🚀 Deployment Guide for BookIt

This guide covers deploying both the backend and frontend of the BookIt application to production.

📋 Pre-Deployment Checklist

- [ ] Code is committed to GitHub
- [ ] All tests pass locally
- [ ] Environment variables are documented
- [ ] Database schema is finalized
- [ ] API endpoints are tested

🗄️ Backend Deployment (Render/Railway)

Option 1: Render.com (Recommended)

Step 1: Create PostgreSQL Database

1. Go to Render Dashboard
2. Click "New +" → "PostgreSQL"
3. Configure:
   - **Name**: `bookit-db`
   - **Database**: `bookit`
   - **User**: `bookit`
   - **Region**: Choose closest to your users
   - **Plan**: Free tier is fine for testing
4. Click "Create Database"
5. **Copy the Internal Database URL** (starts with `postgresql://`)

Step 2: Deploy Backend Service

1. Click "New +" → "Web Service"
2. Connect your GitHub repository
3. Configure:
   - **Name**: `bookit-backend`
   - **Root Directory**: `backend`

- **Environment**: `Node`
- **Region**: Same as database
- **Branch**: `main`
- **Build Command**: `npm install && npm run build`
- **Start Command**: `npm start`
- **Plan**: Free tier

4. Add Environment Variables:

```
DATABASE_URL = [paste internal database URL from step 1]
NODE_ENV = production
```

5. Click "Create Web Service"

6. Wait for deployment (5-10 minutes)

7. **Copy your backend URL**: `https://bookit-backend.onrender.com`

Step 3: Test Backend

```bash
# Test health endpoint
curl https://bookit-backend.onrender.com/health

# Test experiences endpoint
curl https://bookit-backend.onrender.com/api/experiences
```

Option 2: Railway.app

1. Go to Railway
2. Click "New Project" → "Deploy from GitHub repo"
3. Select your repository
4. Add PostgreSQL plugin
5. Set environment variables:

```
DATABASE_URL = ${{Postgres.DATABASE_URL}}
NODE_ENV = production
```

6. Configure build:
   - **Root Directory**: `backend`
   - **Build Command**: `npm install && npm run build`
   - **Start Command**: `npm start`

🌐 Frontend Deployment (Vercel)

Step 1: Prepare Frontend

1. Update `frontend/.env.production`:

```env
VITE_API_URL=https://bookit-backend.onrender.com/api
```

2. Test production build locally:

```bash
cd frontend
npm run build
npm run preview
```

Step 2: Deploy to Vercel

Method 1: Vercel CLI

1. Install Vercel CLI:

```bash
npm install -g vercel
```

2. Login:

```bash
vercel login
```

3. Deploy from frontend directory:

```bash
```

```
cd frontend
vercel --prod
```

4. Set environment variable in Vercel dashboard:
   - Go to Project Settings → Environment Variables
   - Add: `VITE_API_URL` = `https://bookit-backend.onrender.com/api`

Method 2: Vercel Dashboard

1. Go to Vercel Dashboard
2. Click "Add New" → "Project"
3. Import your GitHub repository
4. Configure:
   - **Framework Preset**: Vite
   - **Root Directory**: `frontend`
   - **Build Command**: `npm run build`
   - **Output Directory**: `dist`
5. Add Environment Variable:

```
VITE_API_URL = https://bookit-backend.onrender.com/api
```

6. Click "Deploy"

Step 3: Test Frontend

Visit your Vercel URL (e.g., `https://bookit.vercel.app`) and test:

- Browse experiences
- Select dates and slots
- Complete a booking
- Apply promo codes

🐳 Alternative: Docker Deployment

Step 1: Build Docker Images

```bash
# Build backend
docker build -f Dockerfile.backend -t bookit-backend .

# Build frontend
docker build -f Dockerfile.frontend -t bookit-frontend .
```

Step 2: Deploy with Docker Compose

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

This will start:

- PostgreSQL on port 5432
- Backend on port 5000
- Frontend on port 3000

Step 3: Deploy to AWS ECS / Google Cloud Run

Follow your cloud provider's Docker deployment guide.

