THOMAS MORE

GYM APPLICATION – GROW KASTERLEE

TEAM: KLA2

User Requirements Specification

Karlis Kalnakarklis,
Rune Lemmens,
Victor Nwachukwu Chukwuma,
Mehmet Görmez,
Denys Herasymchuk,
Umer Umer

Table of contents

1. INTRODUCTION 5

2. REQUIREMENTS ANALYSIS 6

2.1. Functional requirements 7

2.1. Use Case Diagram 7

2.1.2. Use Cases 9

2.1.2.1. Login 9

2.1.2.2. Register Profile 10

2.1.2.3. Purchase Credits 11

2.1.2.4. View Remaining Credits 12

2.1.2.5. View Workout Plan 13

2.1.2.6. View Session Schedule 14

2.1.2.7. Book Session 15

2.1.2.8. Confirm Individual Session 17

2.1.2.9. Confirm Shared Session 18

2.1.2.10. View Booked Sessions 19

2.1.2.12. Manage Settings 20

2.1.2.13. Update Profile Details 22

2.1.2.14. Update Workout History 23

2.1.2.15. Send Booking Reminder 24

2.1.2.16. Manage Booking Schedule 25

2.1.2.17. View Booked Sessions 26

2.1.2.18. Manage Exercise Type 27

2.1.2.19. Manage Workout Plan 28

2.1.2.20. View Member Details 30

2.1.2.21. Track Fitness Progress Details 31

2.1.2.22. Manage Personal Settings 33

2.1.1.23. Generate Invoices 34

2.1.2.24. View Session Attendance 35

2.1.2.25. Manage Workout Equipment 36

2.1.2.26. Manage Session Types 38

2.1.2.27. Manage Users 39

2.2. Non-functional requirements 41

2.2.1. User Interface 41

2.2.2. Performance 41

2.2.3. Security 41

2.2.4. Usability 42

2.2.5. Reliability 42

2.2.6. Scalability 42

2.2.7. Availability 42

2.2.8. Maintainability 43

2.2.9. Compatibility 43

2.2.10. Data Privacy and Compliance 43

2.2.11. Notification System 43

2.2.12. Payment Processing 44

2.2.13. Localization 44
2.2.14. Testing Requirements 44

3. DATA MODEL 45

4. PRIORITY BY FUNCTIONALITY 57

a

Before submitting this document, make sure to check that:

- **the document does not contain any spelling mistakes**
- **all template text (indicated in *red*) is removed, including this box**
- **the document layout is neat (table of contents is updated, formatting is consistent, page breaks are properly set…)**

1. Introduction

This report presents a comprehensive analysis and design for the **Grow Kasterlee Gym Application**, a digital solution developed within the Skills Integration Lab. The project serves as a User Requirements Specification designed to modernize the interaction between the gym and its various stakeholders, including members, trainers, assistants, and administrators. By establishing clear functional guidelines and a structured data architecture, the application aims to streamline the process of managing fitness journeys and facility operations.

The core of the system is defined by an extensive set of functional requirements that prioritize user autonomy and administrative efficiency. Members are provided with tools to register profiles, manage credit balances, and navigate a dynamic session schedule to book individual or shared workouts. For the gym staff, the application facilitates specialized management tasks such as maintaining training schedules, tracking client fitness progress, and generating invoices. These interactions are supported by automated background processes, such as the time-triggered delivery of booking reminders to enhance user engagement.

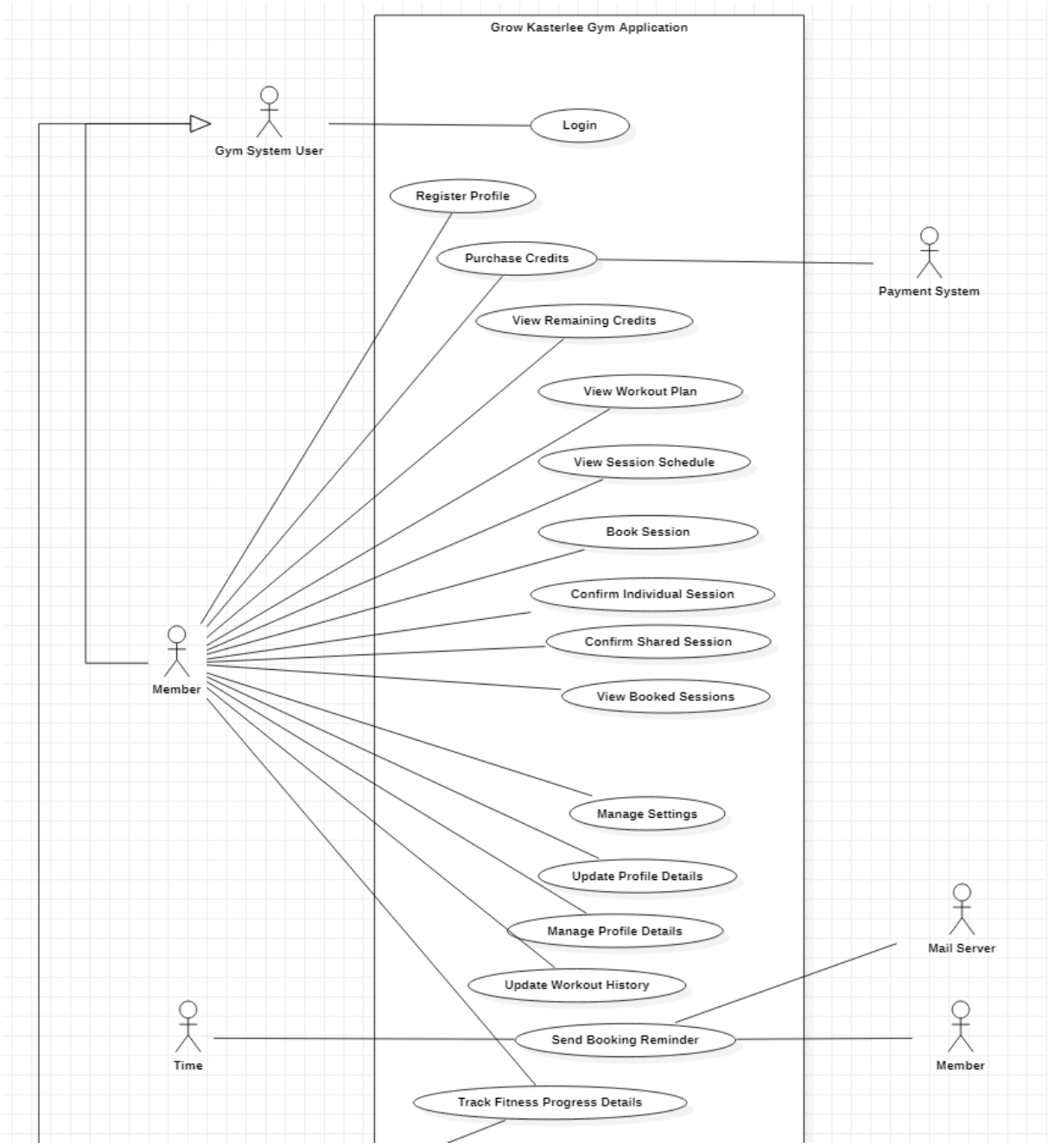
To ensure the system's reliability and scalability, the design is underpinned by a detailed data model. This architecture organizes complex data points ranging from user roles and workout histories to session types and equipment requirements into a cohesive structure that ensures data integrity across all modules. By categorizing these features into a prioritized hierarchy of Must-have through Could-have functionalities, this report provides a strategic roadmap for the successful implementation and deployment of the Grow Kasterlee gym system.

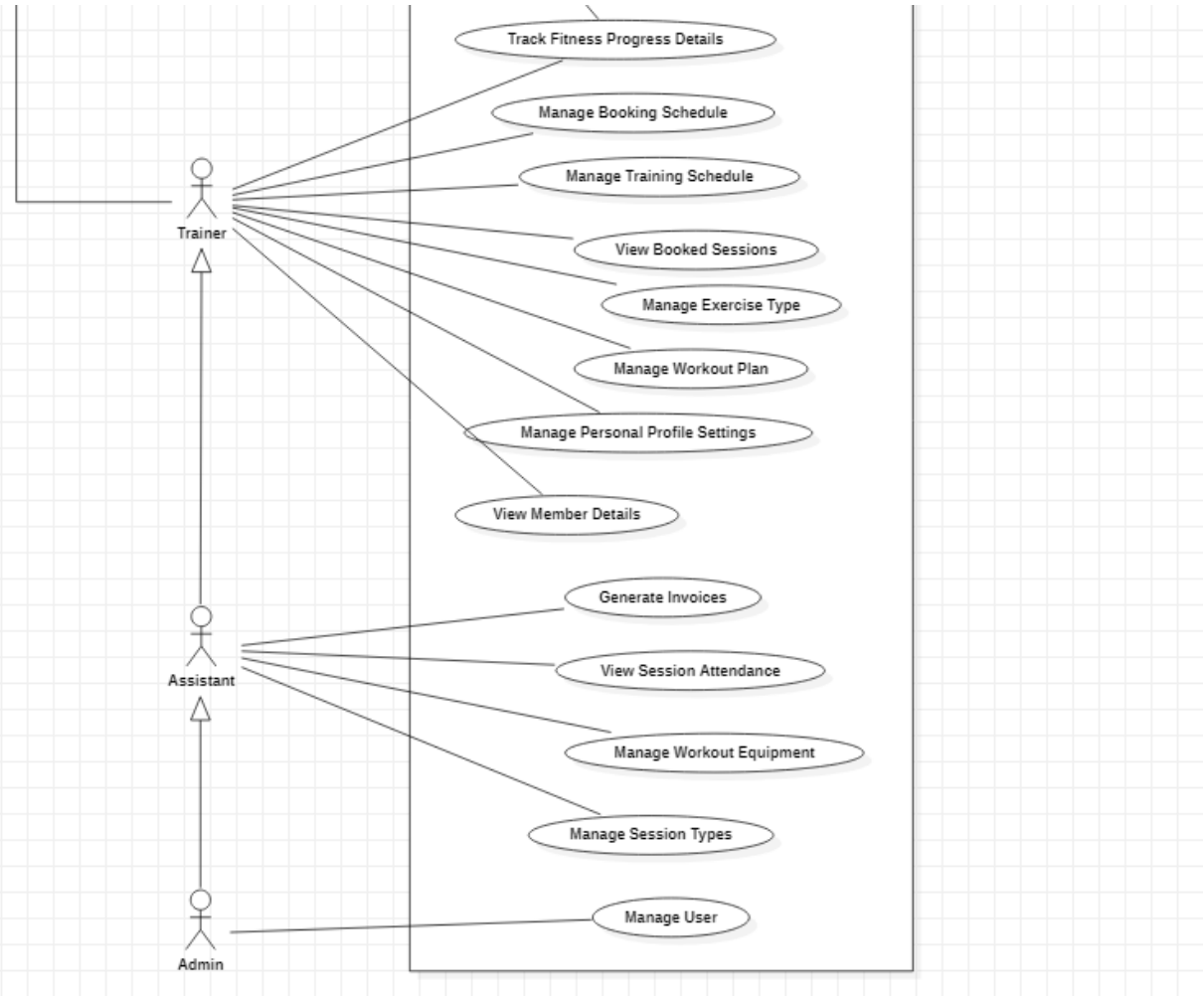
2. Requirements analysis

The following section defines the essential functionalities and operational constraints of the Grow Kasterlee gym application to ensure it meets the needs of all system stakeholders. Through a detailed functional analysis, this report identifies core interactions for members, trainers, and administrative staff, ranging from secure profile management and session booking to fitness progress tracking and automated reminders. By establishing these requirements, the system provides a structured framework for a seamless user experience while maintaining robust data integrity across the platform's various roles and services.

2.1. Functional requirements

2.1. Use Case Diagram



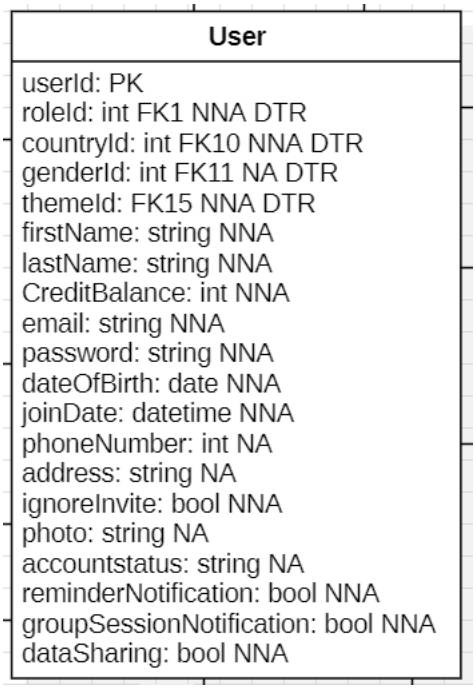

Track Fitness Progress Details
Manage Booking Schedule
Manage Training Schedule
Trainer
View Booked Sessions
Manage Exercise Type
Manage Workout Plan
Manage Personal Profile Settings
View Member Details
Generate Invoices
Assistant
View Session Attendance
Manage Workout Equipment
Manage Session Types
Manage User
Admin

2.1.2. Use Cases

2.1.2.1. Login

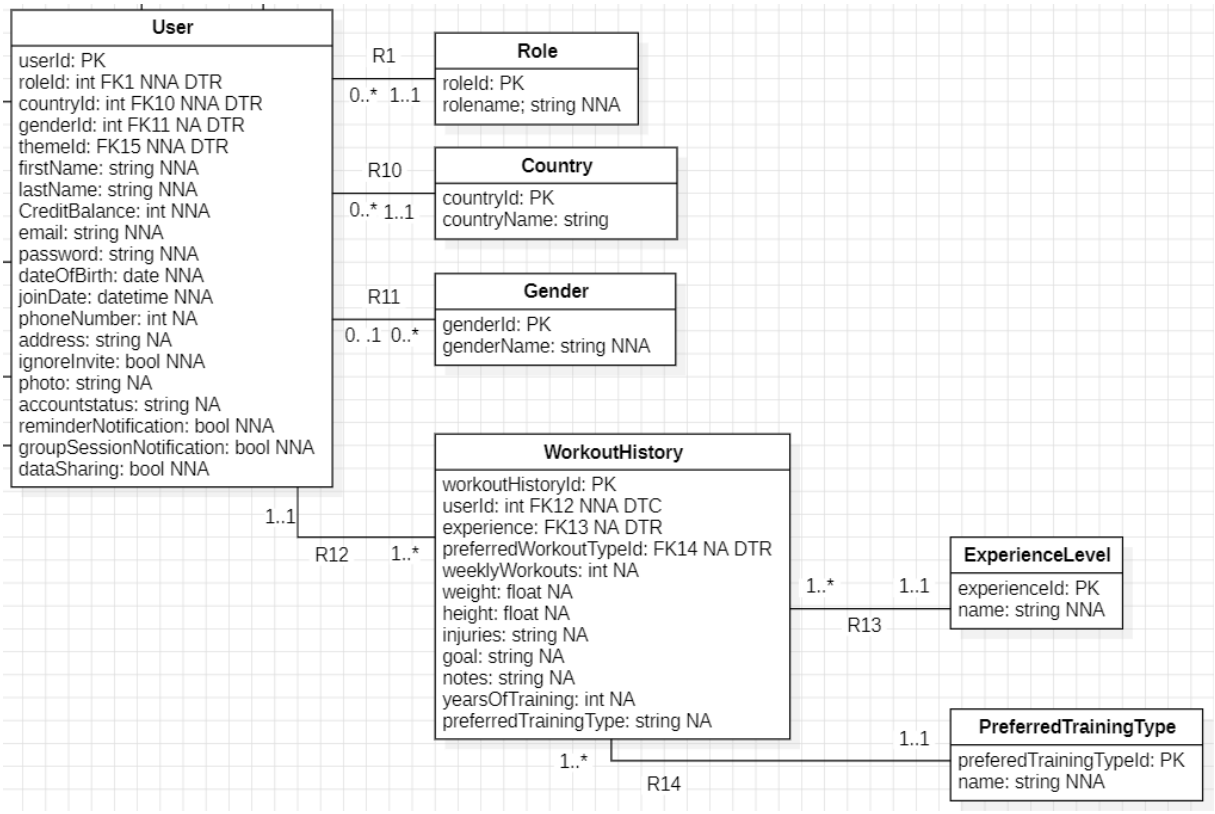
Functionality: As a Gym System User, I can login.
Preconditions: The actor has a registered profile in the gym system.
Normal flow: The system displays a login form with an email and password field and an option to remember the actors account in the browser. The actor enters their credentials. The system grants access to the system.
Alternatives: The password is incorrect; the system shows an error message requesting the actor to try again.

