```
classDiagram
direction LR

%% ================================
%% ENUMS (Defined for schema reference)
%% ================================
class Enums {
  <<enumeration>>
  gender_enum
  playing_hand_enum
  organization_nature_enum
  organization_type_enum
  admin_role_enum
  role_enum
  match_duration_enum
  match_type_enum
  match_gender_enum
  match_status_enum
  match_visibility_enum
  match_format_enum
  facility_type_enum
  country_enum
  sport_enum
  surface_type_enum
  availability_enum
  participant_status_enum
  network_member_role_enum
  membership_status_enum
  join_policy_enum
  network_status_enum
  network_visibility_enum
  day_enum
  period_enum
}
```

```
Enum gender_enum {
  M
  F
  O
  prefer_not_to_say
}


Enum playing_hand_enum {
  right
  left
}

Enum organization_nature_enum {
  public
  private
}

Enum organization_type_enum {
  club
  municipality
  city
  association
}

Enum admin_role_enum {
  super_admin
  moderator
  support
}

Enum role_enum {
  admin
  staff
  player
  coach
  owner
}

Enum match_duration_enum {
  "30"
  "60"
  "90"
  "120"
}

Enum match_type_enum {
  practice
  competitive
  both
}

Enum match_gender_enum {
  male
  female
  any
}

Enum match_status_enum {
  draft
  open
  closed
  canceled
  completed
  in_progress
}

Enum match_visibility_enum {
  public
  private
}

Enum match_format_enum {
  singles
  doubles
  mixed_doubles
  custom
}

Enum match_access_enum {
  open
  request
}

Enum facility_type_enum {
  municipal
  university
  club
  school
  community_center
}

Enum country_enum {
  Canada
  "United States"
}

Enum sport_enum {
  tennis
  pickleball
  badminton
  squash
}

Enum surface_type_enum {
  hard
  clay
  grass
  synthetic
  carpet
  asphalt
  concrete
}

Enum availability_enum {
  available
  under_maintenance
  closed
  reserved
}

Enum participant_status_enum {
  joined
  left
  kicked
  invited
  declined
}
```

```
Enum network_member_role_enum {
  member
  admin
  moderator
}

Enum day_enum {
  monday
  tuesday
  wednesday
  thursday
  friday
  saturday
  sunday
}

Enum period_enum {
  morning   [note: 'Typically 6am–12pm']
  afternoon [note: 'Typically 12pm–6pm']
  evening   [note: 'Typically 6pm–12am']
}

Enum relationship_type_enum {
  favorite
  block
}

Enum contact_source_enum {
  manual          [note: "Manually added by the user"]
  phone_import    [note: "Imported from device contacts"]
  matched_user    [note: "Automatically matched to an existing player"]
}

Enum membership_status_enum {
  pending        [note: "User invited but hasn’t accepted yet"]
  active         [note: "Current active member"]
  suspended      [note: "Temporarily restricted or muted"]
  left           [note: "User voluntarily left the network"]
  removed        [note: "Removed by a moderator/admin"]
  banned         [note: "Permanently banned from rejoining"]
}

Enum join_policy_enum {
  open            [note: "Anyone can join freely"]
  request         [note: "Users must request to join, pending moderator approval"]
  invite_only     [note: "Only added through an invitation"]
  closed          [note: "Joining disabled — network frozen or archived"]
}

Enum network_status_enum {
  active          [note: "Visible and operational"]
  inactive        [note: "Temporarily disabled, not accepting new members"]
  archived        [note: "Read-only; preserved for history"]
  deleted         [note: "Soft-deleted; retained for audit or recovery"]
}

Enum network_visibility_enum {
  public        [note: "Visible to everyone, searchable and discoverable"]
  private       [note: "Visible only to members and invited users"]
  hidden        [note: "Completely hidden; accessible only via direct link or invite"]
