INFORMATION TECHNOLOGY PROJECT (IT2080)

Activity 04_ ITP25_B7.2_168

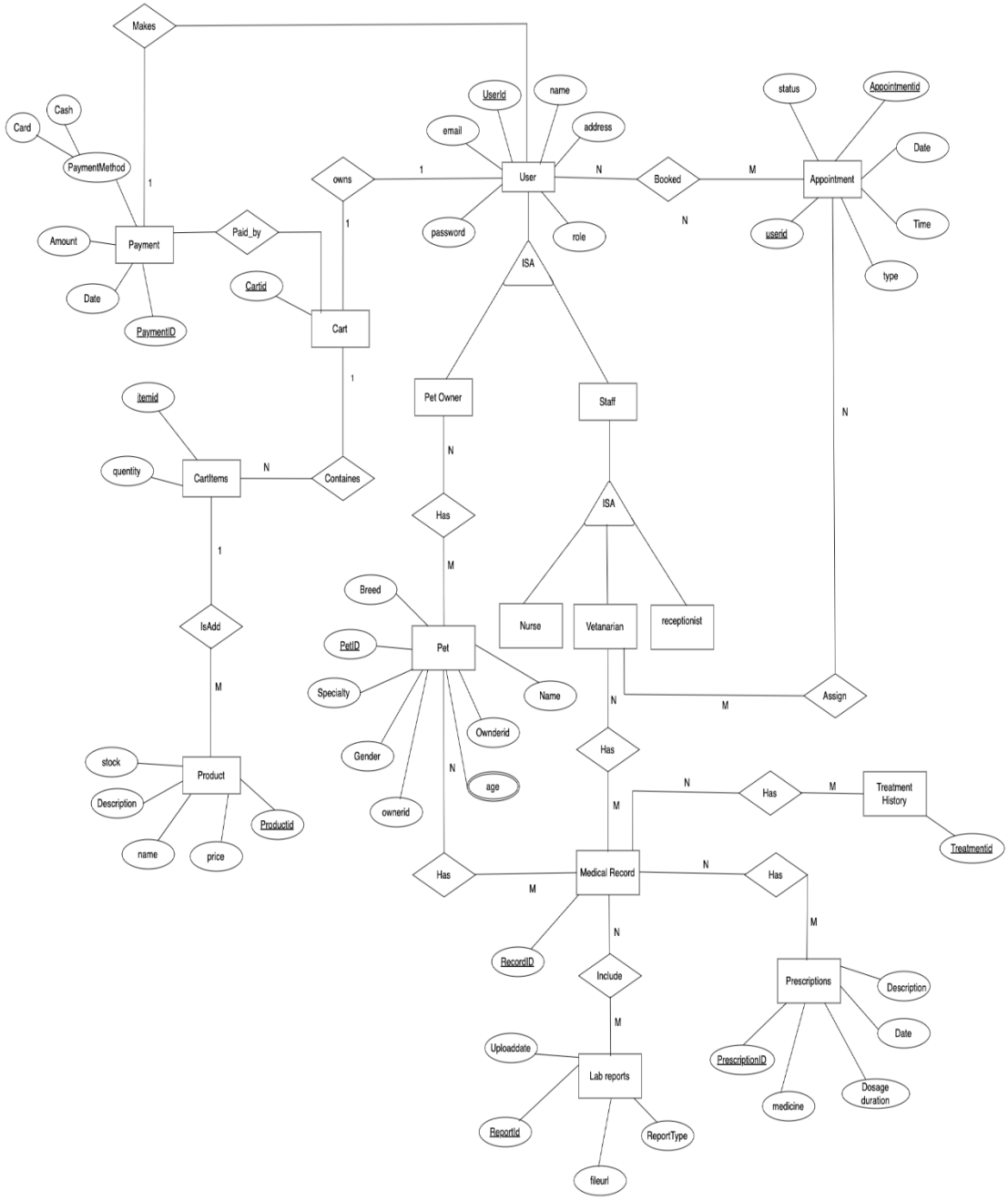
Project Title: Veterinary Management System
(PetIQ.lk)

Group: B7.2_168

Group Member Details:

	IT Number	Student Name	Student E-mail Address	Contact Number
1	IT23640948	Gamlath K. G. V. K. D	it23640948@my.sliit.lk	0702943487
2	IT23413474	Rathnayake W. P. D.D. W	it23413474@my.sliit.lk	0702979383
3	IT23631106	Kavindu J. M. R	it23631106@my.sliit.lk	0703274142
4	IT23631274	Abeysinghe A. M. B.N	it23631274@my.sliit.lk	0761260885
5	IT23594722	Wanasinghe W. M.D.T	it23594722@my.sliit.lk	0774376338

1.Design a comprehensive ER diagram to represent the data model, capturing all entities, attributes, and relationships.



