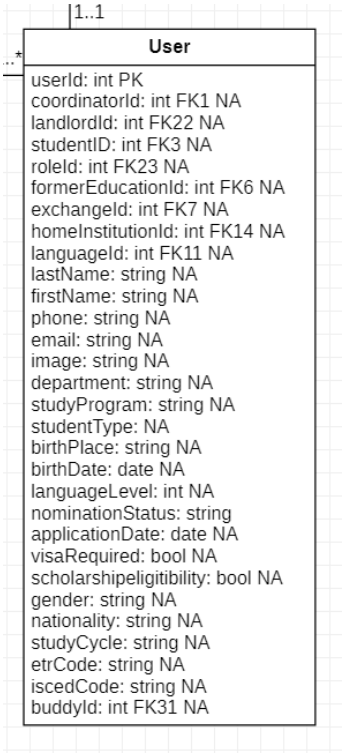
Link to prototype: [Apply for exchange prototype]

1.1.2.12. Apply to be a buddy

Functionality: As an exchange student, I can apply to be a buddy.

Precondition: The student must be an exchange student currently studying at the university.

Data model view:



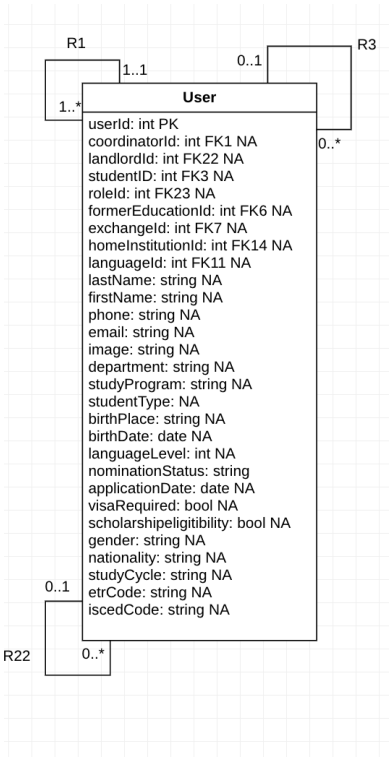
Link to prototype: Apply to be a buddy prototype

1.1.2.13. Assign Buddy

Functionality: As an international office, I can assign buddy.

Precondition: The user must be logged in as international office.

Data model view:



Link to prototype: Assign buddy

Data Model Actions:
Initial State:
READ the Student table to display the list of students and their assigned buddies:
Student Table:

studentId (int PK) - unique identifier for each student

firstName (string NNA) - displayed in the list

lastName (string NNA) - displayed in the list

email (string NNA) - displayed in the list

studyProgram (string NNA) - displayed in the list

buddyId (int FK -> Student.studentId) - references another student acting as their buddy

buddyStatus (string NNA) - displayed as “Assigned” or “Pending”

Actions

Assign Buddy Button

1. READ the Student table to populate the dropdown list:

Filter Student table where studentType is “Buddy” and buddyId is NULL.

Fields to display in dropdown:

studentId (int PK)

firstName (string NNA)

lastName (string NNA)

email (string NNA)

2.On Confirm Button (Assign Buddy):
UPDATE the Student table:

Update the buddyId of the selected student with the studentId of the assigned buddy.

Set buddyStatus to “Assigned.”

Three Dots (View Student Details)

READ the Student table for the selected student:
Fields to display:

firstName (string NNA)

lastName (string NNA)

email (string NNA)

phone (string NNA)

studyProgram (string NNA)

buddyId (int FK -> Student.studentId)

2.READ the Buddy details (via Student table):

Use buddyId to query the assigned buddy’s details.

Fields to display:

firstName (string NNA)

lastName (string NNA)

Edit Button (Change Buddy)

1.READ the Student table to display all available buddies in a dropdown:

Filter Student table where studentType is “Buddy” and buddyId is NULL.

2. On Confirm Button (Reassign Buddy):
UPDATE the Student table:

Update the buddyId of the selected student with the newly selected buddy’s studentId.

Update buddyStatus to “Assigned.”

Save/Confirm Button
UPDATE the Student record:

Update the following fields:

buddyId with the new or existing buddy ID.

Maintain all NNA (Not Null Attribute) constraints.

Delete Buddy Assignment Button

1. REMOVE the Buddy Assignment for the selected student:

Set buddyId to NULL.

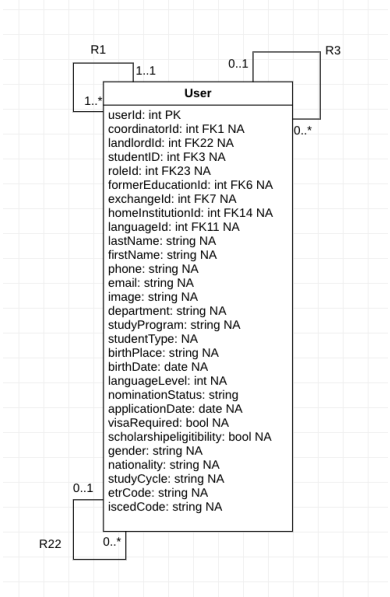
Update buddyStatus to “Pending.”

1.1.2.14. Manage buddy

Functionality: As an international office, I can manage a buddy.

Precondition: The user must be logged in as international office.

Data model view:

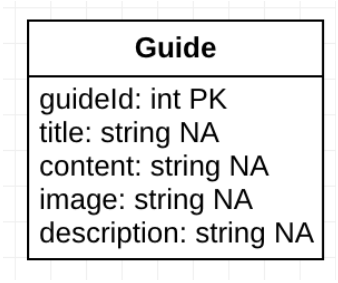


Link to prototype: Manage Buddy

1.1.2.15. Manage guide

Functionality: As an international office, I can manage guide
Precondition: The user must be logged in as international office.

Data model view:



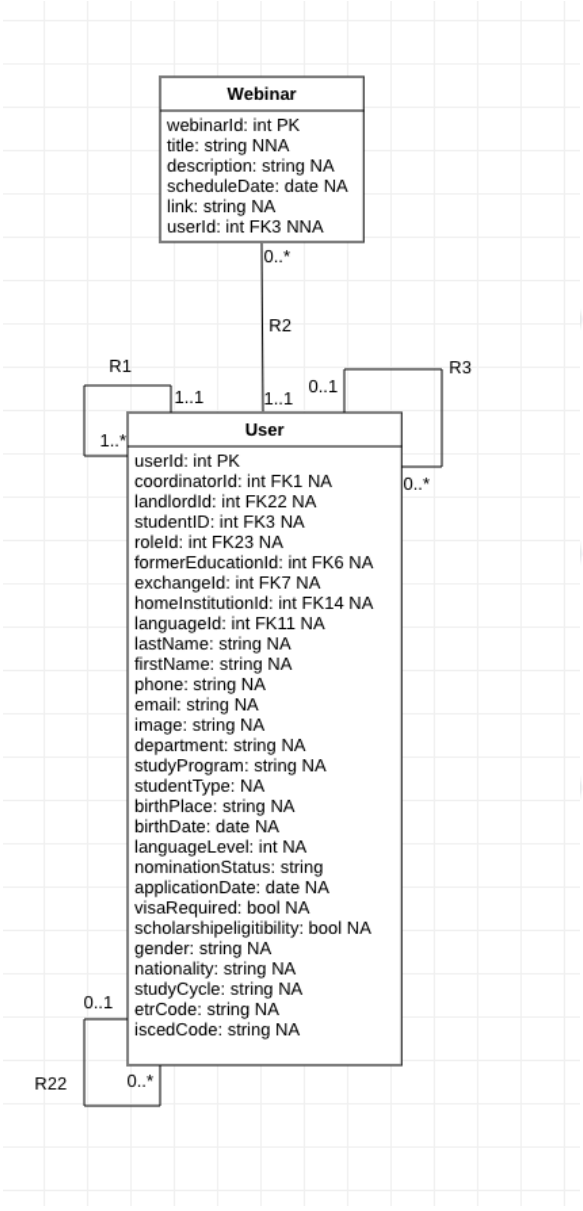
Link to prototype: Manage guide

1.1.2.16. Organise Webinar

Functionality: As an international office, I can organise Webinar

Precondition: The user must be logged in as international office.

Data model view:



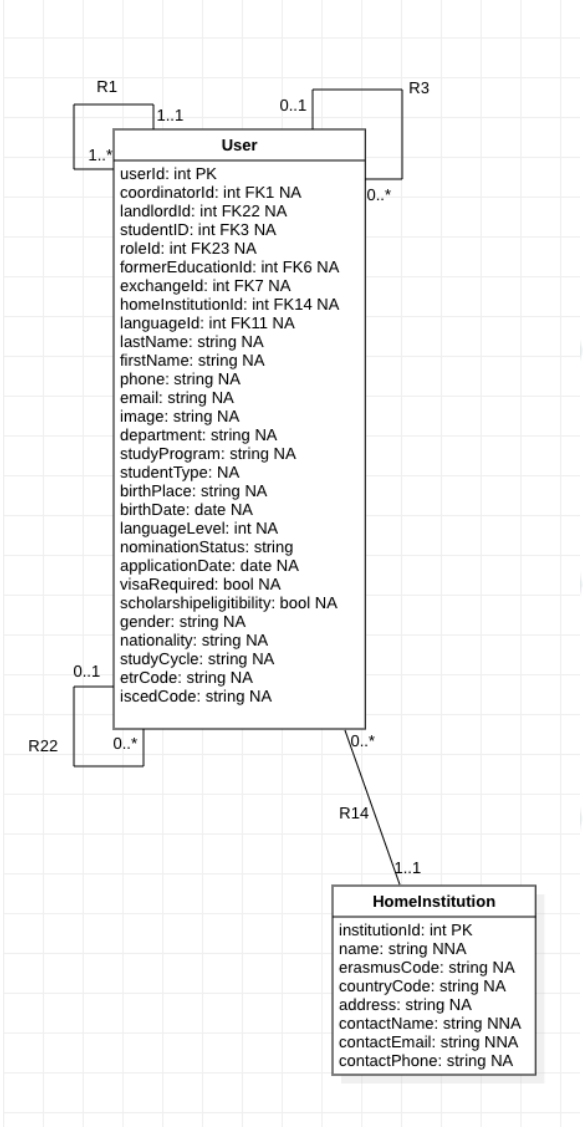
Link to prototype: Organise Webinar

1.1.2.17. Manage partners

Functionality: As an international office, I can manage partners

Precondition: The user must be logged in as international office.

Data model view:



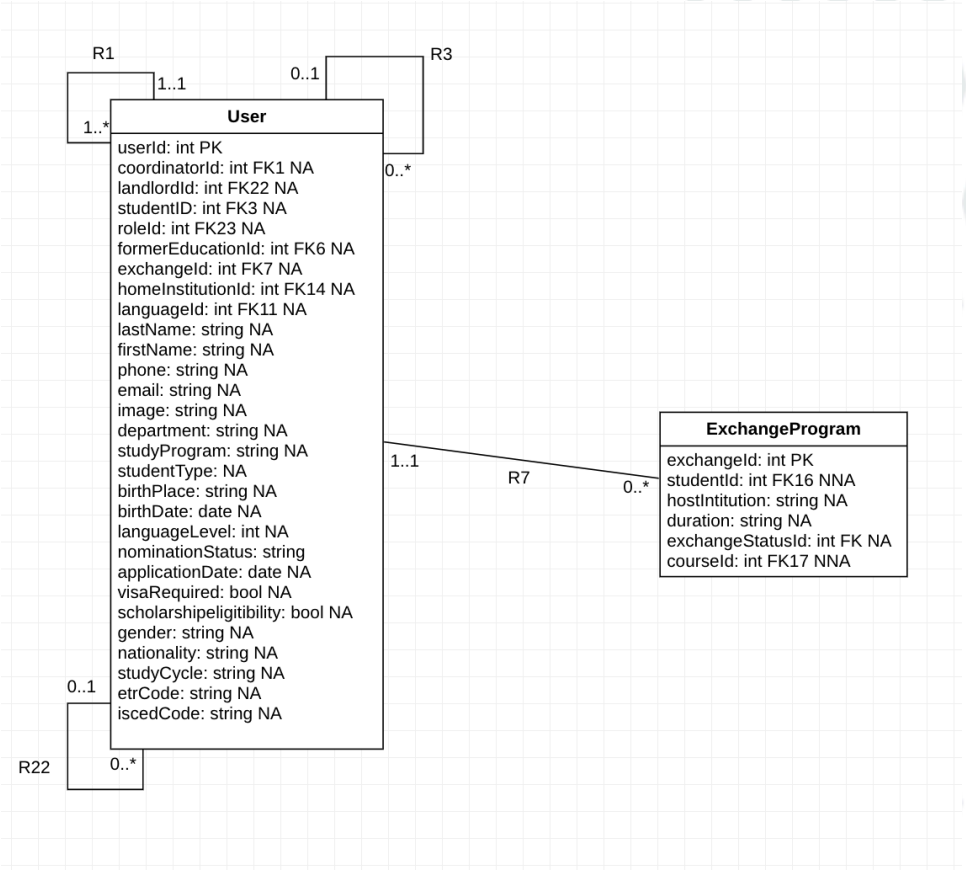
Link to prototype: [Manage Partners]

1.1.2.18. Review nomination

Functionality: As a coordinator, I can review nomination

Precondition: The user must be logged in as coordinator..

