AdventureConnect MVP

A curated marketplace platform connecting specialized travel service providers with travelers seeking unique, authentic experiences.

🚀 Features

Phase 1 MVP Features

For Providers

- ✅ Provider registration and profile management
- ✅ Create and manage trip listings
- ✅ Handle booking inquiries
- ✅ Dashboard with booking analytics
- ✅ Image upload for trips and profiles

For Travelers

- ✅ Browse and search trips
- ✅ Filter by destination, price, dates, and activity type
- ✅ Send booking inquiries
- ✅ Manage bookings
- ✅ Traveler dashboard

Platform Features

- ✅ Secure authentication (JWT)
- ✅ Email notifications
- ✅ Responsive design with Tailwind CSS
- ✅ PostgreSQL database
- ✅ Image upload and storage
- ✅ Amadeus API integration ready

🛠️ Tech Stack

Backend

- Node.js with Express.js
- PostgreSQL database
- JWT authentication

- Multer for file uploads
- Nodemailer for emails
- Amadeus SDK for flight/hotel data

Frontend

- React 18
- React Router v6
- React Query for data fetching
- React Hook Form
- Tailwind CSS
- Axios for API calls

📋 Prerequisites

- Node.js (v14 or higher)
- PostgreSQL (v12 or higher)
- npm or yarn
- Git

🏃 Quick Start

1. Clone the repository

```bash
git clone <repository-url>
cd adventureconnect-mvp
```

2. Set up the Backend

```bash
cd backend
npm install
```

Create a `.env` file in the backend directory:

env

```env
# Database
DATABASE_URL=postgresql://postgres:password@localhost:5432/adventureconnect
DB_HOST=localhost
DB_PORT=5432
DB_NAME=adventureconnect
DB_USER=postgres
DB_PASSWORD=password

# Server
PORT=5000
NODE_ENV=development

# JWT
JWT_SECRET=your-super-secret-jwt-key-change-this-in-production
JWT_EXPIRE=7d

# Email (Using Gmail as example)
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-app-password

# Amadeus API
AMADEUS_CLIENT_ID=Bd76Zxmr3DtsAgSCNVhRlgCzzFDROM07
AMADEUS_CLIENT_SECRET=Onw33473vAI1CTHS
AMADEUS_BASE_URL=https://test.api.amadeus.com

# Frontend URL
FRONTEND_URL=http://localhost:3000

# File Upload
MAX_FILE_SIZE=5242880
UPLOAD_PATH=uploads/
```

Create the database:

bash

```bash
createdb adventureconnect
```

Run database migrations:

bash

```bash
npm run db:migrate
```

Start the backend server:

bash

```bash
npm run dev
```

3. Set up the Frontend

In a new terminal:

bash

```bash
cd frontend
npm install
```

Create a `.env` file (optional, as proxy is configured):

env

```env
REACT_APP_API_URL=http://localhost:5000/api
```

Install Tailwind CSS dependencies (if not already installed):

bash

```bash
npm install -D tailwindcss postcss autoprefixer
npm install @tailwindcss/forms @tailwindcss/typography @tailwindcss/aspect-ratio
```

Start the frontend development server:

bash

```bash
npm start
```

4. Access the Application

- Frontend: http://localhost:3000
- Backend API: http://localhost:5000/api
- API Health Check: http://localhost:5000/api/health

🐳 Docker Setup (Alternative)

If you prefer using Docker:

```bash
docker-compose up -d
```

This will start:

- PostgreSQL database on port 5432
- Redis cache on port 6379
- Backend API on port 5000
- Frontend on port 3000

📚 API Documentation

Authentication Endpoints

```
POST   /api/auth/register    - Register new user
POST   /api/auth/login       - Login user
GET    /api/auth/me          - Get current user
```

Provider Endpoints

```
GET    /api/providers        - List all providers
GET    /api/providers/:id    - Get provider profile
PUT    /api/providers/profile - Update provider profile (auth required)
```

Trip Endpoints

```
GET    /api/trips            - List all trips
GET    /api/trips/:id        - Get trip details
POST   /api/trips            - Create new trip (provider only)
PUT    /api/trips/:id        - Update trip (provider only)
GET    /api/trips/provider/my-trips - Get provider's trips
POST   /api/trips/:id/dates  - Add trip dates (provider only)
```