Data model view



2.1.2.2. Register Profile

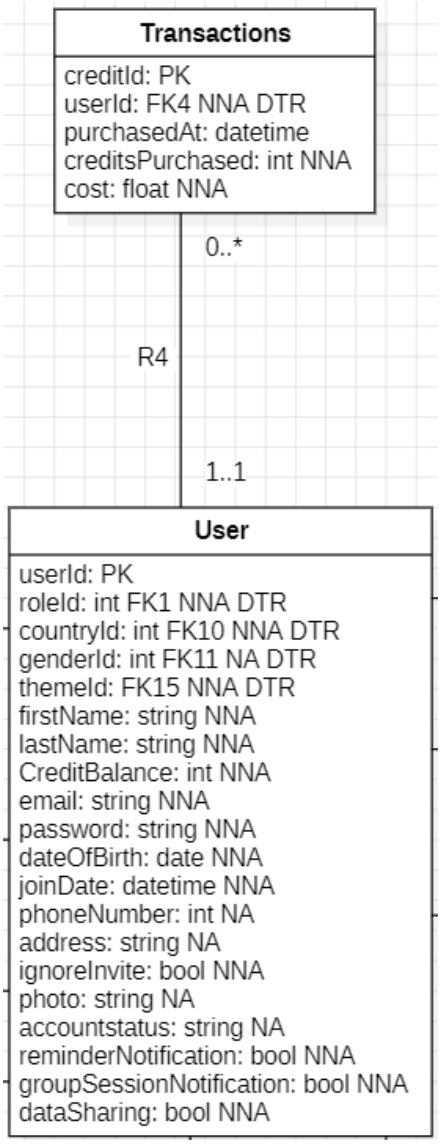
Functionality: As a Member, I can register profile.
Normal flow: The system displays a registration form with fields for first name, last name, email, phone number, date of birth, gender and password, and a checkbox that the actor consents to the terms of service. The actor enters their details and confirms. The system displays a workout history window with fields for weight, height, expected weekly workout days, experience level, if they have any injuries, their main goal, any extra notes, preferred training type and years that they have been training. The actor enters their details and confirms their workout history. The system displays fields for the preferred session type the actor will train in (e.g. individual, group, or couple). The actor selects one of the shared training types (e.g. group or couple). The system displays a search bar where the actor searches for their training partner by first and last name. The actor selects their training partners and confirms their selection. The system saves the settings of the actor's profile and sends a confirmation link to the user that the actor selected for their shared session. The system redirects the actor to the profile details page.
Alternatives: The email address is already registered; the system displays an error message indicating that there is already an account registered under this email.
Data model view:



2.1.2.3. Purchase Credits

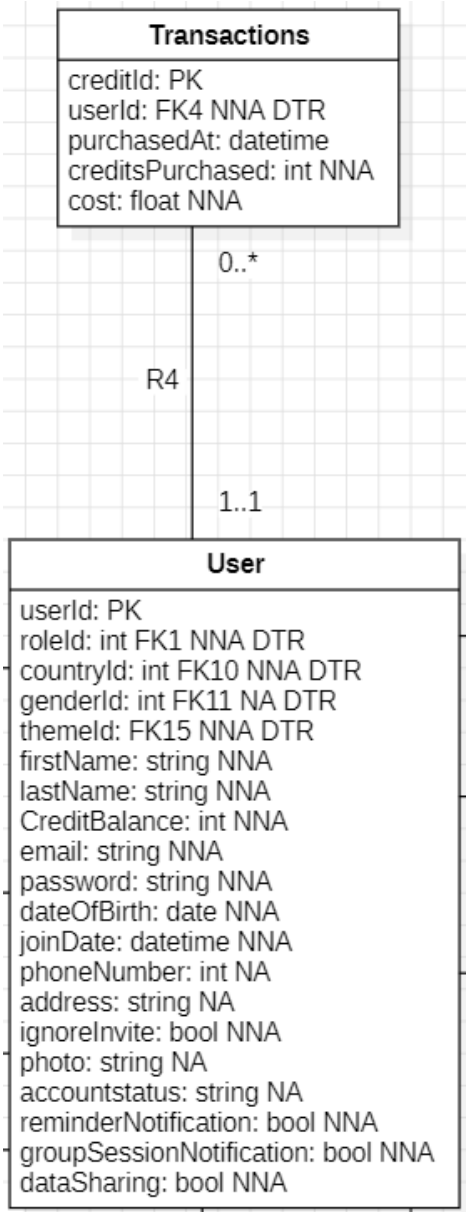
Functionality: As a Member, I can purchase credits.
Preconditions: The actor is logged in.
Normal flow: The system displays the credit purchase screen showing the current credit balance and the fixed price of €1 per credit, the actor enters the number of credits to purchase and confirms the action, the payment is handled externally outside the application, and upon successful payment the system adds the purchased credits to the actor's balance and displays a confirmation message with the updated balance.

Data model view:



2.1.2.4. View Remaining Credits

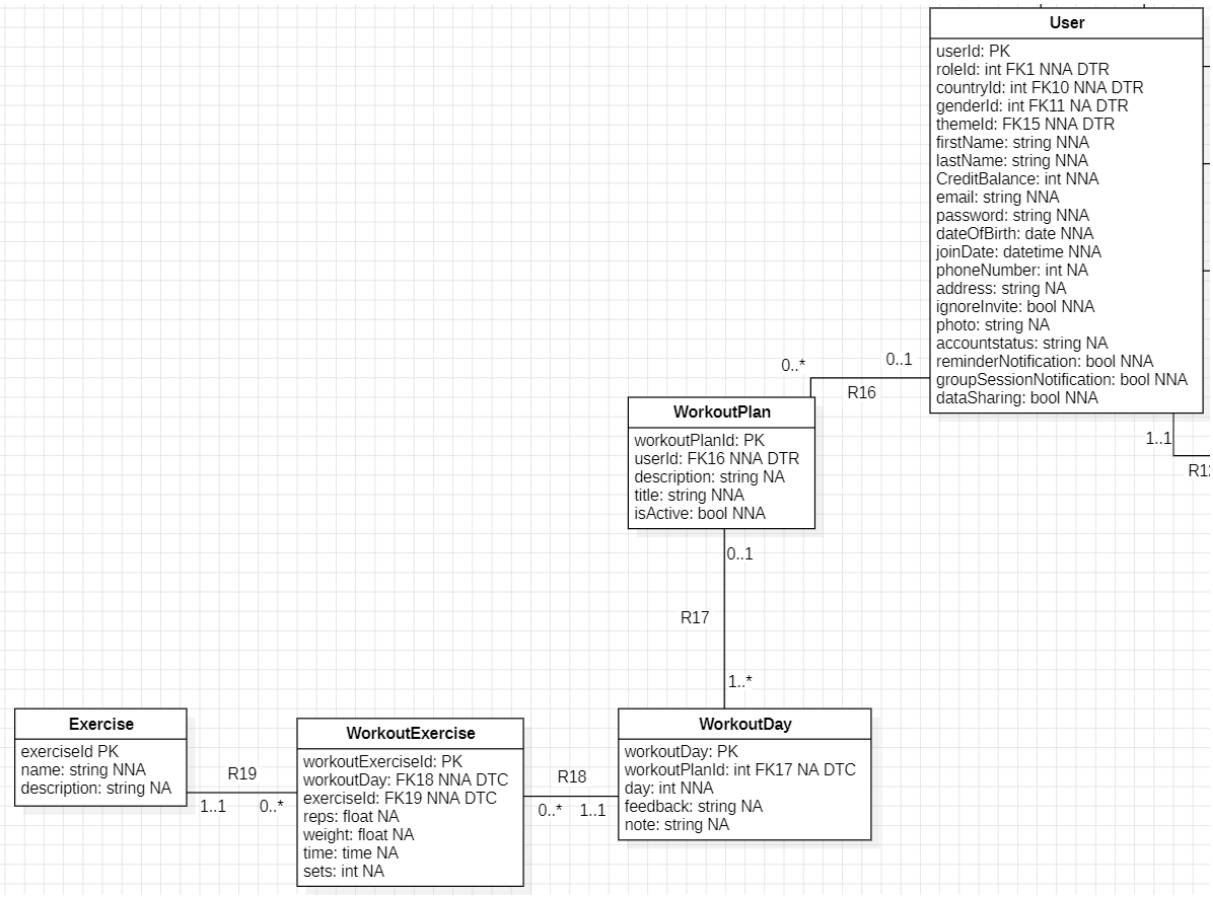
Functionality: As a **Member**, I can view remaining credits.
Preconditions: The actor is logged in.
Normal flow: The system retrieves the actor's current credit balance. The system displays the remaining credits on the screen.
Data model view:



2.1.2.5. View Workout Plan

Functionality: As a Member, I can view workout plan.
Preconditions: The actor is logged in and has at least one assigned workout plan.
Normal flow: The system first displays an overview of all workout plans associated with their profile, clearly distinguishing between active plans and past plans. The actor selects any plan they want to review. Once a plan is chosen, the system retrieves its detailed structure, including the exercises it contains, the number of sets and repetitions for each exercise, and the days on which the exercises are scheduled. The system then presents this information in an organized and readable layout.

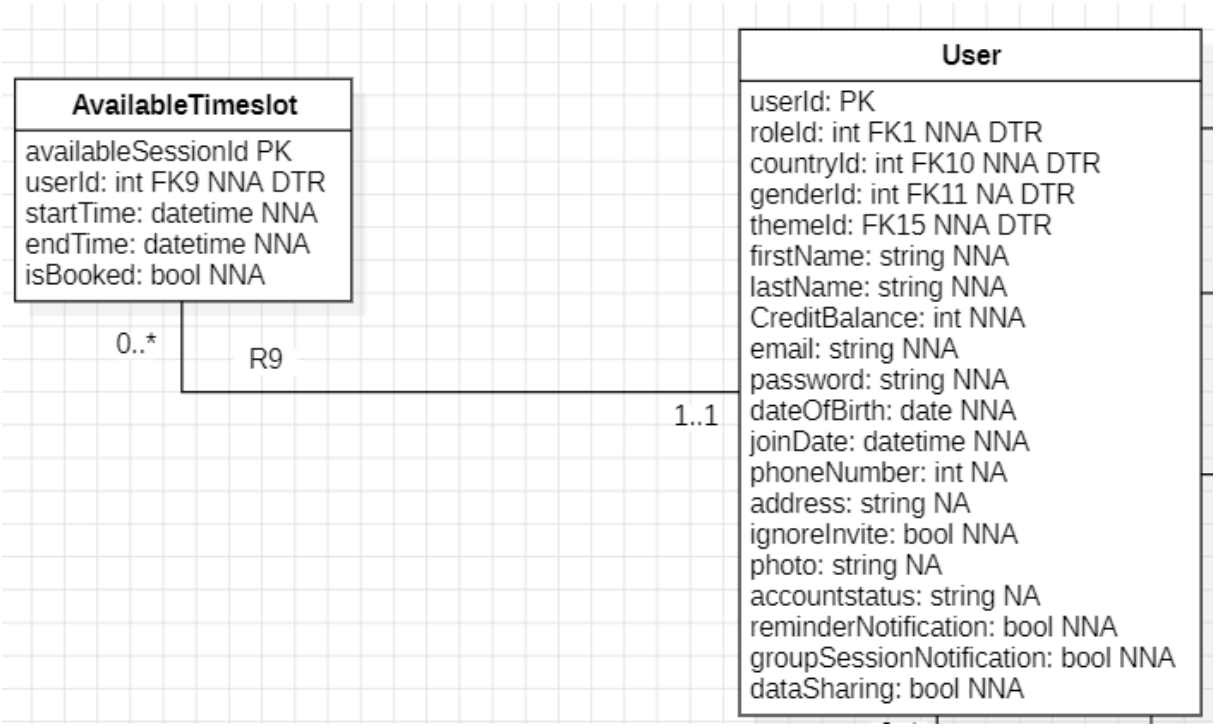
Data model view:



2.1.2.6. View Session Schedule

Functionality: As a Member, I can view session schedule.
Preconditions: The actor is logged in.
Normal flow: The system displays a monthly calendar showing all days within the selected period. Each day in the calendar visually indicates whether sessions are available. When the actor selects a specific day on this calendar, the system retrieves session information for that date, such as the assigned trainer, the scheduled time, and the remaining available slots.

Data model view:



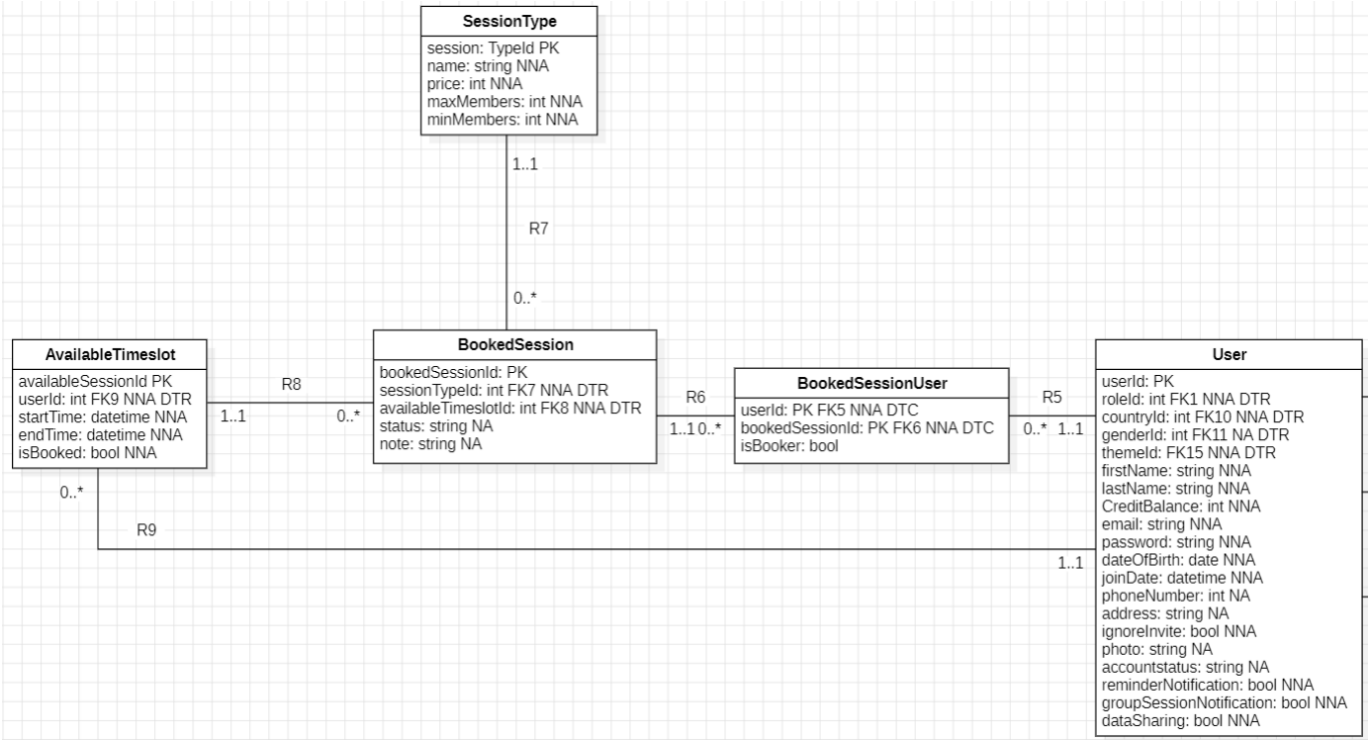
2.1.2.7. Book Session

Functionality: As a Member, I can book sessions.
Preconditions: The actor is logged in.
Normal flow: The system displays the session booking screen showing the selected date and time, trainer information, session type options (Individual or Shared) and an additional notes field. The actor selects the Individual option and, if desired, enters notes. The system verifies the availability of the selected time slot, deducts the required credits from the actor's account, saves the session booking, and then displays a booking confirmation with the session details.
Alternatives:

Shared sessions: The actor selects the Shared option and the system shows the notes field along with a partner selection area that includes search functionality and a list of registered participants. The actor selects one or more partners, up to the system limit. The system verifies slot availability and the availability of all selected participants, processes the required credit deductions for the actor and for any applicable partners, saves the booking including the partner list and then displays the booking confirmation.

