🦕 Dino Reserve - Restaurant Table Reservation System

A full-stack web application for managing restaurant table reservations with a playful dinosaur theme.

🎯 Features

- **5 Dinosaur-themed restaurants** with unique mascots
- **25 tables per restaurant** with dynamic status (available/reserved)
- **Real-time reservation management** (create, update, cancel)
- **Manager-facing interface** with intuitive UI
- **Cute dino-themed design** with emojis and pastel colors

🧬 Tech Stack

Frontend

- **React 18** with TypeScript
- **Tailwind CSS** for styling
- **Radix UI** for accessible components
- **Vite** for build tooling

Backend

- **FastAPI** (Python 3.9+)
- **SQLAlchemy** ORM
- **PostgreSQL** database
- **Uvicorn** ASGI server

📋 Prerequisites

- **Node.js** 18+ and npm
- **Python** 3.9+
- **PostgreSQL** 14+

🚀 Quick Start

1. Database Setup

```bash
# Install PostgreSQL (if not installed)
# macOS: brew install postgresql@14
# Ubuntu: sudo apt install postgresql postgresql-contrib

# Start PostgreSQL service
# macOS: brew services start postgresql@14
# Ubuntu: sudo systemctl start postgresql

# Create database
createdb dinoreserve

# Create user (optional)
psql -d dinoreserve -c "CREATE USER dinouser WITH PASSWORD 'dinopass123';"
psql -d dinoreserve -c "GRANT ALL PRIVILEGES ON DATABASE dinoreserve TO dinouser;"
```

2. Backend Setup

```bash
```

```bash
# Navigate to backend directory
cd backend

# Create virtual environment
python -m venv venv

# Activate virtual environment
# macOS/Linux:
source venv/bin/activate
# Windows:
venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Update database URL in main.py if needed
# DATABASE_URL = "postgresql://dinouser:dinopass123@localhost/dinoreserve"

# Run the FastAPI server
python main.py

# Server will start at http://localhost:8000
# API docs available at http://localhost:8000/docs
```

3. Frontend Setup

```bash
# Navigate to frontend directory
cd frontend

# Install dependencies
npm install

# Start development server
npm run dev

# App will open at http://localhost:5173
```

📁 Project Structure

```
dino-reserve/
├── backend/
│   ├── main.py              # FastAPI application
│   ├── requirements.txt     # Python dependencies
│   └── README.md
│
├── frontend/
│   ├── src/
│   │   ├── components/
│   │   │   ├── ui/          # Shadcn UI components
│   │   │   ├── LoginPage.tsx
│   │   │   ├── RestaurantSelection.tsx
│   │   │   └── TableLayout.tsx
│   │   ├── App.tsx          # Main app component
│   │   ├── main.tsx         # Entry point
│   │   └── index.css        # Tailwind styles
│   ├── package.json
│   └── vite.config.ts
```

```
└── README.md
```

🗄️ Database Schema

Tables

restaurants

- `id` (Primary Key)
- `name` (Unique)
- `location`
- `dino_type` (trex, bronto, raptor, stego, ptero)

tables

- `id` (Primary Key)
- `restaurant_id` (Foreign Key)
- `table_number` (1-25)
- `capacity` (2, 4, or 6 people)

reservations

- `id` (Primary Key)
- `table_id` (Foreign Key)
- `customer_name`
- `phone`
- `party_size`
- `reservation_time`
- `status` (reserved/cancelled)
- `created_at`

🔌 API Endpoints

Restaurants

- `GET /restaurants` - List all restaurants
- `GET /restaurants/{id}` - Get restaurant details
- `GET /restaurants/{id}/tables` - Get all tables with status

Reservations

- `POST /reservations` - Create new reservation
- `PUT /reservations/{id}` - Update reservation
- `DELETE /reservations/{id}` - Cancel reservation
- `GET /reservations` - List all reservations (with filters)

🎨 Dino Theme

Restaurant Types & Colors

- **T-Rex Tavern** 🦖 - Red/Orange gradient
- **Bronto Bistro** 🦕 - Green/Emerald gradient
- **Raptor Restaurant** 🦎 - Orange/Amber gradient
- **Stego Steakhouse** 🦕 - Yellow/Orange gradient
- **Pterodactyl Pub** 🐦 - Purple/Pink gradient

UI Elements

- **Available Table** → Hungry Dino 🦕
- **Reserved Table** → Dino Eating 🍴🦖
- **Confirm Button** → "Feed Dino" 🦖
- **Sync Indicator** → "Dino Cave synced" with green pulse

🔐 Authentication

Current implementation uses simple client-side authentication. For production:

1. Implement JWT tokens
2. Add user roles (manager, admin)
3. Secure API endpoints with authentication middleware
4. Use environment variables for secrets

🧪 Testing

Backend

```bash
# Install pytest
pip install pytest pytest-asyncio httpx

# Run tests
pytest
```

Frontend

```bash
# Run tests (if configured)
npm test
```

🚢 Deployment

Backend (FastAPI)

```bash
# Using Gunicorn + Uvicorn workers
pip install gunicorn
gunicorn main:app -w 4 -k uvicorn.workers.UvicornWorker --bind 0.0.0.0:8000
```

Frontend (React)

```bash
# Build for production
npm run build

# Serve with any static server
# Output will be in dist/
```

Environment Variables

Create `.env` files for configuration:

Backend (.env)

```
DATABASE_URL=postgresql://user:password@localhost/dinoreserve
SECRET_KEY=your-secret-key-here
CORS_ORIGINS=http://localhost:5173,https://yourdomain.com
```

Frontend (.env)

```
VITE_API_URL=http://localhost:8000
```

📝 Seed Data

The application automatically seeds 5 restaurants with 25 tables each on first startup. No manual seeding required!

🦕 Troubleshooting

Database Connection Issues

- Verify PostgreSQL is running: `pg_isready`
- Check connection string in `main.py`
- Ensure database exists: `psql -l`

CORS Errors

- Update `allow_origins` in FastAPI CORS middleware
- Check frontend API URL configuration

Table Status Not Updating

- Verify reservation times are in the future
- Check backend console for errors
- Refresh the page to sync with backend

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Commit your changes
4. Push to the branch
5. Open a Pull Request

📄 License

MIT License - feel free to use this project for learning and commercial purposes.

🦕 Credits

Built with ❤️ and 🦕 by the Dino Reserve team!

Happy Dino Management! 🦕🍴