☁️ Alternative Backend Hosting

Heroku

```bash
```

```
# Login to Heroku
heroku login

# Create app
heroku create bookit-backend

# Add PostgreSQL
heroku addons:create heroku-postgresql:mini

# Deploy
cd backend
git push heroku main

# Set environment variables
heroku config:set NODE_ENV=production
```

DigitalOcean App Platform

1. Connect GitHub repository
2. Select "Node.js" as component type
3. Add PostgreSQL database
4. Configure build commands
5. Deploy

🔍 Post-Deployment Verification

Backend Health Check

```bash
# Health endpoint
curl https://your-backend-url.com/health

# Get experiences
curl https://your-backend-url.com/api/experiences

# Get specific experience
curl https://your-backend-url.com/api/experiences/1

# Get slots
curl "https://your-backend-url.com/api/experiences/1/slots?date=2025-11-10"
```

Frontend Testing

1. Open browser DevTools → Network tab
2. Navigate through the app
3. Verify all API calls succeed (200 status)
4. Test booking flow end-to-end
5. Verify promo codes work

Database Verification

```bash
# Connect to production database
psql $DATABASE_URL

# Check tables
\dt

# Count experiences
SELECT COUNT(*) FROM experiences;

# Check recent bookings
SELECT * FROM bookings ORDER BY booking_date DESC LIMIT 5;
```

🛠️ Troubleshooting

Backend Issues

Issue: Database connection fails

```bash
# Solution: Verify DATABASE_URL format
# Should be: postgresql://user:password@host:port/database
# Check SSL settings for production
```

Issue: CORS errors

```javascript
```

```
// Solution: Update CORS settings in server.ts
app.use(cors({
  origin: ['https://your-frontend-url.com'],
  credentials: true
}));
```

Issue: Build fails on Render

```bash
# Solution: Check Node version
# Add to package.json:
"engines": {
  "node": ">=18.0.0"
}
```

Frontend Issues

Issue: API calls fail with CORS

```typescript
// Solution: Update API base URL in api.ts
const API_BASE_URL = import.meta.env.VITE_API_URL || 'http://localhost:5000/api';
```

Issue: Environment variables not loading

```bash
# Solution: Rebuild and redeploy
vercel --prod --force
```

Issue: 404 on routes

```json
// Add to vercel.json
{
  "rewrites": [
    { "source": "/(.*)", "destination": "/index.html" }
  ]
}
```

📊 Monitoring

Render Monitoring

- View logs: Dashboard → Service → Logs
- Metrics: CPU, Memory, Request count
- Alerts: Set up email notifications

Vercel Analytics

```bash
# Install Vercel Analytics
npm install @vercel/analytics
```

```typescript
// Add to main.tsx
import { Analytics } from '@vercel/analytics/react';

<Analytics />
```

🔒 Security Hardening

Environment Variables

Never commit `.env` files. Use:

- Render: Dashboard → Environment Variables
- Vercel: Project Settings → Environment Variables

Database Security

```sql
-- Create read-only user for analytics
CREATE USER analytics WITH PASSWORD 'secure_password';
GRANT SELECT ON ALL TABLES IN SCHEMA public TO analytics;
```

Rate Limiting

```typescript
```

```
// Add to server.ts
import rateLimit from 'express-rate-limit';

const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100 // limit each IP to 100 requests per windowMs
});

app.use('/api/', limiter);
```

📈 Performance Optimization

Backend

```typescript
// Add caching
import NodeCache from 'node-cache';
const cache = new NodeCache({ stdTTL: 600 });

app.get('/api/experiences', async (req, res) => {
  const cached = cache.get('experiences');
  if (cached) return res.json(cached);

  const experiences = await getExperiences();
  cache.set('experiences', experiences);
  res.json(experiences);
});
```

Frontend

```typescript
// Code splitting
const CheckoutPage = lazy(() => import('./pages/CheckoutPage'));

// Use Suspense
<Suspense fallback={<Loading />}>
  <CheckoutPage />
</Suspense>
```

🚀 Continuous Deployment

GitHub Actions

Create `.github/workflows/deploy.yml`:

```yaml
name: Deploy

on:
  push:
    branches: [main]

jobs:
  deploy-backend:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: npm install
      - run: npm run build
      # Add deployment steps

  deploy-frontend:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: npm install
      - run: npm run build
      # Add deployment steps
```

📝 Final Checklist

- [ ] Backend deployed and accessible
- [ ] Frontend deployed and accessible
- [ ] Database migrations run successfully
- [ ] Sample data populated
- [ ] All API endpoints working
- [ ] CORS configured correctly
- [ ] Environment variables set
- [ ] SSL certificates active (HTTPS)
- [ ] Error logging configured

- [ ] Monitoring set up
- [ ] Domain name configured (optional)
- [ ] Documentation updated with live URLs

🎉 You're Live!

Your application is now deployed! Share these URLs:

- **Frontend**: https://bookit.vercel.app
- **Backend API**: https://bookit-backend.onrender.com
- **GitHub Repo**: https://github.com/username/bookit

Need Help? Check the main README.md or open an issue on GitHub.