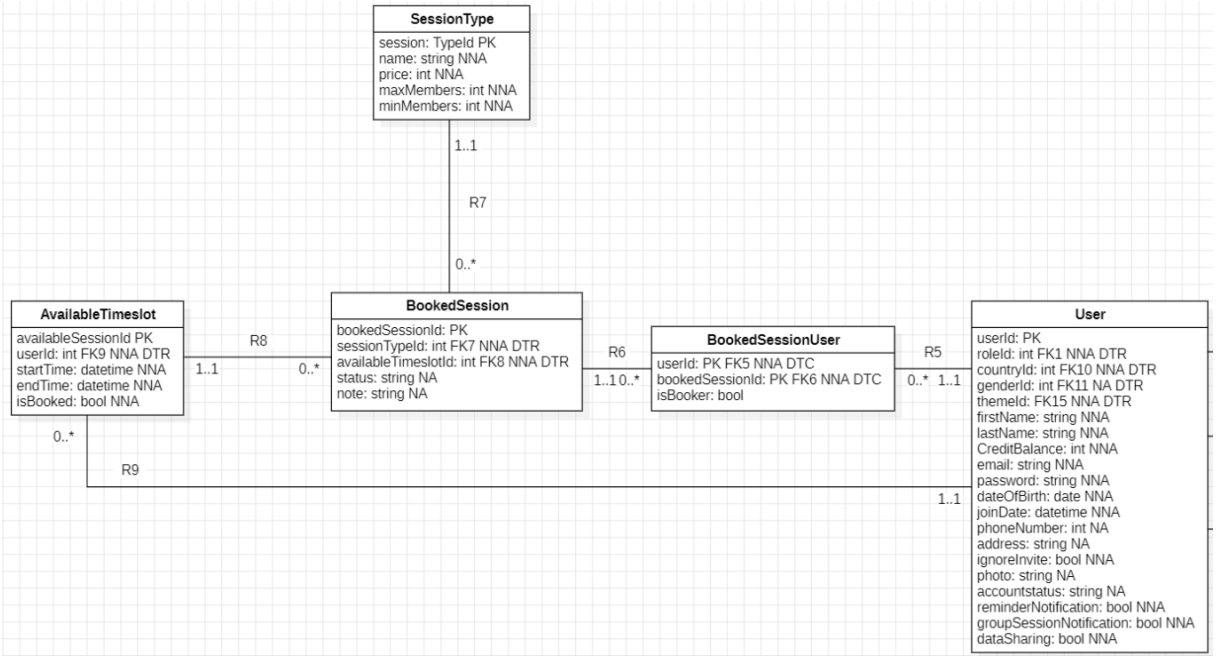
Insufficient credits: The system detects that the actor does not have enough credits during the deduction step. The system notifies the actor and suggests purchasing additional credits before continuing, after which the actor may cancel or proceed with the credit purchase.

Data model view:



2.1.2.8. Confirm Individual Session

Functionality: As a Member, I can confirm individual sessions.
Preconditions: The actor is logged in and has sufficient credits for the selected session.
Normal flow: The system displays the individual session details on the screen, including the session time, date, trainer's name, and the credit cost of the session. Beneath this information, the system shows a confirmation checkbox with the statement: "I accept that 2 credits will be deducted from my balance." The actor reviews the details, selects the checkbox to confirm acceptance of the credit deduction and clicks the Book button. The system verifies the actor's credit balance, deducts the required credits, saves the booking and displays a confirmation message with the session details.
Alternatives: If the actor does not have enough credits, the system displays a notification explaining that the booking cannot proceed and suggests purchasing additional credits.
Data model view:



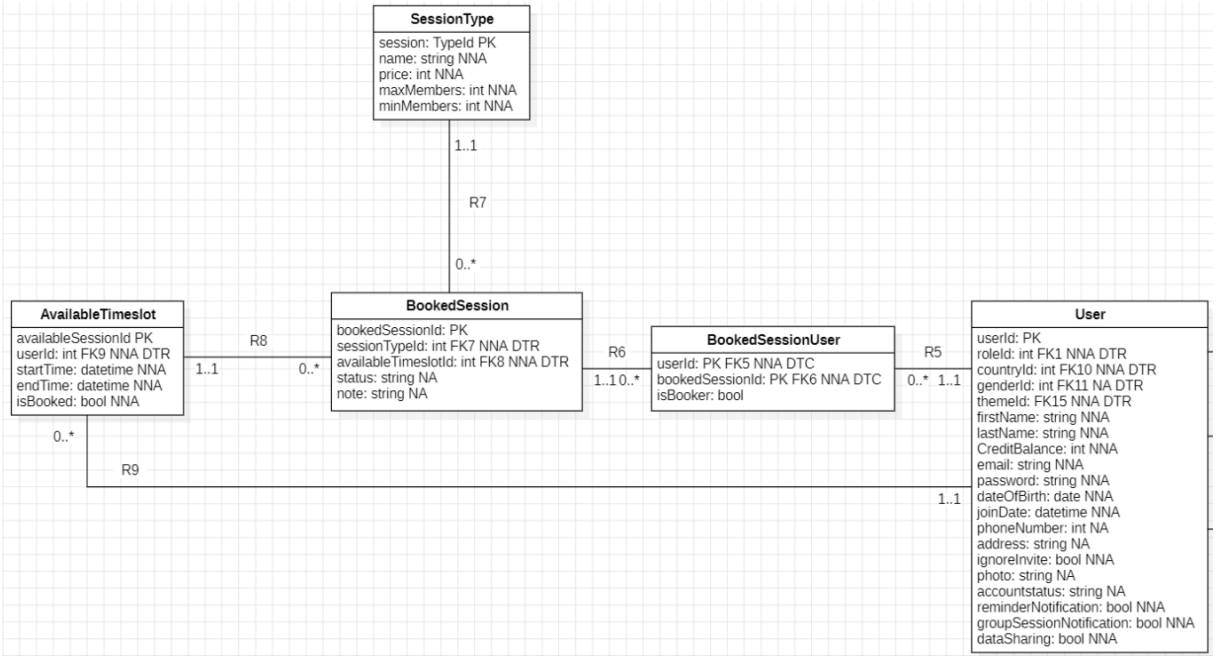
2.1.2.9. Confirm Shared Session

Functionality: As a **Member**, I can confirm shared sessions.
Preconditions: The actor is a registered user who has received an invitation, is logged in, and has sufficient credits.
Normal flow: The invited actor receives a system notification or email containing the session details. The actor can confirm participation directly through the system. Once confirmed, the system deducts the required credits from the actor's account, marks their status as confirmed, and updates the session participant list. Both the actor and the trainer are notified of the confirmation.
Alternatives:

Invitation Declined: If an invited actor declines the invitation or does not confirm attendance within the validity window (to be determined), the system automatically releases the reserved spot and updates the session's availability.

Insufficient credits: If they lack sufficient credits, the system notifies them and provides the option to purchase additional credits before confirming their attendance.

Data model view:

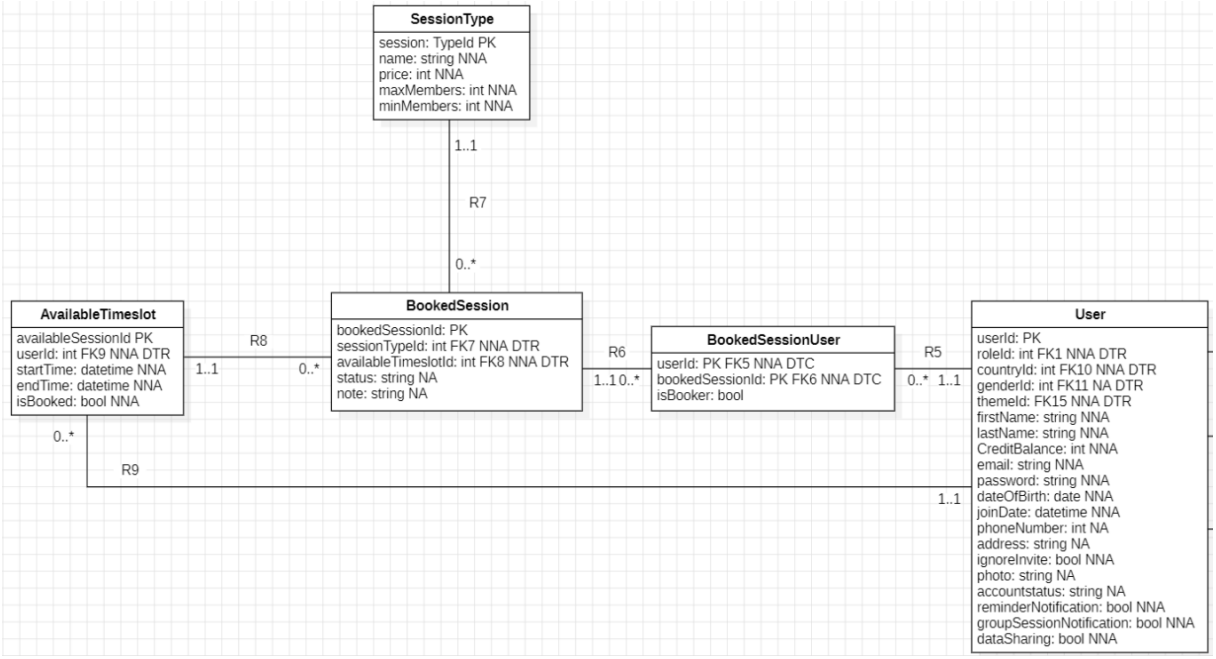


2.1.2.10. View Booked Sessions

Functionality: As a Member, I can view booked sessions.
Preconditions: The actor is logged in.
Normal flow: The system displays a list of all booked sessions for the actor. Each item in the list shows the session date, time, session type, people involved, status and the credit cost. The list allows the actor to review all upcoming and past bookings in a clear and organized overview.
Alternatives:

Cancel Session: For each session in the list, the system displays a menu button represented by three dots. When the actor clicks this button, a pop-up appears asking if they want to cancel the selected session. If the actor approves the action, the system sends a cancellation request for that session and informs the actor that the request has been submitted. If the actor declines, the system closes the pop-up without making any changes.

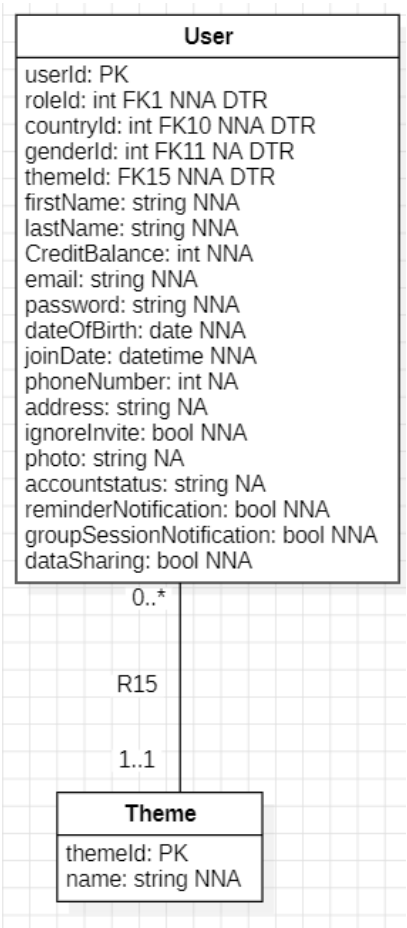
Data model view:



2.1.2.12. Manage Settings

Functionality: As a Member, I can manage settings.
Preconditions: The actor is logged in.
Normal flow: The system displays the settings page, The actor can adjust several types of preferences. The actor may change the visual theme by selecting a preferred color mode under the Appearance section. In the Notifications section, the actor can enable or disable reminder notifications for upcoming sessions and toggle group session notifications to receive updates when someone books a session in which they participate. In the Privacy & Security section, the actor can choose whether to allow data sharing for analytics purposes and may open the change password flow if they wish to update their account password. After making changes, the actor clicks the Save Changes button, and the system updates the stored settings and confirms that the preferences have been successfully saved. The actor may also choose Reset to Defaults, in which case the system restores all settings to their original default values.
Alternatives:
If the actor attempts to save changes but a required field is incomplete or invalid, the system prevents the update and displays an explanatory message. If the actor cancels or leaves the page without saving, the system discards any unsaved changes.

Data model view:



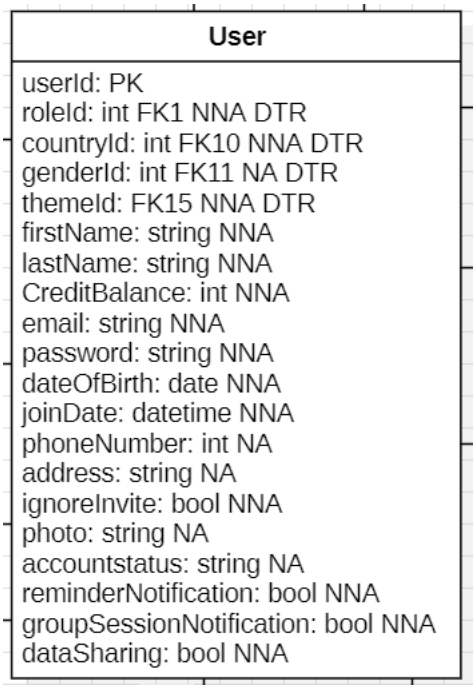
2.1.2.13. Update Profile Details

Functionality: As a Member, I can update profile details.
Preconditions: The actor is logged in.
Normal flow: The system displays the profile page, showing editable fields such as first name, second name, email, phone number, country, address and date of birth. The actor updates any of these fields and confirms the changes. The system validates the input, saves the updated information and notifies the actor that the profile has been successfully updated.
Alternatives:

Change password: When the actor selects the change password option, the system opens a dialog with fields for the old password, the new password and its confirmation. After the actor submits the form, the system verifies the old password, checks that the new passwords match and validates the new password. If everything is correct, the system updates the password and shows confirmation. If not, the system requests corrections.

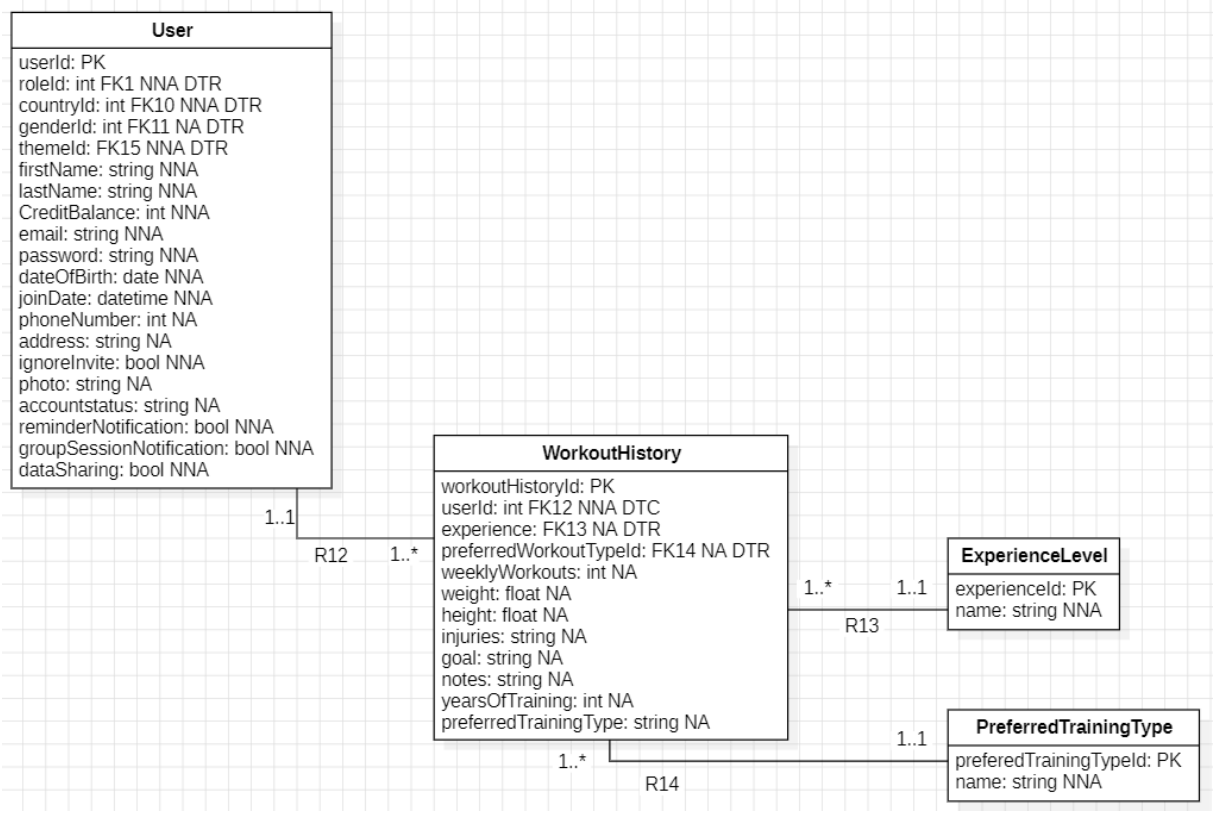
Set Preferred Partners: When the actor chooses to set preferred partners, the system opens a screen with a searchable list of registered participants and shows the maximum number of partners allowed. The actor selects or deselects partners and confirms. The system saves the updated list and shows a confirmation message.

