**Project Specification ID Chips**

**Project Overview**

**Purpose**

Develop a comprehensive platform (public website and intranet) to maintain and expose data about companion animals, facilitating identification, tracking, and information management.

All developments must be available in 4 languages:  
English  
Français  
Nederlands  
Deutsch

**Target Audience**

* Companion animal owners
* Veterinarians and veterinary clinics
* Government authorities and regulatory bodies
* Animal shelters and rescue organizations
* Pet adoption agencies

**Core Objectives**

1. Work with existing SQL Database
2. Provide secure access to animal records for authorized stakeholders and public access for some features
3. Facilitate animal registration, updates, and ownership transfers
4. Support lost animal recovery processes
5. Enable reporting and analytics for authorities

**Technical Stack**

**Frontend**

* **Framework**: React with Vite & Typescript
* **UI Library**: Material UI
* **State Management**: React Query
* **Styling**: Styled Components (with Tailwind CSS as needed)
* **API Integration**: Axios/React Query
* **Form Handling**: React Hook Form with Yup validation
* **Testing**: Jest and React Testing Library

**Backend**

* **API**: RESTful API using ASP.NET Core
* **ORM**: Entity Framework Core
* **Database**: Microsoft SQL Server
* **Authentication**: JWT with ASP.NET Identity on an azure sql server database
* **API Documentation**: Swagger/OpenAPI
* **Logging**: Serilog

**DevOps & Infrastructure**

* **Hosting**: Azure App Service
* **Database Hosting**: Azure SQL
* **CI/CD**: Azure DevOps or GitHub Actions
* **Monitoring**: Application Insights
* **Security**: HTTPS, CORS, input validation, SQL injection prevention

**Functional Requirements**

**Public Website Features**

1. **Animal Registration**
   * Registration form for new animals
   * Assignment of unique identification numbers
2. **Owner Portal**
   * Secure login for animal owners
   * View and update animal information
   * Report lost/found animals
   * Transfer ownership
3. **Lost & Found System**
   * Public registry of lost animals
   * Reporting mechanism for found animals
4. **Veterinarian Access NB in our application Veterinarian is knowed as Identifier**
   * Secure professional portal
   * Update medical records
   * Record microchip registration

**Intranet Features**

1. **Administrative Dashboard**
   * User management
   * Role-based access control
   * System configuration
2. **Data Management**
   * CRUD operations for all entities
   * Bulk import/export capabilities
   * Data validation rules
3. **Reporting System**
   * Statistical reports on registrations
   * Geographic distribution
   * Species/breed analytics
   * Custom report generation

**Data Model (Core Entities)**

1. **Animals**
2. **Owners**
3. **Veterinarians known as identifiers in our app**
4. **Identification entity holds data about animal identification like chip number with a link to Identifier entity.   
   Can have several identifications for one animal**

**Non-Functional Requirements**

1. **Performance**
   * Page load times under 2 seconds
   * Database queries optimized for large datasets
   * Support for concurrent users (100)
2. **Security**
   * Data encryption at rest and in transit
   * Regular security audits
   * GDPR compliance
   * Privacy by design
   * Captcha security for client
3. **Scalability**
   * Horizontal scaling capability
   * Database partitioning strategy
   * Caching implementation
4. **Accessibility**
   * WCAG 2.1 AA compliance
   * Screen reader compatibility
   * Keyboard navigation

Step 00001

I would like first to start with website development with 2 pages Home & contacts based on [www.apple.com](http://www.apple.com) with a rich theme and i18n implementation the simplest way but with full folder structure.  
I would like to have a left vertical responsive navigation and a language switcher on right of top bar

**Specifications for ID-Chips Website Development**

**1. Overview**

This document outlines the specifications for developing a web application that will interface with the existing Idc API endpoints. The website will provide a user-friendly interface for managing animal identification chips and related data.

**2. System Architecture**

**2.1 Frontend**

* **Framework**: React.js with TypeScript Vite
* **UI Library**: Material-UI or Tailwind CSS if needed
* **State Management**: React Query
* **Routing**: React Router
* **API Communication**: Axios

**2.2 Backend Integration**

* The website will consume the existing Idc API endpoints
* Authentication will use the existing token-based system
* No new backend development is required in the initial phase

**3. User Roles and Permissions**

**3.1 Anonymous Users**

* View public information about the ID-chip system
* Register for an account
* Log in to the system

**3.2 Authenticated Users**

* View and manage their profile information
* Register new animals and identification chips
* View and update their registered animals
* Generate reports for their animals

**3.3 Administrators**

* Manage all users, animals, and identifications
* Access system-wide reports and statistics
* Configure system settings

**4. Feature Requirements**

**4.1 Authentication and User Management**

* User login using existing API authentication endpoints
* User registration form
* Password reset functionality
* User profile management
* Session management (timeout, remember me)

**4.2 Dashboard**

* Overview of user's animals and identifications
* Quick access to common actions
* Notifications for important events (e.g., registration confirmations)
* Summary statistics relevant to the user

**4.3 Animal Management**

* List view of all animals registered by the user
* Detailed view of individual animal information
* Form for registering new animals
* Ability to update animal information
* Search and filter functionality

