

Web Application using Google OAuth

Student Names: Mr.N.Jeyapiriyana, Mr.A.Poozithan

Reg-No: MS21926662, MS22905468

Lecturer in charge: Dr. Dharshana Kasthurirathna

Assignment 2 of IE5042 - Software Security is submitted as a partial requirement of the module to test the knowledge in OAuth 2.0 framework

M.Sc. in IT (Specialized in Cyber Security)

Sri Lanka Institute of Information Technology



06 May 2022

Introduction about the system

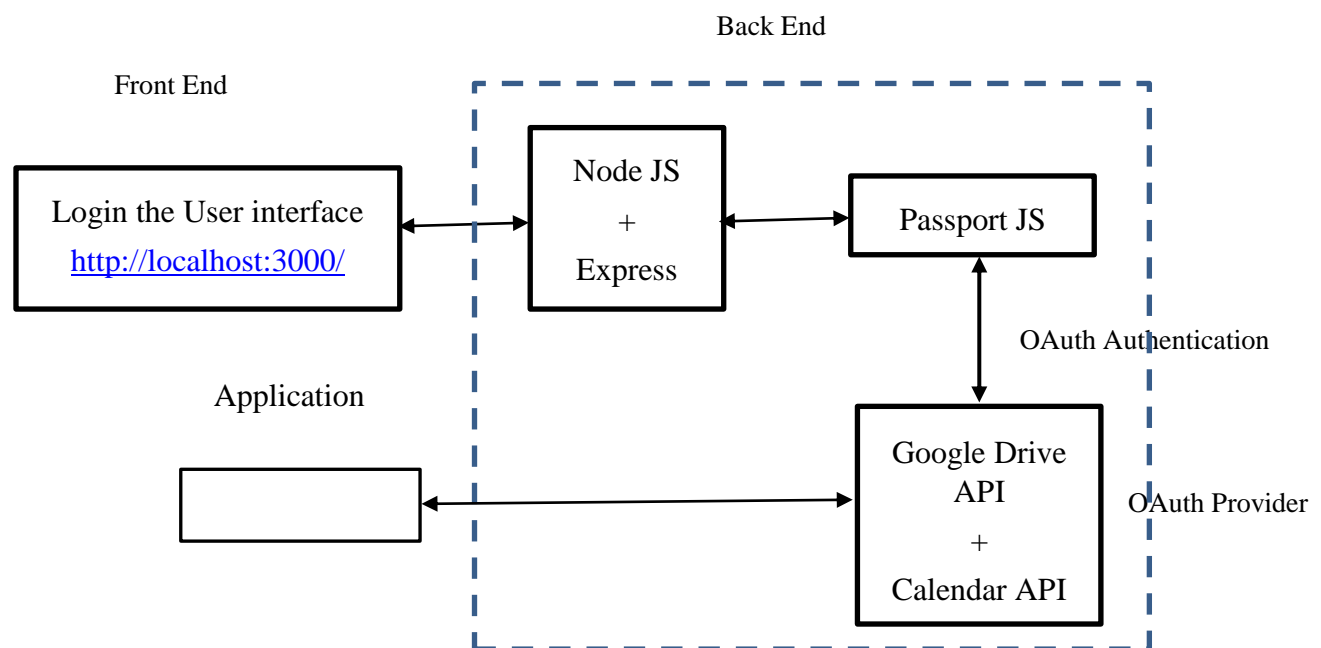
Our application has been developed by using the NodeJS server side Programming Language. This application uses the google OAuth for user authentication. It expresses the knowledge of OAuth 2.0 frame work and provides followings services.

- Retrieve the User profile name
- Upload the files to the google drive
- View the upcoming event to schedule the google calendar

OAuth Authentication

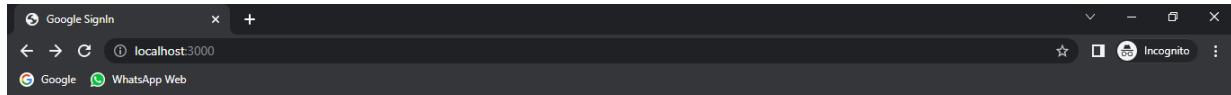
OAuth stands for "Open Authorization," and it's usually used to allow access to a collection of resources, such as distant APIs or a user's data. OAuth 2.0 gives approved access and limits what the client app may do on behalf of the user's resources without ever disclosing the user's credentials. Although it does not provide a precise format for Access Tokens. However, in some contexts, the JSON Web Token (JWT) format is often used

