Specification - Hunting Log

The goal of this web application is to provide hunters and hunting ground managers with an efficient tool for tracking hunting activities. The application will allow users to log visits, track hunting and manage detailed information about hunting grounds, lookout towers and feeding areas. Primary users of the application are hunters, who record their visits and game hunting and hunting ground administrators, who have extended permissions to manage areas, hunting lookouts and overall game reserve administration.

Functional requirements

Roles

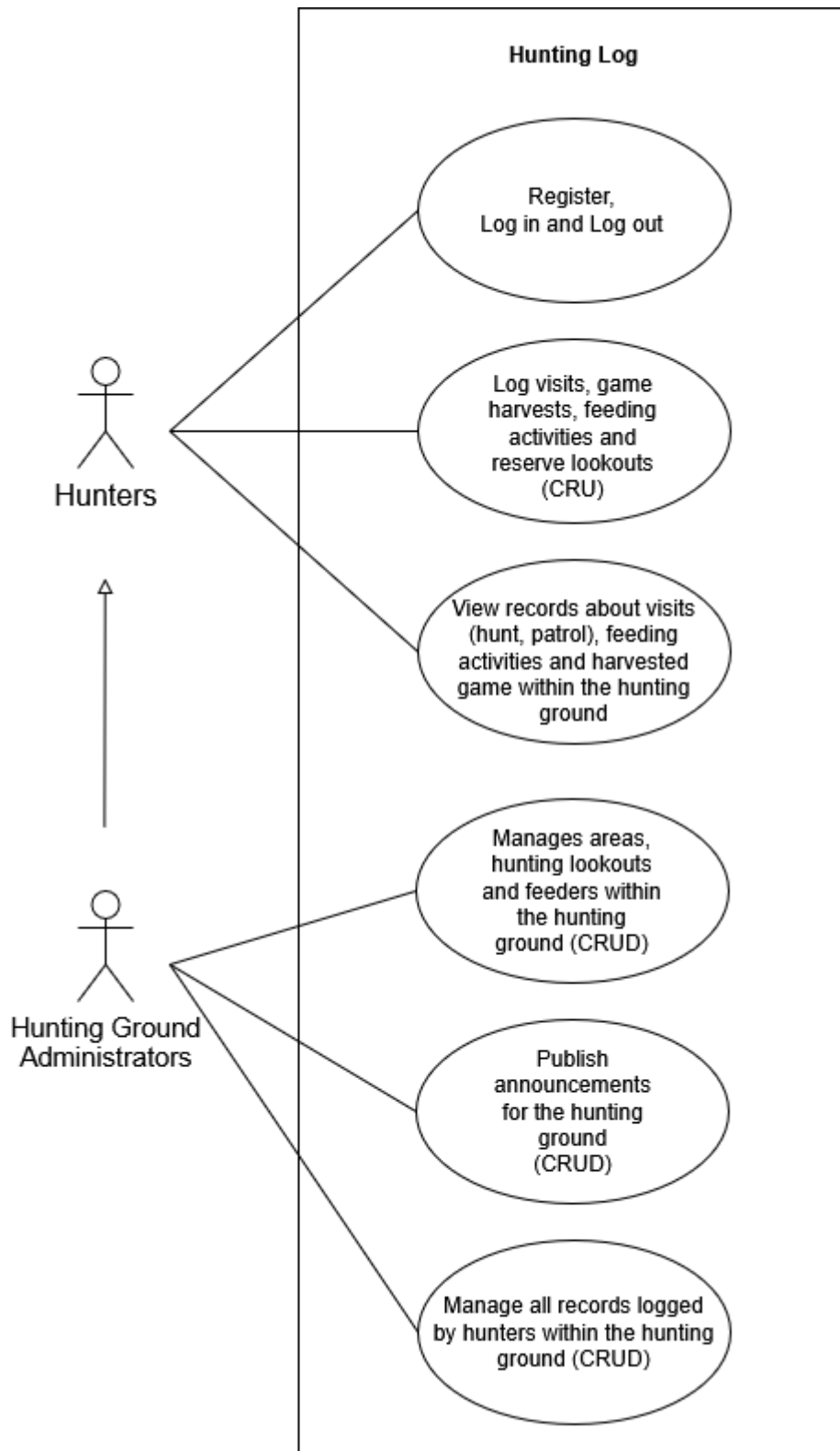The application distinguishes two roles:

- **Hunters** - regular members of hunting grounds who log their activities, including visits, hunting records and feeding.

