OAuth 2.0 Authentication using Sessions

Index

1. Project Overview

This application demonstrates OAuth 2.0 authentication combined with server-side session management. OAuth is used for identity, sessions for state.

2. Learning Outcomes

- Understand OAuth 2.0 login flow
- Differentiate OAuth from JWT and sessions
- Maintain authenticated users using sessions

3. Scope

In scope: OAuth login, callback handling, session persistence.
Out of scope: refresh tokens, advanced scopes.

4. Tech Stack & Versions

• Node.js
• Express.js
• Passport.js
• OAuth Provider
• express-session

5. Folder Structure

```
oauth-session-app/
├── app.js
├── auth/oauth.js
├── routes/protected.js
├── config/passport.js
├── package.json
└── README.md
```

6. High-Level Architecture

```
Browser → OAuth Provider → Callback → Express → Session Store
```

7. OAuth Authentication Flow

```
1. Login request
2. Redirect to OAuth provider
3. Provider authenticates user
4. Callback received
```

```
5. User stored in session
```

8. Session Management

```
Session created after OAuth success
Session ID stored in HttpOnly cookie
```

9. API Contract

```
GET /auth/login
GET /auth/callback
GET /profile
GET /logout
```

10. Configuration & Environment Variables

```
OAUTH_CLIENT_ID
OAUTH_CLIENT_SECRET
SESSION_SECRET
```

11. How to Run

```
npm install
node app.js
```

12. Common Mistakes

- Wrong callback URL
- Not saving session
- Leaking OAuth secrets

13. Debugging Techniques

- Check provider dashboard
- Enable passport debug logs
- Inspect cookies

Appendix A: Source Code

.env

```
PORT=5000
MONGO_URI=mongodb://127.0.0.1:27017/oauth_session_db
SESSION_SECRET=super_session_secret

GOOGLE_CLIENT_ID=YOUR_OAUTH_CLIENT_ID

GOOGLE_CLIENT_SECRET=YOUR_OAUTH_CLIENT_SECRET
```

package.json

```
{
  "name": "oauth-session-app",
  "version": "1.0.0",
  "scripts": {
    "start": "node src/server.js"
  },
  "dependencies": {
    "connect-mongo": "^5.1.0",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "express-session": "^1.17.3",
    "mongoose": "^8.3.4",
    "passport": "^0.7.0",
    "passport-google-oauth20": "^2.0.0"
  }
}
```

public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>OAuth Session App</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background: #f5f5f5;
      display: flex;
      height: 100vh;
      align-items: center;
      justify-content: center;
    }
    .card {
      background: white;
      padding: 30px;
      border-radius: 8px;
      box-shadow: 0 4px 10px rgba(0,0,0,0.1);
      text-align: center;
```

```
    }
    a {
      display: inline-block;
      margin-top: 20px;
      padding: 12px 20px;
      background: #4285F4;
      color: white;
      text-decoration: none;
      border-radius: 4px;
      font-weight: bold;
    }
    a:hover {
      background: #3367D6;
    }
  </style>
</head>
<body>
  <div class="card">
    <h2>OAuth Session App</h2>
    <p>Login using Google OAuth</p>

    <a href="/auth/google">Login with Google</a>
  </div>
</body>
</html>
```

public/profile.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Profile</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background: #f5f5f5;
      padding: 40px;
    }
    .card {
      background: white;
      padding: 30px;
      max-width: 400px;
      margin: auto;
      border-radius: 8px;
      box-shadow: 0 4px 10px rgba(0,0,0,0.1);
      text-align: center;
    }
    button {
      margin-top: 20px;
      padding: 10px 16px;
      border: none;
      background: #d9534f;
```

```
      color: white;
      cursor: pointer;
      border-radius: 4px;
    }
    button:hover {
      background: #c9302c;
    }
  </style>
</head>
<body>
  <div class="card">
    <h2>Profile</h2>
    <p><strong>Email:</strong> <span id="email">Loading...</span></p>
    <p><strong>Role:</strong> <span id="role">Loading...</span></p>

    <button onclick="logout()">Logout</button>
  </div>

  <script>
    async function loadProfile() {
      const res = await fetch("/profile");

      if (res.status === 401) {
        window.location.href = "/";
        return;
      }

      const data = await res.json();
      document.getElementById("email").textContent = data.email;
      document.getElementById("role").textContent = data.role;
    }

    async function logout() {
      await fetch("/logout");
      window.location.href = "/";
    }

    loadProfile();
  </script>
  </body>
</html>
```

readme.txt

```
1. Install dependencies
npm install

2. Configure Google OAuth on Google Cloud Platform
Refer documentation

3. Update .env:
GOOGLE_CLIENT_ID=your_client_id_here
GOOGLE_CLIENT_SECRET=your_client_secret_here
```

```
4. npm start
Server runs at:
http://localhost:5000

5. Login
http://localhost:5000

Login via Google → redirected to profile.html.

6.Click Logout

Request to GET /logout
```

src/models/User.js

```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  googleId: String,
  email: String,
  role: {
    type: String,
    enum: ["user", "admin"],
    default: "user"
  }
});

module.exports = mongoose.model("User", userSchema);
```

src/passport.js

```
const passport = require("passport");
const GoogleStrategy = require("passport-google-oauth20").Strategy;
const User = require("./models/User");

passport.use(
  new GoogleStrategy(
    {
      clientID: process.env.GOOGLE_CLIENT_ID,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,
      callbackURL: "/auth/google/callback"
    },
    async (accessToken, refreshToken, profile, done) => {
      let user = await User.findOne({ googleId: profile.id });

      if (!user) {
        user = await User.create({
          googleId: profile.id,
          email: profile.emails[0].value,
          role: "user"
        });
      }
```

```
      done(null, user);
    }
  )
);

passport.serializeUser((user, done) => {
  done(null, user.id);
});

passport.deserializeUser((id, done) => {
  User.findById(id).then(user => done(null, user));
});
```

src/routes/authRoutes.js

```
const express = require("express");
const passport = require("passport");

const router = express.Router();

router.get("/google", (req, res, next) => {
  console.log("□ /auth/google route HIT");
  next();
}, passport.authenticate("google", { scope: ["profile", "email"] }));

router.get("/google/callback",
  passport.authenticate("google", { failureRedirect: "/" }),
  (req, res) => {
    //res.redirect("/profile");
    res.redirect("/profile.html");
  }
);

module.exports = router;
```

src/server.js

```
const express = require("express");
const mongoose = require("mongoose");
const session = require("express-session");
const MongoStore = require("connect-mongo");
const passport = require("passport");
require("dotenv").config();

require("./passport");

const authRoutes = require("./routes/authRoutes");

const app = express();
app.use(express.json());
app.use(express.static("public"));

mongoose.connect(process.env.MONGO_URI)
  .then(() => console.log("MongoDB connected"));
```

```
app.use(
  session({
    name: "oauth-session-id",
    secret: process.env.SESSION_SECRET,
    resave: false,
    saveUninitialized: false,
    store: MongoStore.create({ mongoUrl: process.env.MONGO_URI }),
    cookie: {
      httpOnly: true,
      maxAge: 60 * 60 * 1000
    }
  })
);

app.use(passport.initialize());
app.use(passport.session());

app.use("/auth", authRoutes);

app.get("/profile", (req, res) => {
  if (!req.user) return res.status(401).send("Not logged in");
  res.json(req.user);
});

app.get("/logout", (req, res) => {
   req.logout(() => {
     res.redirect("/");
   });
 });

app.listen(process.env.PORT, () => {
  console.log(`Server running on port ${process.env.PORT}`);
});
```