TitleTesterPro OAuth Fix - Complete Installation Guide

Prerequisites

- Node.js 16+ installed
- PostgreSQL database
- Access to Google Cloud Console
- Existing TitleTesterPro codebase

Step 1: Install Required Packages

bash

```bash
npm install --save \
  passport@^0.7.0 \
  passport-google-oauth20@^2.0.0 \
  @google-auth-library/googleapis@^1.0.0 \
  express-session@^1.17.3 \
  connect-pg-simple@^9.0.1

# If using TypeScript
npm install --save-dev \
  @types/passport@^1.0.16 \
  @types/passport-google-oauth20@^2.0.14 \
  @types/express-session@^1.17.10
```

Step 2: Update Environment Variables

Add/update these in your `.env` file:

```env
# Google OAuth
GOOGLE_CLIENT_ID=your-client-id-here
GOOGLE_CLIENT_SECRET=your-client-secret-here

# URLs (update for production)
NEXTAUTH_URL=http://localhost:3000
GOOGLE_REDIRECT_URI=http://localhost:3000/auth/google/callback

# Session Secret (generate a random string)
SESSION_SECRET=generate-a-long-random-string-here

# Database
DATABASE_URL=postgresql://user:password@localhost:5432/titletesterpro
```

Step 3: Google Cloud Console Setup

1. Go to Google Cloud Console
2. Select your project or create a new one
3. Enable these APIs:
    - YouTube Data API v3
    - YouTube Analytics API
    - Google+ API (for profile data)
4. Configure OAuth Consent Screen:
    - Go to "APIs & Services" > "OAuth consent screen"
    - Add these scopes:

      ```
      openid
      email
      profile
      https://www.googleapis.com/auth/youtube.readonly
      https://www.googleapis.com/auth/youtube.force-ssl
      https://www.googleapis.com/auth/youtubepartner
      https://www.googleapis.com/auth/yt-analytics.readonly
      ```

5. Create OAuth Credentials:
    - Go to "APIs & Services" > "Credentials"
    - Create OAuth 2.0 Client ID
    - Application type: Web application

- Add authorized redirect URIs:

```
http://localhost:3000/auth/google/callback
https://yourdomain.com/auth/google/callback
```

Step 4: Database Migration

Create migration file `prisma/migrations/add_google_auth/migration.sql`:

sql

```sql
-- Add Google auth fields to User
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "googleId" TEXT UNIQUE;
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "authVersion" TEXT DEFAULT 'v1';
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "youtubeChannelThumbnail" TEXT;
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "tokenExpiresAt" TIMESTAMP;
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "migrationStatus" TEXT;
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "migratedAt" TIMESTAMP;
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "migratedToEmail" TEXT;
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "lastLogin" TIMESTAMP;

-- Create Teams table
CREATE TABLE IF NOT EXISTS "Team" (
  "id" TEXT NOT NULL PRIMARY KEY,
  "name" TEXT NOT NULL,
  "description" TEXT,
  "createdAt" TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- Create TeamMember table
CREATE TABLE IF NOT EXISTS "TeamMember" (
  "id" TEXT NOT NULL PRIMARY KEY,
  "userId" TEXT NOT NULL,
  "teamId" TEXT NOT NULL,
  "role" TEXT NOT NULL DEFAULT 'MEMBER',
  "joinedAt" TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT "TeamMember_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User"("id") ON DELETE RE
  CONSTRAINT "TeamMember_teamId_fkey" FOREIGN KEY ("teamId") REFERENCES "Team"("id") ON DELETE
  UNIQUE("userId", "teamId")
);

-- Create Channel table
CREATE TABLE IF NOT EXISTS "Channel" (
  "id" TEXT NOT NULL PRIMARY KEY,
  "channelId" TEXT NOT NULL UNIQUE,
  "channelTitle" TEXT NOT NULL,
  "channelThumbnail" TEXT,
  "teamId" TEXT NOT NULL,
  "addedAt" TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT "Channel_teamId_fkey" FOREIGN KEY ("teamId") REFERENCES "Team"("id") ON DELETE RESTR
);

-- Create MigrationLog table
```

```
CREATE TABLE IF NOT EXISTS "MigrationLog" (
  "id" TEXT NOT NULL PRIMARY KEY,
  "userId" TEXT NOT NULL,
  "type" TEXT NOT NULL,
  "details" JSONB NOT NULL,
  "timestamp" TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT "MigrationLog_userId_fkey" FOREIGN KEY ("userId") REFERENCES "User"("id") ON DELETE RES
);

-- Add team relations to Test
ALTER TABLE "Test" ADD COLUMN IF NOT EXISTS "teamId" TEXT;
ALTER TABLE "Test" ADD COLUMN IF NOT EXISTS "channelId" TEXT;
ALTER TABLE "Test" ADD COLUMN IF NOT EXISTS "userEmail" TEXT;

-- Create indexes
CREATE INDEX IF NOT EXISTS "User_googleId_idx" ON "User"("googleId");
CREATE INDEX IF NOT EXISTS "TeamMember_userId_idx" ON "TeamMember"("userId");
CREATE INDEX IF NOT EXISTS "TeamMember_teamId_idx" ON "TeamMember"("teamId");
CREATE INDEX IF NOT EXISTS "Channel_channelId_idx" ON "Channel"("channelId");
CREATE INDEX IF NOT EXISTS "Channel_teamId_idx" ON "Channel"("teamId");
CREATE INDEX IF NOT EXISTS "Test_teamId_idx" ON "Test"("teamId");
CREATE INDEX IF NOT EXISTS "Test_channelId_idx" ON "Test"("channelId");
```

Run the migration:

bash

```bash
npx prisma migrate dev --name add_google_auth
npx prisma generate
```

