# FindMyPet Application - Technical Requirements Document (TRD)

## 1. Introduction

### 1.1 Purpose

This document outlines the technical requirements and architectural design for the FindMyPet application. It serves as a guide for developers implementing the system.

### 1.2 Scope

This document covers the technical implementation details of the FindMyPet web application, including frontend and backend components, data storage, security requirements, and integration points.

### 1.3 Definitions, Acronyms, and Abbreviations

* **JWT**: JSON Web Token
* **API**: Application Programming Interface
* **CRUD**: Create, Read, Update, Delete
* **UI**: User Interface
* **UX**: User Experience
* **REST**: Representational State Transfer

## 2. System Architecture

### 2.1 High-Level Architecture

The FindMyPet application follows a client-server architecture with the following main components: - Frontend: React.js based single-page application - Backend: Flask-based RESTful API server - Database: SQL database for structured data storage - Storage: File storage solution for pet images

### 2.2 Component Diagram

┌─────────────────┐ ┌───────────────────┐ ┌─────────────────┐  
│ │ │ │ │ │  
│ React Frontend │◄────┤ Flask Backend │◄────┤ SQL Database │  
│ │ │ │ │ │  
└─────────────────┘ └───────────────────┘ └─────────────────┘  
 │ │ │  
 │ │ │  
 ▼ ▼ ▼  
┌─────────────────┐ ┌───────────────────┐ ┌─────────────────┐  
│ Browser/User │ │ File Storage │ │ Map Service │  
│ Interface │ │ │ │ │  
└─────────────────┘ └───────────────────┘ └─────────────────┘

## 3. Technical Requirements

### 3.1 Frontend Requirements

#### 3.1.1 Technologies

* **Framework**: React.js
* **State Management**: React Context API
* **UI Framework**: Bootstrap 5
* **Mapping Library**: Leaflet
* **HTTP Client**: Axios
* **Build Tool**: Webpack (via Create React App)

#### 3.1.2 Components

1. **Authentication Components**
   * Login form
   * Registration form
   * Password recovery
   * JWT token management
2. **Pet Listing Components**
   * Listing creation form
   * Listing edit form
   * Listing card
   * Detail view
   * Search and filter interface
3. **Map Components**
   * Location picker (for posting listings)
   * Location viewer (for viewing listings)
   * Interactive map interface
4. **Communication Components**
   * Comment system
   * Comment form
   * Comment list
   * Chat interface
   * Contact information display
5. **Common UI Components**
   * Navigation
   * Footer
   * Modal dialogs
   * Alerts and notifications
   * Loading indicators

### 3.2 Backend Requirements

#### 3.2.1 Technologies

* **Framework**: Flask (Python)
* **Authentication**: JWT tokens
* **Database ORM**: SQLAlchemy
* **API Design**: RESTful endpoints
* **File Management**: Flask-Upload

#### 3.2.2 API Endpoints

1. **Authentication**
   * POST /api/auth/register
   * POST /api/auth/login
   * POST /api/auth/refresh
   * GET /api/auth/profile
   * PUT /api/auth/profile
2. **Pet Listings**
   * GET /api/pets (with query parameters for search/filter)
   * GET /api/pets/:id
   * POST /api/pets
   * PUT /api/pets/:id
   * DELETE /api/pets/:id
   * POST /api/pets/:id/images
3. **Comments**
   * GET /api/pets/:id/comments
   * POST /api/pets/:id/comments
   * PUT /api/comments/:id
   * DELETE /api/comments/:id
4. **Messages**
   * GET /api/messages
   * POST /api/messages
   * GET /api/messages/:id

### 3.3 Database Schema

#### 3.3.1 Users Table

* id (Primary Key)
* username
* email
* password (hashed)
* created\_at
* updated\_at

#### 3.3.2 Pets Table

* id (Primary Key)
* user\_id (Foreign Key)
* title
* type (dog, cat, etc.)
* breed
* description
* status (lost, found)
* latitude
* longitude
* address
* contact\_phone
* created\_at
* updated\_at

#### 3.3.3 Images Table

* id (Primary Key)
* pet\_id (Foreign Key)
* filename
* path
* created\_at

#### 3.3.4 Comments Table

* id (Primary Key)
* pet\_id (Foreign Key)
* user\_id (Foreign Key)
* text
* created\_at
* updated\_at

#### 3.3.5 Messages Table

* id (Primary Key)
* sender\_id (Foreign Key)
* receiver\_id (Foreign Key)
* pet\_id (Foreign Key)
* text
* read\_at
* created\_at

### 3.4 Security Requirements

#### 3.4.1 Authentication

* JWT-based token authentication
* Password hashing using bcrypt
* Token refresh mechanism
* Secure storage of tokens in HttpOnly cookies

#### 3.4.2 Authorization

* Role-based access control for admin functions
* Object-level permissions (users can only modify their own data)
* API endpoint protection

#### 3.4.3 Data Protection

* HTTPS for all communications
* Input validation and sanitization
* Protection against common vulnerabilities (XSS, CSRF, SQL Injection)

## 4. Integration Requirements

### 4.1 Map Integration

* Integration with Leaflet.js for map rendering
* Geocoding for address lookup
* Reverse geocoding for retrieving addresses from coordinates

### 4.2 Image Storage

* Secure file upload implementation
* Image optimization for web display
* Storage of images with appropriate access controls

### 4.3 External APIs

* Optional integration with social media platforms for sharing
* Email service integration for notifications

## 5. Performance Requirements

### 5.1 Response Time

* API response time: < 500ms for 95% of requests
* Page load time: < 2 seconds for initial load
* Map interaction response: < 200ms

### 5.2 Scalability

* Support for at least 1000 concurrent users
* Database optimization for large datasets
* Code structure that allows for horizontal scaling

### 5.3 Reliability

* System uptime: 99.9%
* Data backup procedures
* Error handling and logging

## 6. Deployment Requirements

### 6.1 Environment Setup

* Development environment
* Testing environment
* Production environment

### 6.2 Deployment Process

* Automated build process
* Continuous integration
* Deployment scripts

### 6.3 Hosting Requirements

* Web server requirements
* Database server requirements
* File storage requirements

## 7. Testing Requirements

### 7.1 Unit Testing

* Framework: Jest for frontend, pytest for backend
* Coverage requirements: 80% minimum code coverage

### 7.2 Integration Testing

* API endpoint testing
* Frontend-backend integration testing

### 7.3 User Acceptance Testing

* Test scenarios based on user stories
* UI/UX testing

## 8. Monitoring and Maintenance

### 8.1 Logging

* Application logs
* Error tracking
* User activity logs

### 8.2 Monitoring

* Server health monitoring
* Performance monitoring
* User activity analytics

### 8.3 Maintenance

* Regular security updates
* Bug fixing process
* Feature enhancement process

## 9. Development Guidelines

### 9.1 Coding Standards

* JavaScript/React coding standards
* Python/Flask coding standards
* Documentation requirements

### 9.2 Version Control

* Git-based version control
* Branch management strategy
* Pull request and code review process

### 9.3 Documentation

* Code documentation
* API documentation
* User documentation

## 10. Approval

This document requires approval from the technical team and project stakeholders before implementation begins.