🌐 CORS Optimization Guide for AWIBI MEDTECH

🎯 Overview

This guide provides a comprehensive solution to CORS (Cross-Origin Resource Sharing) issues when deploying the AWIBI MEDTECH application with: - **Frontend**: Vercel deployment - **Backend**: Render deployment

🚨 Common CORS Issues and Solutions

Issue 1: "Access to fetch blocked by CORS policy"

Cause: Backend not configured to accept requests from frontend domain **Solution**: Enhanced CORS configuration in backend

Issue 2: "Credentials include but CORS not configured"

Cause: Frontend sending credentials but backend not configured for it **Solution**: Set `credentials: true` in CORS config

Issue 3: "Preflight request failed"

Cause: OPTIONS requests not handled properly **Solution**: Explicit OPTIONS handling

🔧 Backend CORS Configuration (Render)

Current Enhanced Configuration

```
// Enhanced CORS configuration for production deployment
app.use(cors({
  origin: function(origin, callback) {
    // Allow requests with no origin (mobile apps, Postman, etc.)
    if (!origin) return callback(null, true);

    // Allow all Vercel domains (including preview deployments)
    if (origin.includes('.vercel.app') || origin.includes('vercel.app')) {
      return callback(null, true);
    }

    // Allow Netlify domains (backup deployment option)
    if (origin.includes('.netlify.app') || origin.includes('netlify.app')) {
      return callback(null, true);
    }

    // Allow localhost for development (any port)
    if (origin.includes('localhost') || origin.includes('127.0.0.1')) {
      return callback(null, true);
    }

    // Allow specific production domains from environment variable
    const allowedOrigins = process.env.ALLOWED_ORIGINS ?
      process.env.ALLOWED_ORIGINS.split(',').map(origin => origin.trim()) : [];

    if (allowedOrigins.includes(origin)) {
      return callback(null, true);
    }

    // Allow custom domains
    const customDomains = process.env.CUSTOM_DOMAINS ?
      process.env.CUSTOM_DOMAINS.split(',').map(domain => domain.trim()) : [];

    if (customDomains.some(domain => origin.includes(domain))) {
      return callback(null, true);
    }

    // Default allow all in development
    if (process.env.NODE_ENV !== 'production') {
      return callback(null, true);
    }

    // Log blocked origins in production for debugging
    console.warn(`🚫 CORS blocked origin: ${origin}`);
    callback(new Error(`Origin ${origin} not allowed by CORS policy`));
  },
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', 'PATCH'],
  allowedHeaders: [
    'Content-Type',
    'Authorization',
    'X-Requested-With',
    'Accept',
    'Origin',
    'Access-Control-Allow-Origin',
    'Access-Control-Allow-Headers',
    'Access-Control-Allow-Methods'
  ],
  exposedHeaders: ['X-Total-Count', 'X-Page-Count'],
  optionsSuccessStatus: 200,
  preflightContinue: false
}));

// Handle preflight requests explicitly
app.options('*', cors());
```

🌐 Frontend Configuration (Vercel)

API Configuration

```
// src/lib/api.js
import axios from 'axios';

const API_BASE_URL = import.meta.env.VITE_API_URL || 'http://localhost:5000';

// Create axios instance with CORS-friendly config
const api = axios.create({
  baseURL: API_BASE_URL,
  timeout: 30000,
  withCredentials: true, // Important for CORS with credentials
  headers: {
    'Content-Type': 'application/json',
  },
});
```

📋 Environment Variables Configuration

Backend (.env for Render)

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

# Database Configuration
MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.net/awibi-medtech

# JWT Configuration
JWT_SECRET=your-super-secure-jwt-secret-for-production-change-this
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-backend-
url.onrender.com/api/auth/google/callback

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

# Session Configuration
SESSION_SECRET=your-super-secure-session-secret-for-production-change-this
```

Frontend (.env for Vercel)

```
# API Configuration
VITE_API_URL=https://your-backend-url.onrender.com

# Google OAuth Configuration
VITE_GOOGLE_CLIENT_ID=your-google-oauth-client-id

# App Configuration
VITE_APP_NAME=AWIBI MEDTECH
VITE_APP_VERSION=1.0.0
```

🚀 Deployment Steps

Step 1: Deploy Backend to Render

1. **Create Web Service**
2. Repository: Connect your GitHub repo

3. Branch: main
4. Root Directory: `backend`
5. Environment: Node
6. Build Command: `npm install`
7. Start Command: `npm start`
8. **Set Environment Variables** `NODE_ENV=production PORT=10000 MONGODB_URI=your-mongodb-connection-string JWT_SECRET=your-production-jwt-secret FRONTEND_URL=https://your-app-name.vercel.app GOOGLE_CLIENT_ID=your-google-client-id GOOGLE_CLIENT_SECRET=your-google-client-secret SESSION_SECRET=your-production-session-secret`
9. **Deploy and Note URL**
10. Example: `https://awibi-medtech-backend.onrender.com`

Step 2: Deploy Frontend to Vercel

1. **Import Project**
2. Repository: Connect your GitHub repo
3. Framework: Vite
4. Root Directory: `frontend/awibi-medtech-frontend`
5. Build Command: `npm run build`
6. Output Directory: `dist`
7. **Set Environment Variables** `VITE_API_URL=https://your-backend-url.onrender.com VITE_GOOGLE_CLIENT_ID=your-google-client-id`
8. **Deploy and Note URL**
9. Example: `https://awibi-medtech.vercel.app`

Step 3: Update Backend CORS

1. **Update Environment Variables in Render** `FRONTEND_URL=https://awibi-medtech.vercel.app ALLOWED_ORIGINS=https://awibi-medtech.vercel.app`

2. **Redeploy Backend Service**

🧪 Testing CORS Configuration

Test 1: Basic CORS Test