Data model view:



Link to prototype: [Review nomination prototype]

1.1.2.19. Review Application

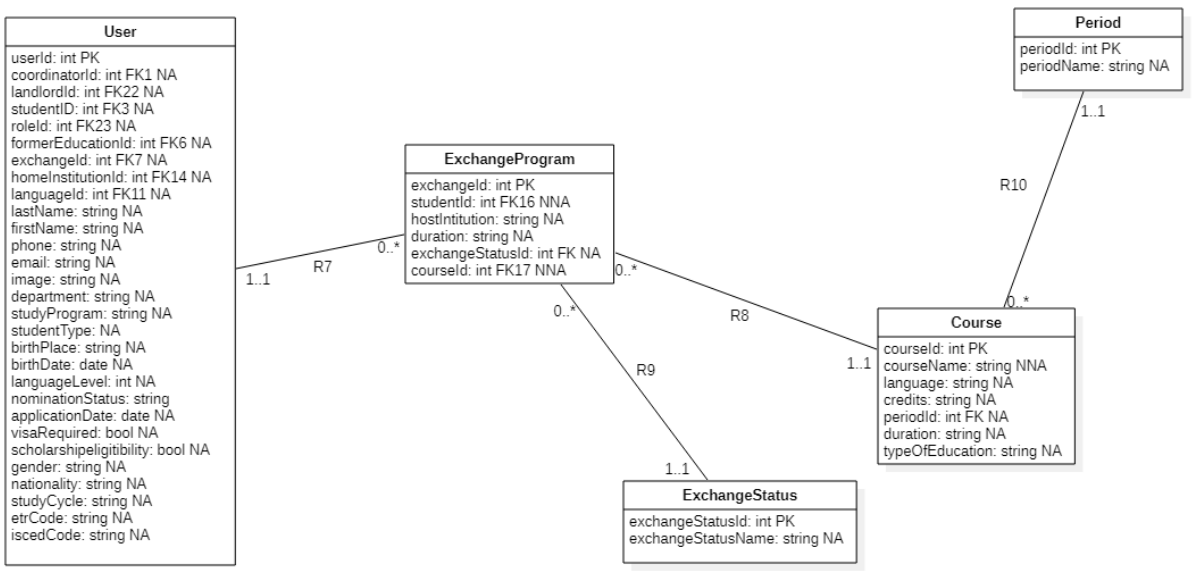
Functionality: As a coordinator, I can review applications for exchange programs.

Precondition: The application must have been submitted by a student enrolled in a university offering an exchange program.

Data model action:

- Retrieve the application details from the system.
- Update the **exchange status** based on the review decision (e.g., Pending → Approved/Rejected).
- Store this change in the **application database**, ensuring the new status is linked to the correct student and exchange program.
- Log the review action for record-keeping and future tracking.

Data model view:

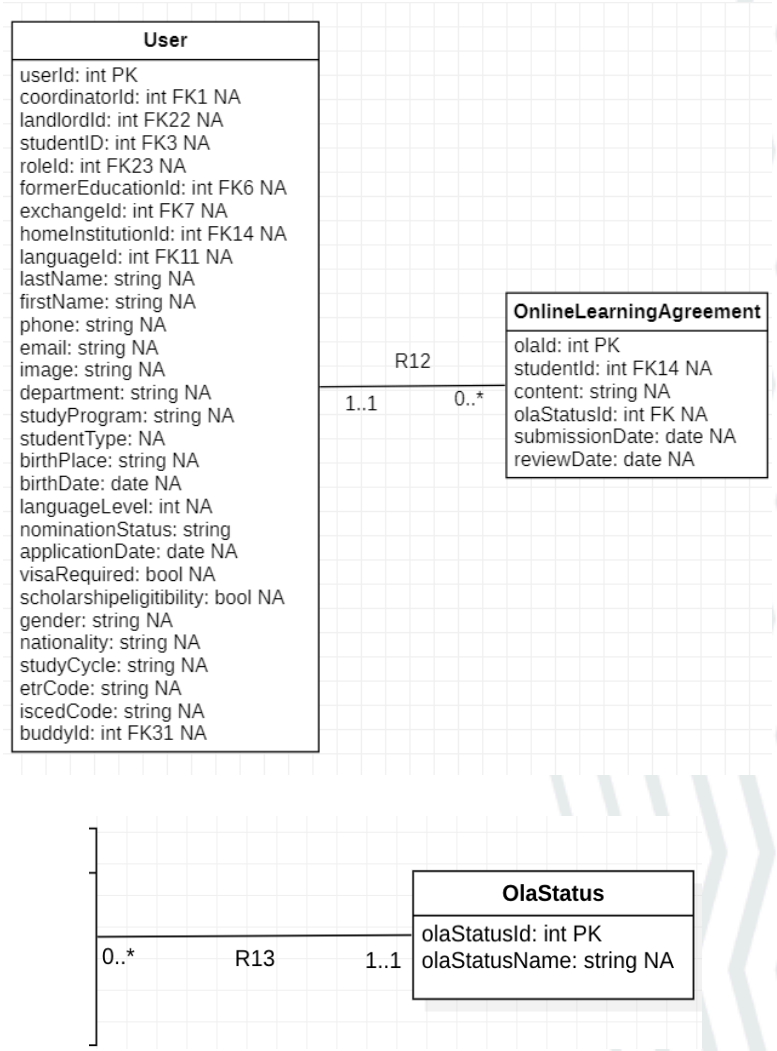


Link to prototype: [Review application prototype]

1.1.2.20. Review OLA

Functionality: As a coordinator, I can review OLA

Data model view:

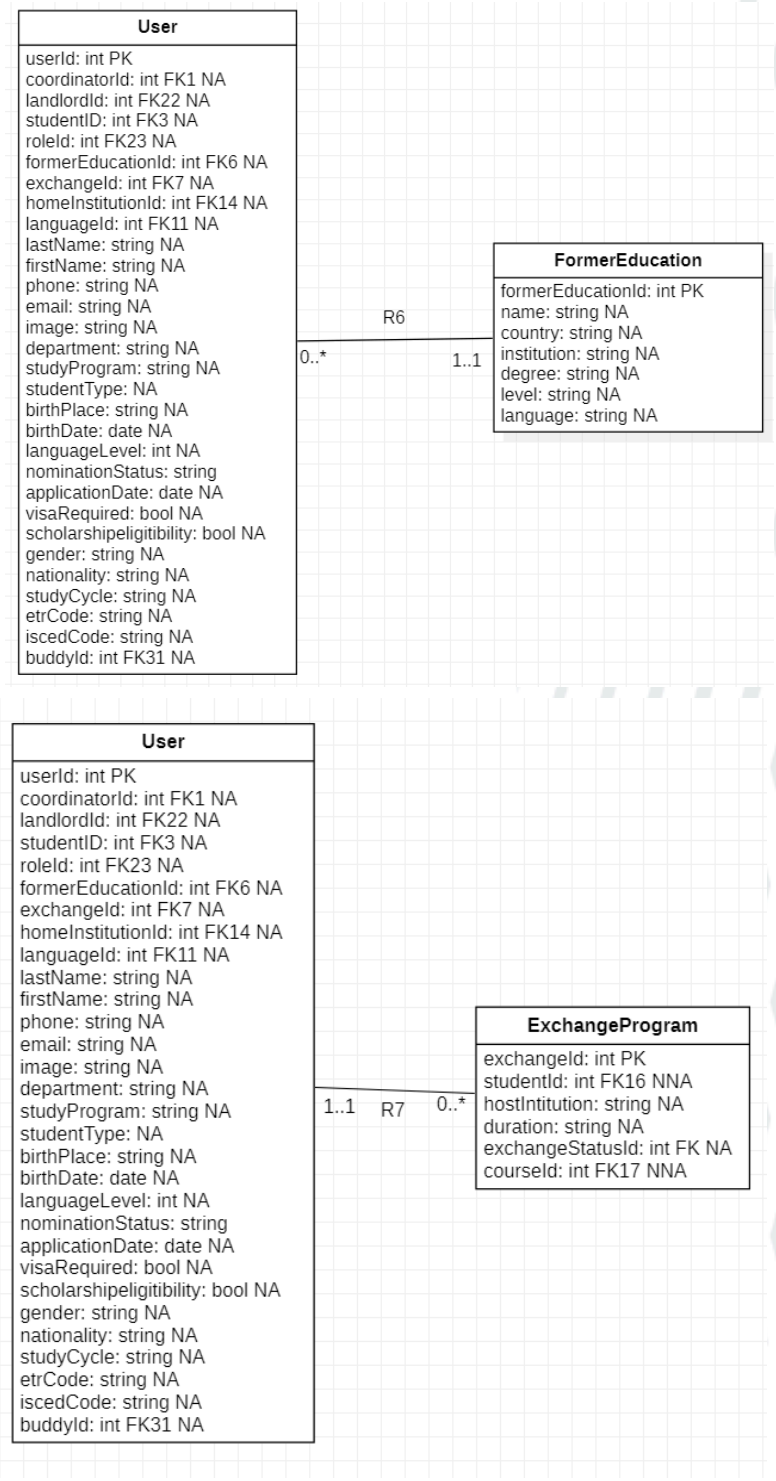


Link to prototype: [Review OLA prototype]

1.1.2.21. *Manage students*

Functionality: As a coordinator, I can manage students

Data model view:

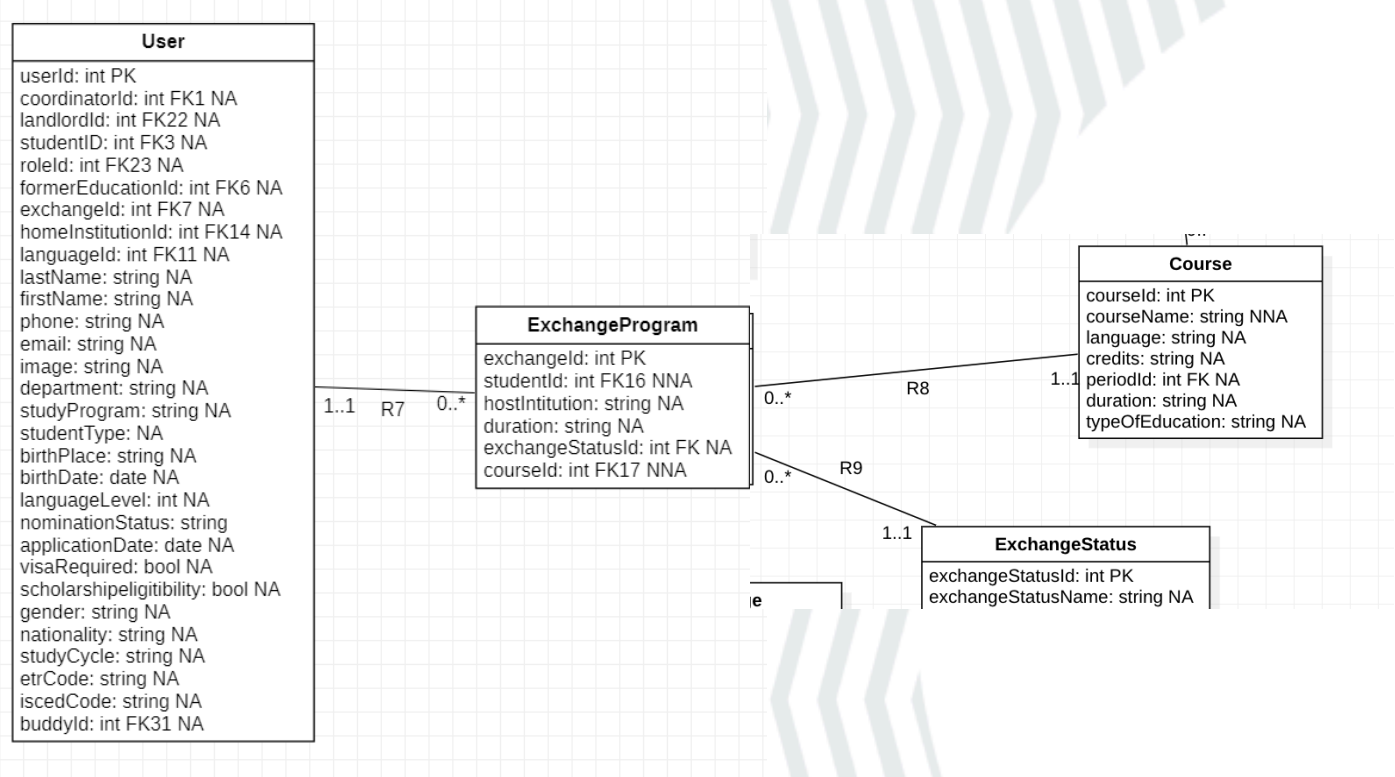


Link to prototype: [Manage Students prototype]

1.1.2.22. View progress

Functionality: As a coordinator, I can view progress

Data model view:



Link to prototype: [View progress prototype]

Data model actions:
READ from ExchangeProgram Table.
Retrieve all exchange program entries associated with the specific studentId to display information such as institution, duration, and exchangeStatusId.