Step 5: Create Required Files

5.1 Create `src/lib/session.ts`:

typescript

```typescript
import session from 'express-session';
import connectPgSimple from 'connect-pg-simple';

const PgSession = connectPgSimple(session);

export const sessionMiddleware = session({
  store: new PgSession({
    conString: process.env.DATABASE_URL,
    tableName: 'user_sessions'
  }),
  secret: process.env.SESSION_SECRET!,
  resave: false,
  saveUninitialized: false,
  cookie: {
    maxAge: 30 * 24 * 60 * 60 * 1000, // 30 days
    httpOnly: true,
    secure: process.env.NODE_ENV === 'production',
    sameSite: 'lax'
  }
});
```

5.2 Update `src/app/api/auth/[...auth]/route.ts`:

typescript

```typescript
import { NextRequest, NextResponse } from 'next/server';
import passport from '../../../../config/passport';

export async function GET(
  request: NextRequest,
  { params }: { params: { auth: string[] } }
) {
  const [action, ...rest] = params.auth;

  if (action === 'google') {
    // Initiate Google OAuth
    return NextResponse.redirect(
      `https://accounts.google.com/o/oauth2/v2/auth?` +
      `client_id=${process.env.GOOGLE_CLIENT_ID}&` +
      `redirect_uri=${encodeURIComponent(process.env.GOOGLE_REDIRECT_URI!)}&` +
      `response_type=code&` +
      `scope=${encodeURIComponent([
        'openid',
        'email',
        'profile',
        'https://www.googleapis.com/auth/youtube.readonly',
        'https://www.googleapis.com/auth/youtube.force-ssl',
        'https://www.googleapis.com/auth/youtubepartner',
        'https://www.googleapis.com/auth/yt-analytics.readonly'
      ].join(' '))}&` +
      `access_type=offline&` +
      `prompt=consent`
    );
  }

  return NextResponse.json({ error: 'Invalid auth action' }, { status: 400 });
}
```

5.3 Create test file `src/test/auth-test.ts`:

typescript

```typescript
import { YouTubeAuthService } from '../services/youtube-auth.service';

async function testAuth() {
  console.log('Testing Google OAuth configuration...');

  // Test 1: Check environment variables
  const requiredEnvVars = [
    'GOOGLE_CLIENT_ID',
    'GOOGLE_CLIENT_SECRET',
    'GOOGLE_REDIRECT_URI',
    'DATABASE_URL',
    'SESSION_SECRET'
  ];

  const missingVars = requiredEnvVars.filter(v => !process.env[v]);
  if (missingVars.length > 0) {
    console.error('❌ Missing environment variables:', missingVars);
    return false;
  }

  console.log('✅ All environment variables present');

  // Test 2: Database connection
  try {
    const { PrismaClient } = require('@prisma/client');
    const prisma = new PrismaClient();
    await prisma.$connect();
    console.log('✅ Database connection successful');
    await prisma.$disconnect();
  } catch (error) {
    console.error('❌ Database connection failed:', error);
    return false;
  }

  console.log('\n🎉 All tests passed! Ready to implement OAuth fix.');
  return true;
}

// Run the test
testAuth();
```

Step 6: Implementation Steps

1. **Stop your development server**
2. **Backup your database**:

   ```bash
   pg_dump your_database > backup_$(date +%Y%m%d_%H%M%S).sql
   ```

3. **Copy all the artifact files** from the previous response into your project:
   - `src/config/passport.ts`
   - `src/services/youtube-auth.service.ts`
   - `src/services/auth-migration.service.ts`
   - `src/routes/auth.routes.ts`
   - `src/components/MigrationBanner.tsx`
   - `src/components/ChannelSelector.tsx`
   - Update `prisma/schema.prisma`
4. **Run the test**:

   ```bash
   npx ts-node src/test/auth-test.ts
   ```

5. **Start with migration mode**:

   ```bash
   # Add to your .env
   ENABLE_MIGRATION_MODE=true

   npm run dev
   ```

Step 7: Testing the Fix

1. **Test new user flow**:
   - Log out completely
   - Click "Sign in with Google"
   - Grant all permissions
   - Verify YouTube channel appears
2. **Test existing user migration**:

    - Log in with existing account
    - Click migration banner
    - Complete migration process
    - Verify all tests/data transferred
3. **Test API functionality**:
    - Create a new test
    - Verify title rotation works
    - Check analytics fetching

Step 8: Monitoring & Rollback

Monitor for:

- Authentication failures in logs
- Missing YouTube channel data
- Token refresh failures
- API quota issues

Rollback plan:

```bash
# 1. Restore database
psql your_database < backup_TIMESTAMP.sql

# 2. Revert code changes
git revert HEAD

# 3. Restart services
pm2 restart all
```

Step 9: Production Deployment

1. **Test in staging first**
2. **Enable gradual rollout**:

```typescript
// Feature flag
const useNewAuth = process.env.ENABLE_NEW_AUTH === 'true' ||
                   Math.random() < 0.1; // 10% of users
```

3. **Monitor error rates**
4. **Gradually increase rollout percentage**

Troubleshooting

"Profile not showing" issue:

- Check if `youtubeChannelId` is being fetched
- Verify scopes include YouTube permissions
- Check browser console for errors

"Invalid grant" errors:

- Tokens expired - implement refresh
- User revoked access - re-authenticate
- Wrong redirect URI - check Google Console

Migration issues:

- Check `MigrationLog` table for errors
- Verify email matching logic
- Test with different account types

Support

If you encounter issues:

1. Check logs: `tail -f logs/auth.log`
2. Run diagnostics: `npm run diagnose:auth`
3. Check migration status: `npm run migration:status`

Remember to update your privacy policy and terms of service to reflect the change from YouTube to Google authentication.