Social Media Authentication Service - Implementation Guide  
GitHub Repo: **https://github.com/psu-edu/social-auth-app1.git**

# Project Overview

This project implements a Node.js-based web application that enables users to log in using their Google or Facebook accounts. It leverages OAuth 2.0 protocol through Passport.js strategies, with user data stored in MongoDB. Sessions are managed with express-session and stored persistently using connect-mongo.

A group of logos with text

AI-generated content may be incorrect.

# Project Structure

A screen shot of a computer

AI-generated content may be incorrect.

# Step By Step Implementation

## Environment Setup

* MongoDB installed and running.
* Node.js installed.
* Created .env file with sensitive keys:

GOOGLE\_CLIENT\_ID=real\_google\_client\_id

GOOGLE\_CLIENT\_SECRET=real\_google\_client\_secret

FACEBOOK\_APP\_ID=real\_facebook\_app\_id

FACEBOOK\_APP\_SECRET=real\_facebook\_app\_secret

SESSION\_SECRET=real\_session\_secret

MONGO\_URI=mongodb://localhost:27017/social\_auth

PORT=3000

## User Model

A screenshot of a computer program

AI-generated content may be incorrect.

## Google Authentication

A screenshot of a computer

AI-generated content may be incorrect.

Used passport-google-oauth20 strategy.

Checks existing user or creates a new user with profile data.

## Facebook Authentication

https://developers.facebook.com/  
A screenshot of a computer

AI-generated content may be incorrect.

In App Dashboard in the"Facebook Login" 🡪 "Settings" added the

Valid OAuth Redirect URIs:

http://localhost:3000/auth/facebook/callback

Then in the left side at App🡪 Settings🡪 Basics

A screenshot of a login screen

AI-generated content may be incorrect.

Used passport-facebook strategy.

Handles missing email fields gracefully.

## Database

Download and Start MongoDB locally,   
https://www.mongodb.com/products/platform/atlas-database

## Backend Server (Node.js)

The backend server is built using Node.js and Express.js and is responsible for handling user authentication via social login providers Google /Facebook, managing sessions, connecting to the MongoDB database, and rendering user profile data upon successful login.

### Key Responsibilities:

Set up and configure Express server.

Initialize and manage Passport.js strategies for OAuth 2.0 with Google and Facebook.

Connect securely to a MongoDB database using Mongoose.

Maintain sessions using express-session and connect-mongo.

Define routes for:

Google and Facebook login and callback.

User profile access.

Logout functionality.

### Middleware and Libraries Used:

express for server and routing.

passport for authentication.

passport-google-oauth20 and passport-facebook for OAuth strategy.

mongoose for MongoDB connection and schema definition.

express-session for session management.

connect-mongo to store sessions in MongoDB.

dotenv to manage environment variables securely.

### File Overview:

index.js: Entry point. Sets up the server, middleware, sessions, and Passport strategies.

auth/google.js and auth/facebook.js: Configure Passport strategies for social login.

models/User.js: Mongoose schema for storing authenticated user data.

routes/auth.js: Auth routes for login and callback handling.

routes/profile.js: Routes for the user home and profile page after login.

### Logging:

Basic logging is performed using console.log for:

MongoDB connection status.

Authentication events (can be enhanced using winston or morgan).

## Express Service Setup

* Integrated Passport middleware.
* Session configured with connect-mongo to persist sessions.
* Routes defined for authentication and profile viewing.

## Routes

* /auth/google, /auth/facebook initiate login.
* /auth/google/callback, /auth/facebook/callback handle redirects.
* /profile shows logged-in user info.
* /logout logs user out.

## Social Media Authentication Flow Diagrams

A screenshot of a computer screen

AI-generated content may be incorrect.

A diagram of a social media authentication

AI-generated content may be incorrect.

Express App OAuth Strategy Flow (Google/Facebook)  
Application Internal Flow (Passport + DB)

A diagram of a software flow

AI-generated content may be incorrect.

## Error Handling and Loggins

### Error Handling

All Passport strategies have try-catch blocks for async calls.

Errors passed to done(err, null) to prevent app crashes.

Middleware uses Passport’s built-in failureRedirect for failed login.  
routes > auth.js  
A computer screen shot of text

AI-generated content may be incorrect.

### Logging

Console logs are used for MongoDB connection status and strategy callbacks.

Can be extended using winston or morgan for production.

A black screen with white text

AI-generated content may be incorrect.

## Security Considerations

### Secrets Management

All sensitive credentials are stored in a .env file and excluded via .gitignore.

### Session Security

Sessions are not stored in memory (used MongoDB store).

SESSION\_SECRET ensures integrity of session data.

### Passport Handling

User serialization uses minimal data (MongoDB ID only).

OAuth tokens are not stored permanently.

### HTTPS Recommendation

In production, HTTPS is essential to protect token exchange.

### Recommendations For Future

Rate limiting and CSRF protection middleware (e.g. helmet, express-rate-limit).

Validate email existence where applicable (especially Facebook).

## Conclusion

This project demonstrates a working social login system using Google and Facebook OAuth 2.0, Node.js, Passport.js, and MongoDB. The structure is modular and production-ready with recommended enhancements. The implementation guide and diagrams provide clear instructions for future extension or deployment.

### **Outputs Screen shots**

A computer screen shot of a black screen

AI-generated content may be incorrect.

A black screen with colorful lines

AI-generated content may be incorrect.

A screen shot of a computer

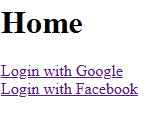
AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

A close up of a website

AI-generated content may be incorrect.

Logout from Google will bring back to home screen.  


Facebook Login:

A screenshot of a computer

AI-generated content may be incorrect.

A close up of a website

AI-generated content may be incorrect.