Step 1: Set Up the React App

1.1 Create a new React app

```bash
npx create-react-app google-auth-app
cd google-auth-app
```

1.2 Install necessary dependencies

```bash
npm install @react-oauth/google @mui/material @emotion/react @emotion/styled
# or if you prefer Bootstrap:
npm install react-bootstrap bootstrap
```

Step 2: Set Up Google Cloud Console

2.1 Create a Google Cloud Project

1. Go to Google Cloud Console
2. Click "Select a project" → "New Project"
3. Name your project and click "Create"

2.2 Configure OAuth Consent Screen

1. Go to "APIs & Services" → "OAuth consent screen"
2. Select "External" (for testing) or "Internal" (for G Suite)
3. Fill in:
   - App name: Your app name
   - User support email: Your email
   - Developer contact email: Your email
4. Add scopes (for now, just keep the default)
5. Add test users (your email address)

2.3 Create OAuth 2.0 Credentials

1. Go to "APIs & Services" → "Credentials"
2. Click "Create Credentials" → "OAuth client ID"
3. Application type: "Web application"
4. Name: "React Google Auth"
5. Add authorized JavaScript origins:
   - `http://localhost:3000`
6. Add authorized redirect URIs:
   - `http://localhost:3000`
   - `http://localhost:3000/login`
7. Click "Create"
8. **Copy your Client ID** - you'll need this later

Step 3: Create React Components

3.1 Create a `.env` file in the root

```text
REACT_APP_GOOGLE_CLIENT_ID=your-client-id-here
```

3.2 Update `src/App.js`

```jsx
import React from 'react';
import { GoogleOAuthProvider } from '@react-oauth/google';
import Login from './components/Login';
import Dashboard from './components/Dashboard';
import { useState } from 'react';
import './App.css';

function App() {
  const [user, setUser] = useState(null);

  const handleLoginSuccess = (response) => {
    console.log('Login Success:', response);
    // Decode the JWT token to get user info
    const userObject = parseJwt(response.credential);
    setUser(userObject);
  };

  const handleLogout = () => {
    setUser(null);
    // Clear any stored tokens
    localStorage.removeItem('google_token');
  };

  // Helper function to parse JWT
  const parseJwt = (token) => {
    try {
      const base64Url = token.split('.')[1];
      const base64 = base64Url.replace(/-/g, '+').replace(/_/g, '/');
      const jsonPayload = decodeURIComponent(
        atob(base64)
          .split('')
          .map((c) => '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2))
          .join('')
      );
      return JSON.parse(jsonPayload);
    } catch (e) {
      console.error('Error parsing JWT:', e);
      return null;
    }
  };

  return (
    <GoogleOAuthProvider clientId={process.env.REACT_APP_GOOGLE_CLIENT_ID}>
      <div className="App">
```

```
        <header className="App-header">
          <h1>Google Authentication App</h1>
        </header>
        <main>
          {user ? (
            <Dashboard user={user} onLogout={handleLogout} />
          ) : (
            <Login onSuccess={handleLoginSuccess} />
          )}
        </main>
      </div>
    </GoogleOAuthProvider>
  );
}

export default App;
```

3.3 Create `src/components/Login.js`

```jsx
import React from 'react';
import { GoogleLogin } from '@react-oauth/google';
import { useGoogleLogin } from '@react-oauth/google';
import { Button, Container, Typography, Box, Paper } from '@mui/material';
// Or for Bootstrap:
// import { Container, Button, Card } from 'react-bootstrap';

const Login = ({ onSuccess }) => {
  const login = useGoogleLogin({
    onSuccess: (response) => {
      console.log('Login Success:', response);
      onSuccess(response);
    },
    onError: (error) => {
      console.log('Login Failed:', error);
    },
    flow: 'implicit', // or 'auth-code' for server-side
  });

  // Alternative: Using GoogleLogin component (renders Google's button)
  const handleSuccess = (response) => {
    console.log('Login Success:', response);
    onSuccess(response);
  };

  const handleError = () => {
    console.log('Login Failed');
  };

  return (
    <Container maxWidth="sm">
      <Paper elevation={3} sx={{ p: 4, mt: 5 }}>
        <Box sx={{ textAlign: 'center' }}>
          <Typography variant="h4" gutterBottom>
            Welcome
          </Typography>
          <Typography variant="body1" color="textSecondary" paragraph>
            Please sign in with your Google account to continue
          </Typography>

          {/* Option 1: Custom button with useGoogleLogin hook */}
          <Button
            variant="contained"
            onClick={() => login()}
            sx={{
              backgroundColor: '#4285F4',
              color: 'white',
              '&:hover': {
                backgroundColor: '#357AE8',
              },
              mb: 2,
            }}
          >
            Sign in with Google
          </Button>

          <Typography variant="body2" sx={{ mb: 2 }}>OR</Typography>

          {/* Option 2: Google's official button */}
          <GoogleLogin
            onSuccess={handleSuccess}
            onError={handleError}
            useOneTap
            theme="filled_blue"
            size="large"
            shape="rectangular"
          />

          <Box sx={{ mt: 3 }}>
            <Typography variant="caption" color="textSecondary">
              By signing in, you agree to our Terms of Service and Privacy Policy
            </Typography>
          </Box>
        </Box>
      </Paper>
    </Container>
  );
};

export default Login;
```

