Team & User Hierarchy - MVP PRD (Brief)

Goal: Build dashboard for automatic access inheritance system
Scale: 500 users, 50 teams, unlimited managers per user
Core Logic: User joins team → managers inherit access; Resource assigned to team → all members access it

Key Features from Wireframes

Client Management (Primary View)

- **Client-centric dashboard** showing access patterns
- **"Users with access" counter** for each client
- **Multi-team client assignment** (same client in multiple teams)
- **"Show clients without teams" filter** for orphaned clients
- **Client sorting** by access count and creation date
- **Popup detail view** showing teams and user access breakdown

Role-Based Hierarchy

- **Banking roles:** RM, Senior RM, Head of RM
- **Role-based permissions** and responsibilities
- **"Create new role" functionality** for custom positions
- **Role dropdown** for quick assignments

Team Assignment Interface

- **"Auto-assign clients" checkbox** for teams
- **Direct members vs Manager access** clear separation
- **Visual inheritance indicators** (granted via X)
- **Bulk user selection** with checkboxes
- **"Invite new user" integration** within team management

Manager Hierarchy Visualization

- **"Line Managers" section** showing reporting chain
- **Subordinates section** for direct reports
- **Manager type classification** (implied from context)
- **Remove buttons** for relationship management

Access Pattern Display

- **Direct vs Manager access badges** throughout UI
- **"Granted via" attribution** showing inheritance source
- **Team membership counts** in parentheses
- **Client access summaries** per team

Data Model (Enhanced for Banking Context)

sql

```
-- Core entities
users {
  id UUID PRIMARY KEY,
  email VARCHAR UNIQUE,
  name VARCHAR,
  role VARCHAR DEFAULT 'RM'  -- 'RM', 'Senior RM', 'Head of RM'
}

teams {
  id UUID PRIMARY KEY,
  name VARCHAR,
  auto_assign_clients BOOLEAN DEFAULT false  -- From wireframe checkbox
}

resources {
  id UUID PRIMARY KEY,
  name VARCHAR,
  type VARCHAR DEFAULT 'client',
  segment VARCHAR  -- 'Private', 'Corporate', 'Retail'
}

-- Relationships
user_managers {
  user_id UUID REFERENCES users(id),
  manager_id UUID REFERENCES users(id),
  manager_type VARCHAR DEFAULT 'line_manager',  -- 'line_manager', 'functional', 'dotted_line'
  created_at TIMESTAMP
}

team_members {
  team_id UUID REFERENCES teams(id),
  user_id UUID REFERENCES users(id),
  access_type ENUM('direct', 'manager'),
  granted_via UUID REFERENCES users(id),  -- NULL for direct, user_id for manager
  joined_at TIMESTAMP
}

team_resources {
  team_id UUID REFERENCES teams(id),
  resource_id UUID REFERENCES resources(id),
  assigned_at TIMESTAMP
}
```

Key Enhancements:

- Added `role` field for banking positions (RM, Senior RM, Head of RM)
- Added `auto_assign_clients` for teams (from wireframe checkbox)
- Added `segment` field for client categorization
- Added `manager_type` to distinguish different manager relationships

Core Operations

1. Add User to Team (with auto-inheritance)

```
Input: user_id, team_id
Process:
  1. Add user as 'direct' member
  2. Get user's managers (recursive, max 3 levels)
  3. Add each manager as 'manager' member
Output: List of added users with access types
```

2. Assign Manager

```
Input: user_id, manager_id
Process:
  1. Validate: no self-mgmt, no cycles, max depth
  2. Add to user_managers table
  3. Inherit all user's current teams
Output: Teams inherited by new manager
```

3. Assign Resource to Team

```
Input: team_id, resource_id
Process:
  1. Add to team_resources table
  2. All team members now have access
Output: Users who gained access
```

4. Remove User from Team

```
Input: user_id, team_id
Process:
  1. Remove user's direct membership
  2. Check managers - remove if no other path
Output: Users who lost access
```

Business Rules

1. **No self-management:** user_id ≠ manager_id
2. **No circular chains:** A→B→C→A forbidden
3. **Max 3 hierarchy levels:** Performance limit
4. **Direct takes precedence:** If user is direct member, don't add as manager
5. **Multiple managers OK:** User can have unlimited managers
6. **Access through teams only:** All resource access via team membership