Booking Endpoints

```
POST   /api/bookings/inquiry     - Create booking inquiry (traveler only)
GET    /api/bookings/traveler    - Get traveler's bookings
GET    /api/bookings/provider    - Get provider's bookings
GET    /api/bookings/:id         - Get booking details
PUT    /api/bookings/:id/status  - Update booking status (provider only)
```

Amadeus Integration (Future)

```
GET    /api/amadeus/flights/search       - Search flights
GET    /api/amadeus/hotels/search        - Search hotels
GET    /api/amadeus/hotels/:hotelId      - Get hotel details
POST   /api/amadeus/flights/confirm-price - Confirm flight price
GET    /api/amadeus/locations/search     - Search locations
```

🧪 Testing the Application

1. Register as a Provider

1. Go to http://localhost:3000/register
2. Select "Offer travel services/experiences (Provider)"
3. Fill in the registration form
4. You'll be redirected to complete your profile

2. Create a Trip

1. Click "Create Trip" in the navbar or dashboard
2. Fill in all required fields
3. Add images, dates, and itinerary
4. Publish the trip

3. Register as a Traveler

1. Open an incognito window or different browser
2. Register as a traveler
3. Browse trips at http://localhost:3000/trips
4. Click on a trip and send a booking inquiry

4. Manage Bookings

1. As a provider, check your bookings dashboard
2. Respond to the inquiry (confirm or decline)
3. As a traveler, check your booking status

📁 Project Structure

```
adventureconnect-mvp/
├── backend/
│   ├── src/
│   │   ├── config/         # Database, auth, Amadeus config
│   │   ├── controllers/    # Route controllers
│   │   ├── middleware/     # Auth, upload middleware
│   │   ├── models/         # Database models
│   │   ├── routes/         # API routes
│   │   ├── utils/          # Email utilities
│   │   └── app.js          # Express app setup
│   ├── uploads/            # Uploaded images
│   ├── server.js           # Server entry point
│   └── package.json
├── frontend/
│   ├── src/
│   │   ├── components/     # Reusable components
│   │   ├── contexts/       # React contexts (Auth)
│   │   ├── pages/          # Page components
│   │   ├── services/       # API services
│   │   ├── App.js          # Main app component
│   │   └── index.js        # Entry point
│   ├── public/
│   └── package.json
├── docker-compose.yml
└── README.md
```

🔒 Security Considerations

- Change the JWT secret in production
- Use environment variables for sensitive data
- Enable HTTPS in production
- Implement rate limiting (already configured)
- Validate and sanitize all inputs
- Use prepared statements for database queries

🚀 Deployment

Backend Deployment (Heroku example)

```bash
cd backend
heroku create adventureconnect-api
heroku addons:create heroku-postgresql:hobby-dev
heroku config:set JWT_SECRET=your-production-secret
git push heroku main
```

Frontend Deployment (Netlify example)

```bash
cd frontend
npm run build
netlify deploy --prod --dir=build
```

Update the `REACT_APP_API_URL` environment variable to point to your production API.

📈 Future Enhancements (Phase 2+)

- [ ] Real-time chat with Socket.IO
- [ ] Advanced trip customization engine
- [ ] AI-powered trip recommendations
- [ ] Payment processing integration
- [ ] Review and rating system
- [ ] Mobile applications
- [ ] Multi-currency support
- [ ] Advanced analytics dashboard

🤝 Contributing

1. Fork the repository
2. Create your feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

📝 License

This project is licensed under the MIT License.

🆘 Troubleshooting

Common Issues

1. **Database connection error**
   - Ensure PostgreSQL is running
   - Check database credentials in `.env`
   - Run `createdb adventureconnect` if database doesn't exist
2. **Email not sending**
   - For Gmail, enable 2FA and create an app password
   - Check EMAIL_USER and EMAIL_PASS in `.env`
3. **Images not uploading**
   - Ensure `uploads/` directory exists in backend
   - Check file permissions
4. **Amadeus API errors**
   - Verify API credentials are correct
   - Check if you're using test environment credentials

📧 Support

For support, email support@adventureconnect.com or create an issue in the repository.

Built with ❤️ for the adventure travel community