3.4 Create `src/components/Dashboard.js`

jsx

```jsx
import React from 'react';
import {
  Container,
  Paper,
  Typography,
  Button,
  Avatar,
  Box,
  Grid,
} from '@mui/material';
// Or for Bootstrap:
// import { Container, Card, Button, Image } from 'react-bootstrap';

const Dashboard = ({ user, onLogout }) => {
  return (
    <Container maxWidth="md">
      <Paper elevation={3} sx={{ p: 4, mt: 5 }}>
        <Box sx={{ textAlign: 'center', mb: 4 }}>
          <Typography variant="h4" gutterBottom>
            Welcome to Dashboard
          </Typography>
          <Typography variant="body1" color="textSecondary">
            You are successfully authenticated!
          </Typography>
        </Box>

        <Grid container spacing={3} alignItems="center">
          <Grid item xs={12} md={4} sx={{ textAlign: 'center' }}>
            <Avatar
              src={user.picture}
              alt={user.name}
              sx={{ width: 120, height: 120, mx: 'auto', mb: 2 }}
            />
            <Typography variant="h6">{user.name}</Typography>
            <Typography variant="body2" color="textSecondary">
              {user.email}
            </Typography>
          </Grid>

          <Grid item xs={12} md={8}>
            <Paper variant="outlined" sx={{ p: 3 }}>
              <Typography variant="h6" gutterBottom>
                User Information
              </Typography>
              <Box sx={{ mb: 2 }}>
                <Typography variant="body2" color="textSecondary">
                  Full Name
                </Typography>
                <Typography variant="body1">{user.name}</Typography>
              </Box>
              <Box sx={{ mb: 2 }}>
                <Typography variant="body2" color="textSecondary">
                  Email Address
                </Typography>
                <Typography variant="body1">{user.email}</Typography>
              </Box>
              <Box sx={{ mb: 2 }}>
                <Typography variant="body2" color="textSecondary">
                  Email Verified
                </Typography>
                <Typography variant="body1">
                  {user.email_verified ? 'Yes' : 'No'}
                </Typography>
              </Box>
              <Box>
                <Typography variant="body2" color="textSecondary">
                  User ID
                </Typography>
                <Typography variant="body1" sx={{ wordBreak: 'break-all' }}>
                  {user.sub}
                </Typography>
              </Box>
            </Paper>
          </Grid>
        </Grid>

        <Box sx={{ mt: 4, textAlign: 'center' }}>
          <Button
            variant="outlined"
            color="error"
            onClick={onLogout}
            size="large"
          >
            Sign Out
          </Button>
        </Box>

        <Box sx={{ mt: 4, p: 2, bgcolor: 'grey.50', borderRadius: 1 }}>
          <Typography variant="body2" color="textSecondary">
            <strong>Note:</strong> This is a client-side only implementation.
            For production, you should validate tokens on your backend server.
          </Typography>
        </Box>
      </Paper>
    </Container>
  );
};

export default Dashboard;
```

3.5 Update `src/App.css`

```css
.App {
  min-height: 100vh;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
}

.App-header {
  padding: 20px;
  text-align: center;
  color: white;
}

main {
  padding: 20px;
}

/* For better mobile responsiveness */
@media (max-width: 600px) {
  .App-header h1 {
    font-size: 1.5rem;
  }
}
```

Step 4: Backend Validation (Optional but Recommended)

For production, you should validate the token on your backend. Here's a simple Node.js/Express example:

4.1 Create a backend server

javascript

```javascript
// server.js
const express = require('express');
const cors = require('cors');
const { OAuth2Client } = require('google-auth-library');
require('dotenv').config();

const app = express();
const PORT = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

const CLIENT_ID = process.env.GOOGLE_CLIENT_ID;
const client = new OAuth2Client(CLIENT_ID);

app.post('/api/verify-token', async (req, res) => {
  const { token } = req.body;

  try {
    const ticket = await client.verifyIdToken({
      idToken: token,
      audience: CLIENT_ID,
    });

    const payload = ticket.getPayload();
    res.json({
      success: true,
      user: {
        id: payload.sub,
        email: payload.email,
        name: payload.name,
        picture: payload.picture,
        email_verified: payload.email_verified
      }
    });
  } catch (error) {
    console.error('Token verification failed:', error);
    res.status(400).json({ success: false, error: 'Invalid token' });
  }
});

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

4.2 Install backend dependencies

```bash
npm install express cors google-auth-library dotenv
```

4.3 Update React app to validate token with backend

```javascript
// In your Login component or App.js
const validateTokenWithBackend = async (token) => {
  try {
    const response = await fetch('http://localhost:5000/api/verify-token', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ token }),
    });

    const data = await response.json();
    if (data.success) {
      setUser(data.user);
    } else {
      console.error('Token validation failed');
    }
  } catch (error) {
    console.error('Error validating token:', error);
  }
};
```

Step 5: Run Your Application

5.1 Start the React app

```bash
npm start
```

5.2 Start the backend server (if created)

```bash
node server.js
```

Step 6: Testing

1. Open `http://localhost:3000` in your browser
2. Click "Sign in with Google"

3. Select your Google account
4. Grant permissions if prompted
5. You should see the dashboard with your user information

Security Considerations for Production

1. **Always validate tokens on backend** in production
2. **Use environment variables** for sensitive data
3. **Implement proper session management**
4. **Add CSRF protection**
5. **Set up proper CORS policies**
6. **Use HTTPS in production**
7. **Implement rate limiting**
8. **Log authentication events**
9. **Regularly review OAuth consent screen in Google Cloud Console**

Additional Features You Can Add

1. **Persistent login** using localStorage or cookies
2. **Protected routes** with React Router
3. **Refresh token handling**
4. **Multiple authentication providers**
5. **User profile management**
6. **Role-based access control**

This implementation provides a complete Google Authentication system for your React app. Remember to replace the placeholder Client ID with your actual Google Client ID from the Google Cloud Console.