---

Key Queries

sql

```sql
-- User's accessible resources
SELECT r.name, t.name as via_team, tm.access_type
FROM resources r
JOIN team_resources tr ON r.id = tr.resource_id
JOIN team_members tm ON tr.team_id = tm.team_id
JOIN teams t ON tm.team_id = t.id
WHERE tm.user_id = ?

-- Team members with access details
SELECT u.name, tm.access_type, via.name as granted_via
FROM team_members tm
JOIN users u ON tm.user_id = u.id
LEFT JOIN users via ON tm.granted_via = via.id
WHERE tm.team_id = ?

-- User's managers (recursive)
WITH RECURSIVE mgr_chain AS (
  SELECT manager_id, 1 as level FROM user_managers WHERE user_id = ?
  UNION ALL
  SELECT um.manager_id, mc.level + 1
  FROM mgr_chain mc
  JOIN user_managers um ON mc.manager_id = um.user_id
  WHERE mc.level < 3
)
SELECT manager_id FROM mgr_chain
```

API Endpoints (Enhanced for Wireframes)

typescript

```
// Core CRUD
POST   /api/users          { email, name, role }
POST   /api/teams          { name, auto_assign_clients }
POST   /api/clients        { name, type, segment }
POST   /api/roles          { name, permissions }

// User Management (Wireframe 4)
POST   /api/users/:id/managers       { manager_id, manager_type }
PUT    /api/users/:id/role           { role }
GET    /api/users/:id/subordinates   // Direct reports
GET    /api/users/:id/line-managers  // Manager chain

// Team Management (Wireframe 3)
POST   /api/teams/:id/members        { user_id }
POST   /api/teams/:id/clients        { client_id }
PUT    /api/teams/:id/auto-assign    { auto_assign_clients }
DELETE /api/teams/:teamId/members/:userId
DELETE /api/teams/:teamId/clients/:clientId

// Client Views (Wireframes 1 & 2)
GET    /api/clients                  // List with filters
GET    /api/clients/:id/teams        // Teams assigned to client
GET    /api/clients/:id/users        // Users with access
GET    /api/clients/unassigned       // Clients without teams

// Dashboard Queries
GET    /api/teams/:id/members        // With access_type and granted_via
GET    /api/teams/stats              // User/client counts for list view
GET    /api/users/:id/accessible-clients  // Clients user can access
GET    /api/dashboard/client-access/:clientId  // Who can access specific client

// Bulk Operations
POST   /api/teams/:id/bulk-assign-clients  { client_ids[] }
POST   /api/teams/:id/bulk-add-members     { user_ids[] }
POST   /api/users/invite                   { email, name, role, team_ids[] }
```

Dashboard Screens (Based on Wireframes)

1. Main Dashboard - Clients View (Wireframe 1)

- **Left Panel:** Navigation (Users/Teams)
- **Center:** Client list with filtering
    - Client ID (ABC123, ABC456)
    - Name (Yummy, Nummy)

    - Type (Natural Person)
    - Segment (Private)
    - Users with access count
- **Right Panel:** Client detail popup
    - Shows teams assigned (Private RM Team 1, Private RM Team 2)
    - Direct Members list (Shan, Yusuf, Osama)
    - Manager Access list (DK - Manager of Shan, Roger, Piyush)
    - Teams dropdown for assignment
- **Key Features:**
    - "Show clients without teams" checkbox
    - Client sorting by user access count and creation date
    - Team assignment interface

2. Teams List View (Wireframe 2)

- **Simple table layout:**
    - Team ID (ABC123, ABC456)
    - Name (Yummy, Nummy)
    - Users count (15, 7)
    - Clients count (20, 30)
- **Navigation:** Clients/Teams tab toggle

3. Team Detail with Assignment (Wireframe 3)

- **Team info section:**
    - Team name and auto-assign checkbox
    - "Automatically assign all clients to this team"
- **Direct Members (5) section:**
    - List of direct team members (Shan - Senior RM, Yusuf - Head of RM, Osama - Head of RM)
    - ✅ remove buttons for each member
- **Manager Access (6) section:**
    - Auto-inherited managers (Shan - Senior RM, Yusuf - Head of RM, Osama - Head of RM)
    - Shows inheritance chain
- **Clients (2) section:**
    - Assigned clients list
    - ✅ remove buttons
- **Right panel:** User selection
    - Checkboxes for available users
    - "Invite new user" button at bottom

4. User Detail/Role Management (Wireframe 4)

- **User profile section:**
    - Name (ABC123), client count (10 clients)
    - Role dropdown (RM, Senior RM, Head of RM)
    - "Create new role" button
- **Line Managers (3) section:**
    - Manager hierarchy (Shan - Senior RM, Yusuf - Head of RM, Osama - Head of RM)
    - ✅ remove buttons
- **Teams (2) section:**
    - Team assignments (Private RM Team - 5 clients | 7 users)
    - ✅ remove buttons
- **Subordinates (3) section:**
    - Direct reports (Shan - Senior RM)
- **Right panels:**
    - Role assignment interface
    - User invitation/team assignment
    - Team selection checkboxes