```
curl -H "Origin: https://your-app-name.vercel.app" \
     -H "Access-Control-Request-Method: POST" \
     -H "Access-Control-Request-Headers: X-Requested-With" \
     -X OPTIONS \
     https://your-backend-url.onrender.com/api/test-cors
```

Test 2: Authentication Test

```
// Frontend test
const testAuth = async () => {
  try {
    const response = await fetch(`${API_URL}/api/auth/login`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      credentials: 'include',
      body: JSON.stringify({
        email: 'test@example.com',
        password: 'testpassword'
      })
    });

    const data = await response.json();
    console.log('Auth test result:', data);
  } catch (error) {
    console.error('CORS Error:', error);
  }
};
```

Test 3: Preflight Request Test

```javascript
 // Test preflight for complex requests
const testPreflight = async () => {
  try {
    const response = await fetch(`${API_URL}/api/users`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer your-token'
      },
      credentials: 'include',
      body: JSON.stringify({ firstName: 'Test' })
    });

    console.log('Preflight test passed');
  } catch (error) {
    console.error('Preflight failed:', error);
  }
};
```

🔍 Debugging CORS Issues

Check 1: Browser Developer Tools

1. Open Network tab
2. Look for failed requests
3. Check response headers:
4. `Access-Control-Allow-Origin`
5. `Access-Control-Allow-Credentials`
6. `Access-Control-Allow-Methods`

Check 2: Backend Logs

```
// Add debugging to CORS configuration
app.use(cors({
  origin: function(origin, callback) {
    console.log('🌐 CORS Request from origin:', origin);

    // Your CORS logic here

    if (isAllowed) {
      console.log('✅ CORS allowed for:', origin);
      callback(null, true);
    } else {
      console.log('❌ CORS blocked for:', origin);
      callback(new Error('Not allowed by CORS'));
    }
  },
  // ... rest of config
}));
```

Check 3: Environment Variables

```
// Add environment variable logging
console.log('Environment Check:');
console.log('FRONTEND_URL:', process.env.FRONTEND_URL);
console.log('ALLOWED_ORIGINS:', process.env.ALLOWED_ORIGINS);
console.log('NODE_ENV:', process.env.NODE_ENV);
```

🛠️ Advanced CORS Solutions

Solution 1: Dynamic Origin Validation

```
const isDevelopment = process.env.NODE_ENV !== 'production';
const isVercelDomain = (origin) => {
  return origin && (
    origin.includes('.vercel.app') ||
    origin.includes('vercel.app') ||
    origin.match(/^https:\/\/.*-.*\.vercel\.app$/)
  );
};

const isLocalhost = (origin) => {
  return origin && (
    origin.includes('localhost') ||
    origin.includes('127.0.0.1') ||
    origin.includes('0.0.0.0')
  );
};
```

Solution 2: Whitelist Management

```
const getWhitelistedOrigins = () => {
  const origins = [
    process.env.FRONTEND_URL,
    process.env.FRONTEND_DOMAIN,
    'https://awibi-medtech.vercel.app',
    'https://awibi-medtech-preview.vercel.app'
  ].filter(Boolean);

  if (process.env.ALLOWED_ORIGINS) {
    origins.push(...process.env.ALLOWED_ORIGINS.split(',').map(o => o.trim()));
  }

  return origins;
};
```

Solution 3: Fallback CORS for Development

```
const corsOptions = {
  origin: isDevelopment ? true : function(origin, callback) {
    // Production CORS logic
  },
  credentials: true,
  // ... rest of config
};
```

📊 CORS Monitoring

Health Check Endpoint

```
app.get('/api/cors-health', (req, res) => {
  res.json({
    cors: {
      origin: req.headers.origin,
      allowed: true,
      timestamp: new Date().toISOString(),
      environment: process.env.NODE_ENV,
      frontendUrl: process.env.FRONTEND_URL
    }
  });
});
```

CORS Analytics

```javascript
let corsStats = {
  allowed: 0,
  blocked: 0,
  origins: new Set()
};

// Track CORS requests
const trackCorsRequest = (origin, allowed) => {
  corsStats.origins.add(origin);
  if (allowed) {
    corsStats.allowed++;
  } else {
    corsStats.blocked++;
  }
};
```

🚨 Emergency CORS Fix

If CORS is completely broken, use this temporary fix:

```javascript
// EMERGENCY CORS FIX - Use only temporarily
app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Credentials', 'true');
  res.header('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE,OPTIONS');
  res.header('Access-Control-Allow-Headers', 'Origin,X-Requested-With,Content-Type,Accept,Authorization');

  if (req.method === 'OPTIONS') {
    res.sendStatus(200);
  } else {
    next();
  }
});
```

✅ CORS Checklist

- [ ] Backend CORS configuration includes Vercel domains
- [ ] Frontend API calls include `credentials: 'include'`
- [ ] Environment variables are set correctly
- [ ] Preflight requests are handled
- [ ] HTTPS is used in production

- [ ] Origin validation is working
- [ ] Error logging is in place
- [ ] Health check endpoint is accessible

🎯 Final Verification

1. **Test from Vercel domain**: Make API calls from deployed frontend
2. **Check browser console**: No CORS errors
3. **Verify authentication**: Login/logout works
4. **Test all HTTP methods**: GET, POST, PUT, DELETE
5. **Check preflight requests**: Complex requests work
6. **Monitor logs**: Backend receives requests properly

This configuration ensures bulletproof CORS handling for Vercel + Render deployment!