}
```

```
%% ==================================
%% AUTH & CORE USER (Profiles)
%% ==================================
class AuthUsers {
  +id: UUID (PK)
  +aud: varchar
  +role: varchar
  +email: varchar
  +email_confirmed_at: timestamptz
  +phone: varchar
  +phone_confirmed_at: timestamptz
  +confirmed_at: timestamptz
  +last_sign_in_at: timestamptz
  +app_metadata: jsonb
  +user_metadata: jsonb
  +identities: jsonb
  +created_at: timestamptz
  +updated_at: timestamptz
  +is_anonymous: boolean
}

class Profile {
  +id: UUID (PK, FK > AuthUsers.id)
  +full_name: varchar
  +display_name: varchar
  +avatar_url: varchar
  +birth_date: date
  +locale: varchar
  +timezone: varchar
  +two_factor_enabled: boolean
  +last_active_at: timestamptz
  +is_active: boolean
  +role: enum [player, admin]
  +created_at: timestamptz
  +updated_at: timestamptz
  Methods:
    updateProfile()
    deactivate()
}

class Player {
  +profile_id: UUID (PK, FK > Profile.id)
  +username: varchar
  +gender: gender_enum
  +profile_photo_url: varchar
  +address: varchar
  +postal_code: varchar
  +reputation_score: float
  +verified: boolean
  +playing_hand: playing_hand_enum
  +max_travel_distance: int (default: 6, max: 50) <--- NEW ATTRIBUTE
  Methods:
    updateSkillLevel()
    getReputation()
    requestVerification()
    joinSport(sportId)
}

class Admin {
  +profile_id: UUID (PK, FK > Profile.id)
  +admin_role: admin_role_enum
  Methods:
    banUser(userId)
    viewStats()
    refundPayment(paymentId)
    resolveSuggestion(suggestionId)
}

AuthUsers "1" -- "1" Profile : manages_profile
Profile <|-- Player
Profile <|-- Admin

%% ================================
%% ORGANIZATION ENTITY & MEMBERSHIP
%% ================================
class Organization {
  +id: UUID (PK)
  +creator_id: UUID (FK > Profile.id)
  +name: varchar
  +organization_type: organization_type_enum
  +organization_nature: organization_nature_enum
  +description: text
  +contact_email: varchar
  +phone_number: varchar
  +logo_url: varchar
  +website: varchar
  Methods:
    updateInfo()
    deactivate()
}

class OrganizationMember {
  +id: UUID
  +organization_id: UUID (FK > Organization.id)
  +profile_id: UUID (FK > Profile.id)
  +role: role_enum
  +permissions: jsonb
  +joined_at: timestamptz
  +left_at: timestamptz
  +invited_by: UUID (FK > Profile.id)
  Methods:
    grantPermission(permission)
    revokePermission(permission)
    leave()
}
```

```
Profile "1" -- "*" Organization : creates_owns
Organization "1" -- "*" OrganizationMember : has_members
Profile "1" -- "*" OrganizationMember : belongs_to

%% ================================
%% SPORTS, RATINGS & AVAILABILITY
%% ================================
class Sport {
  +id: UUID
  +name: varchar
  +description: text
  +is_active: boolean
  Methods:
    activate()
    deactivate()
    updateDescription()
}

class PlayerSport {
  +id: UUID
  +player_id: UUID (FK > Player.profile_id)
  +sport_id: UUID (FK > Sport.id)
  +preferred_match_duration: match_duration_enum
  +preferred_match_type: match_type_enum
  +created_at: timestamptz
  Methods:
    setPreferences(duration, type)
    archive()
}

class PlayerAvailability {
  +id: UUID (PK)
  +player_id: UUID (FK > Player.profile_id)
  +day_of_week: day_enum
  +time_period: period_enum
  +is_active: boolean
  +created_at: timestamptz
  +updated_at: timestamptz
  Methods:
    setAvailable()
    setUnavailable()
}
```

```
class Rating {
  +id: UUID
  +sport_id: UUID
  +name: varchar
  +description: text
  Methods:
    addLevel(value, label)
}

class RatingScore {
  +id: UUID
  +rating_id: UUID
  +value: float
  +label: varchar
  +description: text
  Methods:
    updateLabel()
}

class PlayerRatingScore {