- **Hunting ground administrators -** users with extended permissions who manage hunting ground areas, create and edit lookout towers, assign zones. They can also publish important announcements visible to all hunters in the hunting ground. Hunting ground administrators are also hunters and can log their own hunting activities. There can only be one administrator per hunting ground, but they have the ability to transfer administration rights to another hunter if needed.

Use Case Diagram

Notes:

- CRUD = Create, Read, Update, Delete
- Register: Unauthenticated users can create a new account. If they register as a hunter, they must select an existing hunting ground. If they register as a hunting ground administrators, they create a new hunting ground and are automatically assigned to it.
- Log in and Log out are not primary user goals, but they are required to access all other features. Only registered users can log in. After logging in, users can log visits, record hunting activity or manage the hunting ground based on their assigned role.

- Delete operations: Some delete functionalities for the hunting ground administrator role may be implemented as a soft delete to maintain data integrity due to existing references. The exact implementation details will be determined during the development phase.

Hunting Log

Hunters

- Register, Log in and Log out
- Log visits, game harvests, feeding activities and reserve lookouts (CRU)
- View records about visits (hunt, patrol), feeding activities and harvested game within the hunting ground

Hunting Ground Administrators

- Manages areas, hunting lookouts and feeders within the hunting ground (CRUD)
- Publish announcements for the hunting ground (CRUD)
- Manage all records logged by hunters within the hunting ground (CRUD)

Data model

The following conceptual data model includes entities, attributes and their relationships.

Entities and Attributes:

Users

- Represents hunters and hunting ground administrators.
- Constraints and rules:
    - A user can belong to only one hunting ground.
    - The user's role (hunter or admin) is not stored here, but in the User_Hunting_Ground table.
- Attributes:
    - id - Unique identifier for each user.
    - username - Unique login name.
    - email - Unique email address.
    - password_hash - Securely stored hashed password.
    - first_name - User's first name.
    - last_name - User's last name.
    - street - Street name.
    - house_number - House number.
    - postal_code - Postal code.
    - city - City.

Hunting_Grounds

- Represents individual hunting ground.
- Attributes:
    - id - Unique identifier.
    - name - Name of the hunting ground.
    - description - Additional details about the hunting ground.

User_Hunting_Ground

- Defines the relationship between users and hunting ground.
- Constraints & Rules:
    - A user can belong to only one hunting ground.
- A user can have only one role per hunting ground (either hunter or admin).
- Attributes:
    - user_id – (Foreign Key → Users.id).
    - hunting_ground_id - (Foreign Key → Hunting_Grounds.id) Assigned hunting ground.
    - role - (ENUM: Hunter, Admin) Defines the user's role in that hunting ground.

Hunting Areas

- Each hunting ground consists of multiple areas where hunts take place.
- Attributes:
    - id - Unique identifier.
    - hunting_ground_id - (Foreign Key → Hunting_Grounds.id)
      The hunting ground where the area belongs.
    - name - Name of the area.

Structures

- Represents lookout towers, feeding stations and other structures inside an area.
- Attributes:
    - id - Unique identifier.
    - hunting_area_id - (Foreign Key → Hunting_Areas.id) The area where the structure is located.
    - name - Name of the structure.
    - type - (ENUM: Lookout Tower, Feeding Station, Shelter/Lodge).
    - notes - (Nullable) Additional details.