Data model view:



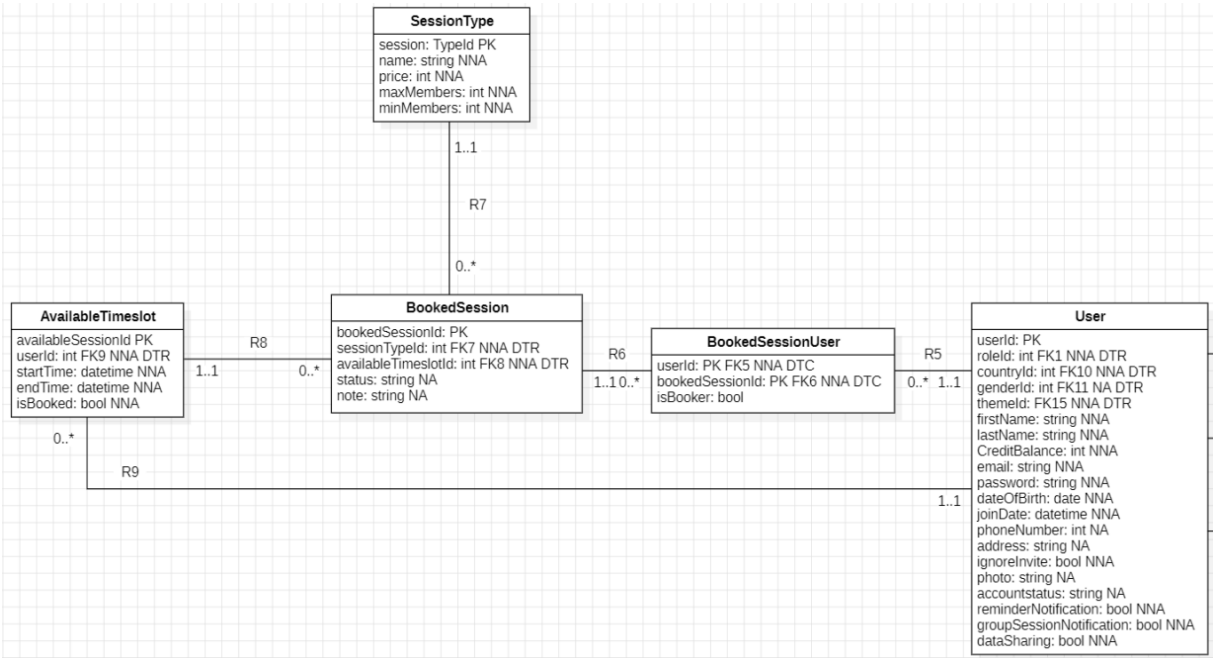
2.1.2.14. Update Workout History

Functionality: As a **Member**, I can update my workout history details.
Preconditions: The actor is logged in and has an existing workout history record.
Normal flow: The system first displays the current workout history information, including values such as weight, height, injuries, goals, weekly workout frequency, years of training and any notes. The actor reviews these details and enters updated information where needed. Once the changes are submitted, the system validates all inputs to ensure that numerical values and text fields meet the required constraints. When validation is successful, the system saves the updated data and confirms that the workout history has been successfully modified.
Alternatives: If the actor provides invalid or incomplete information, the system identifies the incorrect fields and shows an error message explaining what must be corrected. After adjusting the information, the actor may submit the form again.
Data model view:



2.1.2.15. Send Booking Reminder

Functionality: As **Time**, I can send booking reminders.
Preconditions: There is at least one upcoming booked session with reminders enabled for the member.
Normal flow: The system checks for upcoming sessions and automatically sends a reminder to the member by email or notification (as per the member preference) 24hours in advance.
Data model view:



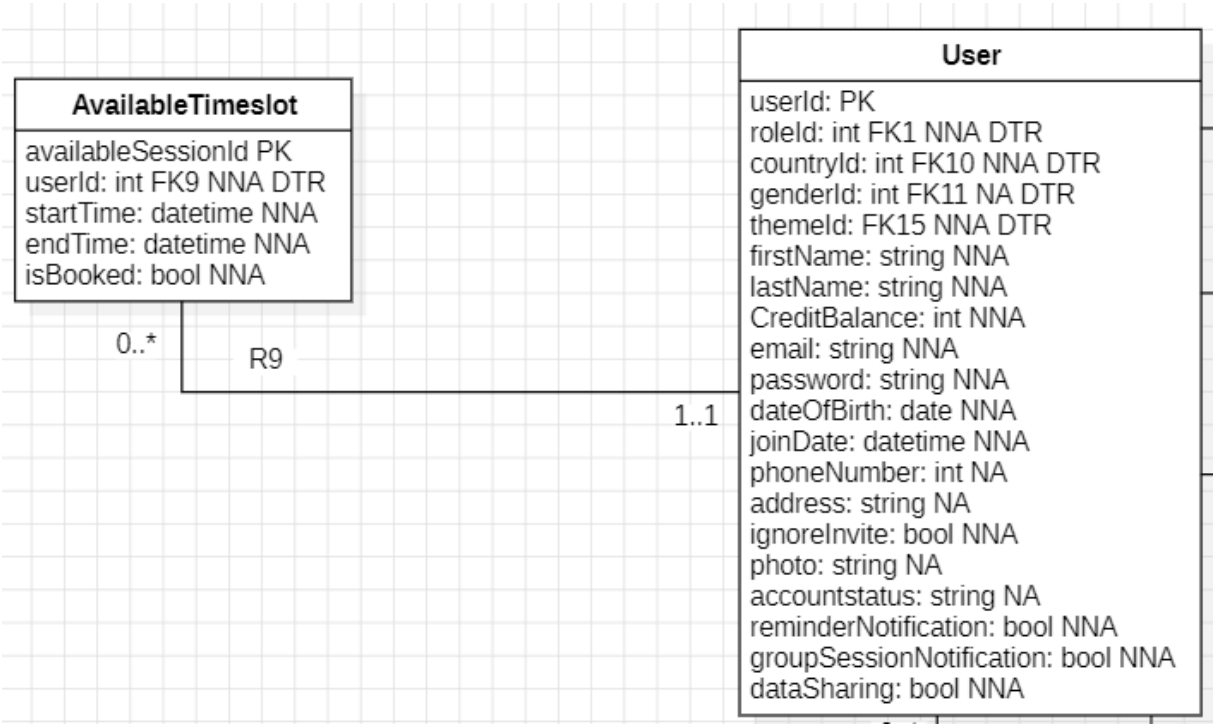
2.1.2.16. Manage Booking Schedule

Functionality: As a **Trainer**, I can manage a training schedule.
Preconditions: The actor is logged in.
Normal Flow:
The system displays the actor's current availability calendar. The actor selects dates and time slots to indicate availability, which can be configured for up to two months in advance. The system saves and updates the availability slots and makes them visible to members for booking sessions. In special cases, the Actor may also directly schedule sessions for specific members.
Alternative Flows:

Update: The Actor selects an existing availability slot, modifies the date, time, or duration, and confirms the changes. The system saves updates and refreshes the schedule to reflect the modifications.

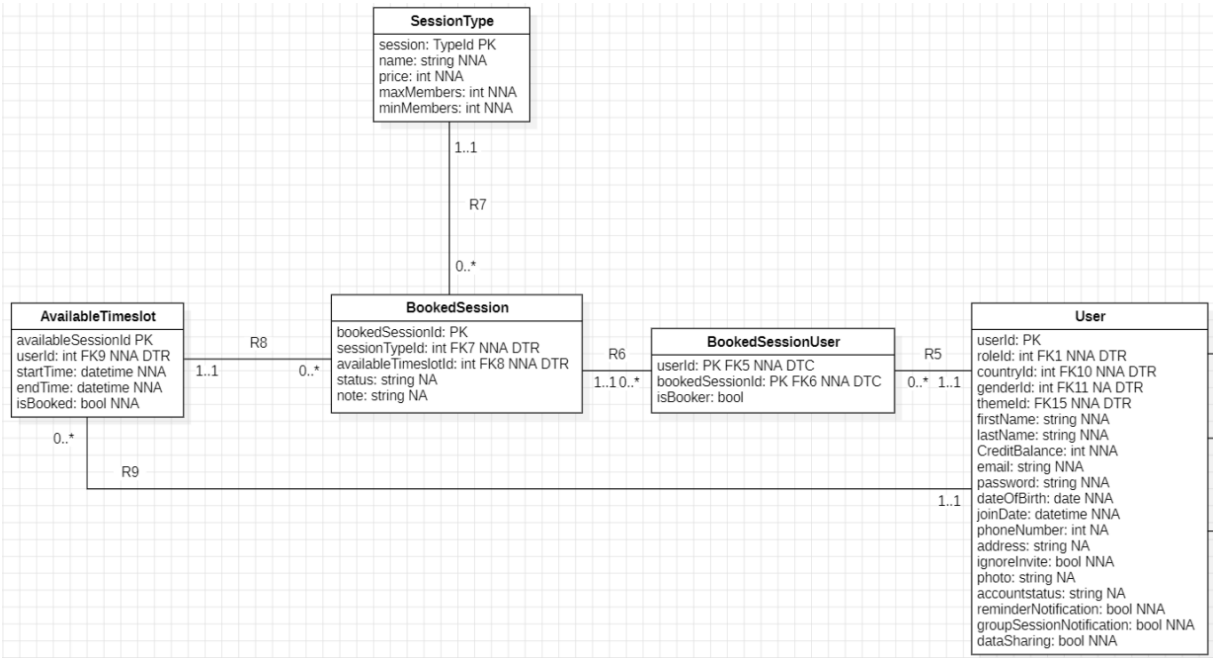
Delete: The Actor selects an availability slot to remove. The system prompts confirmation. Upon approval, the system deletes the selected slot and updates the schedule accordingly.

Data model view:



2.1.2.17. View Booked Sessions

Functionality: As a **Trainer**, I can view booked sessions.
Preconditions: The actor is logged in.
Normal flow: The system displays all booked sessions in chronological order. The interface provides two views: one showing all sessions booked with trainers and another displaying the actor's personal booked sessions. For each session, the actor can view details including the session date, time and duration, number of attendees, and session type. By selecting a participant's name, the actor can view the participant's details. Additionally, the actor can add sessions to their personal calendar by selecting the ***Add to My Calendar*** option.
Data model view:



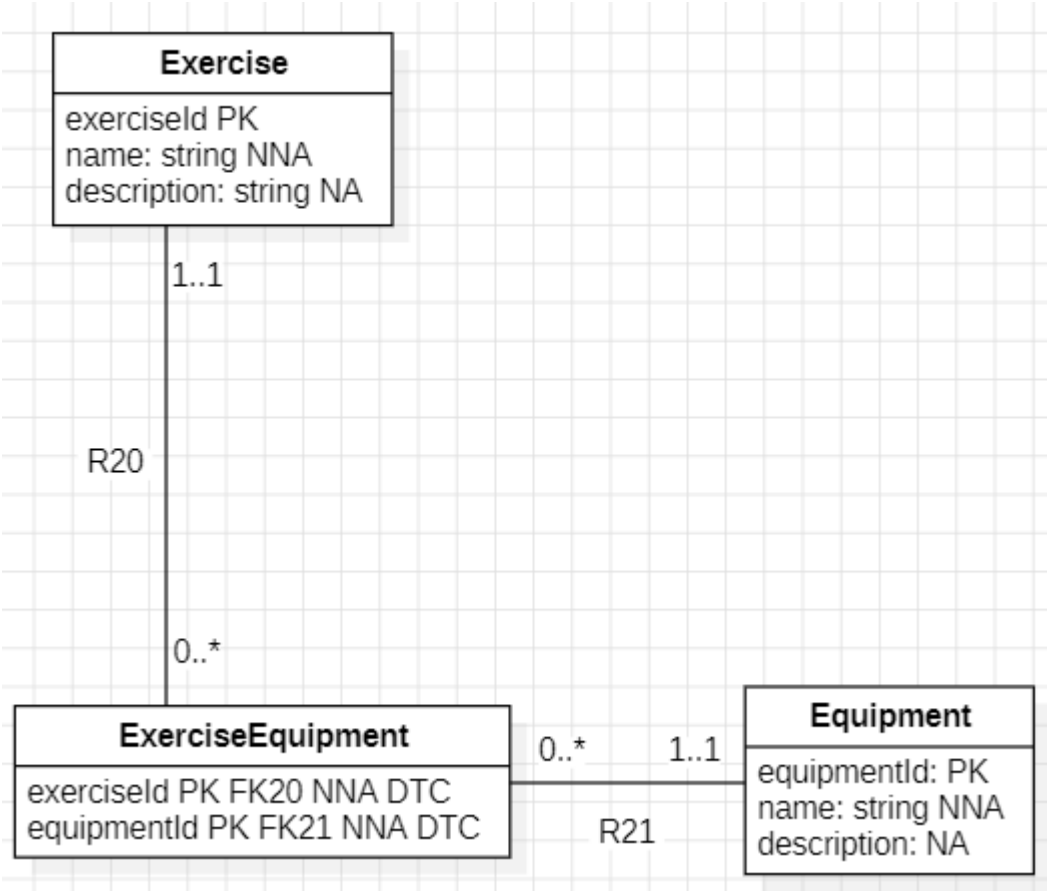
2.1.2.18. Manage Exercise Type

Functionality: As a **Trainer**, I can manage exercises.
Preconditions: The actor is logged in.
Normal flow: The system displays the current list of exercises. The actor can add new exercises by providing relevant details such as name and description. The system saves the information and refreshes the list.
Alternatives:

Update: The actor selects an existing exercise, modifies the necessary details, and confirms the changes. The system saves the update and displays the refreshed list.

Delete: The actor selects an exercise to remove. The system asks for confirmation, deletes the exercise upon approval, and updates the list accordingly.

Data model view:



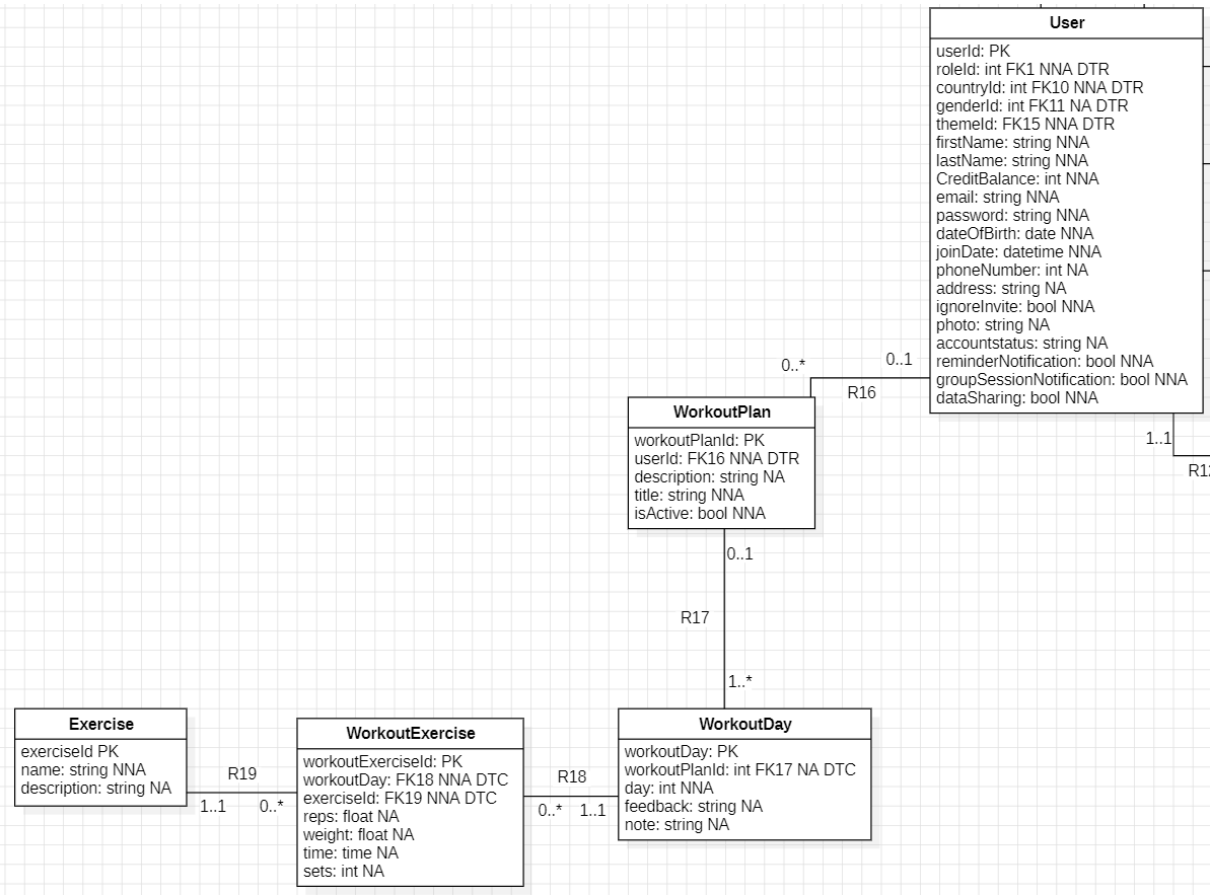
2.1.2.19. Manage Workout Plan

Functionality: As a **Trainer**, I can manage workout plans for clients.
Preconditions: The actor is logged in.
Normal flow: The system displays the current list of workout plans. The actor creates a new workout plan by entering relevant details such as plan name, exercises, schedule, and targeted goals. The system saves the information and displays the updated list of workout plans.
Alternatives:

Update: The actor selects an existing workout plan, modifies the necessary details, and confirms the changes. The system saves updates and refreshes the list.

