LOGIN USER WITH GOOGLE AND ISSUE JWT TOKEN

Reference: https://docs.nestjs.com/recipes/passport

AuthController

```
@Get('/google/login')
@UseGuards(GoogleGuard)
login(@Req() req) {}
```

```
@Get('/google/redirect')
@UseGuards(GoogleGuard)
Codeium: Refactor | Explain | Generate JSDoc | ×
redirect(@Req() req, @Res() resp) {
  const jwt = this.authService.createJwt(req);
  // redirect to the UI with the token.
  resp.redirect(`http://localhost:3000/api/oauth?token=${jwt.access_token}`);
}
```

If we are redirected then, user has provided consense to use his account and scope for this app.

GoogleGuard

GoogleStrategy

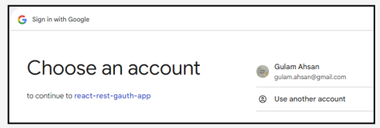
```
export class GoogleStrategy extends PassportStrategy(Strategy) {
  Codeium: Refactor | Explain | Generate JSDoc | ×
  constructor() {
    super({
      clientID: configuration().googole.clientId,
      clientSecret: configuration().googole.clientSecret,
      callbackURL: 'http://localhost:3001/api/auth/google/redirect',
      scope: ['profile', 'email'],
    });
  }
```

```
createJwt(req) {
  console.log(`Redirected and Validated Request: ${JSON.stringify(req.user)}`);
  return {
    access_token: this.jwtService.sign(req.user),
  };
}
```

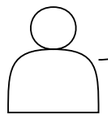
Use the req object[which contains user email, firstName, lastName, accessToken etc] to generate a jwt toekn and send him to a UI endpoint or any Valid end Point

passport-google-oauth20 would pass request for user's consense for the application identified by clientId and clientSecret

Google Consense UI



Google Authentication Service



User

VALIDATE JWT TOKEN IN ALL INCOMMING REQUEST

AmaalController

```
@UseGuards(JwtGuard)
@Get('/')
Codeium: Refactor | Explain | Generate JSDoc | ×
async amaalByDateRange(
  @Req() req,
  @Query() params: any,
): Promise<DayAmaalDto[]> {
```

JwtGuard

JWTStrategy

```
export class JwtStrategy extends PassportStrategy(Strategy) {
  Codeium: Refactor | Explain | Generate JSDoc | ×
  constructor() {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      ignoreExpiration: false,
      secretOrKey: configuration().jwt.secret,
    });
  }
```

Now any endpoint call from UI must have jwt token that is validated. for this we use passport-jwt strategy and custom super() call in this strategy that will validate the token and extract user for validate() method to check user existence. validate() is not shown here.

User Interface