Adventurous Travel Express

A modern, full-stack travel booking platform inspired by booking.com, featuring tours, flights, parks, and taxi services.

Features

- **User Authentication**: Secure JWT-based authentication with role-based access control
- **Tour Booking**: Browse and book guided tours with ratings and reviews
- **Flight Booking**: Search flights with seat selection and class preferences
- **National Parks**: Explore and book national park visits with activities
- **Taxi Services**: Book taxi rides with vehicle selection and pricing
- **Payment Processing**: Secure payment handling with multiple payment methods
- **User Dashboard**: Manage bookings, view history, and track payments

Tech Stack

Backend

- Node.js & Express.js
- MongoDB with Mongoose
- JWT Authentication
- Bcrypt for password hashing
- CORS enabled

Frontend

- HTML5 & CSS3
- Vanilla JavaScript
- Font Awesome icons
- Responsive design

Installation

Prerequisites

- Node.js (v14 or higher)
- MongoDB (local or cloud instance)

Setup

1. **Clone the repository**

   ```
   git clone <repository-url>
   cd adventurous-travel-express
   ```

2. **Install dependencies**

   ```
   npm install
   ```

3. **Configure environment variables**

   - Copy `.env.example` to `.env`
   - Update the following variables:
     - `MONGO_URI`: Your MongoDB connection string
     - `JWT_SECRET`: A secure random string for JWT signing
     - Other optional configurations as needed

4. **Start MongoDB**

   - If using local MongoDB:

     ```
     mongod
     ```

   - Or use MongoDB Atlas cloud connection

5. **Run the application**

   Development mode (with auto-restart):

   ```
   npm run dev
   ```

   Production mode:

   ```
   npm start
   ```

6. **Access the application**

   - Backend API: `http://localhost:5000`
   - Frontend: Open `frontend/index.html` in a browser or serve with a local server

API Endpoints

Authentication

- `POST /api/auth/signup` - Register new user
- `POST /api/auth/login` - Login user
- `GET /api/auth/me` - Get current user (protected)

Tours

- `GET /api/tours` - Get all tours
- `GET /api/tours/:id` - Get single tour
- `POST /api/tours` - Create tour (admin/operator)
- `PUT /api/tours/:id` - Update tour (admin/operator)
- `DELETE /api/tours/:id` - Delete tour (admin/operator)

Flights

- `GET /api/flights` - Get all flights
- `GET /api/flights/:id` - Get single flight
- `POST /api/flights` - Create flight (admin/operator)
- `PUT /api/flights/:id` - Update flight (admin/operator)
- `DELETE /api/flights/:id` - Delete flight (admin/operator)

Parks

- `GET /api/parks` - Get all parks
- `GET /api/parks/:id` - Get single park
- `POST /api/parks` - Create park (admin/operator)
- `PUT /api/parks/:id` - Update park (admin/operator)
- `DELETE /api/parks/:id` - Delete park (admin/operator)

Taxis

- `GET /api/taxis` - Get all taxis
- `GET /api/taxis/:id` - Get single taxi
- `POST /api/taxis` - Create taxi (admin/operator)
- `POST /api/taxis/:id/book` - Book a taxi (authenticated)

Bookings

- `GET /api/bookings` - Get user bookings (authenticated)
- `GET /api/bookings/:id` - Get single booking (authenticated)
- `POST /api/bookings` - Create booking (authenticated)
- `PUT /api/bookings/:id` - Update booking (authenticated)
- `PUT /api/bookings/:id/cancel` - Cancel booking (authenticated)

Payments

- `GET /api/payments` - Get user payments (authenticated)
- `GET /api/payments/:id` - Get single payment (authenticated)
- `POST /api/payments` - Create payment (authenticated)

Users

- `GET /api/users` - Get all users (admin)
- `GET /api/users/:id` - Get single user (admin)
- `PUT /api/users/:id` - Update user (admin)
- `DELETE /api/users/:id` - Delete user (admin)

Project Structure

```
adventurous-travel-express/
├── backend/
│   ├── config/
│   │   └── db.js              # Database configuration
│   ├── controllers/           # Route controllers
│   ├── middleware/            # Custom middleware
│   ├── models/                # Mongoose models
│   ├── routes/                # API routes
│   ├── services/              # Business logic services
│   ├── utils/                 # Utility functions
│   └── server.js              # App entry point
├── frontend/
│   ├── assets/                # Images and media
│   ├── css/
│   │   └── style.css          # Styles
│   ├── js/                    # Frontend JavaScript
│   └── *.html                 # HTML pages
├── .env                       # Environment variables
├── .env.example               # Environment template
├── package.json               # Dependencies
└── README.md                  # Documentation
```

User Roles

- **User**: Can browse and book tours, flights, parks, and taxis
- **Operator**: Can create and manage tours, flights, parks, and taxis
- **Admin**: Full access to all features and user management

Development

Adding Sample Data

You can add sample data through the API endpoints or directly through MongoDB:

```
// Example: Create a tour
POST /api/tours
{
  "name": "Amazon Rainforest Adventure",
  "description": "Explore the heart of the Amazon",
  "price": 1299,
  "duration": 7,
  "maxGroupSize": 15,
  "difficulty": "moderate",
  "location": {
    "type": "Point",
    "coordinates": [-60.0217, -3.4653],
    "city": "Manaus",
    "country": "Brazil"
  }
}
```

Running Tests

(Tests to be implemented)

```
npm test
```

Security

- Passwords are hashed using bcrypt
- JWT tokens for secure authentication
- Role-based access control
- Environment variables for sensitive data
- CORS enabled for cross-origin requests

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

Support

For issues and questions, please open an issue on GitHub or contact the development team.

Future Enhancements

- Email notifications for bookings
- Payment gateway integration (Stripe)
- Real-time availability updates
- User reviews and ratings
- Advanced search filters
- Mobile application
- Admin dashboard
- Analytics and reporting