Delete: The actor selects a workout plan to remove. The system asks for confirmation, deletes the plan upon approval, and updates the list accordingly.

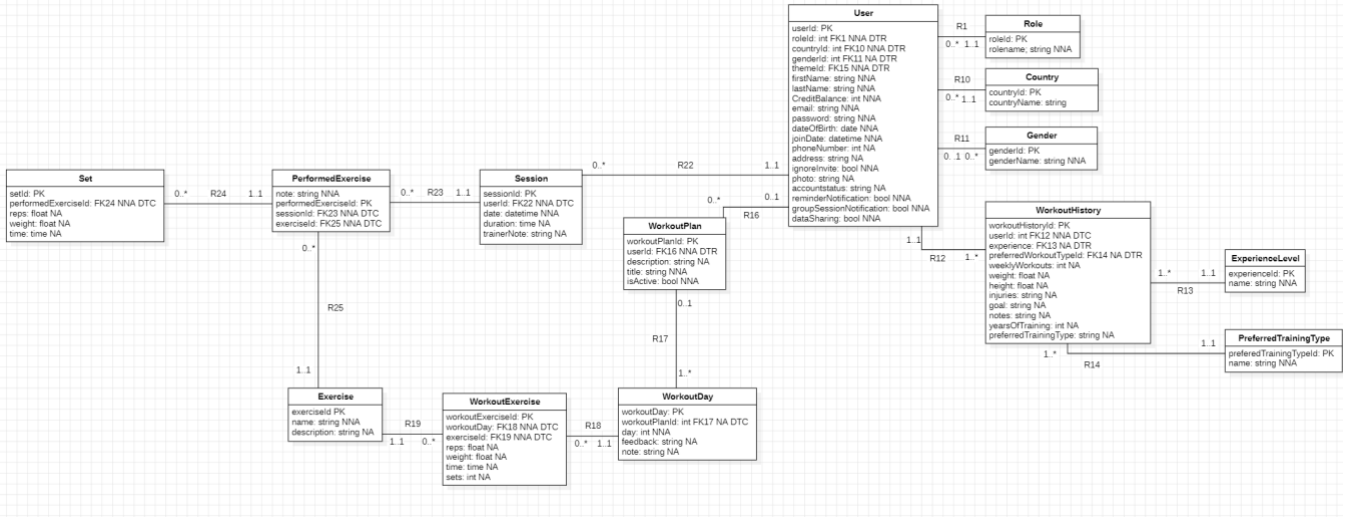
Data model view:



2.1.2.20. View Member Details

Functionality: As a **Trainer**, I can view member details
Preconditions: The actor is logged in.
Normal flow: The system displays a search bar to search for a member by name. The actor enters the name of the member he wants to search for. The system displays the member info, workout history, assigned workout plans, and progress details to the actor. Each field contains information from the actors' settings.

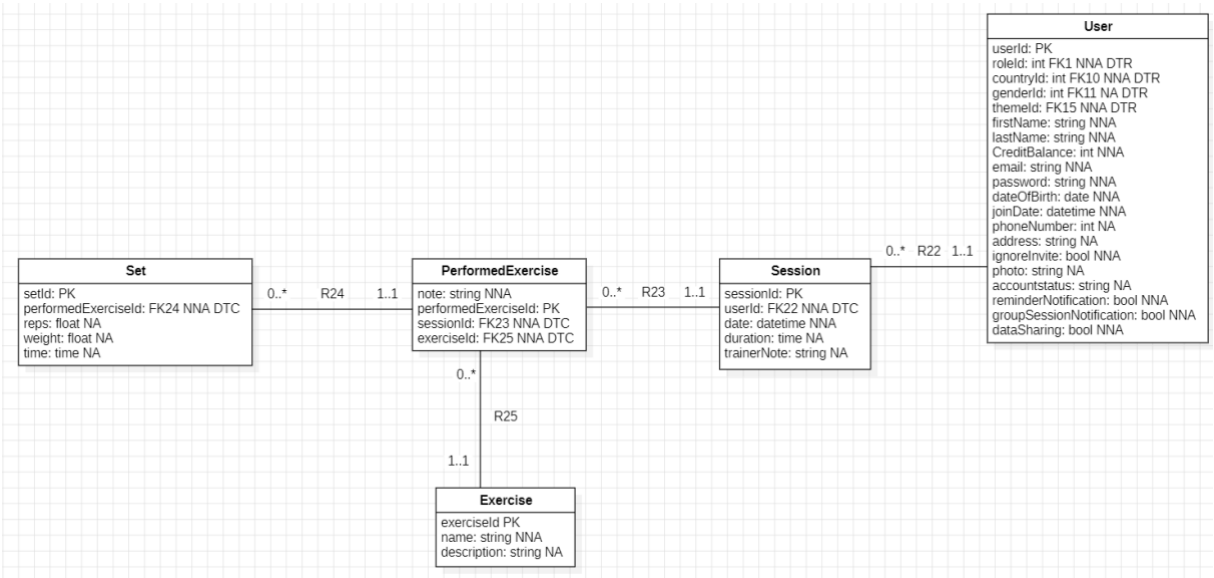
Data model view:



2.1.2.21. Track Fitness Progress Details

Functionality: As a **Trainer**/**Member**, I can track fitness progress details.
Preconditions: The actor is logged in.
Normal flow: The system displays the fitness progress history containing previous session details, including date, duration, and exercises performed (type, sets, and reps), the actor can select an option to add a new session, enters the required session information, and the system saves the progress, allowing trainers to add trainer notes and members to provide feedback.

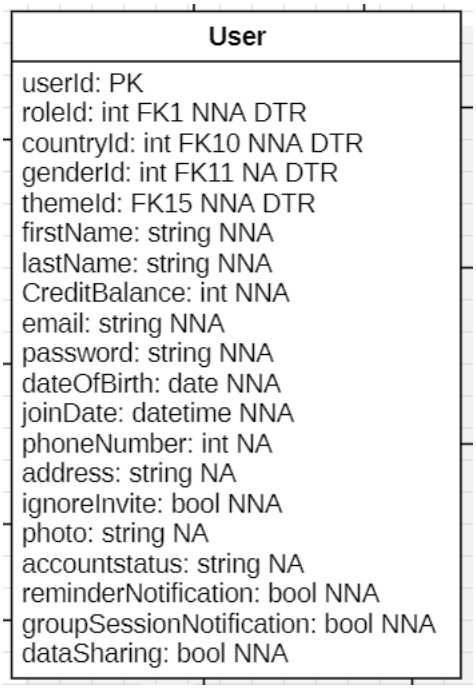
Data model view:



2.1.2.22. Manage Personal Settings

Functionality: As a Trainer, I can Manage/Update Profile Details.
Preconditions: The actor is Logged In.
Normal flow: The system displays all available customizable settings. The actor can modify the application theme, configure notification preferences, manage privacy settings, and update or change the profile password.

Data model view:



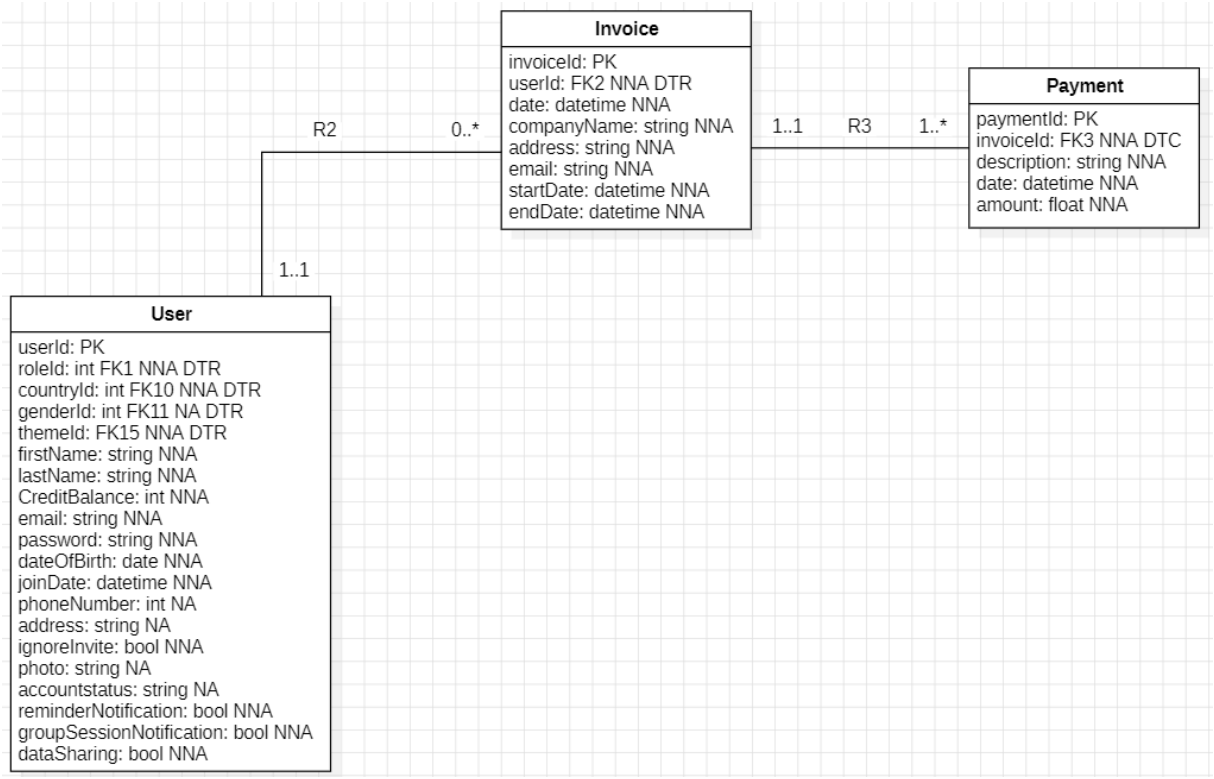
2.1.1.23. Generate Invoices

Functionality: As an **Assistant**, I can generate invoices.
Preconditions: The actor is logged in.
Normal flow: The system displays a member selection and date range input, the actor selects a member and enters a from-date and to-date, the system generates the invoice for the selected period, and the actor chooses to download the invoice as a PDF or send it to the member via email.
Alternatives:

No billable sessions: The system informs the assistant that no billable items exist for the selected date range

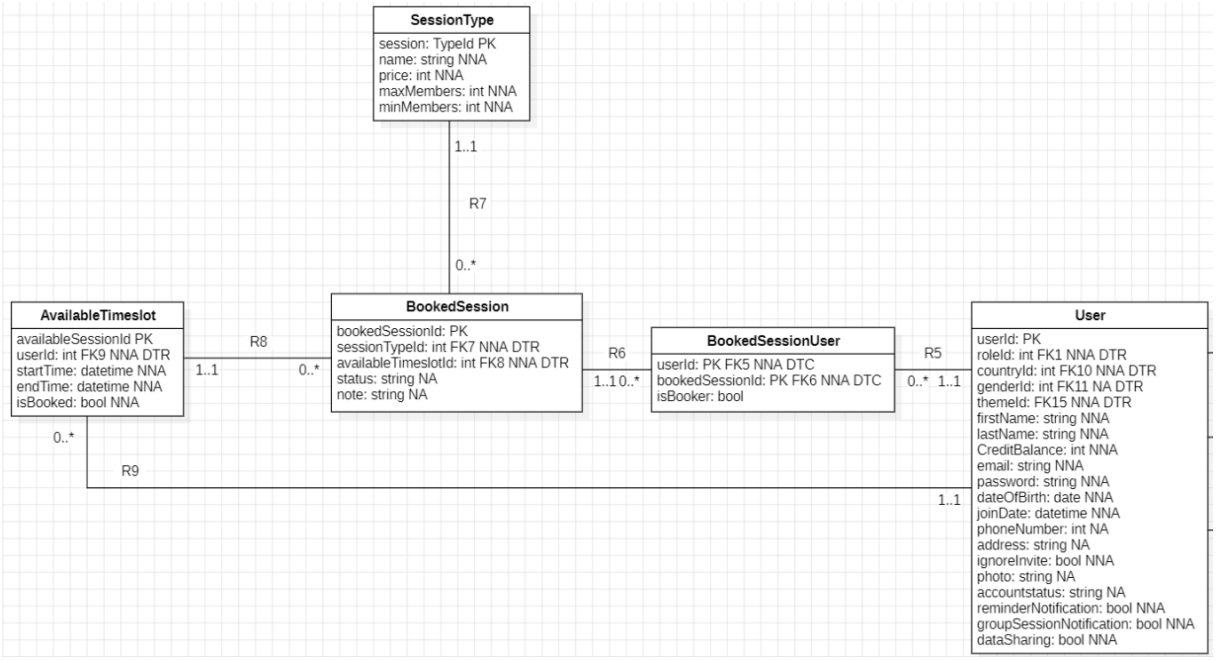
Invalid date range: The system displays an error and requests a valid date range

Data model view:



2.1.2.24. View Session Attendance

Functionality: As an **Assistant**, I can view session attendance.
Preconditions: The actor is logged in.
Normal flow: The system displays options to select a month and year and one or more session types, the actor makes the selection, and the system displays the total attendance counts and compares attendance percentages using a pie chart.
Data model view:



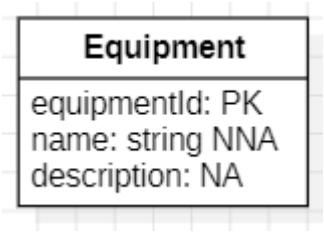
2.1.2.25. Manage Workout Equipment

Functionality: As an **Assistant,** I can manage workout equipment.
Preconditions: The actor is logged in.
Normal flow: The system displays the current list of workout equipment. The actor selects the option to add new equipment and enters details such as name and description. The system validates the information, saves the new equipment, and displays the updated list.
Alternatives:

Update: The actor selects an existing equipment item, edits the necessary fields, and confirms the changes. The system saves the update and refreshes the list.

Delete: The actor selects an equipment item to remove. The system asks for confirmation, deletes the record upon approval, and updates the list accordingly.

Data model view:



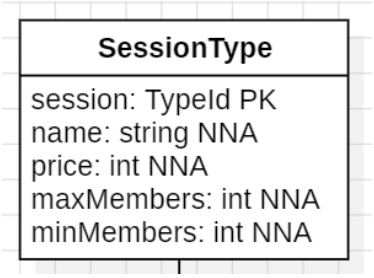
2.1.2.26. Manage Session Types

Functionality: As an **Assistant,** I can manage session types.
Preconditions: The actor is logged in.
Normal flow: The system displays a list of all active session types and provides an option to create a new session type, the actor selects this option and enters the session name, price per member, minimum members, and maximum members, and the system saves the session type and adds it to the list.
Alternatives:

Update: The actor selects an existing session type and modifies the session name, price per member, minimum members, or maximum members, and the system saves the changes.

Delete: The actor selects an option to delete a session type if it is not used in any active sessions or workout plans, and the system removes it from the list.

Data model view:



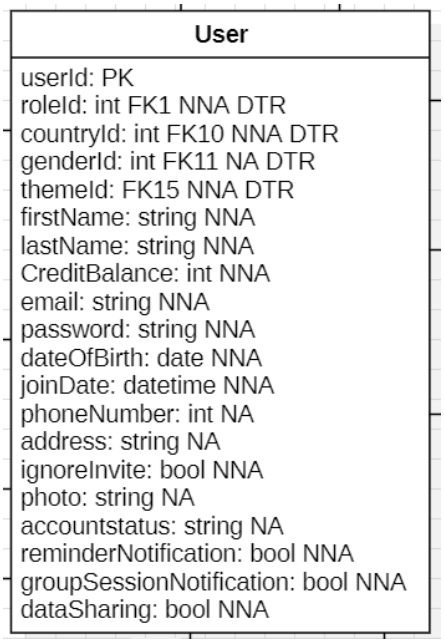
2.1.2.27. Manage Users

Functionality: As an **Administrator**, I can manage users.
Preconditions: The actor is logged in as the system administrator.
Normal flow: The system displays a list of all gym system users containing their assigned role, email, account status (active, inactive or pending), actions to edit or delete the user's account, and an action to create a new user account. The system displays a search bar to search for a user by name and an option to filter users by role. The actor chooses to create a new user. The system prompts the actor to provide a password, role, email address, first name, and last name for the user's account. The actor confirms, and the systems saves the users account as pending status in the list. The system sends a confirmation email to the user's email address. The invited user confirms their account creation in the email. The system changes their account status to active.
Alternatives:

Update: The actor chooses to edit the user's account details. The system displays an option to change the user's first name, last name, email address, password, role, or account status. The actor chooses to change the user's email address and role. The system changes the user's account status to pending. The system sends a confirmation email to the user's new email address to activate their account. The user activates their account, and the system changes the account status to active.