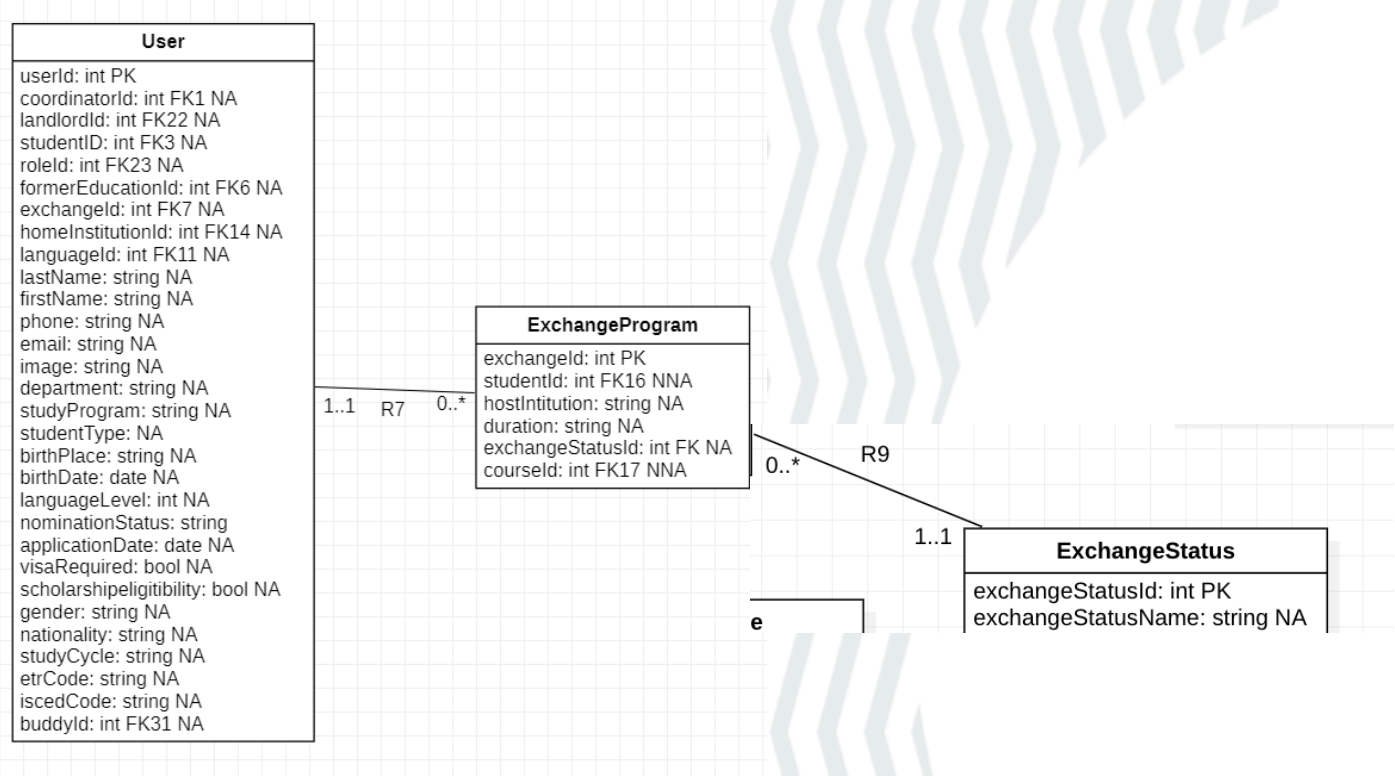
READ from the Course Table.
Retrieve the courses related to the specific exchangeId using courseId. Extract details like courseName, language, credits, and typeOfEducation.

READ from ExchangeStatus Table.
Retrieve the exchangeStatusName for the current exchangeStatusId to provide a clear description of the student's progress in the program.

1.1.2.23. Review exchange request

Functionality: As a coordinator, I can review exchange request

Data model view:

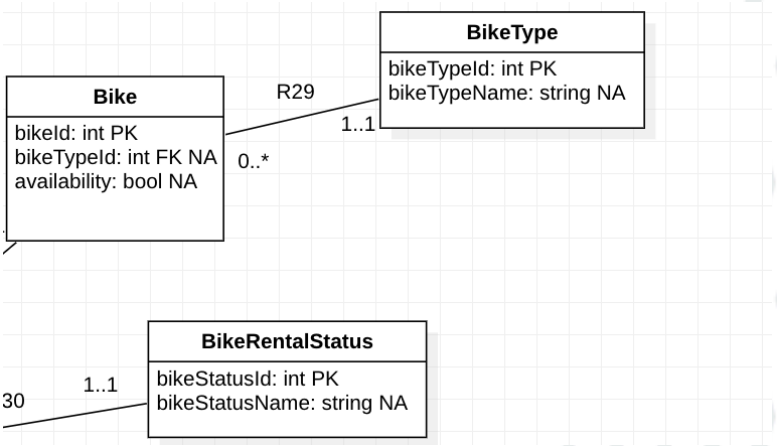


Link to prototype: [Review exchange Request prototype]

1.1.2.24. Manage bikes

Functionality: As student facilities, I can manage bikes

Data model view:

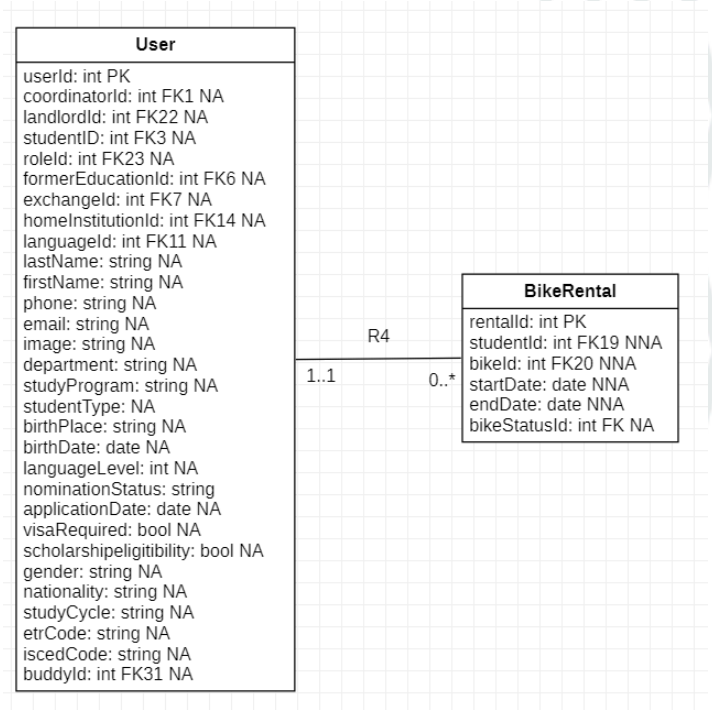


Link to prototype: [Manage bikes prototype]

1.1.2.25. Manage rentals

Functionality: As student facilities, I can manage rentals

Data model view:



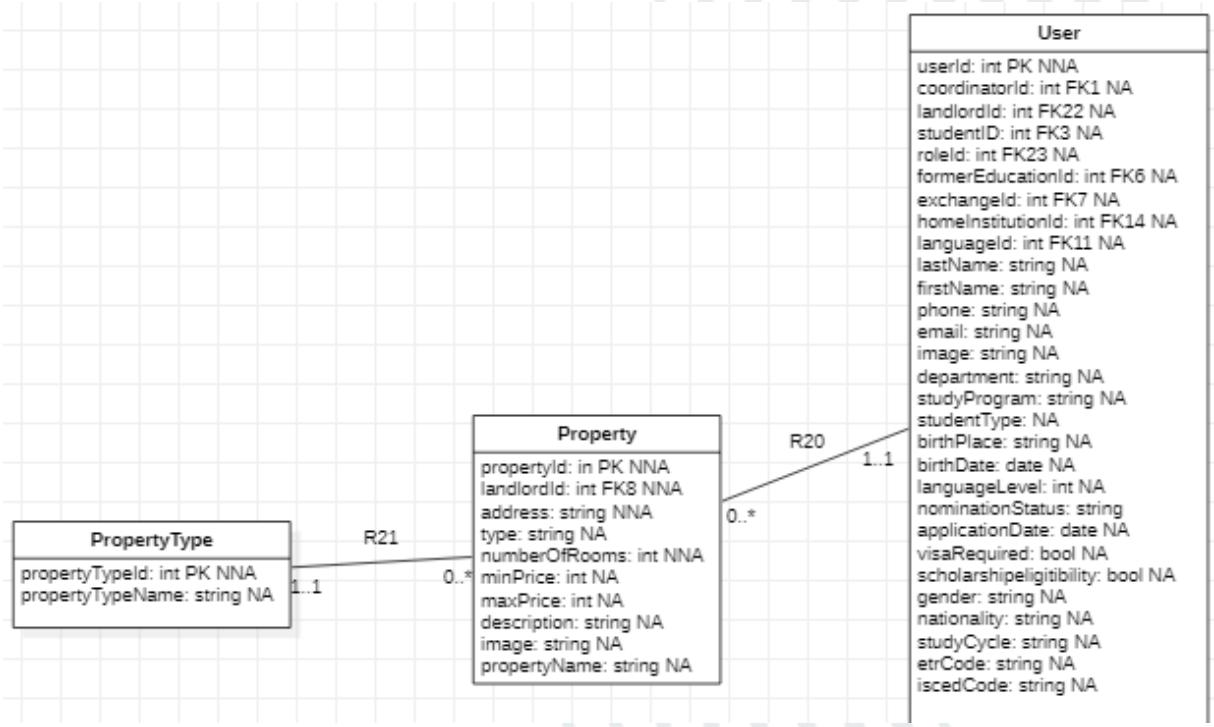
Link to prototype: [Manage rentals]

1.1.2.26. Manage properties

Functionality: As a landlord, I can manage properties

Precondition: Users account should be approved by facility office

Data model view:



Link to prototype: [Manage properties]
Data model action
List of Properties:
READ from Property Table.
Retrieve a list of available properties with attributes like propertyId, address, minPrice, maxPrice, availability, and dateListed to display in a table format.

- Add Property Button:

CREATE new entry in Property Table.
Insert a new property record with details provided by the user, such as address, numberOfRooms, minPrice, maxPrice, description, propertyTypeId, landlordId, and image.

- Edit Property Button:

UPDATE existing entry in Property Table.
Allow the user to update property details for a selected propertyId by modifying attributes such as address, numberOfRooms, minPrice, maxPrice, description, availability, and image.

- Delete Property Button:

DELETE entry from Property Table.
Remove the selected property based on propertyId after user confirmation.

- View Property Details Button:

READ from Property Table.
Retrieve detailed information for a specific property using propertyId. Attributes retrieved include address, numberOfRooms, description, image, minPrice, maxPrice, and availability.

- Filter Properties Button:

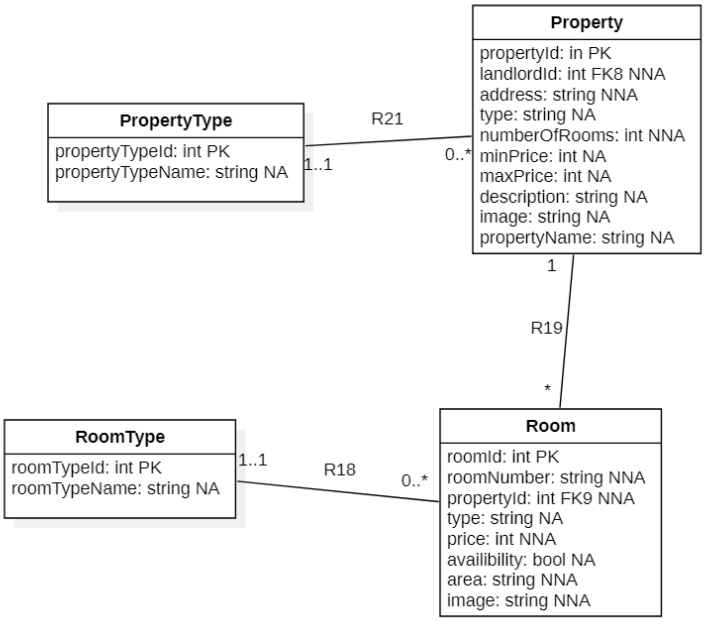
READ filtered entries from Property Table.
Retrieve a subset of properties based on filter criteria like propertyTypeId, minPrice, maxPrice, and availability provided by the user.

