🚀 Render Deployment Guide for AWIBI MEDTECH Backend

📋 Prerequisites

- GitHub account with AWIBI MEDTECH repository
- Render account (free tier is sufficient)
- MongoDB Atlas account (recommended for production)

🔧 Step-by-Step Deployment

Step 1: Prepare MongoDB Database

1. **Create MongoDB Atlas Cluster**
2. Go to https://cloud.mongodb.com
3. Create a new cluster (free tier is fine)
4. Create a database user with read/write permissions
5. Whitelist all IP addresses (0.0.0.0/0) for Render access
6. **Get Connection String** `mongodb+srv://username:password@cluster.mongodb.net/awibi-medtech?retryWrites=true&w=majority`

Step 2: Deploy to Render

1. **Go to Render Dashboard**
2. Visit https://render.com
3. Sign in with your GitHub account
4. **Create New Web Service**
5. Click "New" → "Web Service"
6. Connect your GitHub repository
7. Select the AWIBI MEDTECH repository
8. **Configure Service Settings**
9. **Name**: `awibi-medtech-backend`
10. **Environment**: `Node`
11. **Region**: Choose closest to your users
12. **Branch**: `main`

13. **Root Directory**: `backend`
14. **Build Command**: `npm install`
15. **Start Command**: `npm start`

Step 3: Set Environment Variables

In the Render dashboard, go to your service → Environment:

```
# Server Configuration
NODE_ENV=production
PORT=10000

# Database Configuration
MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.net/awibi-medtech?retryWrites=true&w=majority

# JWT Configuration
JWT_SECRET=awibi-medtech-super-secure-jwt-secret-2025-change-this-in-production
JWT_EXPIRE=7d

# Google OAuth Configuration
GOOGLE_CLIENT_ID=your-google-oauth-client-id
GOOGLE_CLIENT_SECRET=your-google-oauth-client-secret
GOOGLE_CALLBACK_URL=https://your-service-name.onrender.com/api/auth/google/callback

# CORS Configuration
FRONTEND_URL=https://your-app-name.vercel.app
ALLOWED_ORIGINS=https://your-app-name.vercel.app
CUSTOM_DOMAINS=your-custom-domain.com

# Session Configuration
SESSION_SECRET=awibi-medtech-super-secure-session-secret-2025-change-this-in-production

# Security Configuration
BCRYPT_ROUNDS=12
PASSWORD_MIN_LENGTH=6
ACCOUNT_LOCK_TIME=900000
MAX_LOGIN_ATTEMPTS=5

# Rate Limiting
RATE_LIMIT_WINDOW=900000
RATE_LIMIT_MAX=100
AUTH_RATE_LIMIT_MAX=5
```

Important Notes: - Replace `username:password` with your MongoDB credentials - Replace `your-service-name` with your actual Render service name - Replace `your-app-name` with your Vercel app name - Generate strong, unique secrets for JWT and session

Step 4: Deploy

1. **Click "Create Web Service"**
2. Render will automatically build and deploy your application
3. Wait for the build to complete (usually 3-5 minutes)
4. **Get Your Service URL**
5. Example: `https://awibi-medtech-backend.onrender.com`
6. Note this URL for frontend configuration

Step 5: Verify Deployment

1. **Test Health Endpoint** `bash curl https://your-service-name.onrender.com/health`

2. **Expected Response:** `json { "status": "OK", "message": "AWIBI MEDTECH API is running", "timestamp": "2025-01-09T...", "environment": "production", "version": "3.0.0", "database": { "status": "Connected", "collections": 4, "dataSize": "1MB" } }`

3. **Test CORS Endpoint** `bash curl -H "Origin: https://your-app-name.vercel.app" \ https://your-service-name.onrender.com/api/test-cors`

🔧 Advanced Configuration

Custom Domain Setup

1. **In Render Dashboard**
2. Go to Service → Settings → Custom Domains
3. Add your custom domain
4. Follow DNS configuration instructions
5. **Update Environment Variables** `env GOOGLE_CALLBACK_URL=https://your-custom-domain.com/api/auth/google/callback`

Database Optimization

1. **MongoDB Atlas Configuration**
2. Enable connection pooling
3. Set up database monitoring
4. Configure automated backups
5. **Connection String Optimization** `mongodb+srv://username:password@cluster.mongodb.net/awibi-medtech?retryWrites=true&w=majority&maxPoolSize=10&serverSelectionTimeoutMS=5000&socketTimeoutMS=45000`