Delete: The actor selects an option to delete a user. The system displays the user's name, their email address and role, and a prompt asking the actor if they are sure about deleting that user. The actor confirms. The system removes that users account from the list.

Data model view:


User
userId: PK
roleId: int FK1 NNA DTR
countryId: int FK10 NNA DTR
genderId: int FK11 NA DTR
themeId: FK15 NNA DTR
firstName: string NNA
lastName: string NNA
CreditBalance: int NNA
email: string NNA
password: string NNA
dateOfBirth: date NNA
joinDate: datetime NNA
phoneNumber: int NA
address: string NA
ignoreInvite: bool NNA
photo: string NA
accountstatus: string NA
reminderNotification: bool NNA
groupSessionNotification: bool NNA
dataSharing: bool NNA

2.2. Non-functional requirements

2.2.1. User Interface

The user interface should integrate the signature Grow Gym accent color into its primary color scheme to establish a consistent visual identity and ensure that all interactive elements are instantly recognizable to the user. The interface must be intuitive and require minimal training, with clear navigation paths and consistent design patterns across all modules.

2.2.2. Performance

Response Time: The system must respond to user actions within 2 seconds under normal load conditions for standard operations (e.g., viewing schedules, booking sessions).
Page Load Time: All pages must load within 3 seconds on standard broadband connections.
Concurrent Users: The system must support at least 100 concurrent users without performance degradation.
Database Queries: Complex queries (e.g., generating reports, viewing progress history) must execute within 5 seconds.

2.2.3. Security

Authentication: The system must implement secure password-based authentication with password complexity requirements (minimum 8 characters, including uppercase, lowercase, numbers, and special characters).
Session Management: User sessions must timeout after 30 minutes of inactivity to prevent unauthorized access.
Data Encryption: All sensitive data (passwords, payment information) must be encrypted using industry-standard encryption algorithms (e.g., AES-256).
HTTPS: All communication between client and server must be encrypted using HTTPS/TLS protocols.
Role-Based Access Control: The system must enforce strict role-based permissions ensuring users can only access features appropriate to their role (Member, Trainer, Assistant, Administrator).
SQL Injection Prevention: The system must use parameterized queries to prevent SQL injection attacks.
Payment Security: Payment processing must comply with PCI DSS standards when handling credit card information.

2.2.4. Usability

Accessibility: The application must comply with WCAG 2.1 Level AA standards to ensure accessibility for users with disabilities.
Responsive Design: The interface must be fully responsive and functional across desktop, tablet, and mobile devices with screen sizes ranging from 320px to 2560px width.
Language Support: The system should support multiple languages with easy language switching functionality.
Error Messages: Error messages must be clear, user-friendly, and provide actionable guidance for resolution.
Help Documentation: Context-sensitive help and tooltips should be available throughout the application.

2.2.5. Reliability

Uptime: The system must maintain 99.5% uptime during operational hours (excluding scheduled maintenance).
Error Handling: The system must gracefully handle errors without crashing and provide meaningful error messages to users.
Data Integrity: All transactions (credit purchases, session bookings, credit deductions) must be atomic to prevent data inconsistencies.
Backup: Daily automated backups must be performed with the ability to restore data within 4 hours in case of system failure.

2.2.6. Scalability

User Growth: The system architecture must support horizontal scaling to accommodate up to 1000 registered members without major redesign.
Data Storage: The database must be designed to efficiently handle increasing volumes of workout plans, session bookings, and progress tracking data over time.
Session Management: The booking system must scale to handle multiple simultaneous bookings without conflicts or race conditions.

2.2.7. Availability

Scheduled Maintenance: System maintenance windows must be scheduled during off-peak hours (e.g., 2:00 AM - 4:00 AM) and communicated to users at least 48 hours in advance.

Redundancy: Critical system components (database, authentication service) should have redundant backups to minimize downtime.

2.2.8. Maintainability

Code Quality: The codebase must follow industry best practices and coding standards to ensure readability and maintainability.
Documentation: Comprehensive technical documentation must be maintained for system architecture, APIs, and database schema.
Logging: The system must maintain detailed logs of all critical operations, errors, and user activities for troubleshooting and audit purposes.
Modular Design: The system should be built with a modular architecture allowing independent updates to different components without affecting the entire system.

2.2.9. Compatibility

Browser Support: The application must be compatible with the latest two versions of major browsers (Chrome, Firefox, Safari, Edge).
Operating Systems: The web application must function correctly on Windows, macOS, Linux, iOS, and Android platforms.
API Compatibility: Any external APIs (payment gateways, email services) must be integrated following their documented specifications and versioning guidelines.

2.2.10. Data Privacy and Compliance

GDPR Compliance: The system must comply with GDPR regulations, including the right to access, modify, and delete personal data.
Data Retention: User data must be retained according to legal requirements and the gym's privacy policy.
Consent Management: The system must obtain and record explicit user consent for data collection, processing, and sharing.
Audit Trail: All access to sensitive user data must be logged for compliance and security auditing.

2.2.11. Notification System

Email Delivery: Email notifications (booking reminders, confirmations, invitations) must be delivered within 5 minutes of the triggering event.
Notification Preferences: Users must be able to customize their notification preferences as specified in the Manage Settings use case.

Delivery Reliability: The notification system must have a 95% successful delivery rate.

2.2.12. Payment Processing

Transaction Time: Credit purchase transactions must be processed within 60 seconds.
Payment Gateway Integration: The system must integrate with secure, reliable payment gateways that support multiple payment methods.
Transaction Records: All financial transactions must be logged with complete audit trails including timestamps, amounts, and user details.

2.2.13. Localization

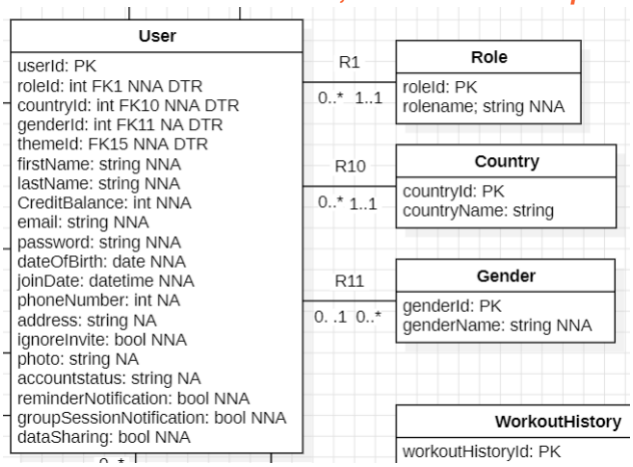
Time Zones: The system must correctly handle time zones for session scheduling and reminders.
Date Formats: Date and time displays must adapt to regional preferences.
Currency: Credit pricing and invoices must display amounts in the appropriate currency format (€).

2.2.14. Testing Requirements

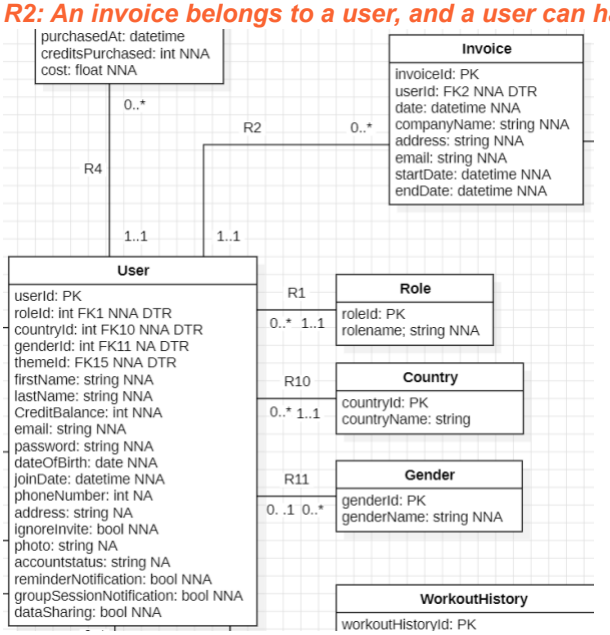
Unit Testing: All critical functions must have unit test coverage of at least 80%.
Integration Testing: Key user workflows must be covered by automated integration tests.
User Acceptance Testing: The system must undergo UAT by representatives from each user role before deployment

3. Data Model

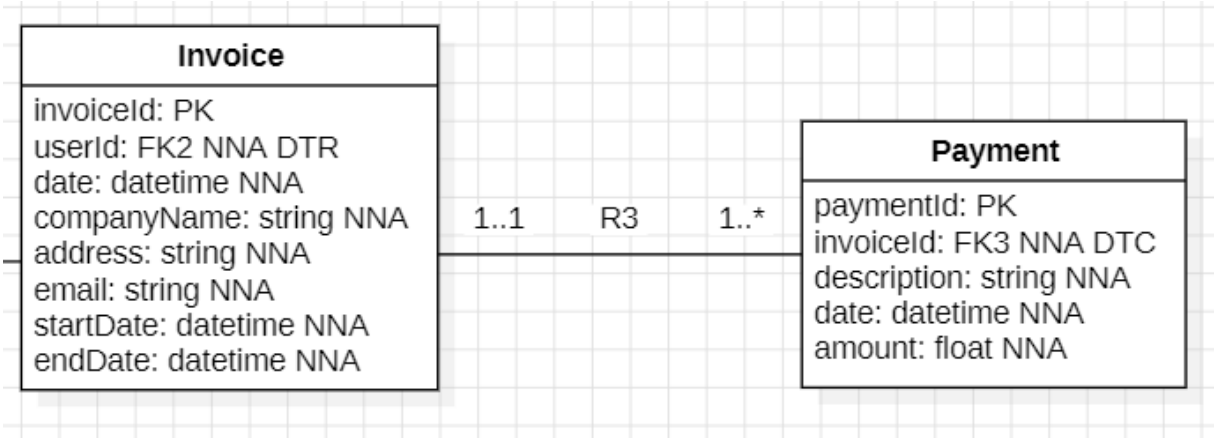
R1: A User must have a role; there can be multiple users with the same role.



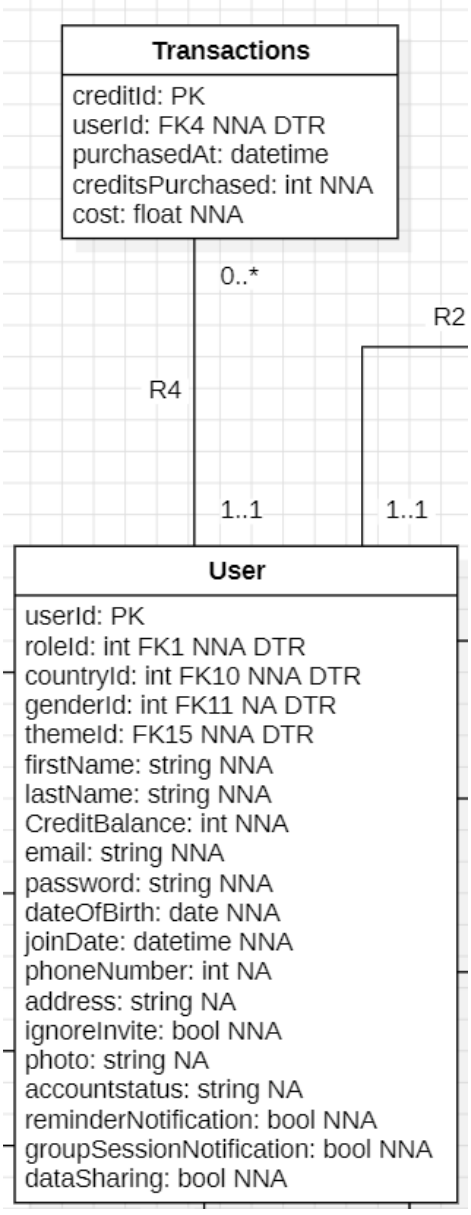
R2: An invoice belongs to a user, and a user can have multiple invoices.



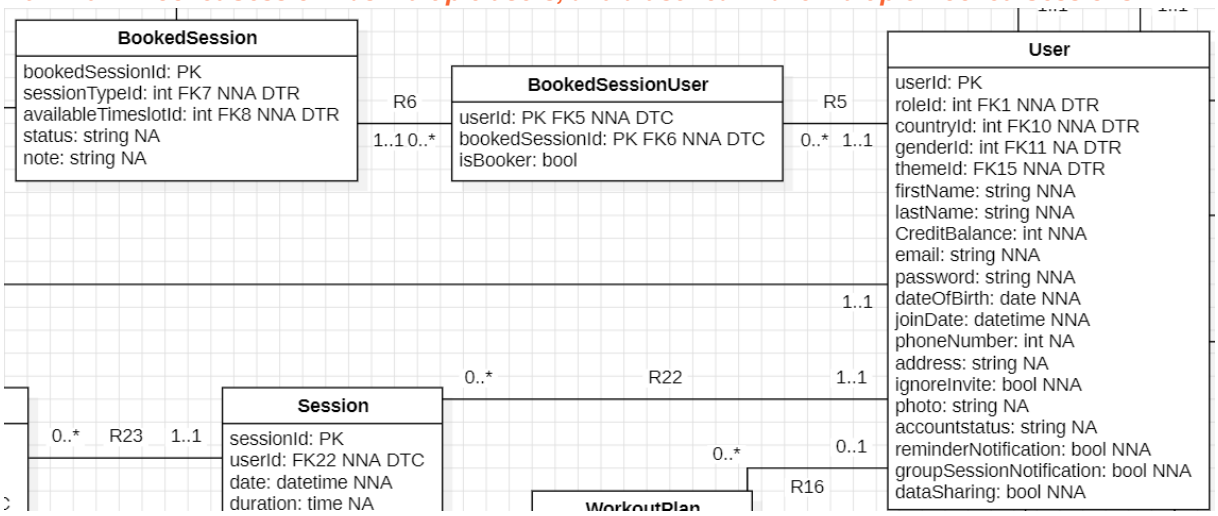
R3: A payment belongs to an invoice, and an invoice can have multiple payments.



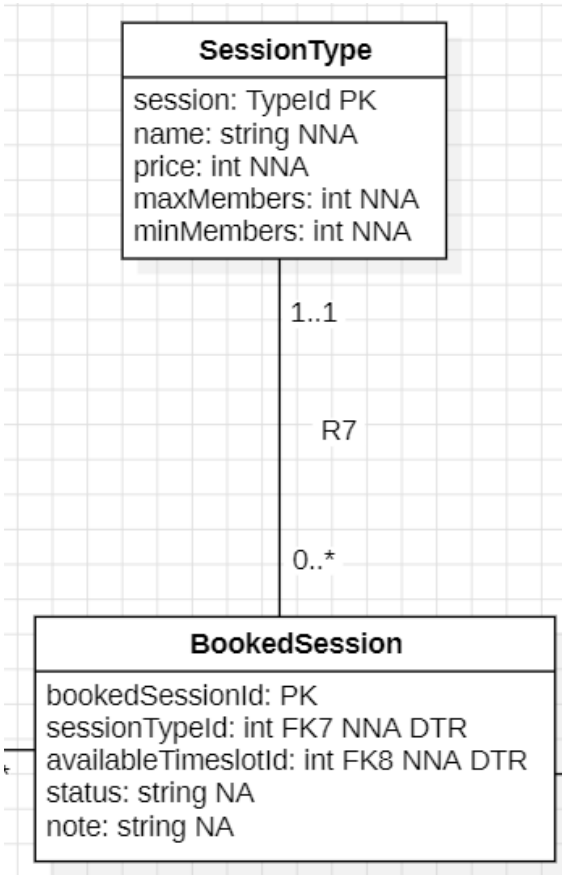
R4: A transaction belongs to a user, and a user can have multiple transactions.



R5 + R6: A BookedSession has multiple users, and a user can have multiple Booked Sessions.



R7: A Booked Session has a Session Type and each Session type can belong to multiple BookedSession.