Visits

- Represents records of visits to hunting areas, including their purpose, time and optionally the structure used during the visit.
- Constraints and rules:
    - Each visit must be assigned to one hunting area.
    - If a structure is specified, it must be located within the same area as the visit.
    - The hunter must belong to the same hunting ground as the area being visited.
- Attributes:
    - id – Unique visit identifier.
    - hunter_id – (Foreign Key → Users.id) The hunter who made the visit.
    - hunting_area_id – (Foreign Key → Hunting_Areas.id) The area visited.
    - structure_id - (Foreign Key → Structures.id, Nullable) The specific lookout tower or feeding station visited.
    - start_datetime - (DATETIME) The exact date and time when the visit started.
    - end_datetime - (DATETIME) The exact date and time when the visit ended.
    - purpose - (ENUM: Hunting, Inspection, Feeding, Repair, Lodge stay).
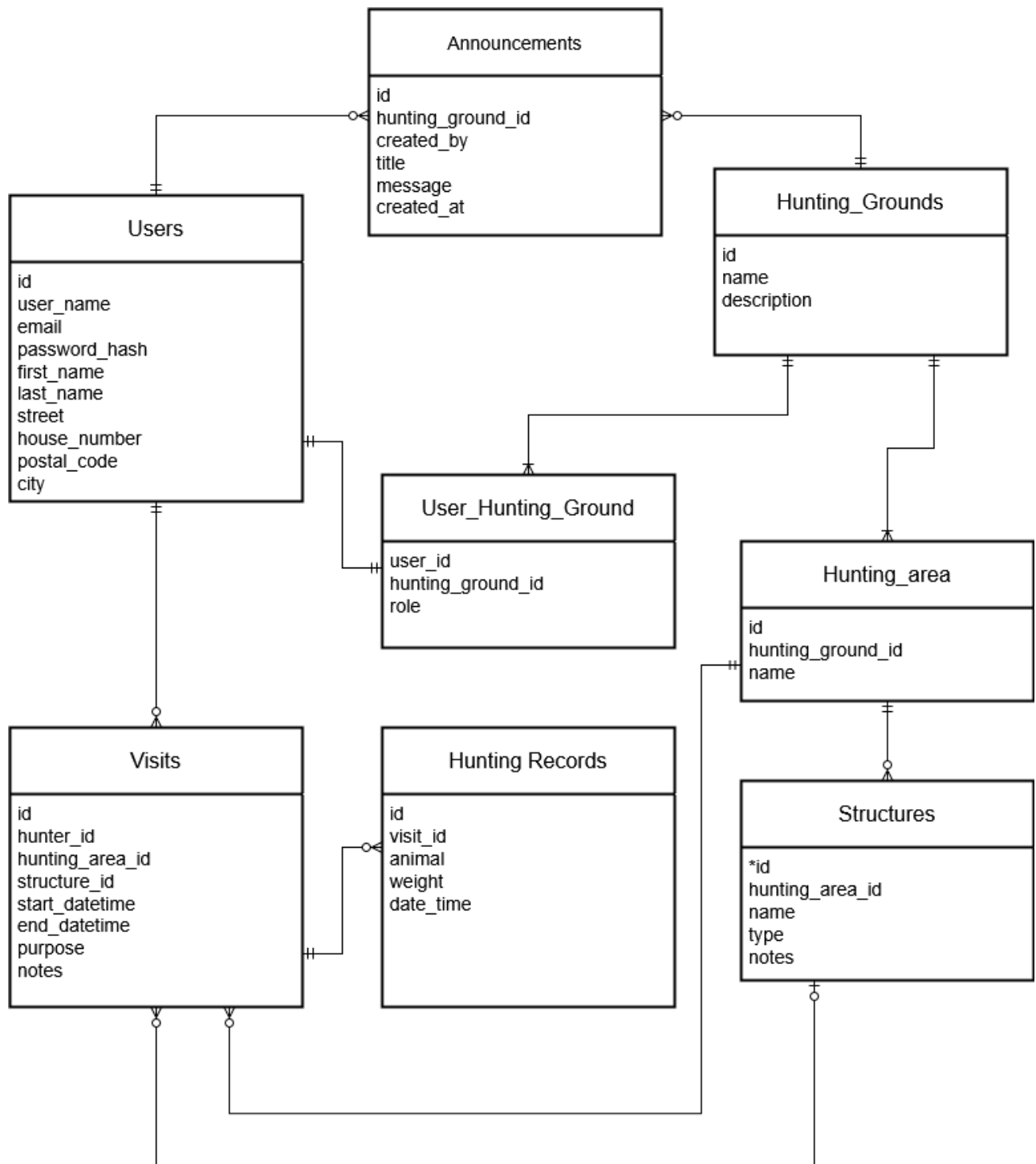   - notes - (Nullable) Additional details.