  +id: UUID
  +rating_score_id: UUID
  +player_id: UUID (FK > Player.profile_id)
  +assigned_at: timestamptz
  +expires_at: timestamptz
  Methods:
    expire()
    renew()
}

Player "1" -- "*" PlayerSport : plays
Player "1" -- "*" PlayerAvailability : is_available
Sport "1" -- "*" PlayerSport : played_by
Sport "1" -- "*" Rating : supports
Rating "1" -- "*" RatingScore : defines_levels
Player "1" -- "*" PlayerRatingScore : rated
RatingScore "1" -- "*" PlayerRatingScore : assigned_to

%% ================================
%% FACILITY & BOOKING
%% ================================
class Facility {
  +id: UUID
  +organization_id: UUID (FK > Organization.id)
  +name: varchar
  +slug: varchar
  +facility_type: facility_type_enum
  +description: text
  +address: varchar
  +city: varchar
  +province: country_enum
  +postal_code: varchar
  +latitude: decimal
  +longitude: decimal
  +phone: varchar
  +email: varchar
  +website: varchar
  +indoor_courts: int
  +outdoor_courts: int
  +total_courts: int
  +is_active: boolean
  +created_at: timestamptz
  +updated_at: timestamptz
  Methods:
    updateDetails()
    archive()
}

class Court {
  +id: UUID
  +facility_id: UUID
  +name: varchar
  +court_number: int
  +sport: sport_enum
  +surface_type: surface_type_enum
  +indoor: boolean
  +lighting: boolean
  +lines_marked_for_multiple_sports: boolean
  +availability_status: availability_enum
  +notes: text
  +is_active: boolean
  +created_at: timestamptz
  +updated_at: timestamptz
  Methods:
    getAvailability()
    updateInfo()
    closeCourt()
}
```

```
class CourtSlot {
  +id: UUID
  +court_id: UUID
  +start_time: timestamptz
  +end_time: timestamptz
  +price: money
  +is_booked: boolean
  +booked_by: UUID (FK > Profile.id)
  Methods:
    reserveSlot(playerId, payment)
    releaseSlot()
}

class Booking {
  +id: UUID
  +court_slot_id: UUID
  +player_id: UUID (FK > Player.profile_id)
  +status: enum
  +payment_id: UUID
  +created_at: timestamptz
  +updated_at: timestamptz
  Methods:
    confirmBooking()
    cancelBooking()
    updateStatus(newStatus)
}

class Payment {
  +id: UUID
  +booking_id: UUID
  +amount: money
  +currency: varchar
  +provider: enum
  +status: enum
  +created_at: timestamptz
  Methods:
    process()
    refund()
    markDispute()
}

Organization "1" -- "*" Facility : owns
Facility "1" -- "*" Court : hosts
```

```
Court "1" -- "*" CourtSlot : has
Booking "*" -- "1" CourtSlot : reserves
Player "1" -- "*" Booking : booked_by
Payment "*" -- "1" Booking : for

%% ================================
%% MATCHES & PARTICIPANTS
%% ================================
class Match {
  +id: UUID
  +creator_id: UUID (FK > Profile.id)
  +court_id: UUID
  +scheduled_at: timestamptz
  +status: match_status_enum
  +visibility: match_visibility_enum
  +match_format: match_format_enum
  +match_gender: match_gender_enum
  +match_access: enum
  +conversation_id: UUID
  Methods:
    invitePlayer(playerId)
    updateScore(scoreData)
    submitFeedback(feedback)
    cancel()
}

class MatchParticipant {
  +id: UUID
  +match_id: UUID
  +player_id: UUID (FK > Player.profile_id)
  +joined_at: timestamptz
  +left_at: timestamptz
  +status: participant_status_enum
  +score: int
  Methods:
    acceptInvite()
    leaveMatch()
    reportNoShow()
}

Player "1" -- "*" Match : organizes
Match "1" -- "*" MatchParticipant : includes
Player "1" -- "*" MatchParticipant : plays
Court "1" -- "*" Match : hosts
```

```
%% =================================
%% NETWORKS (SOCIAL STRUCTURES)
%% =================================
class NetworkType {
  +id: UUID
  +name: varchar
  +num_max_members: int
  +num_max_admins: int
  +member_policy: text
  +description: text
  +is_active: boolean
  Methods:
    updatePolicy()
    deactivate()
}

class Network {
  +id: UUID
  +network_type_id: UUID
  +name: varchar
  +slug: varchar
  +description: text
  +creator_id: UUID (FK > Profile.id)
  +conversation_id: UUID
  +avatar_url: varchar
  +is_public: network_visibility_enum
  +status: network_status_enum
  +join_policy: join_policy_enum
  +created_at: timestamptz
  +updated_at: timestamptz
  +archived_at: timestamptz
  Methods:
    archive()
    addMember(userId, role)
    removeMember(userId)
}

class NetworkMember {
  +id: UUID
  +network_id: UUID
  +profile_id: UUID (FK > Profile.id)
  +role: network_member_role_enum
  +status: membership_status_enum
  +joined_at: timestamptz
  +left_at: timestamptz
  +invited_by: UUID (FK > Profile.id)
  Methods:
    promote(role)
    leave()
}

NetworkType "1" -- "*" Network : types
Network "1" -- "*" NetworkMember : has
Profile "1" -- "*" NetworkMember : in

%% ================================
%% MESSAGING (CHAT & THREADS)
%% ================================
class Conversation {
  +id: UUID
  +type: enum
  +name: varchar
  +avatar_url: varchar
  +last_message_at: timestamptz
  +created_at: timestamptz
  +archived_at: timestamptz
  Methods:
    addParticipant(userId)
    archive()
    sendSystemMessage(content)
}

class ConversationParticipant {
  +id: UUID
  +conversation_id: UUID
  +profile_id: UUID (FK > Profile.id)
  +last_read_at: timestamptz
  +joined_at: timestamptz
  +left_at: timestamptz
  +is_muted: boolean
  Methods:
    mute()
    markRead(time)
    leave()
}

class Message {
  +id: UUID
  +conversation_id: UUID
  +sender_id: UUID (FK > Profile.id)
  +reply_to_id: UUID
  +content: text
  +sent_at: timestamptz
  +edited_at: timestamptz
  +deleted_at: timestamptz
  +is_reported: boolean
  +reported_at: timestamptz
  +reported_by: UUID (FK > Profile.id)
  Methods:
    edit(newContent)
    delete()
    report(byUserId)
}

class MessageAttachment {
  +id: UUID
  +message_id: UUID
  +file_type: enum
  +file_url: varchar
  +thumbnail_url: varchar
  +file_size: bigint
  +mime_type: varchar
  +duration: int
  +width: int
  +height: int
  +original_filename: varchar
  +storage_key: varchar
  +uploaded_at: timestamptz
  Methods:
    generateThumbnail()
}

Conversation "1" -- "*" ConversationParticipant : has
Profile "1" -- "*" ConversationParticipant : in
Conversation "1" -- "*" Message : contains
Profile "1" -- "*" Message : sends
Message "1" -- "*" MessageAttachment : attaches
Network "1" -- "0..1" Conversation : main_chat

%% ================================
%% FEEDBACK, SUGGESTION, NOTIFICATION
%% ================================
class Feedback {
  +id: UUID
  +match_id: UUID
  +from_player_id: UUID (FK > Player.profile_id)
  +to_player_id: UUID (FK > Player.profile_id)
  +punctuality: int
  +respect: int
  +review: int
  +comments: text
  +submitted_at: timestamptz
  Methods:
    computeReputation()
}
Match "1" -- "*" Feedback : collects
Player "1" -- "*" Feedback : gives_receives

class Suggestion {
  +id: UUID
  +from_user_id: UUID (FK > Profile.id)
  +subject: varchar
  +content: text
  +category: varchar
  +priority: varchar
  +created_at: timestamptz
  +updated_at: timestamptz
  +is_resolved: boolean
  +resolved_at: timestamptz
  +resolved_by: UUID (FK > Profile.id)
  +admin_notes: text
  Methods:
    markResolved()
    escalateToAdmin()
}
Profile "1" -- "*" Suggestion : submits

class Notification {
  +id: UUID
  +user_id: UUID (FK > Profile.id)
  +type: enum
  +target_id: UUID
  +title: varchar
  +body: text
  +payload: jsonb
  +is_read: boolean
  +read_at: timestamptz
  +sent_at: timestamptz
  +expires_at: timestamptz
  Methods:
    send()
    markRead()
}
Profile "1" -- "*" Notification : receives
```