2. Normalize the database schema to eliminate redundancy, improve data integrity, and ensure optimal performance.

```
const mongoose = require("mongoose");

const appointmentSchema = new mongoose.Schema({

  ownerName: { type: String, required: true },
  petName: { type: String, required: true },
  petType: { type: String, required: true },
  service: { type: String, required: true },
  price: { type: Number, required: true },
  vet: { type: String, required: true },
  date: { type: String, required: true },
  time: { type: String, required: true },
  createdAt: { type: Date, default: Date.now },

});

module.exports = mongoose.model("Appointment", appointmentSchema);
```

```
import mongoose from "mongoose";
import bcrypt from "bcryptjs";

const employeeSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    role: {
        type: String,
        required: true,
        enum: ['admin', 'veterinarian', 'nurse', 'receptionist'],
        default: 'admin'
    },
    avatarUrl: { type: String },
    isActive: { type: Boolean, default: true },
    createdAt: { type: Date, default: Date.now },
    updatedAt: { type: Date, default: Date.now }
});
```

```
import mongoose from "mongoose";
import bcrypt from "bcryptjs";

const registerSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    avatarUrl: { type: String },
    phone: { type: String },
    addressLine1: { type: String },
    addressLine2: { type: String },
    city: { type: String },
    state: { type: String },
    postalCode: { type: String },
    country: { type: String },
    dateOfBirth: { type: Date },
    isActive: { type: Boolean, default: true },
    createdAt: { type: Date, default: Date.now },
    updatedAt: { type: Date, default: Date.now }
});
```

```jsx
const mongoose = require('mongoose');

const medicalRecordSchema = new mongoose.Schema({
  petName: {type: String,required: [true, 'Pet name is required'],trim: true,maxlength: [50, 'Pet name car
  owner: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: [true, 'Owner is required']},
  ownerName: {type: String,required: [true, 'Owner name is required'],trim: true},
  petType: { type: String, required: [true, 'Pet type is required'],enum: ['Dog', 'Cat', 'Bird', 'Rabbit',
  breed: { type: String, trim: true, maxlength: [50, 'Breed cannot exceed 50 characters']},
  age: { type: String,trim: true, maxlength: [20, 'Age cannot exceed 20 characters']},
  weight: { type: String,trim: true,maxlength: [20, 'Weight cannot exceed 20 characters']},
  symptoms: {type: String,trim: true,maxlength: [1000, 'Symptoms cannot exceed 1000 characters']},
  diagnosis: { type: String, trim: true, maxlength: [1000, 'Diagnosis cannot exceed 1000 characters']},
  treatment: {type: String,trim: true,maxlength: [1000, 'Treatment cannot exceed 1000 characters']},
  prescription: {type: String,trim: true,maxlength: [1000, 'Prescription cannot exceed 1000 characters']},
  notes: {type: String, trim: true, maxlength: [2000, 'Notes cannot exceed 2000 characters']},
  visitDate: {type: Date,required: [true, 'Visit date is required'],default: Date.now},
  veterinarian: {type: mongoose.Schema.Types.ObjectId,ref: 'User',required: [true, 'Veterinarian is requir
  veterinarianName: {type: String,required: [true, 'Veterinarian name is required']},
  attachments: [{filename: String,originalName: String,mimetype: String,size: Number,
  uploadDate: { type: Date, default: Date.now}}],
  status: { type: String,enum: ['active', 'completed', 'archived'],default: 'active'}
}, {
  timestamps: true
});

// Indexes for better performance
```

```jsx
const mongoose = require("mongoose");

const AddressSchema = new mongoose.Schema(
  {
    userId:    { type: String, required: true, index: true, default: "guest" },

    firstName: { type: String, required: true, trim: true },
    lastName:  { type: String, required: true, trim: true },
    phone:     { type: String, required: true, trim: true },

    line1:     { type: String, required: true, trim: true },
    line2:     { type: String, trim: true, default: "" },
    city:      { type: String, required: true, trim: true },

    state:     { type: String, required: true, trim: true },
    postalCode:{ type: String, trim: true, default: "" },
    country:   { type: String, trim: true, default: "Sri Lanka" },
  },
  { timestamps: true }
);

module.exports = mongoose.model("Address", AddressSchema);
```

```
MongoDB connected
```

```javascript
const Card =
  mongoose.models.Card ||
  mongoose.model(
    "Card",
    new mongoose.Schema(
      {
        pmId: { type: String, unique: true, index: true },
        brand: String,
        last4: String,
        exp_month: Number,
        exp_year: Number,
        billing_name: String,
        stripe_customer: String,
      },
      { timestamps: true }
    )
  );

const Tx =
  mongoose.models.Tx ||
  mongoose.model(
    "Tx",
    new mongoose.Schema(
      {
        piId: { type: String, unique: true, index: true },
        amount: Number,
        currency: String,
        status: String,
        source: String,
        ref_id: String,
        description: String,
      },
      { timestamps: true }
    )
  );
```