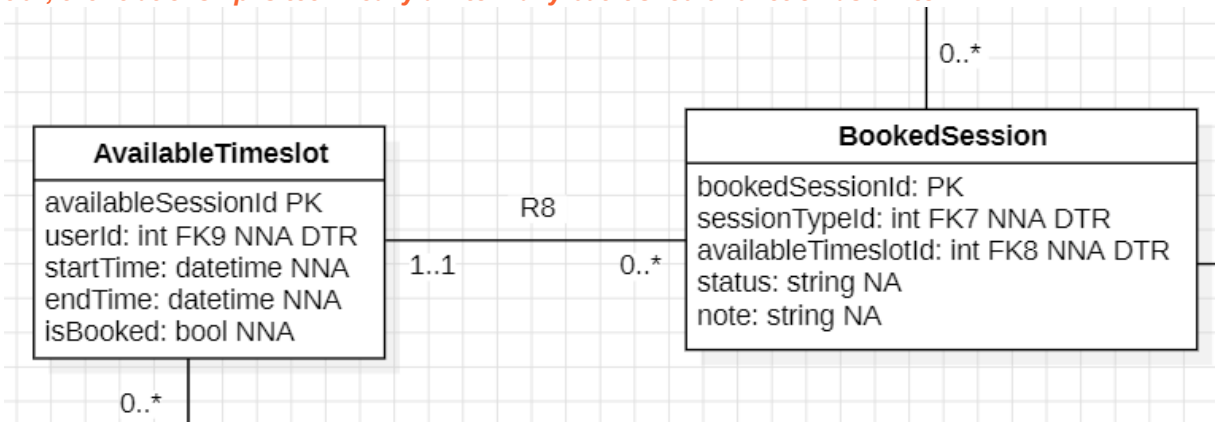


R8: A booked session has a corresponding timeslot.

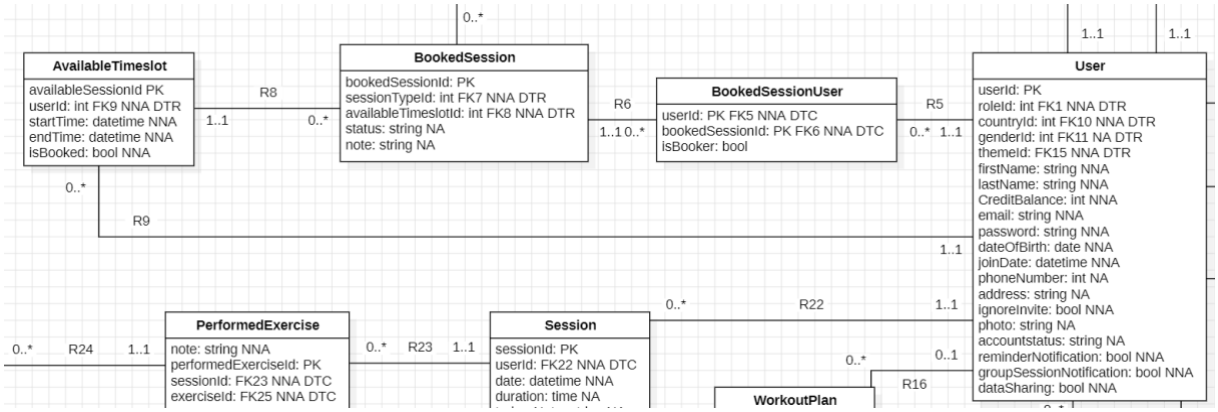
NOTE: Technically R7 would have to be a 1 to 1 relationship because when a session is booked you want to know which time slot it was booked in to be able to mark it as booked but each session can also only take up a single timeslot.

But this is technically impossible to make because the side with the PK is automatically the many sides but the fact that it is technically possible does not matter because it is not possible to book a session if it has already been booked (isBooked set to true)

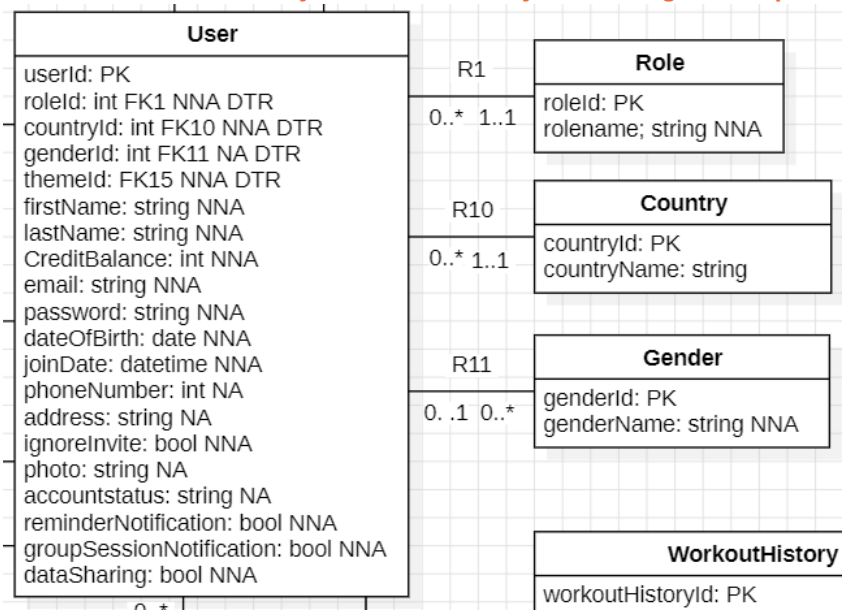
tldr; the relationship is technically a 1 to many but it should function as a 1 to 1.



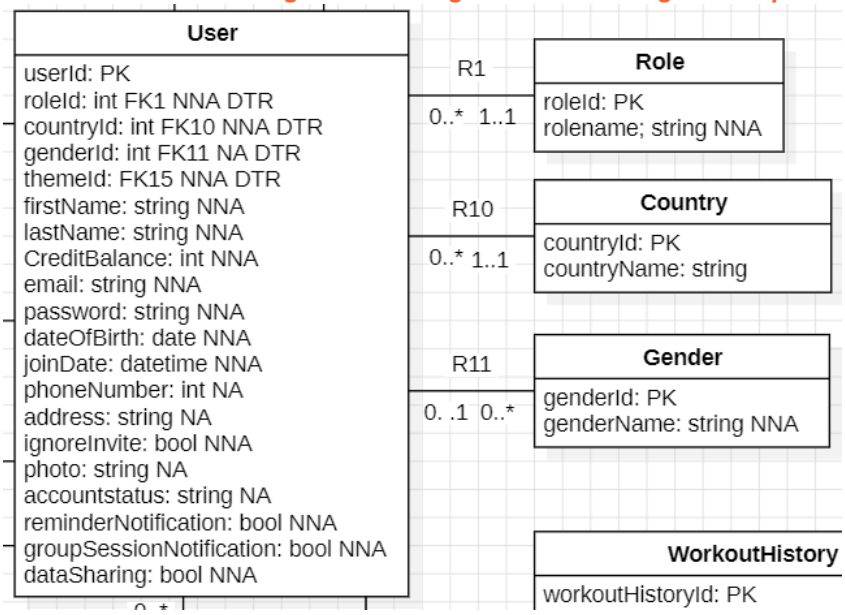
R9: An Available Session has a user which is the trainer and each user can have multiple available sessions.



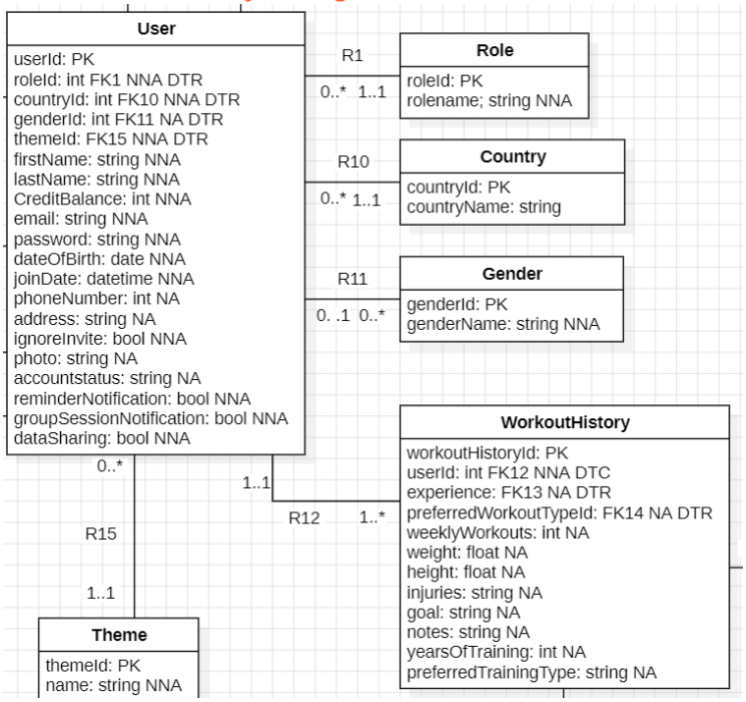
R10: A user has a country, and each country can belong to multiple users.



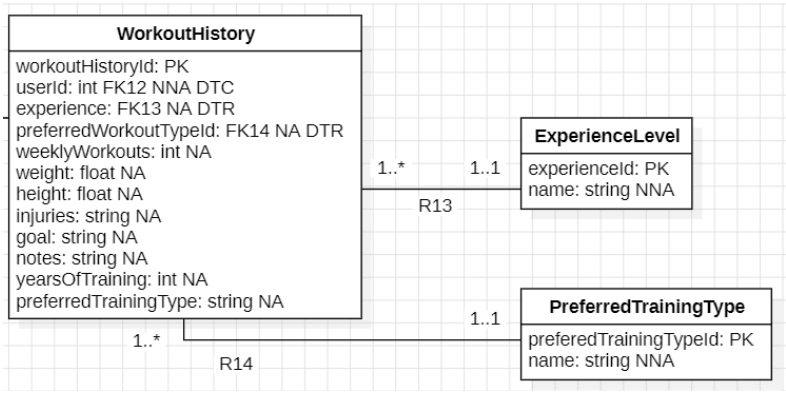
R11: A user can have a gender and a gender can belong to multiple users



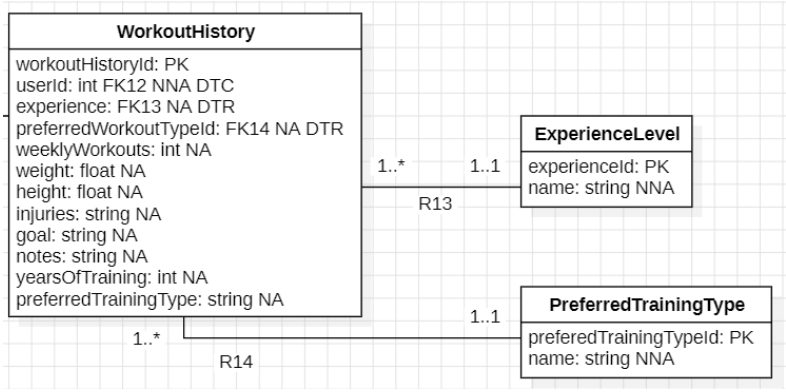
R12: Workout history belongs to a user, and each user can have multiple workout histories.



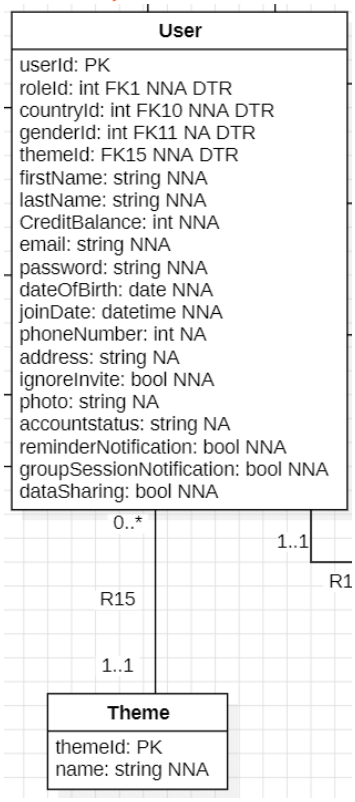
R13: Workout history has an experience level, and each experience level can belong to multiple workout histories.



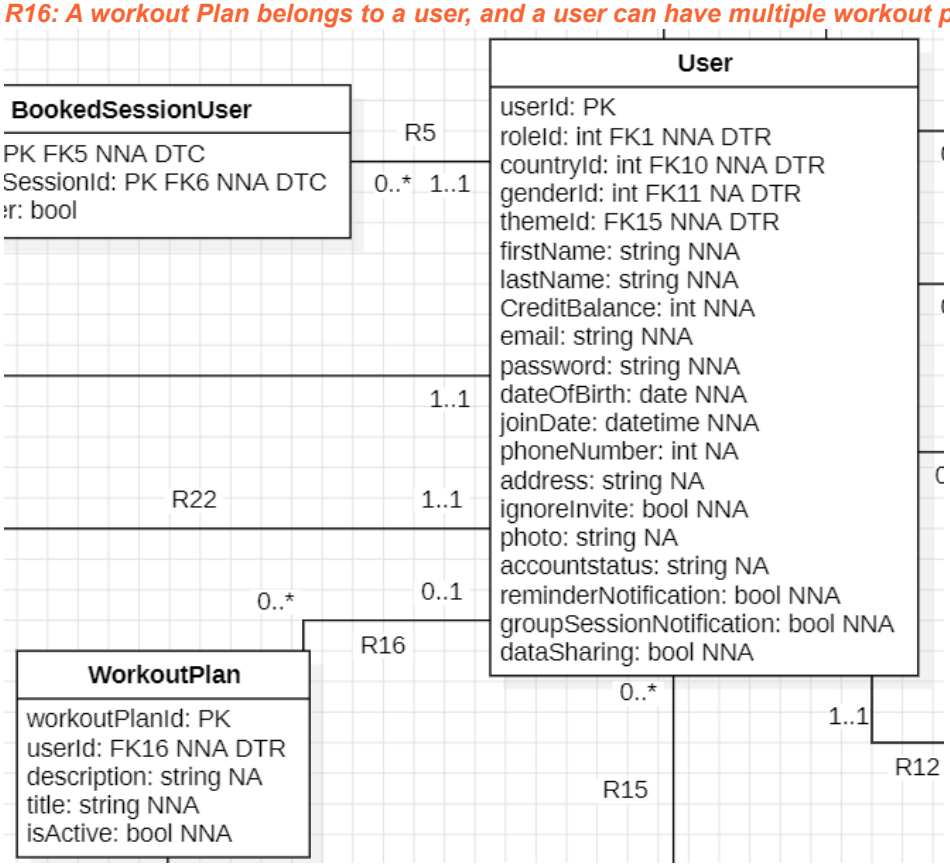
R14: Workout history has a preferred training type, and each preferred training type can belong to multiple workout histories.



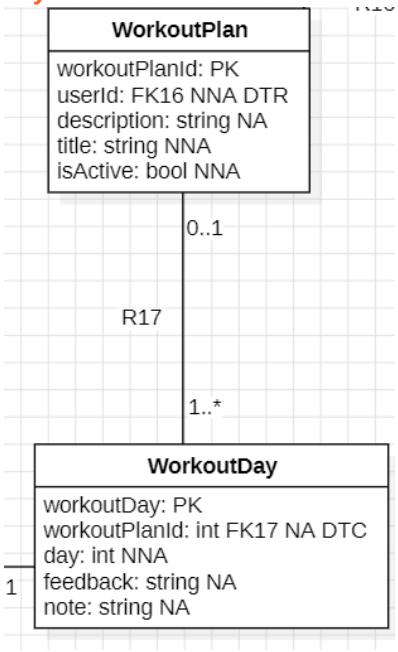
R15: A user has a theme, and a theme can belong to multiple users (this is like light or dark mode or whatever).



R16: A workout Plan belongs to a user, and a user can have multiple workout plans.

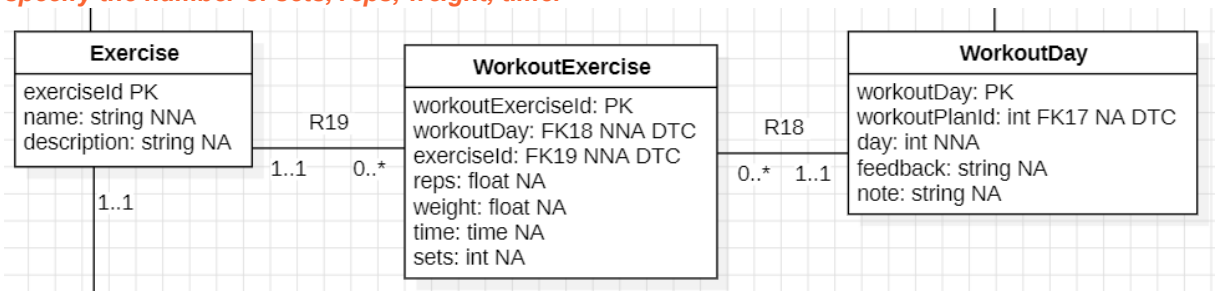


R17: A workout day belongs to a Workout plan, and each workout plan has one or more workout days.

