HAPPY TAILS – A Pet adoption system

**GROUP NAME**: Murali NaiduPPradeepN

TABLE OF CONTENTS

* Overview
* Introduction
* README: Installation guidelines
* Technical specification
* Conceptual design (UML)
* Logical data
* User flow
* Admin flow
* Lessons learnt
* Future work

1. Overview

Our **Pet Shelter Adoption Management System** is a full-stack web application designed to streamline operations for animal shelters while enhancing the pet adoption experience. Built with Next.js (React) and MySQL, this system modernizes traditional shelter workflows by digitizing pet management, adoption applications, veterinary coordination, and community engagement through an intuitive interface.

2. README: Installation Guide

Requirements

1. **Software**:
   * Node.js v18+ [Download](https://nodejs.org/)
   * MySQL 8.0+ [Download](https://dev.mysql.com/downloads/)
   * Next.js 14+ (npm install next)
2. **Libraries**:

npm install bcryptjs

npm install --save-dev @types/bcryptjs

npm install mysql2 bcrypt jsonwebtoken cookie

npm install mysql2

npm install react-hook-form zustand zod @hookform/resolvers

npm install geist

npm install tailwindcss postcss autoprefixer

npm install tw-animate-css --legacy-peer-deps

Setup Instructions

1. **Database Configuration**:
   * Run mysql -u root -p < database\_dump.sql
   * Configure .env:

DB\_HOST=localhost

DB\_USER=root

DB\_PASSWORD=your\_password

DB\_NAME=PET\_SHELTER\_2

JWT\_SECRET=your\_jwt\_secret

How to run:

3. Technical Specifications

| **Component** | **Technology Stack** |
| --- | --- |
| Frontend | Next.js 14, React 18, Tailwind CSS |
| Backend | Next.js API Routes |
| Database | MySQL 8.0 |
| Security | JWT Authentication, AES-256 Encryption |
| Visualization | Chart.js, React Table |