1.1.2.27. Manage Rooms

Functionality: As a landlord, I can manage rooms

Precondition: property is listed and accepted

Data model view:

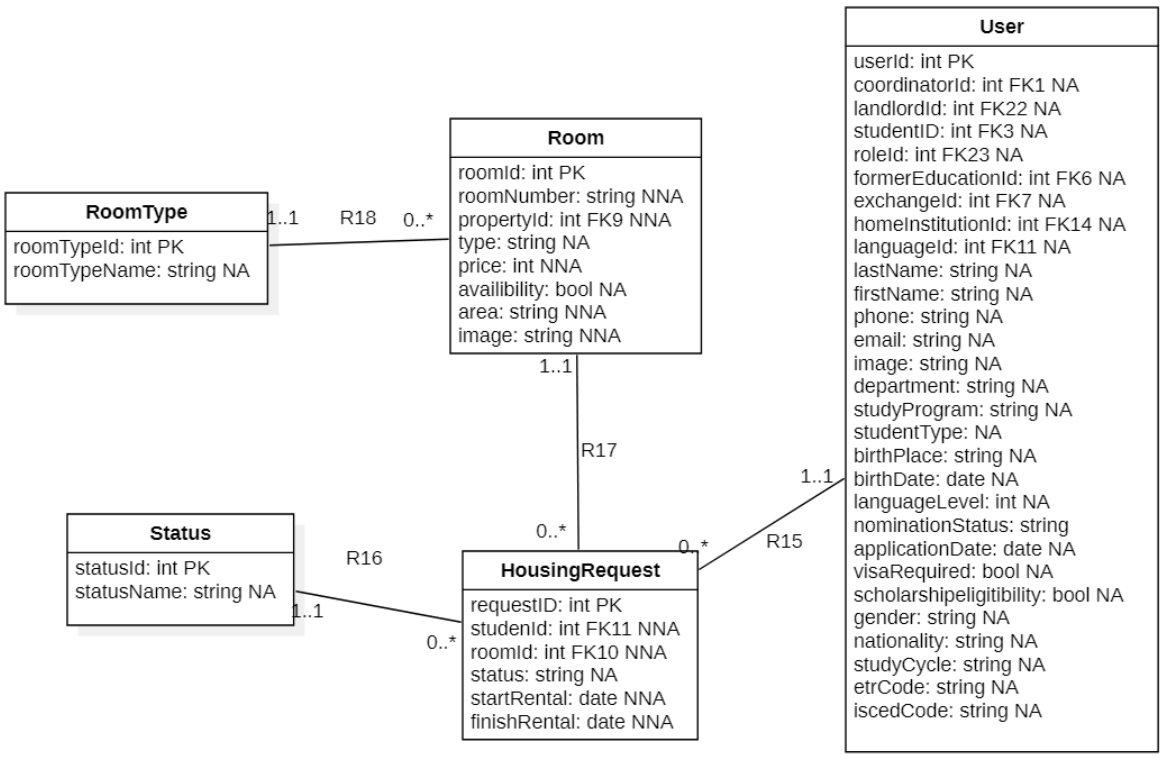


Link to prototype: [Manage Rooms]

1.1.2.28. Manage housing request

Functionality: As a landlord, I can manage housing request

Data model view:

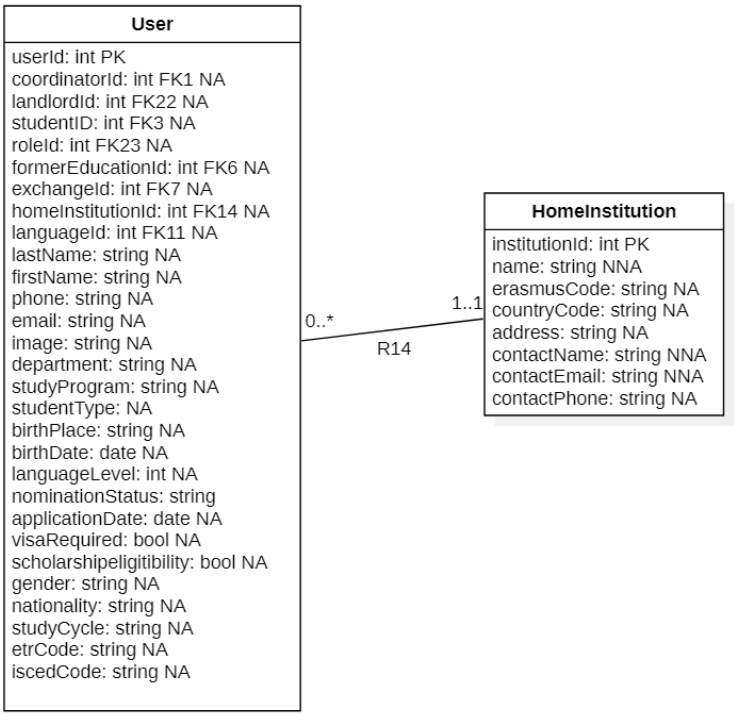


Link to prototype: [Manage Housing request]

1.1.2.29. Submit nominations

Functionality: As a home school employee, I can submit nominations

Data model view:



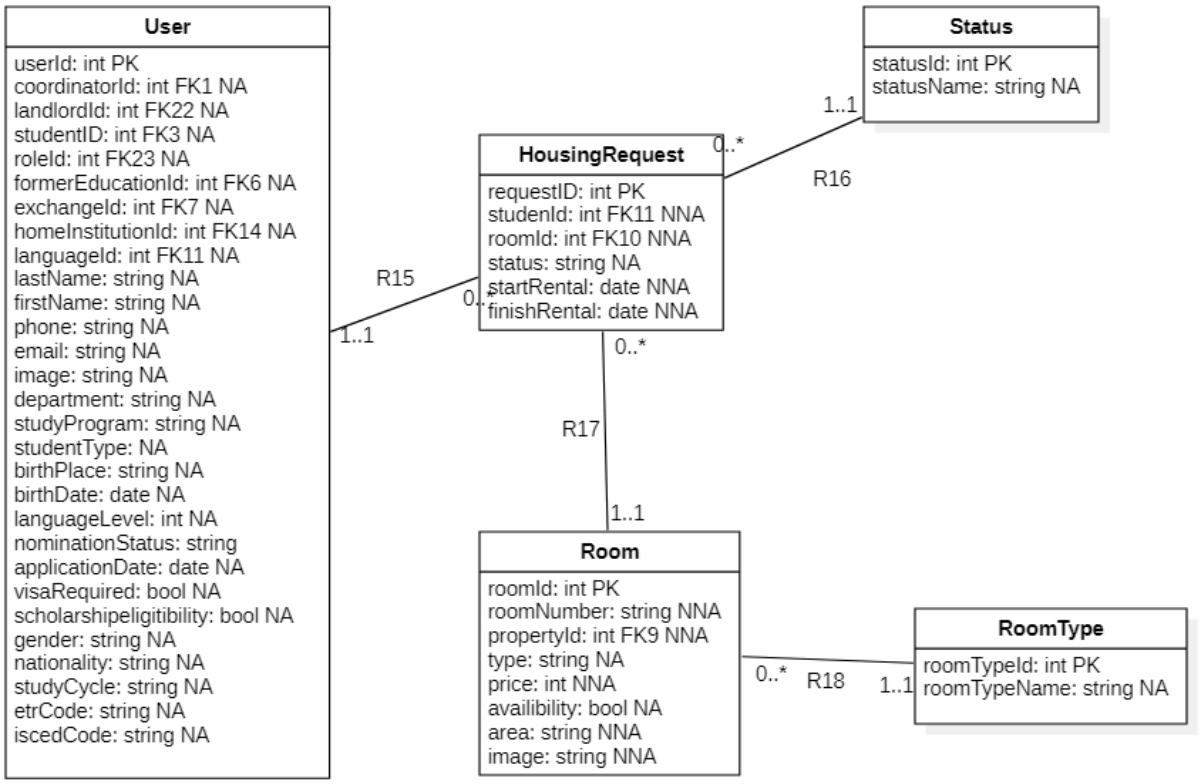
Link to prototype: [submit nominations]

1.1.2.30. Review housing request

Functionality: As a facility office, I can review housing request

Precondition: landlord requested the property listing

Data model view:

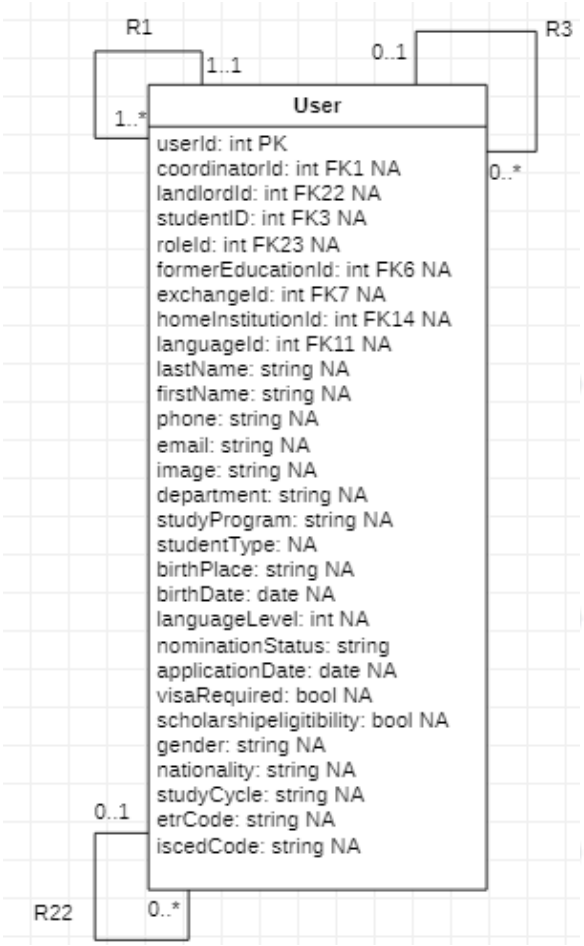


Link to prototype: [Review housing request]

1.1.2.31. Manage Users

Functionality: As an admin, I can manage users

Data model view:

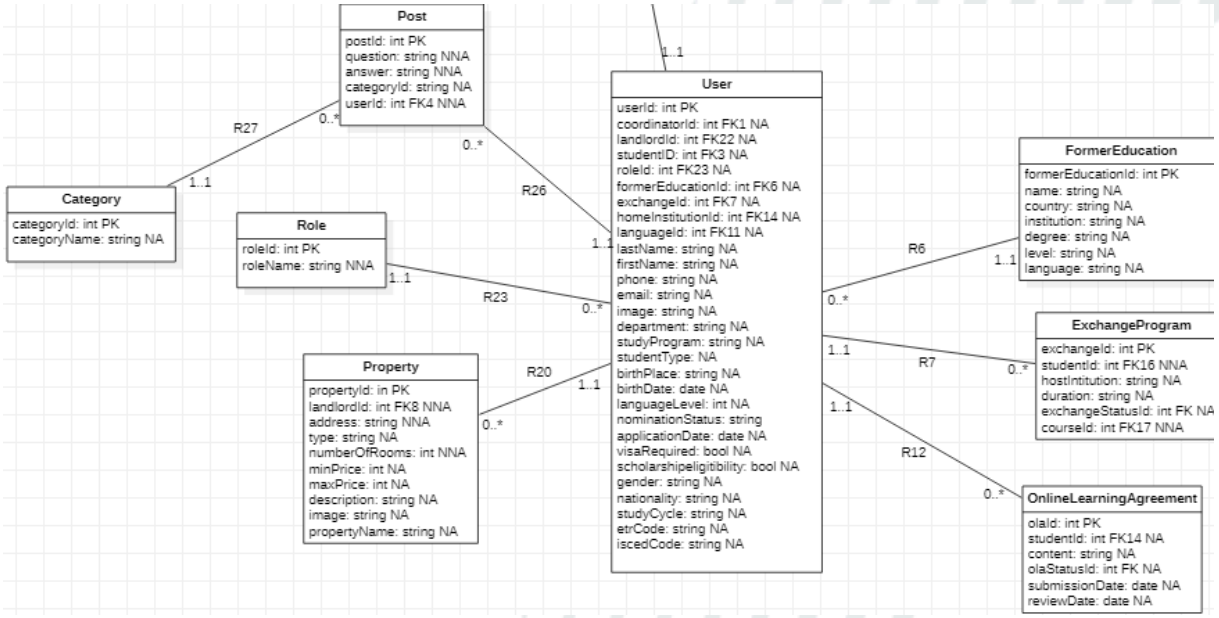


Link to prototype: [Manage users]