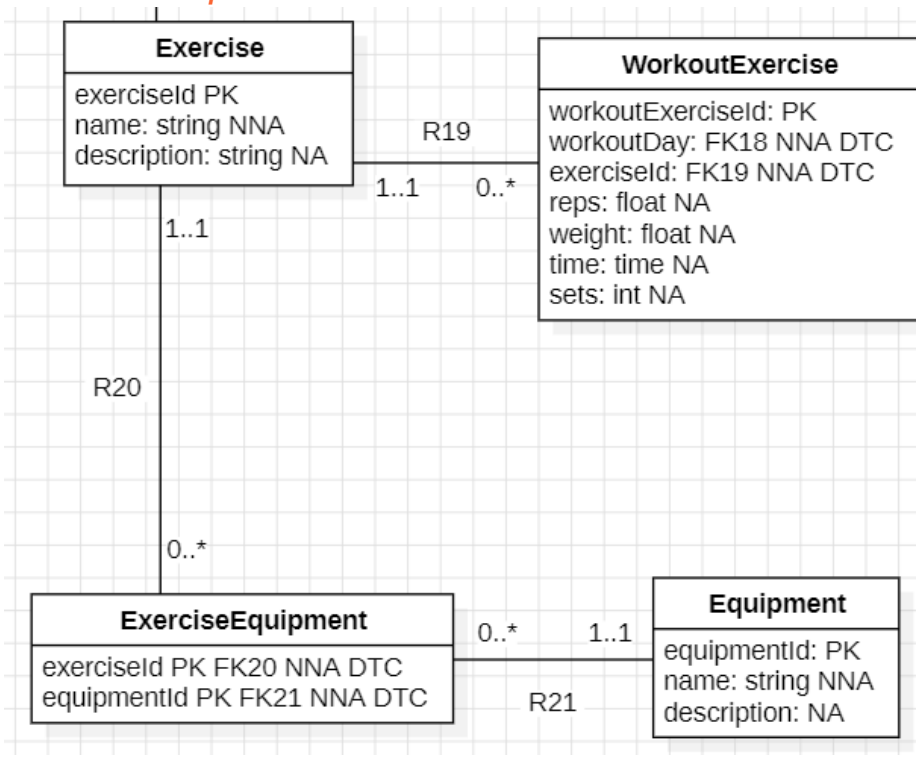


R18 + R 19: Each workout day can have multiple exercises, and each exercise can belong to multiple workout days.

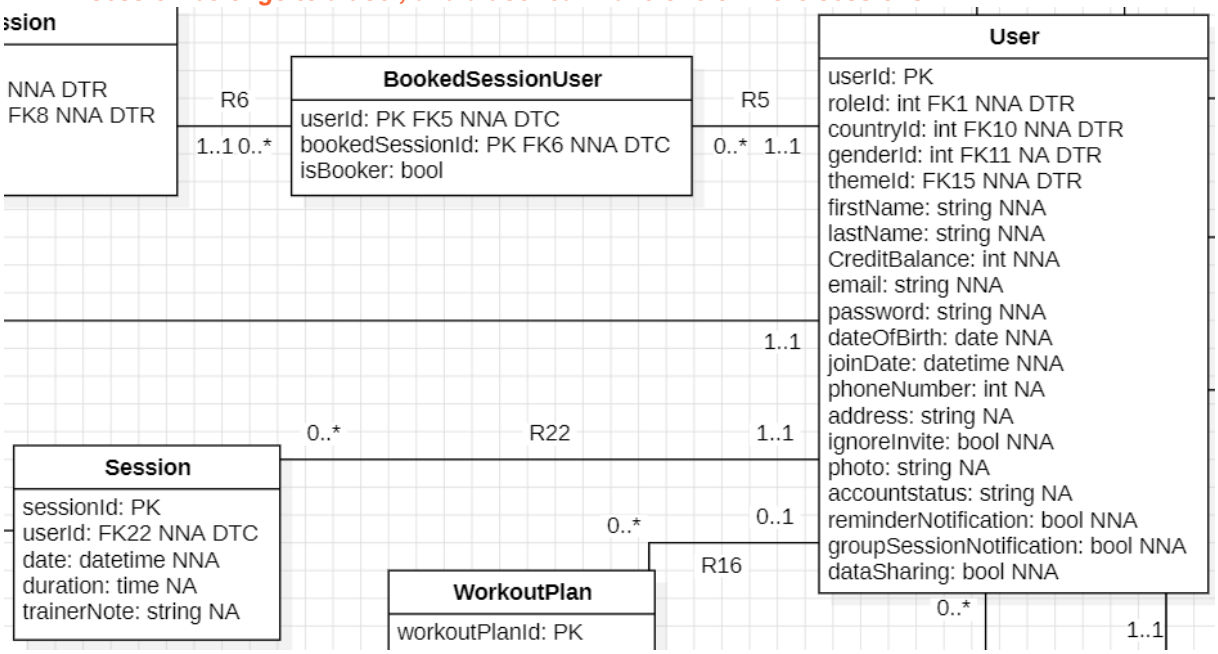
Additionally for each exercise in a workout day there is some important information attached to specify the number of sets, reps, weight, time.



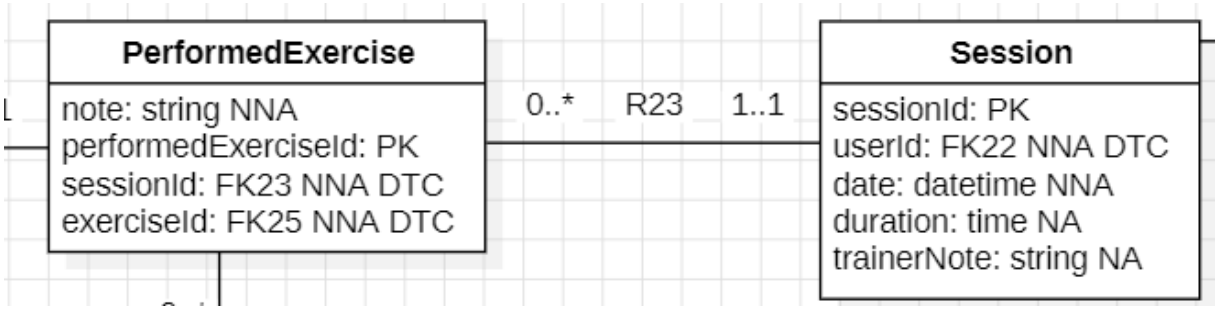
R20 + R21: An exercise can require multiple pieces of equipment while each piece of equipment can be used for multiple exercises.



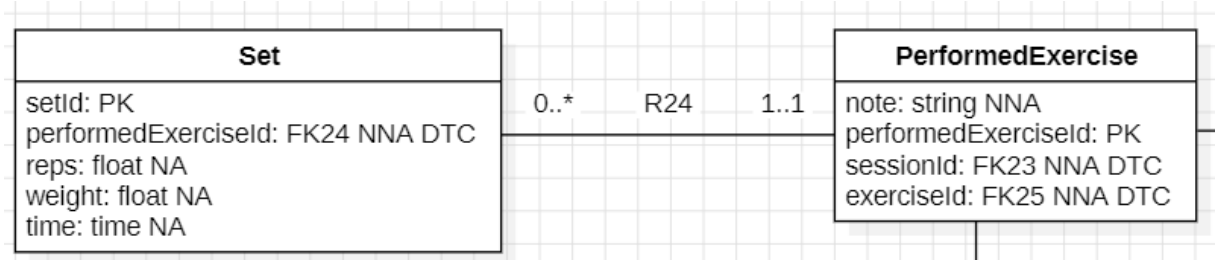
R22: A session belongs to a user, and a user can have one or more sessions.



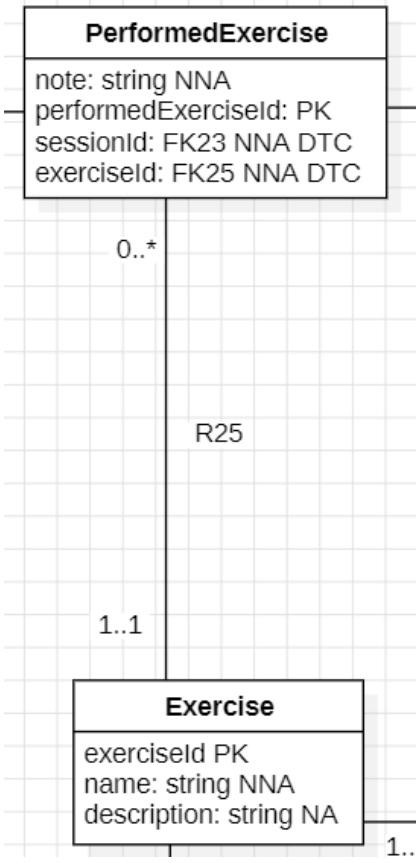
R23: A performed exercise belongs to a session, and each session can have multiple performed exercises.



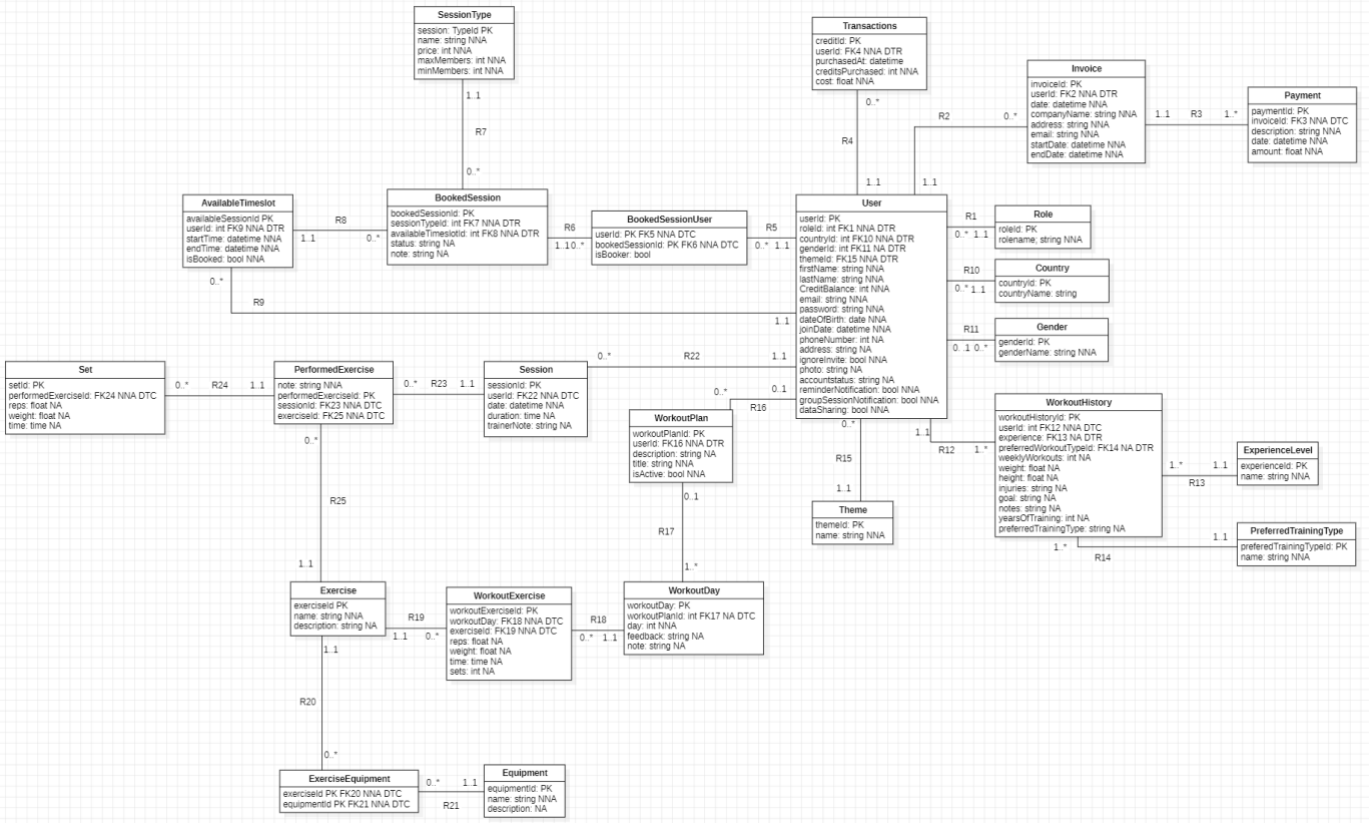
R24: Each set belongs to a performed exercise, and each performed exercise can have multiple sets.



R25: Each performed exercise has an exercise and each exercise can belong to multiple performed exercises.



Whole ERD:


SessionType
session: TypeId PK
name: string NNA
price: int NNA
maxMembers: int NNA
minMembers: int NNA
Transactions
creditId: PK
userId: FK4 NNA DTR
purchasedAt: datetime
creditsPurchased: int NNA
cost: float NNA
Invoice
invoiceId: PK
userId: FK2 NNA DTR
date: datetime NNA
companyName: string NNA
address: string NNA
email: string NNA
startDate: datetime NNA
endDate: datetime NNA
Payment
paymentId: PK
invoiceId: FK3 NNA DTC
description: string NNA
date: datetime NNA
amount: float NNA
AvailableTimeslot
availableSessionId: PK
userId: int FK9 NNA DTR
startTime: datetime NNA
endTime: datetime NNA
isBooked: bool NNA
BookedSession
bookedSessionId: PK
sessionTypeId: int FK7 NNA DTR
availableTimeslotId: int FK8 NNA DTR
status: string NA
note: string NA
BookedSessionUser
userId: PK FK5 NNA DTC
bookedSessionId: PK FK6 NNA DTC
isBooker: bool
User
userId: PK
roleId: int FK1 NNA DTR
countryId: int FK10 NNA DTR
genderId: int FK11 NA DTR
themeId: FK15 NNA DTR
firstName: string NNA
lastName: string NNA
CreditBalance: int NNA
email: string NNA
password: string NNA
dateOfBirth: date NNA
joinDate: datetime NNA
phoneNumber: int NA
address: string NA
ignoreInvite: bool NNA
photo: string NA
accountstatus: string NA
reminderNotification: bool NNA
groupSessionNotification: bool NNA
dataSharing: bool NNA
Role
roleId: PK
rolename: string NNA
Country
countryId: PK
countryName: string
Gender
genderId: PK
genderName: string NNA
Set
setId: PK
performedExerciseId: FK24 NNA DTC
reps: float NA
weight: float NA
time: time NA
PerformedExercise
note: string NNA
performedExerciseId: PK
sessionId: FK23 NNA DTC
exerciseId: FK25 NNA DTC
Session
sessionId: PK
userId: FK22 NNA DTC
date: datetime NNA
duration: time NA
trainerNote: string NA
WorkoutPlan
workoutPlanId: PK
userId: FK16 NNA DTR
description: string NA
title: string NNA
isActive: bool NNA
WorkoutHistory
workoutHistoryId: PK
userId: int FK12 NNA DTC
experience: FK13 NA DTR
preferredWorkoutTypeId: FK14 NA DTR
weeklyWorkouts: int NA
weight: float NA
height: float NA
injuries: string NA
goal: string NA
notes: string NA
yearsOfTraining: int NA
preferredTrainingType: string NA
ExperienceLevel
experienceId: PK
name: string NNA
PreferredTrainingType
preferedTrainingTypeId: PK
name: string NNA
Theme
themeId: PK
name: string NNA
Exercise
exerciseId PK
name: string NNA
description: string NA
WorkoutExercise
workoutExerciseId: PK
workoutDay: FK18 NNA DTC
exerciseId: FK19 NNA DTC
reps: float NA
weight: float NA
time: time NA
sets: int NA
WorkoutDay
workoutDay: PK
workoutPlanId: int FK17 NA DTC
day: int NNA
feedback: string NA
note: string NA
ExerciseEquipment
exerciseId PK FK20 NNA DTC
equipmentId PK FK21 NNA DTC
Equipment
equipmentId: PK
name: string NNA
description: NA

4. Priority by functionality

Must have	Should have	Could have	Won't have
Member			
Login			
Register Profile			
Purchase Credits			
View Remaining Credits			
	View Workout Plan		
View Session Schedule			
Book Session			
Confirm Individual Session			
Confirm Booked Session			
	Update Workout History		
			View Nutrition Schedule
Time			
	Send Booking Reminder		
Trainer			
Manage Booking Schedule			
View Booked Sessions			
Manage Exercise Type			
	Manage Workout Plan		
	Track Fitness Progress Details		
			Manage Nutrition Schedule
Assistant			
Manage Invoices			
		View Session Attendance	
	Manage Workout Equipment		
Manage Session types			
Admin			
Manage Users			