4. Conceptual Design (UML)

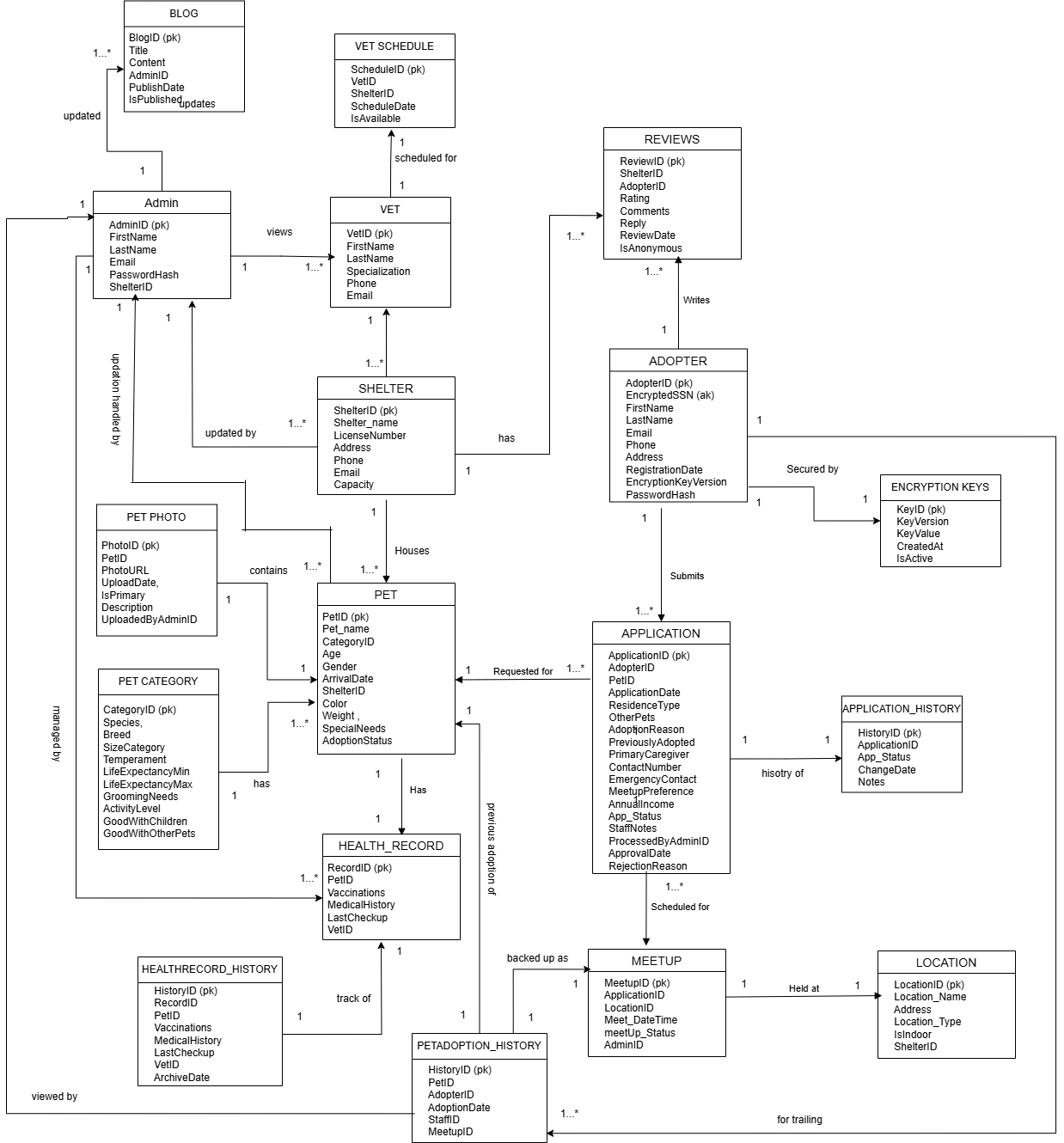
Sample key Tables

| **Table** | **Columns** | **Relationships** |
| --- | --- | --- |
| Pet | PetID (PK), CategoryID (FK)... | 1:M with HealthRecord |
| Application | AppID (PK), PetID (FK)... | M:1 with Adopter, Meetup |
| HealthRecordHistory | RecordID (PK), PetID (FK)... | Temporal versioning pattern |

Schema Features

* **3NF Compliance**: All tables decomposed to eliminate transitive dependencies
* **Indexes**: 15+ indexes on frequently queried fields (e.g., Pet.AdoptionStatus)
* **Constraints**: foreign keys with ON DELETE CASCADE/SET NULL policies

Below is the conceptual UML diagram:

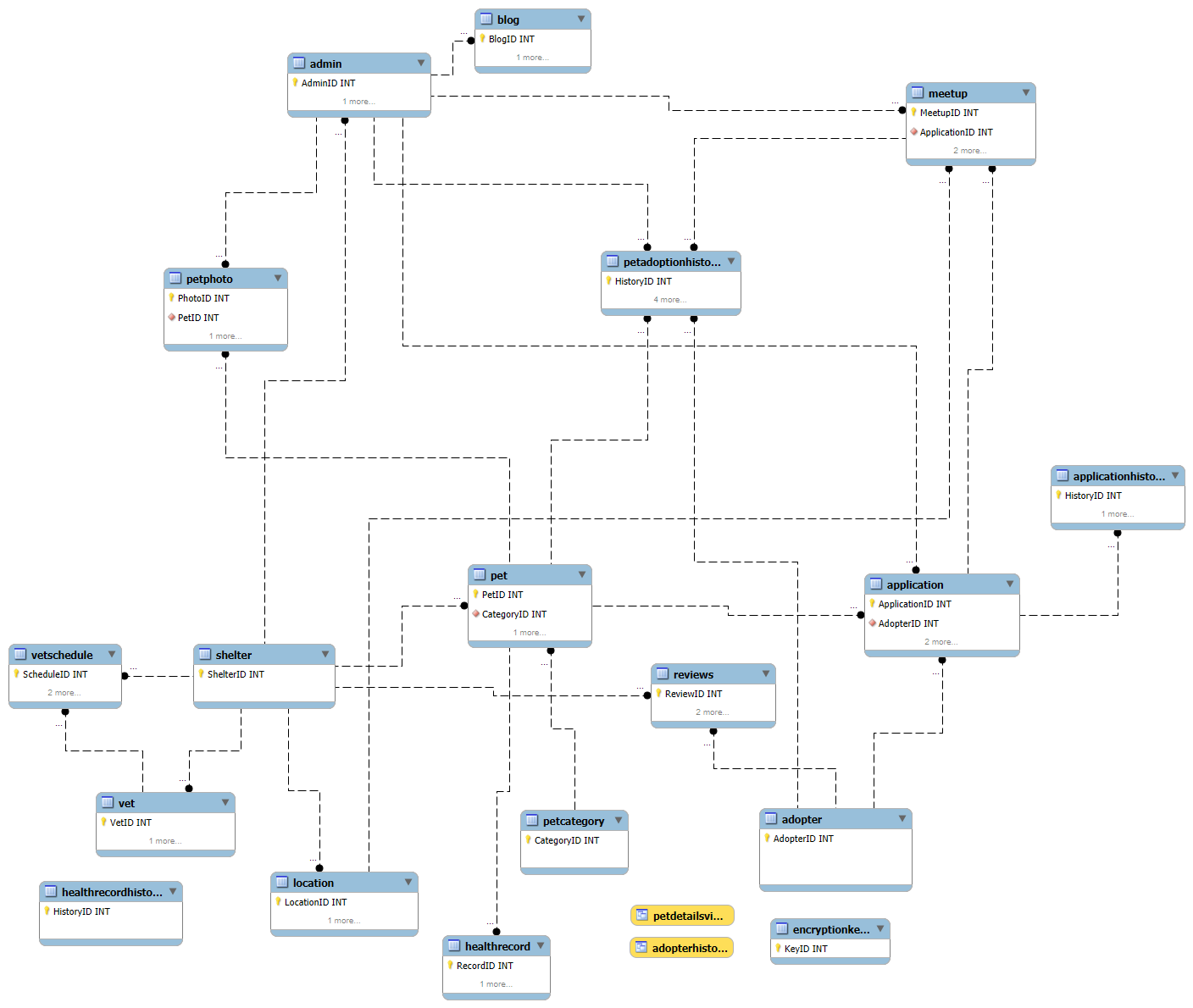


5. Logical Design

Schema Reverse engineered from the database.

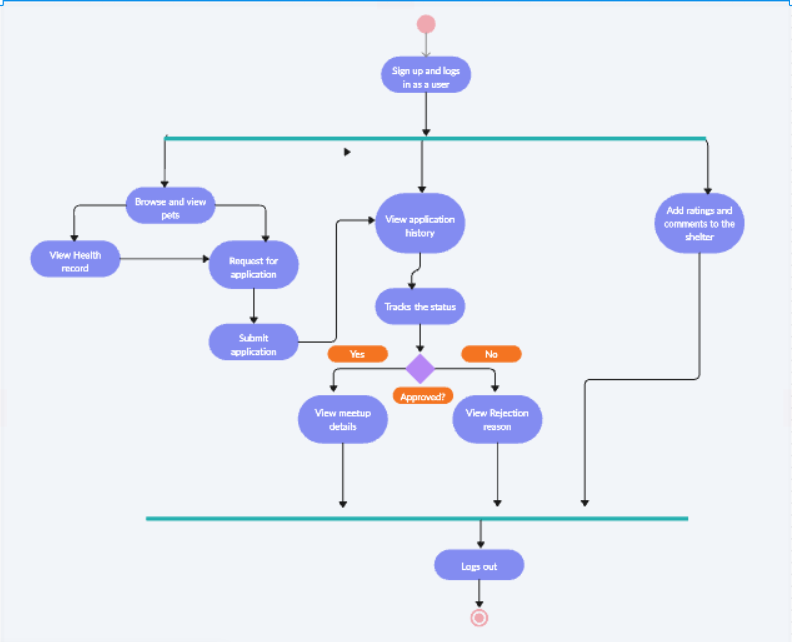
Object Notation: **Workbench (PKs and FKs only)**