**4.4 Identification Management**

* Register new identification chips
* Associate/disassociate chips with animals
* View identification history
* Update identification information
* Search for identifications by various criteria

**4.5 Reporting**

* Generate standard reports (e.g., animal registration certificates)
* Export data in common formats (PDF, CSV)
* Custom report generation based on user-selected criteria

**5. API Integration Points**

**5.1 Identity Endpoints**

* POST api/identity/login - User authentication
* PATCH api/identity/user - Update user details

**5.2 Identifier Endpoints**

* GET api/identifiers - Retrieve all identifiers
* GET api/identifiers/GetById/{id} - Get specific identifier
* POST api/identifiers - Create new identifier
* PATCH api/identifiers/{id} - Update identifier
* DELETE api/identifiers/{id} - Delete identifier

**5.3 Identification Endpoints**

* GET api/identifications - List identifications
* GET api/identifications/{id} - Get specific identification
* POST api/identifications - Create new identification
* PATCH api/identifications/{id} - Update identification
* DELETE api/identifications/{id} - Delete identification

**5.4 EPN Sync Endpoints**

* GET api/epnsynclog - Get synchronization logs
* GET api/epnsynclog/{id} - Get specific sync log

**6. User Interface Requirements**

**6.1 General UI Requirements**

* Responsive design (mobile, tablet, desktop)
* Accessible according to WCAG 2.1 AA standards
* Consistent branding and styling
* Intuitive navigation
* Loading indicators for asynchronous operations
* Clear error messages and validation feedback

**6.2 Key Screens**

1. **Login/Registration**
   * Clean, simple forms with validation
   * Password strength indicator
   * Terms of service acceptance
2. **Dashboard**
   * Card-based layout for key information
   * Quick action buttons
   * Recent activity feed
   * Summary statistics
3. **Animal List**
   * Tabular view with sorting and filtering
   * Search functionality
   * Pagination
   * Quick actions (view, edit, delete)
4. **Animal Details**
   * Comprehensive information display
   * Associated identifications
   * Edit capabilities
   * History/audit log
5. **Identification Management**
   * Registration form for new chips
   * Association with animals
   * Status tracking
   * History view
6. **User Profile**
   * Personal information management
   * Password change
   * Notification preferences
   * Account settings

**7. Non-Functional Requirements**

**7.1 Performance**

* Page load time < 2 seconds
* API response handling with appropriate loading states
* Efficient data caching strategy
* Optimized assets (images, CSS, JavaScript)

**7.2 Security**

* HTTPS for all communications
* XSS and CSRF protection
* Input validation on all forms
* Secure handling of authentication tokens
* Protection against common web vulnerabilities

**7.3 Scalability**

* Component-based architecture for reusability
* Code splitting for optimized loading
* Efficient state management

**7.4 Maintainability**

* Comprehensive documentation
* Consistent coding standards
* Unit and integration tests
* Clear component structure

**8. Development Process**

**8.1 Development Phases**

1. **Design Phase**
   * UI/UX design approval
   * Component library setup
   * API integration planning
2. **Implementation Phase**
   * Core functionality development
   * API integration
   * Unit testing
3. **Testing Phase**
   * Integration testing
   * User acceptance testing
   * Performance testing
4. **Deployment Phase**
   * Staging environment deployment
   * Final testing
   * Production deployment

**8.2 Development Practices**

* Git-based version control
* Feature branch workflow
* Code reviews for all pull requests
* Continuous integration
* Automated testing

**9. Deliverables**

1. Source code repository
2. Development documentation
3. User documentation
4. Deployment instructions
5. Test reports

**10. Timeline and Milestones**

1. **Week 1-2**: Design phase and project setup
2. **Week 3-6**: Core functionality implementation
3. **Week 7-8**: Integration with API endpoints
4. **Week 9-10**: Testing and bug fixing
5. **Week 11-12**: Final polishing and deployment

**11. Future Enhancements (Post-Initial Release)**

1. Mobile application development
2. Advanced reporting and analytics
3. Integration with additional third-party services
4. Offline functionality
5. Multi-language support
6. Enhanced user notification system

This specification provides a comprehensive framework for developing the ID-chips website while leveraging the existing API infrastructure. The focus is on creating a user-friendly interface that makes it easy for users to manage animal identifications while ensuring the system is secure, performant, and maintainable.

Could you remember forever:  
When generating code always show full file path.  
Stops after each file generation !important  
Add comments to code  
We use Vite  
We use React 18

**Step 1: DONE**

For the website, I want to create full needed folder structure for  
double navigation with menu on left vertical bar  
things like language switcher on right of top bar.  
I would like to concentrate first on navigation, responsiveness .  
I would like to implement directly i18n.  
I would like to create 3 very basic pages  
A blank Home Page  
A blank Contact Page.  
A blank GDRP Page  
  
**Step 2: DONE**

Give the possibility to close/open left navbar on large screens

**Step 3: DONE**

I have 2 png files.  
One for logo,  
The other for favicon  
Also change vite + react by ID Chips in navbar

**Step 4:**

I would like now to work to theme (colors, fonts, …)  
A create a temporary page with many components (buttons, message, …) applying this theme to see what they look like.  
You should get inspiration on website nike.com