Hunting Records

- Represents successful hunts recorded during visits.
- Constraints and rules:
    - Each hunting record must be linked to a specific visit.
    - The visit must have purpose = 'Hunting'.
    - The date_time of the record must fall within the start_datetime and end_datetime of the visit.
- Attributes:
    - id – Unique hunting record identifier.
    - visit_id – (Foreign Key → Visits.id) The visit during which the hunt occurred.
    - animal – Type of animal hunted.
    - weight – Weight of the animal.
    - date_time – Timestamp of the hunt.

Announcements

- Represents messages published by hunting ground administrators to inform users about updates, planned events, or important notices related to the hunting ground.
- Constraints and rules:
    - Only users with the role hunting ground administrators can create announcements.
    - Each announcement is assigned to exactly one hunting ground.
    - Announcements are visible to all users who belong to that hunting ground.
- Attributes:
    - id – Unique identifier of the announcement.
    - hunting_ground_id – (Foreign Key → Hunting_Grounds.id) The hunting ground the announcement belongs to.
    - created_by – (Foreign Key → Users.id) The administrator who created the announcement.
    - title – Title of the announcement.
    - message – Content of the announcement.
    - created_at – Date and time when the announcement was created.

ER model

Announcements
- id
- hunting_ground_id
- created_by
- title
- message
- created_at

Users
- id
- user_name
- email
- password_hash
- first_name
- last_name
- street
- house_number
- postal_code
- city

Hunting_Grounds
- id
- name
- description

User_Hunting_Ground
- user_id
- hunting_ground_id
- role

Hunting_area
- id
- hunting_ground_id
- name

Visits
- id
- hunter_id
- hunting_area_id
- structure_id
- start_datetime
- end_datetime
- purpose
- notes

Hunting Records
- id
- visit_id
- animal
- weight
- date_time

Structures
- *id
- hunting_area_id
- name
- type
- notes

Architecture

The application will be based on the client-server architecture and it will use the SPA (Single Page Application) approach.

Technological requirements

- Client-side: React 19, JavaScript, HTML5, CSS3, Bootstrap v5.3.2
- Server-side: node.js 23, express.js 4.21.2, JavaScript
- Database: PostgreSQL 16
- Interface client - server: Rest API
- Hosting: render.com
- Supported browsers: Chrome, Firefox

Future work

As the application evolves, the following enhancements could be implemented:

- Display the hunting ground on a map with lookouts, feeding stations and other key locations to improve terrain orientation and simplify log entries.
- Allow a hunter to be registered in multiple hunting ground under a single account.
- Enable hunting ground administrators to manage multiple hunting ground instead of being limited to a single one.
- Implement an internal chat or messaging system within the hunting ground to allow hunters and administrators to coordinate activities.

Time schedule

Week 4

- Define the visual layout and structure of core application screens (login, dashboard, registration form).
- Implement initial page components with temporary content and working navigation.
- Probably about 6 - 8 h.

Week 5

- Implement user authentication with secure registration, login and password hashing.
- Set up role-based access control to ensure hunters and administrators can access only the sections of the application relevant to their assigned roles.
- Probably about 8 - 10 hours.

Week 6

- Implement backend logic for managing visit logs and hunting records, including data validation and saving to the database.
- Build frontend components to allow users to add and view visit logs and hunting activity.
- Ensure data flows correctly between user input, the server and the database.
- Probably about 8 - 9 hours.

Week 7

- Develop the administration interface for managing the hunting ground: create, edit areas and structures (lookout towers, feeders, lodges, etc.).
- Add basic access checks so that only administrators can use this part of the system.
- Probably about 8 - 10 hours.

Week 8

- Implement search filters and sorting options for visit and hunting record pages.
- Improve the overall user experience to make navigation and interaction more intuitive.
- Probably about 6 - 8 hours.

Week 9 (Beta version)

- Finalize core functionality: visit logs, hunting records, user authentication and admin management of areas and structures.
- Perform testing to ensure all selected features work properly.
- Deploy the working Beta version to a public hosting platform for review and feedback.
- Probably about 10 hours.

Week 10

- Gather feedback from testing, fix major bugs and optimize performance.
- Add any last-minute improvements based on user feedback.
- Probably about 6 - 8 hours.

Week 11 (Final version)

- Finalize the last improvements and make sure the application looks and works well.
- Deploy the final version, do final testing and prepare for the project presentation.
- Probably about 6 - 8 hours.