1.1.2.32. Manage Settings

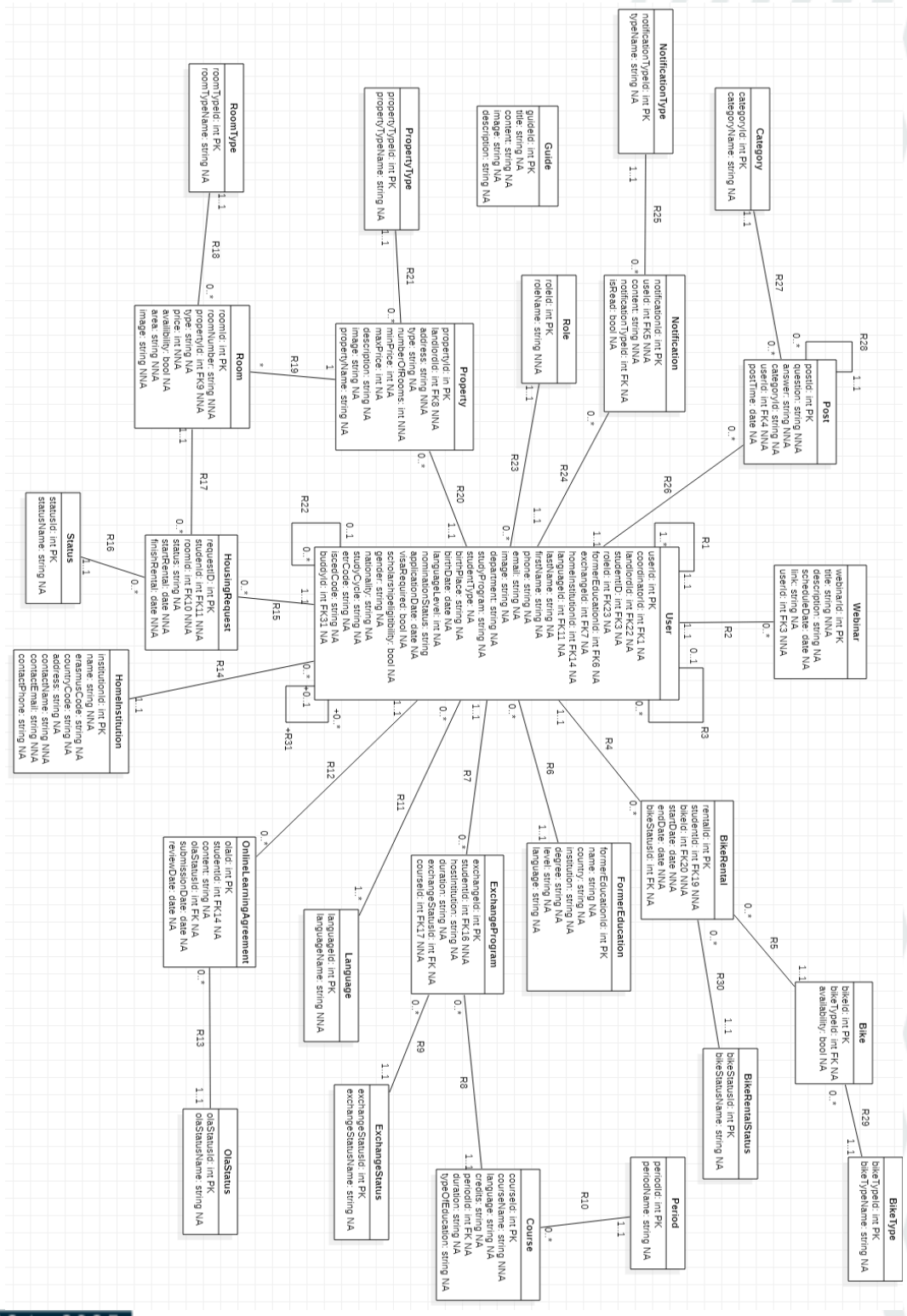
Functionality: As an admin, I can manage settings

Data model view:

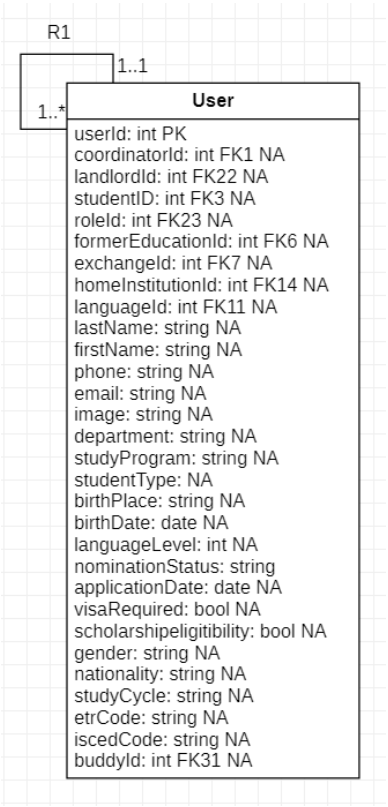


Link to prototype: [Manage settings]

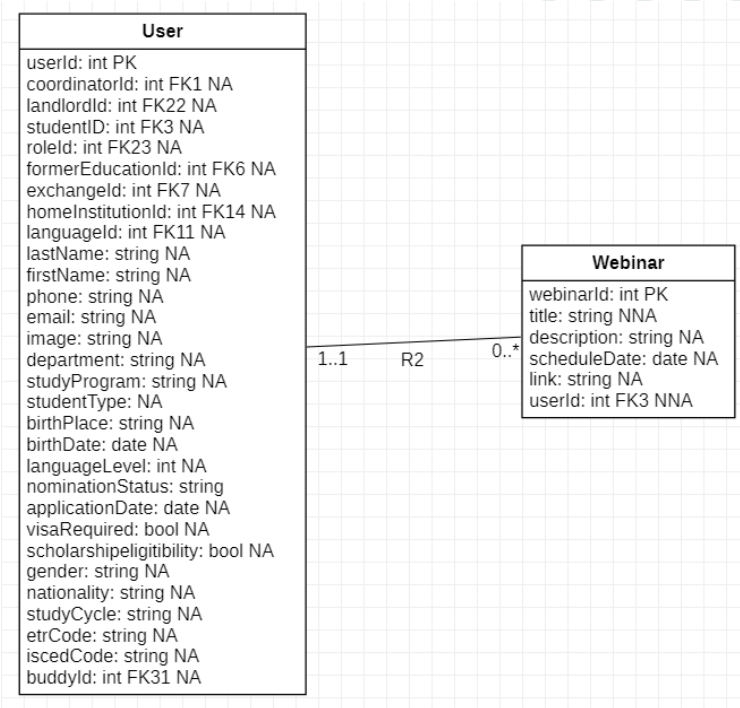
2. Data Model



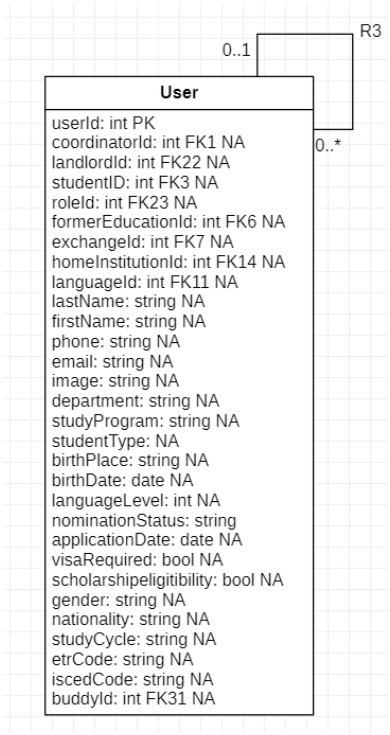
2024 - 2025



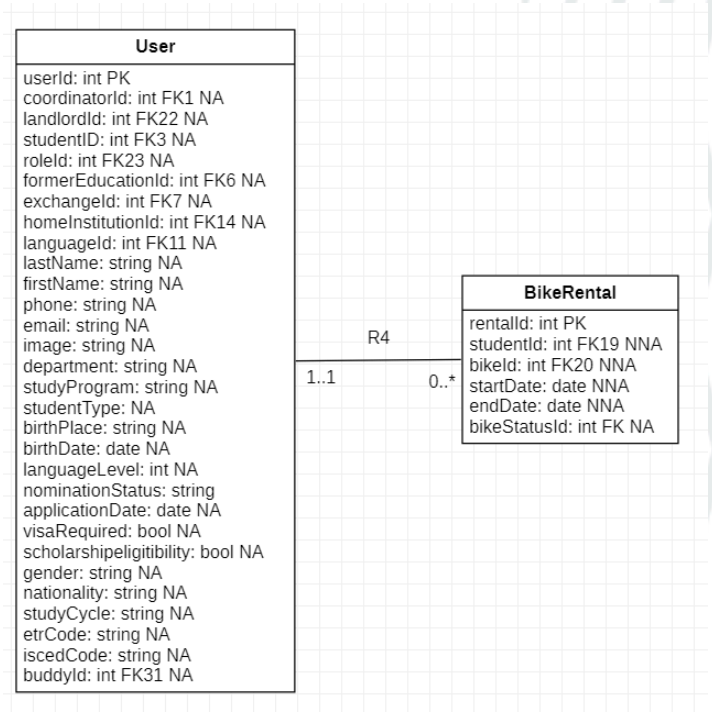
R1: Student can have one coordinator, Each coordinator can be assigned to many students.



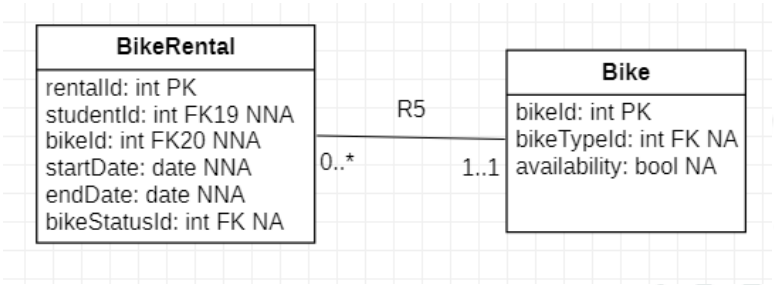
R2: A webinar is hosted by one user, and a user can host multiple webinars.



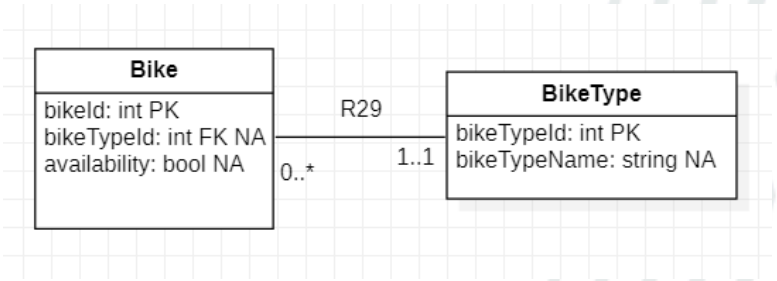
R3: A User can have multiple bike rentals, and each bike rental belongs to one user.



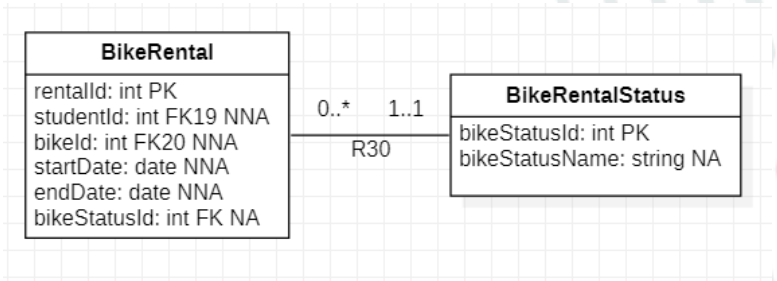
R4: A student can rent multiple bikes, and each bike rental belongs to one student.