Performance Optimization

1. **Enable Compression**
2. Already configured in the backend code
3. Reduces response sizes by ~70%
4. **Database Indexing** `javascript // Indexes are already configured in the schemas userSchema.index({ email: 1 }); chapterSchema.index({ slug: 1 }); eventSchema.index({ date: 1 });`

Monitoring and Logging

1. **Render Metrics**
2. CPU usage
3. Memory usage
4. Response times

5. Error rates

6. **Custom Logging** `javascript // Already implemented in the backend app.use(morgan(process.env.NODE_ENV === 'production' ? 'combined' : 'dev'));`

🐛 Troubleshooting

Build Failures

1. **Check build logs in Render dashboard**
2. **Common issues**:
3. Missing dependencies: Check package.json
4. Node version mismatch: Specify in package.json
5. Build timeout: Optimize build process

Database Connection Issues

1. **Check MongoDB Atlas**:
2. Verify connection string
3. Check IP whitelist (should include 0.0.0.0/0)

4. Verify database user permissions

5. **Test connection locally**: `bash node -e " const mongoose = require('mongoose'); mongoose.connect('your-connection-string') .then(() => console.log('Connected')) .catch(err => console.error('Error:', err)); "`

CORS Issues

1. **Check environment variables:** `bash # In Render dashboard, verify: FRONTEND_URL=https://your-app-name.vercel.app ALLOWED_ORIGINS=https://your-app-name.vercel.app`

2. **Test CORS manually**: `bash curl -H "Origin: https://your-app-name.vercel.app" \ -H "Access-Control-Request-Method: POST" \ -H "Access-Control-Request-Headers: Content-Type,Authorization" \ -X OPTIONS \ https://your-service-name.onrender.com/api/auth/login`

Performance Issues

1. **Check service metrics** in Render dashboard
2. **Optimize database queries**
3. **Enable caching** where appropriate
4. **Consider upgrading** to paid plan for better performance

Memory Issues

1. **Monitor memory usage** in Render dashboard
2. **Optimize code** for memory efficiency

3. **Consider upgrading** to higher memory plan

📊 Monitoring and Maintenance

Health Monitoring

1. **Set up uptime monitoring**: `bash # Use services like UptimeRobot or Pingdom # Monitor: https://your-service-name.onrender.com/health`

2. **Database monitoring**:

3. MongoDB Atlas provides built-in monitoring
4. Set up alerts for connection issues

Log Management

1. **Access logs** in Render dashboard
2. **Set up log aggregation** for production
3. **Monitor error patterns**

Security Updates

1. **Regular dependency updates**: `bash npm audit npm update`

2. **Security headers** (already configured):

3. Helmet.js for security headers
4. Rate limiting for API protection
5. CORS for cross-origin security

Backup Strategy

1. **MongoDB Atlas automated backups**
2. **Code backups** via GitHub
3. **Environment variable backups** (securely stored)

🔒 Security Best Practices

Environment Variables

1. **Use strong, unique secrets**
2. **Rotate secrets regularly**
3. **Never commit secrets to code**

Database Security

1. **Use strong database passwords**

2. **Enable MongoDB Atlas security features**
3. **Regular security audits**

API Security

1. **Rate limiting** (already configured)
2. **Input validation** (already implemented)
3. **Authentication** (JWT + OAuth)

🚀 Deployment Checklist

- [ ] MongoDB Atlas cluster is created and configured
- [ ] Repository is up to date on GitHub
- [ ] Environment variables are set correctly
- [ ] Build completes successfully
- [ ] Health endpoint returns OK
- [ ] Database connection is working
- [ ] CORS is configured correctly
- [ ] Authentication endpoints work
- [ ] API endpoints respond correctly
- [ ] Error handling is working
- [ ] Logging is enabled
- [ ] Security headers are set
- [ ] Rate limiting is active
- [ ] Custom domain is configured (if applicable)
- [ ] Monitoring is set up

📞 Support

If you encounter issues:

1. **Check Render documentation**: https://render.com/docs
2. **Review service logs** in Render dashboard
3. **Test locally** before deploying
4. **Check MongoDB Atlas** for database issues
5. **Contact Render support** if needed

Your AWIBI MEDTECH backend should now be successfully deployed on Render! 🎉

🔗 Next Steps

1. **Update frontend** with your Render backend URL

2. **Deploy frontend** to Vercel
3. **Test full application** end-to-end
4. **Set up monitoring** and alerts
5. **Configure custom domains** if needed