3. Build the database using the chosen technology, ensuring adherence to the designed schema and incorporating all necessary constraints, indexes, and relationships.

```env
MONGO_URI = mongodb+srv://dayanwishwanathr_db_user:RtYVrZDHef1CUWuv@cluster0.23zknzn.mongodb.net/petipDB?retryWrites=true&w=majority&appName=Cluster

PORT = 5000
```

```javascript
import mongoose from'mongoose';

export const connectDB = async () => {
    try {
        const conn = await mongoose.connect(process.env.MONGO_URI);
        console.log(`MongoDB Connected: ${conn.connection.host}`);
    } catch (error) {
        console.error(`Error: ${error.message}`);
        process.exit(1); // process code 1 means exit with failure, 0 means success
    }
}
```

Admin.js > ...

```js
const mongoose = require("mongoose");

const AdminSchema = new mongoose.Schema({
  firstname: { type: String, required: true },
  lastname: { type: String, required: true },
  adminNIC: { type: String, unique: true, required: true },
  password: { type: String, required: true } // hash this before saving
}, { timestamps: true });

module.exports = mongoose.model("Admin", AdminSchema);
```

AddressModel.js

server > Model > AddressModel.js > ...

```js
const mongoose = require("mongoose");

const addressSchema = new mongoose.Schema({
  userId:    { type: String, required: true, index: true, default: "guest" },
  firstName: { type: String, required: true, trim: true },
  lastName:  { type: String, required: true, trim: true },
  phone:     { type: String, required: true, trim: true },
  line1:     { type: String, required: true, trim: true },
  line2:     { type: String, trim: true, default: "" },
  city:      { type: String, required: true, trim: true },
  state:     { type: String, required: true, trim: true },
  postalCode:{ type: String, trim: true, default: "" },
  country:   { type: String, trim: true, default: "Sri Lanka" },
}, { timestamps: true });

module.exports = mongoose.model("Address", addressSchema);
```

AppointmentModel.js

server > Model > AppointmentModel.js > ...

```js
import mongoose from "mongoose";

const appointmentSchema = new mongoose.Schema({

  ownerName: { type: String, required: true },
  petName: { type: String, required: true },
  petType: { type: String, required: true },
  service: { type: String, required: true },
  price: { type: Number, required: true },
  vet: { type: String, required: true },
  date: { type: String, required: true },
  time: { type: String, required: true },
  createdAt: { type: Date, default: Date.now },

});

export default mongoose.model("Appointment", appointmentSchema);
```

JS Employees.js ×

server > Model > JS Employees.js > ...

```javascript
const mongoose = require("mongoose");

const employeeSchema = new mongoose.Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    avatarUrl: { type: String },
});

const Employees = mongoose.model("Employees", employeeSchema);

module.exports = Employees;
```

JS MedicalRecord.js ×

server > Model > JS MedicalRecord.js > ...

```javascript
const medicalRecordSchema = new mongoose.Schema({
  status: {
    default: 'active'
  }
}, {
  timestamps: true
});

// Indexes for better performance
medicalRecordSchema.index({ owner: 1, visitDate: -1 });
medicalRecordSchema.index({ petName: 'text', symptoms: 'text', diagnosis: 'text' });
medicalRecordSchema.index({ veterinarian: 1 });

export default mongoose.model('MedicalRecord', medicalRecordSchema);
```

JS PaymentModel.js

server > Model > JS PaymentModel.js > ...

```javascript
import mongoose from "mongoose";
const cardSchema = new mongoose.Schema({
  pmId: { type: String, unique: true, index: true },
  brand: String,
  last4: String,
  exp_month: Number,
  exp_year: Number,
  billing_name: String,
  stripe_customer: String,
}, { timestamps: true });
const txSchema = new mongoose.Schema({
  piId: { type: String, unique: true, index: true },
  amount: Number,
  currency: String,
  status: String,
  source: String,
  ref_id: String,
  description: String,
}, { timestamps: true });
const Card = mongoose.model("Card", cardSchema);
export const Tx = mongoose.model("Tx", txSchema);
export default Card;
```

ProductModel.js

server > Model > ProductModel.js > productSchema > name > required

```javascript
import mongoose  from "mongoose";
const productSchema = new mongoose.Schema({
    name :{
        type: String,
        required: true},
    price:{
        type: Number,
        required: true},
    description: {
        type: String,
        required: true },
    category: {
        type: String,
        required: false,
        default: "Other"},
    stock: {
        type: Number,
        required: true,
        min: 0 },
    image: {
        type: String,
        required: true},
}, {
    timestamps: true // createdAt, updatedAt
});
const Product = mongoose.model('Product', productSchema);
export default Product;
```