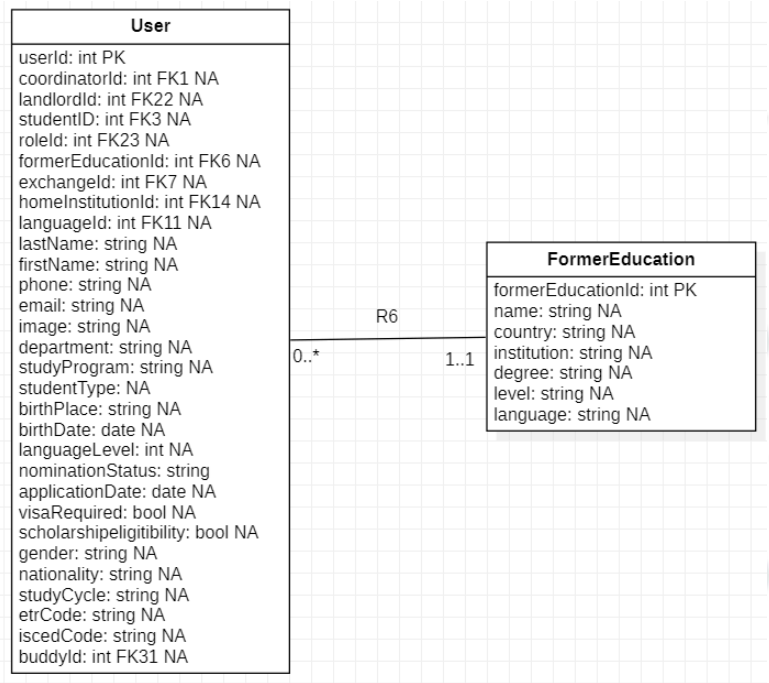
R5: A bike rental is associated with one bike, and each bike can be rented multiple times.



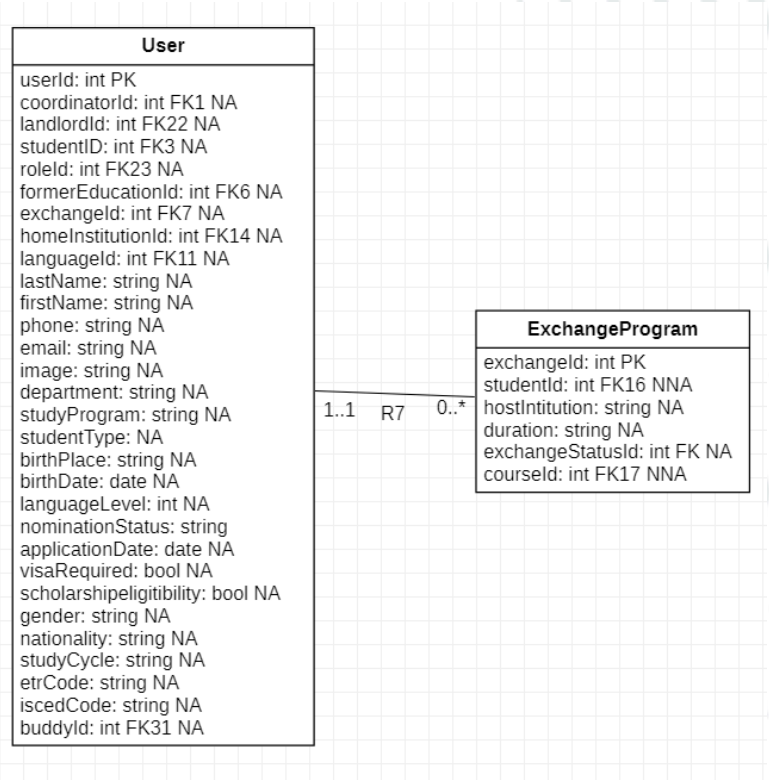
R29: A bike is categorized by one bike type, and a bike type can categorize multiple bikes.



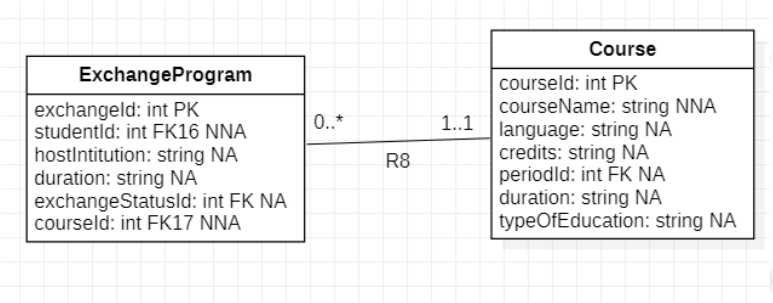
R30: A bike rental is assigned one status, and a status can be used for multiple rentals.



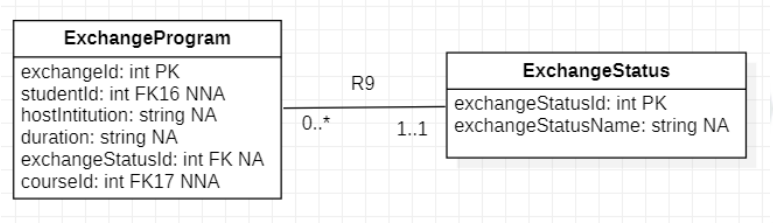
R6: A student can have multiple former education records, and each record belongs to one student.



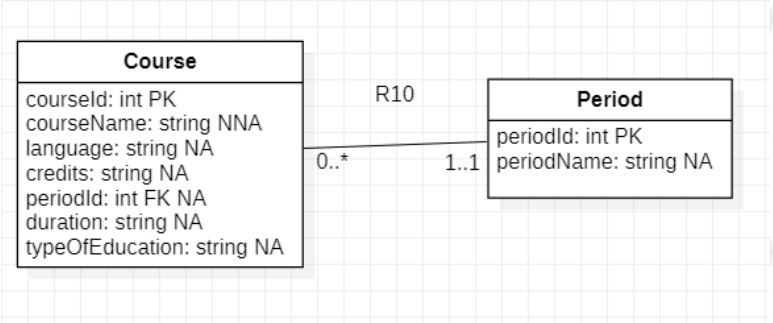
R7: A student can apply to multiple exchange programs, and each exchange program application belongs to one student.



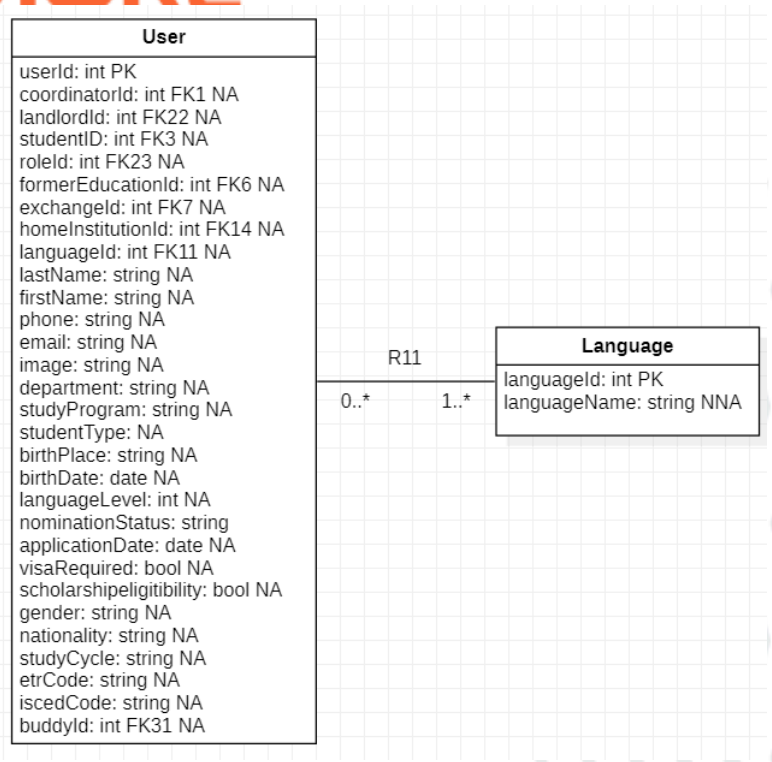
R8: An ExchangeProgram can have multiple courses, and each course belongs to one exchange program.



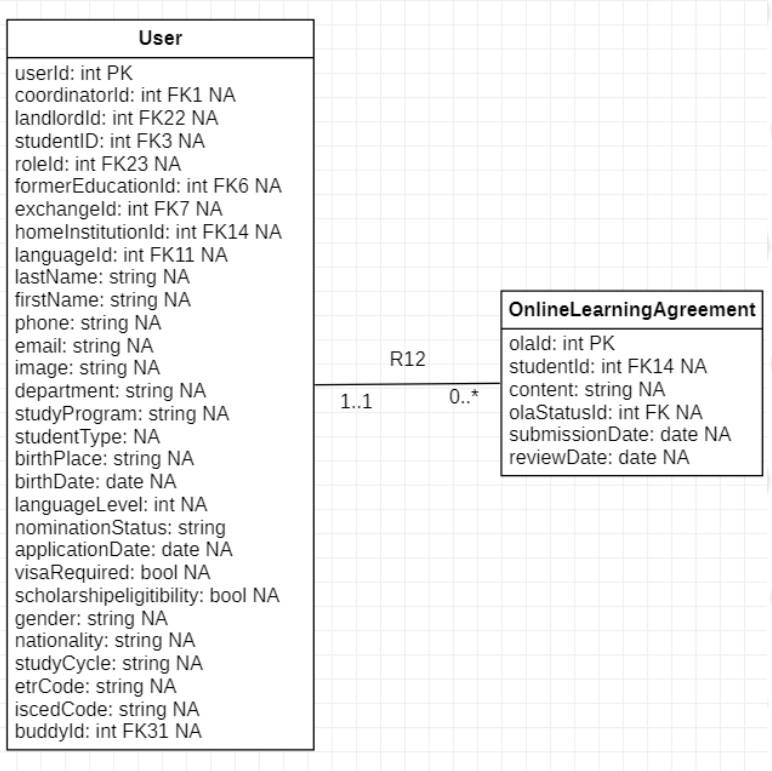
R9: An exchange program has a status, and a status can be assigned to multiple exchange programs.



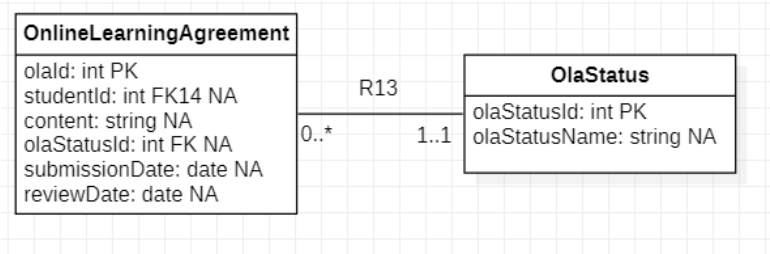
R10: A Period can have multiple courses, and each course belongs to one period



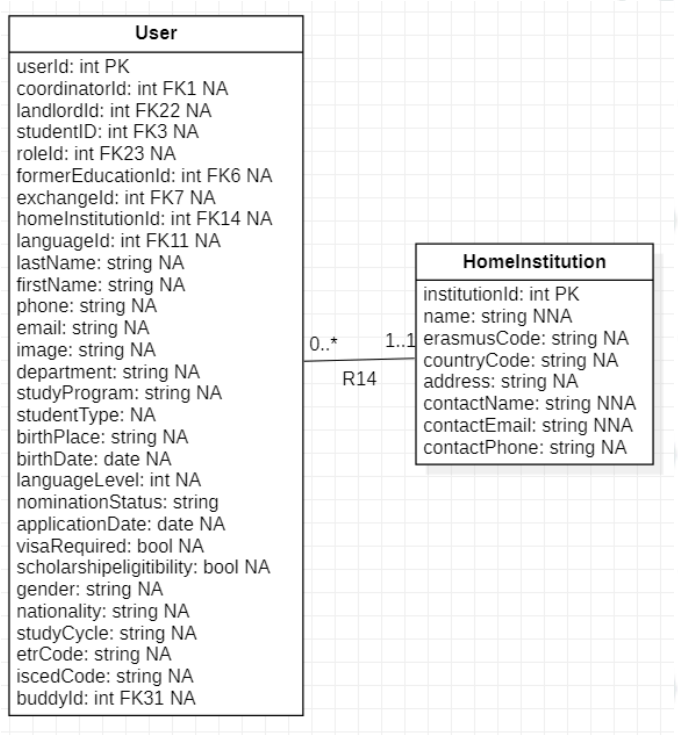
R11: A user can speak multiple languages, and a language can be associated with multiple users.