Relationship Notation: **IDEF1X**



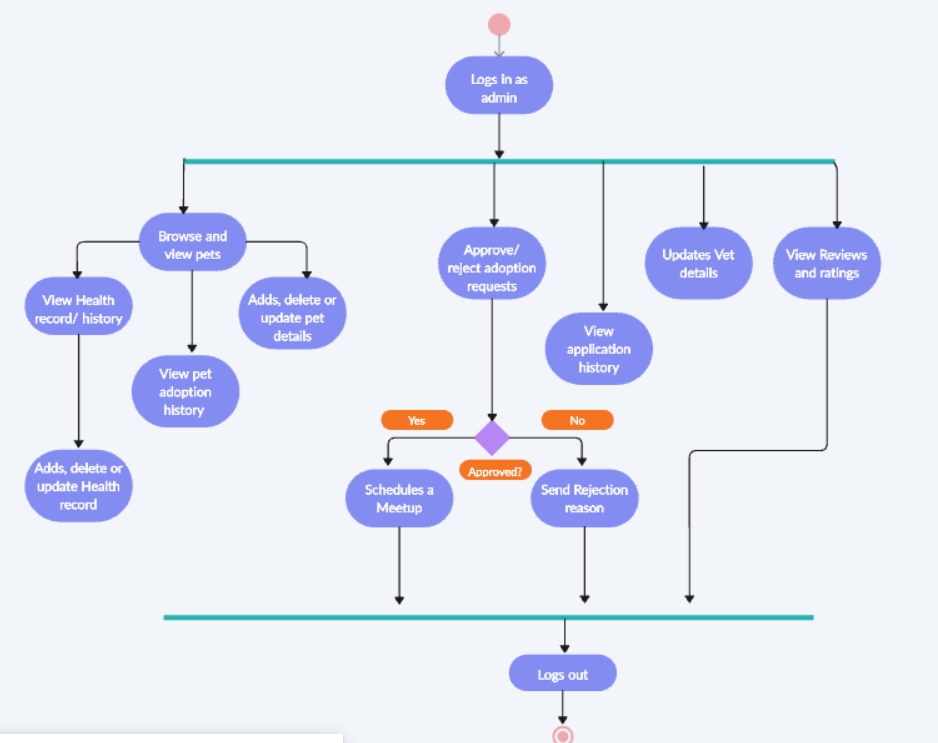
6. User Flow

Below diagram describes the flow of the program for a user:



7. Admin Flow

Below diagram describes the flow of the program for an admin:



8. Lessons Learned

Technical Expertise

1. **Database Encryption**: Implemented AES-256 for sensitive data (SSN)
2. **Complex Joins**: 12-table join for adoption analytics view
3. **Stored Procedures**: Business logic in SQL

Key Insights

* **Time Management**: Prioritized core adoption workflow before implementing reviews
* **Data Challenges**: Pet categorization required multiple normalization iterations

Alternative Approaches Considered

1. **MongoDB**: Rejected due to complex joins in reporting
2. **GraphQL**: Prototyped but chose REST for simpler caching

9. Future Work

Planned Features

| **Feature** | **Complexity** | **Business Value** |
| --- | --- | --- |
| AI Pet Matching | High | faster adoptions |
| Volunteer Management | Medium | Expand capacity |

Justification for Exclusions

* **Mobile App**: Web app responsive enough for most of the users
* **Payment Integration**: Outside project scope per requirements