Sample Data for MVP (Banking Context)

javascript

```
// Users with banking roles
const users = [
  { id: 'u1', email: 'shan@bank.com', name: 'Shan', role: 'Senior RM' },
  { id: 'u2', email: 'yusuf@bank.com', name: 'Yusuf', role: 'Head of RM' },
  { id: 'u3', email: 'osama@bank.com', name: 'Osama', role: 'Head of RM' },
  { id: 'u4', email: 'dk@bank.com', name: 'DK', role: 'Senior RM' },
  { id: 'u5', email: 'roger@bank.com', name: 'Roger', role: 'Head of RM' },
  { id: 'u6', email: 'piyush@bank.com', name: 'Piyush', role: 'Head of RM' }
]

// Teams (Private Banking focus)
const teams = [
  { id: 't1', name: 'Private RM Team 1', auto_assign_clients: false },
  { id: 't2', name: 'Private RM Team 2', auto_assign_clients: false },
  { id: 't3', name: 'Gold Segment', auto_assign_clients: true }
]

// Resources (Clients with segments)
const resources = [
  { id: 'c1', name: 'ABC123 - Yummy', type: 'client', segment: 'Private' },
  { id: 'c2', name: 'ABC456 - Nummy', type: 'client', segment: 'Private' },
  { id: 'c3', name: 'DEF789 - Tech Corp', type: 'client', segment: 'Corporate' }
]

// Manager hierarchy (from wireframes)
const managers = [
  { user_id: 'u1', manager_id: 'u4', manager_type: 'line_manager' }, // DK manages Shan
  { user_id: 'u2', manager_id: 'u3', manager_type: 'line_manager' }, // Osama manages Yusuf
  { user_id: 'u4', manager_id: 'u5', manager_type: 'line_manager' }  // Roger manages DK
]

// Team memberships (showing inheritance)
const team_members = [
  // Private RM Team 1
  { team_id: 't1', user_id: 'u1', access_type: 'direct', granted_via: null },     // Shan direct
  { team_id: 't1', user_id: 'u4', access_type: 'manager', granted_via: 'u1' },  // DK via Shan
  { team_id: 't1', user_id: 'u5', access_type: 'manager', granted_via: 'u1' },  // Roger via Shan
  // Private RM Team 2
  { team_id: 't2', user_id: 'u2', access_type: 'direct', granted_via: null },     // Yusuf direct
  { team_id: 't2', user_id: 'u3', access_type: 'manager', granted_via: 'u2' }   // Osama via Yusuf
]

// Client assignments
const team_resources = [
  { team_id: 't1', resource_id: 'c1' },  // ABC123 to Team 1
  { team_id: 't1', resource_id: 'c2' },  // ABC456 to Team 1
  { team_id: 't2', resource_id: 'c1' }   // ABC123 also to Team 2 (multi-team access)
]
```

Implementation Priority (Based on Wireframes)

Week 1: Core Client & Team Views

- Client list with filtering (Wireframe 1)
- Teams table with stats (Wireframe 2)
- Basic CRUD for users, teams, clients
- Client-team assignment interface

Week 2: Team Detail & Membership

- Team detail view with member management (Wireframe 3)
- Direct vs Manager access indicators
- Add/remove team members
- Auto-inheritance when adding users

Week 3: User Management & Hierarchy

- User detail/role management (Wireframe 4)
- Manager assignment interface
- Role management (RM, Senior RM, Head of RM)
- Subordinates view

Week 4: Advanced Features

- "Auto-assign clients" functionality
- Bulk operations (invite multiple users)
- Client access analysis ("Users with access" counts)
- Polish and edge case handling

Week 5: Integration & Polish

- "Show clients without teams" filter
- Role-based permissions
- Client sorting and search
- Performance optimization for large datasets

Success Criteria

✅ Create 20 users, 10 teams, 15 resources in UI

✅ Add user to team → managers automatically added

✅ Assign resource → all team members can access

✅ UI clearly shows "Direct" vs "Manager" access

✅ Remove user → managers lose access (if no other path)

✅ All operations < 500ms

Tech Notes

- Use React + TypeScript for dashboard
- SQLite/PostgreSQL for database
- Node.js/Express for API
- Recursive SQL queries for manager chains
- Foreign key constraints for data integrity