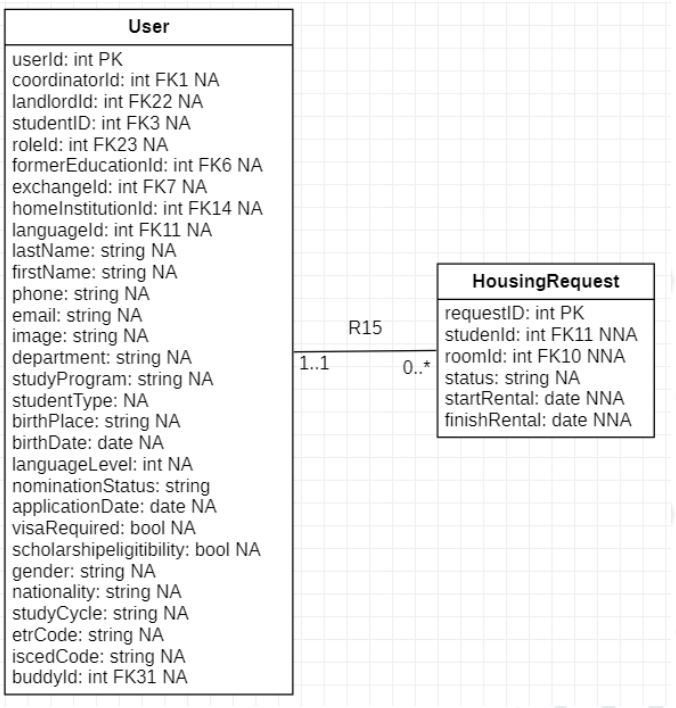
R12: A student can have one or more online learning agreements, and each agreement belongs to one student.



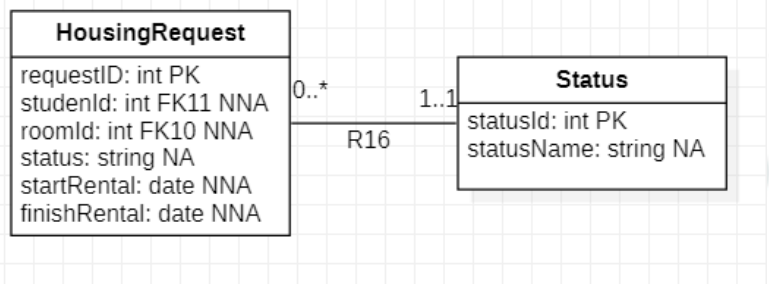
R13: An online learning agreement has one status, and a status can be assigned to multiple agreements.



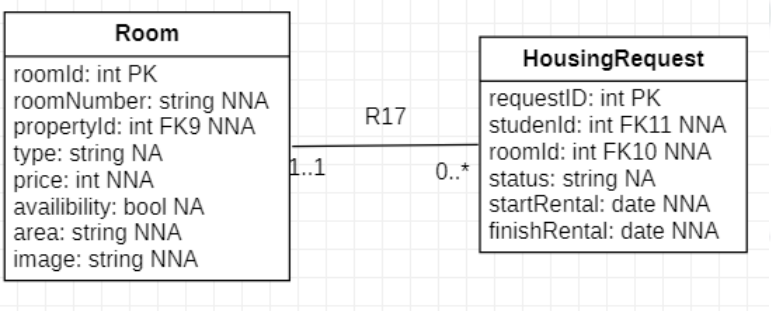
R14: A User has exactly one homeInstitution, and a homeInstitution can have multiple users



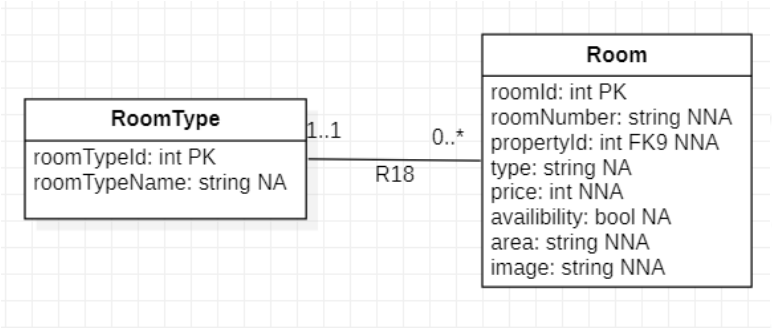
R15: A User can make multiple housing requests, and each housing request belongs to one user.



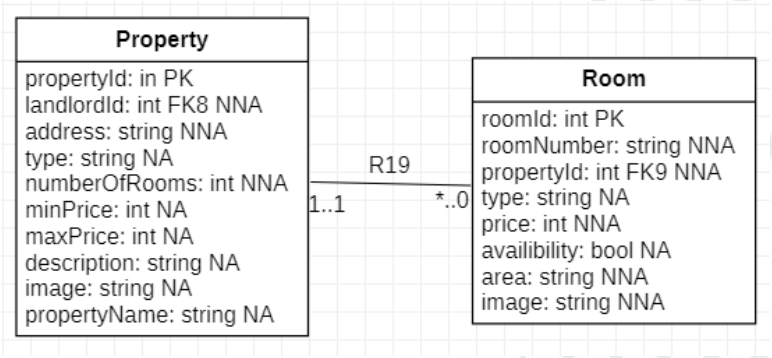
R16: A HousingRequest has exactly one status, and a status has one request.



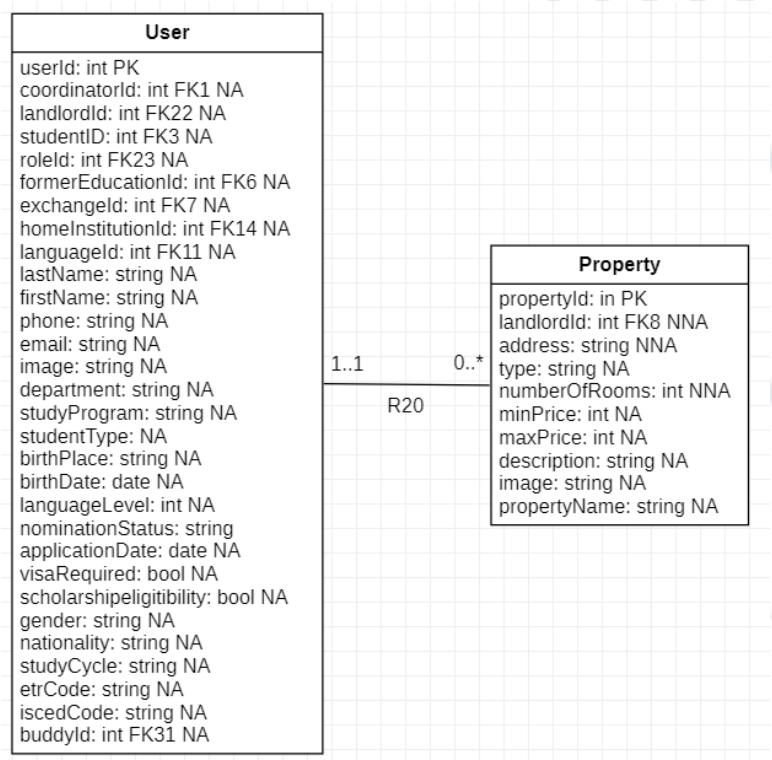
R17: A HousingRequest can be associated with one room, and a room can have multiple requests.



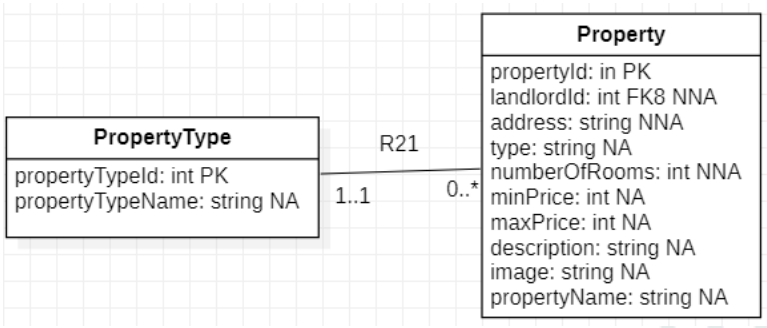
R18: A room is assigned a specific room type, and a room type can be applied to multiple rooms.



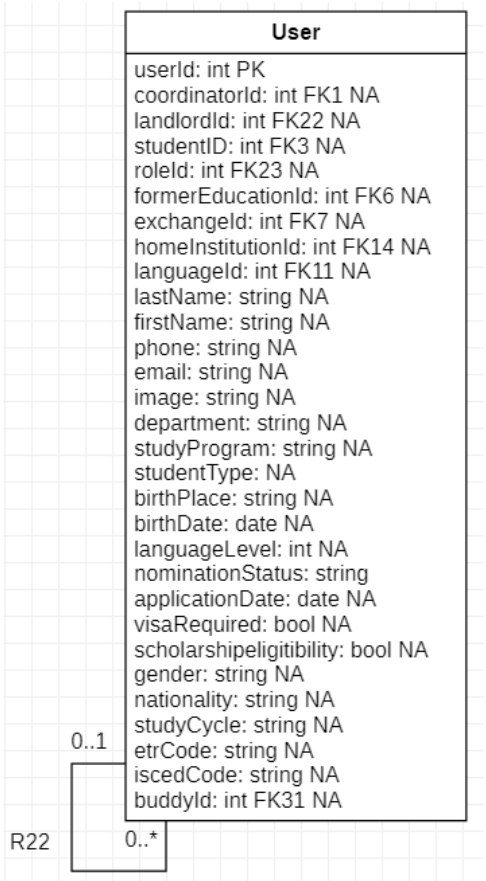
R19: A Room belongs to at most one property, and a property can have multiple rooms.



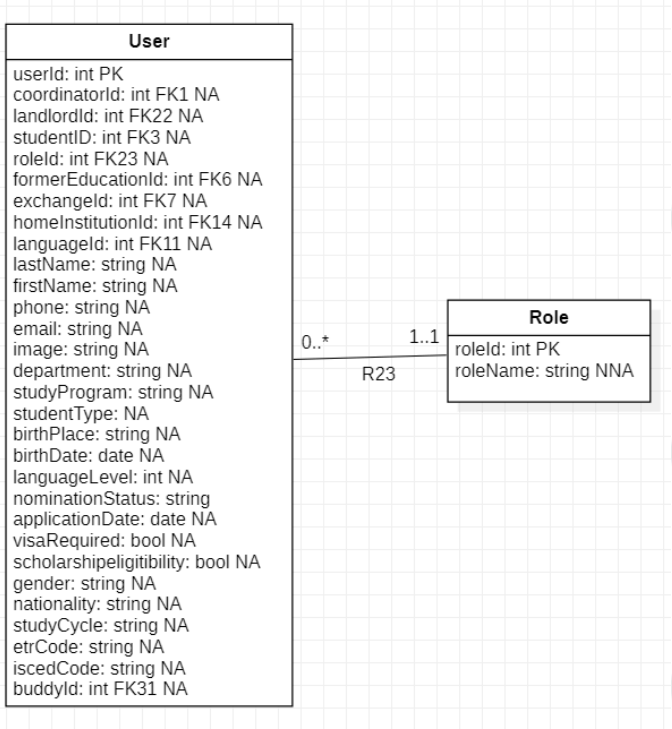
R20: A User can own multiple properties, and each property belongs to one user.



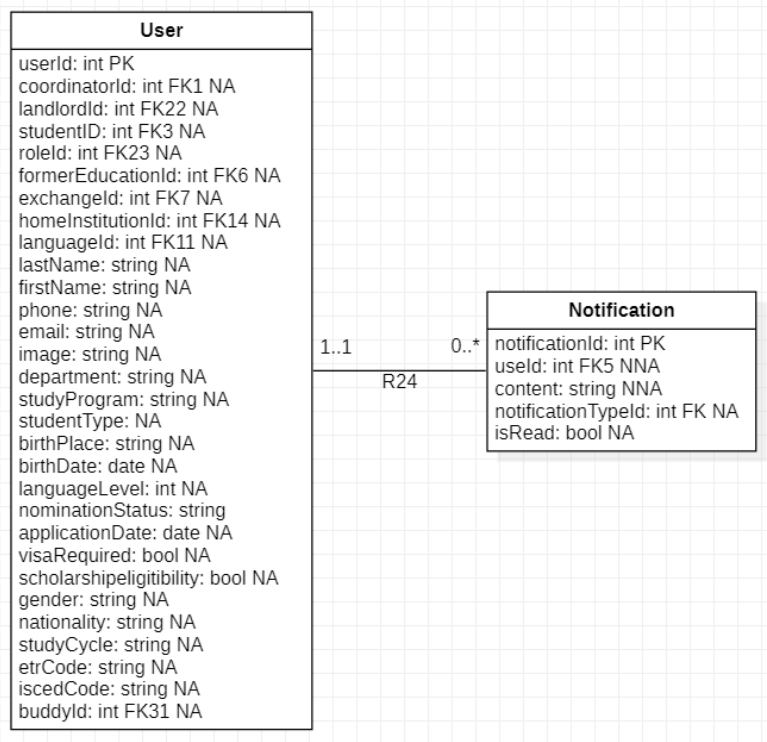
R21: A PropertyType can have multiple properties, and each property has exactly one type.



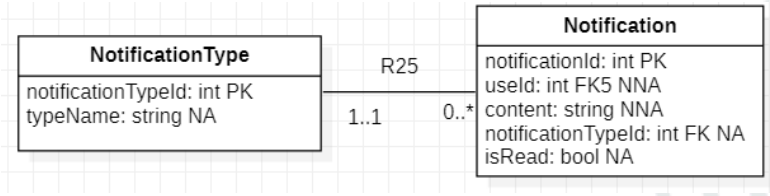
R22: A User has exactly one home institution, and a home institution can have multiple users



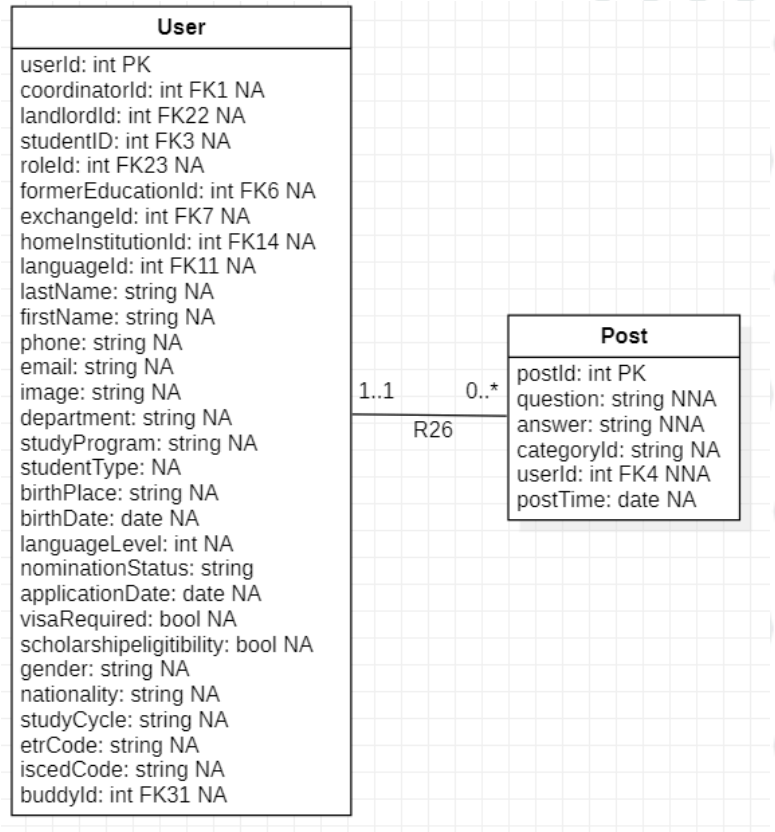
R23: A User can have one role, and a role can be assigned to multiple users



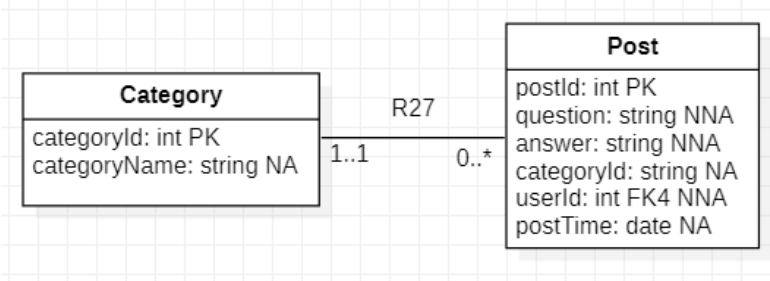
R24: A User can receive multiple notifications, and each notification belongs to one user.



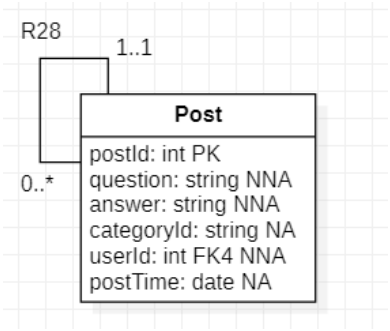
R25: A NotificationType can have multiple notifications, and each notification has exactly one type.



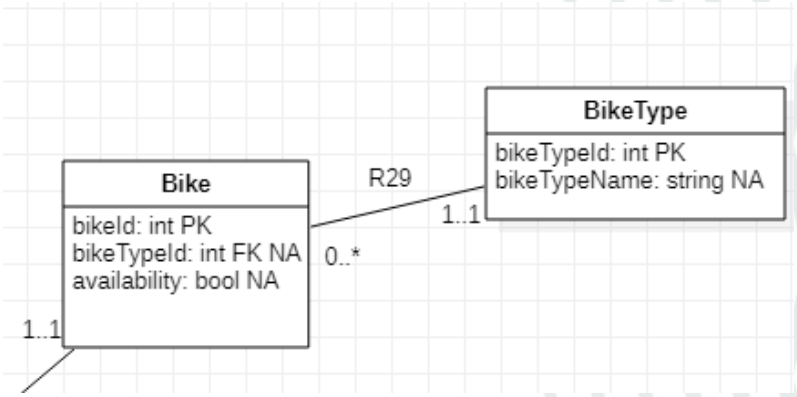
R26: A User can have multiple posts, and each post belongs to one user.



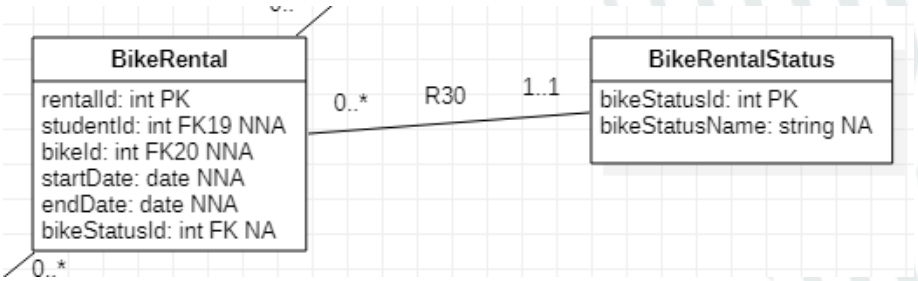
R27: A Post belongs to exactly one category, and a category can have multiple posts.



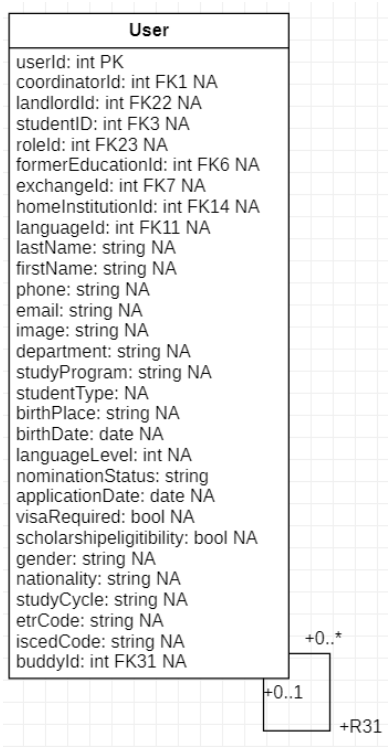
R28: A User can create multiple posts, and each post belongs to one user.



R29: A Bike has exactly one bike type, and a bike type can have multiple bikes.



R30: A BikeRental has exactly one rental status, and a rental status can have multiple rentals.



R31: A HousingRequest belongs to exactly one home institution, and a home institution can have multiple requests.

3. User Stories

Epic: Post question
User Stories:

- As a user, I want to post a question, so that I can seek information or assistance.
- As a user, I want to categorize my question, so that it is easier to find relevant answers.
- As a user, I want to edit my posted question, so that I can make corrections if needed.
- As a user, I want to delete my question, so that I can remove unnecessary or incorrect posts.

Epic: Assign buddy
User Stories:

- As an International Office, I want to assign a buddy to a student, so that they can receive guidance and support.
- As an International Office, I want to view the list of available buddies, so that I can make an appropriate match.
- As an International Office, I want to reassign a buddy, so that I can adjust pairings when needed.
- As an International Office, I want to remove a buddy assignment, so that I can update the system when changes occur.

Epic: Review application
User Stories:

- As a coordinator, I want to review student applications, so that I can determine eligibility.
- As a coordinator, I want to approve or reject an application, so that I can manage admissions.
- As a coordinator, I want to add comments to an application, so that I can provide feedback or request additional information.
- As a coordinator, I want to filter applications by status, so that I can manage pending, approved, and rejected cases efficiently.

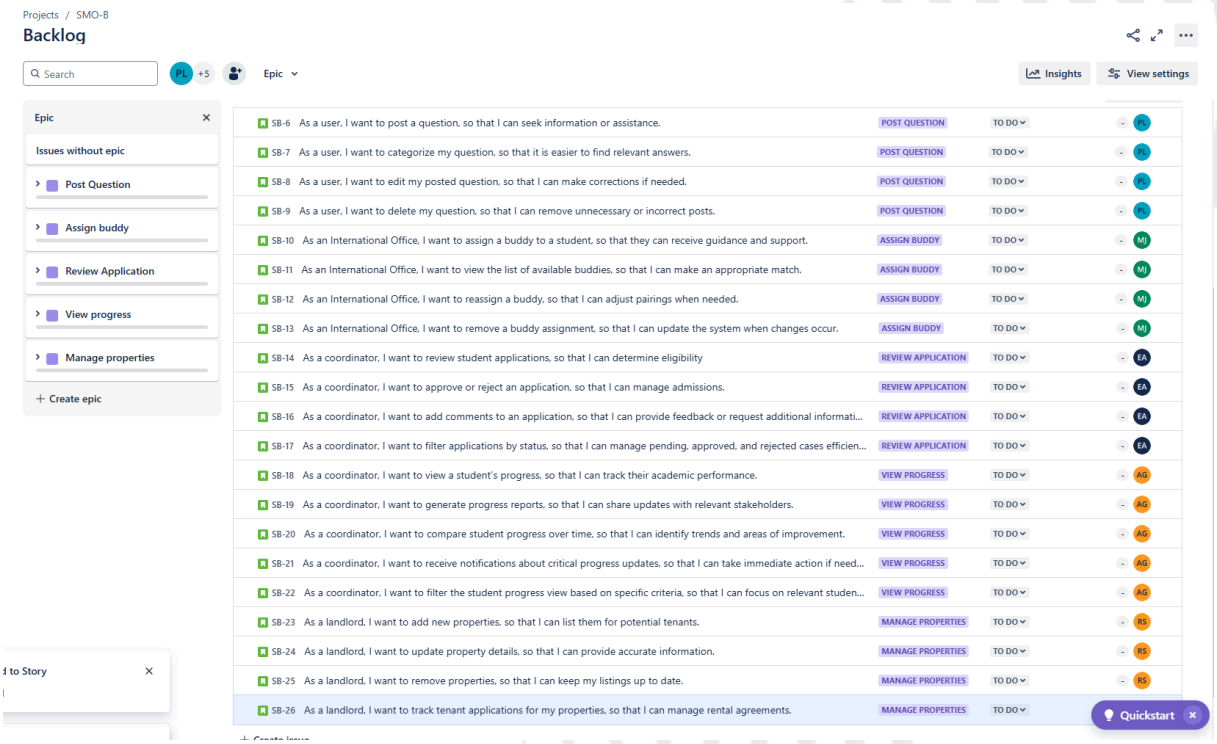
Epic: View progress
User Stories:

- As a coordinator, I want to view a student's progress, so that I can track their academic performance.
- As a coordinator, I want to generate progress reports, so that I can share updates with relevant stakeholders.
- As a coordinator, I want to compare student progress over time, so that I can identify trends and areas of improvement.
- As a coordinator, I want to receive notifications about critical progress updates, so that I can take immediate action if needed.
- As a coordinator, I want to filter the student progress view based on specific criteria, so that I can focus on relevant students.

Epic: Manage properties
User Stories:

- As a landlord, I want to add new properties, so that I can list them for potential tenants.
- As a landlord, I want to update property details, so that I can provide accurate information.
- As a landlord, I want to remove properties, so that I can keep my listings up to date.
- As a landlord, I want to track tenant applications for my properties, so that I can manage rental agreements.

THOMAS MORE



2024 - 2025