Architecture Diagram

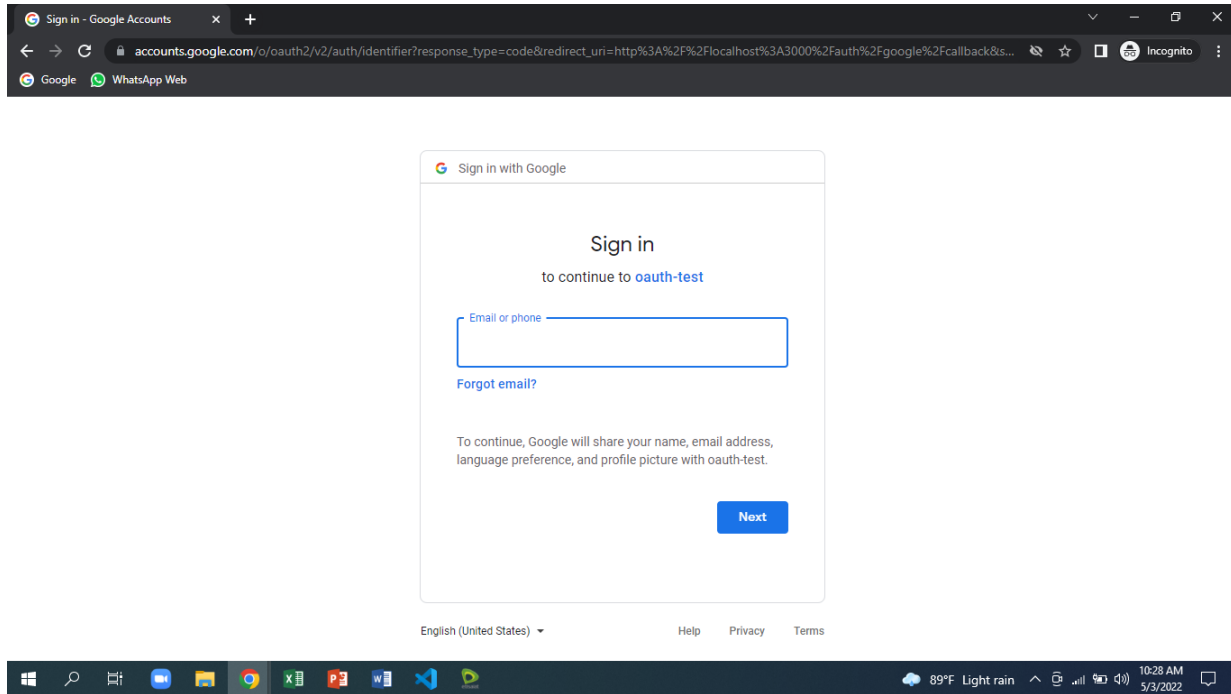


Service 1 : Retrieve the User profile name

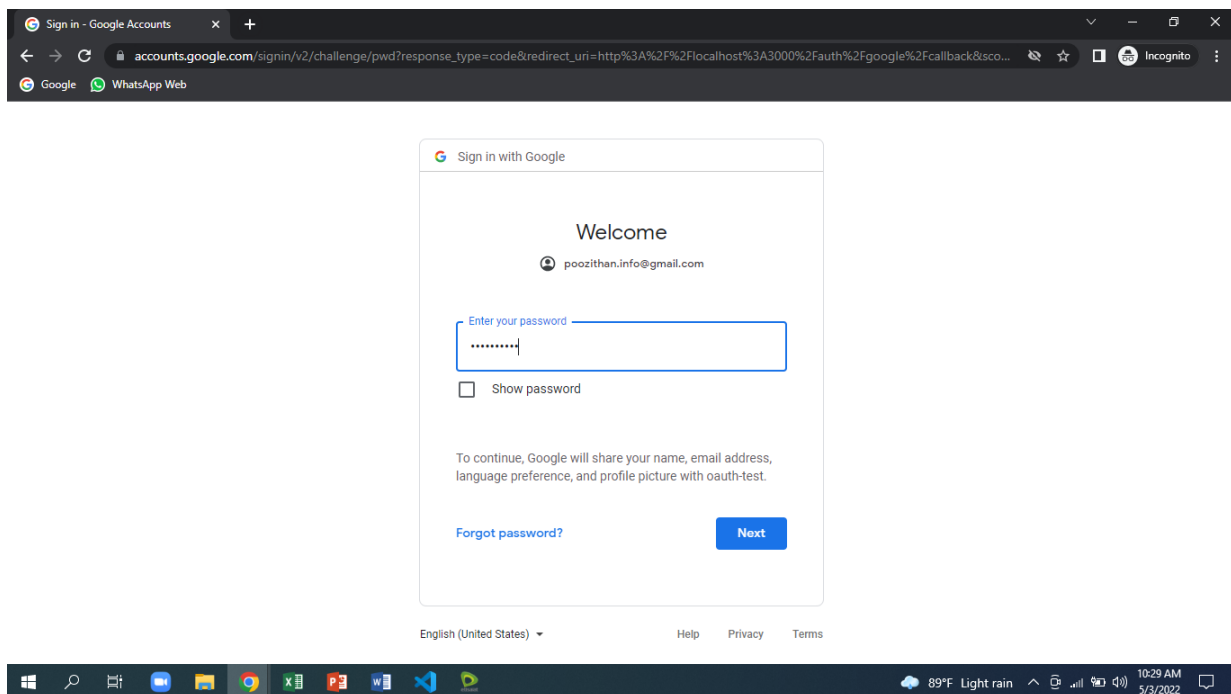
- Go to the Google chrome browser and type “localhost:3000” in the URL bar, the below page will be loaded



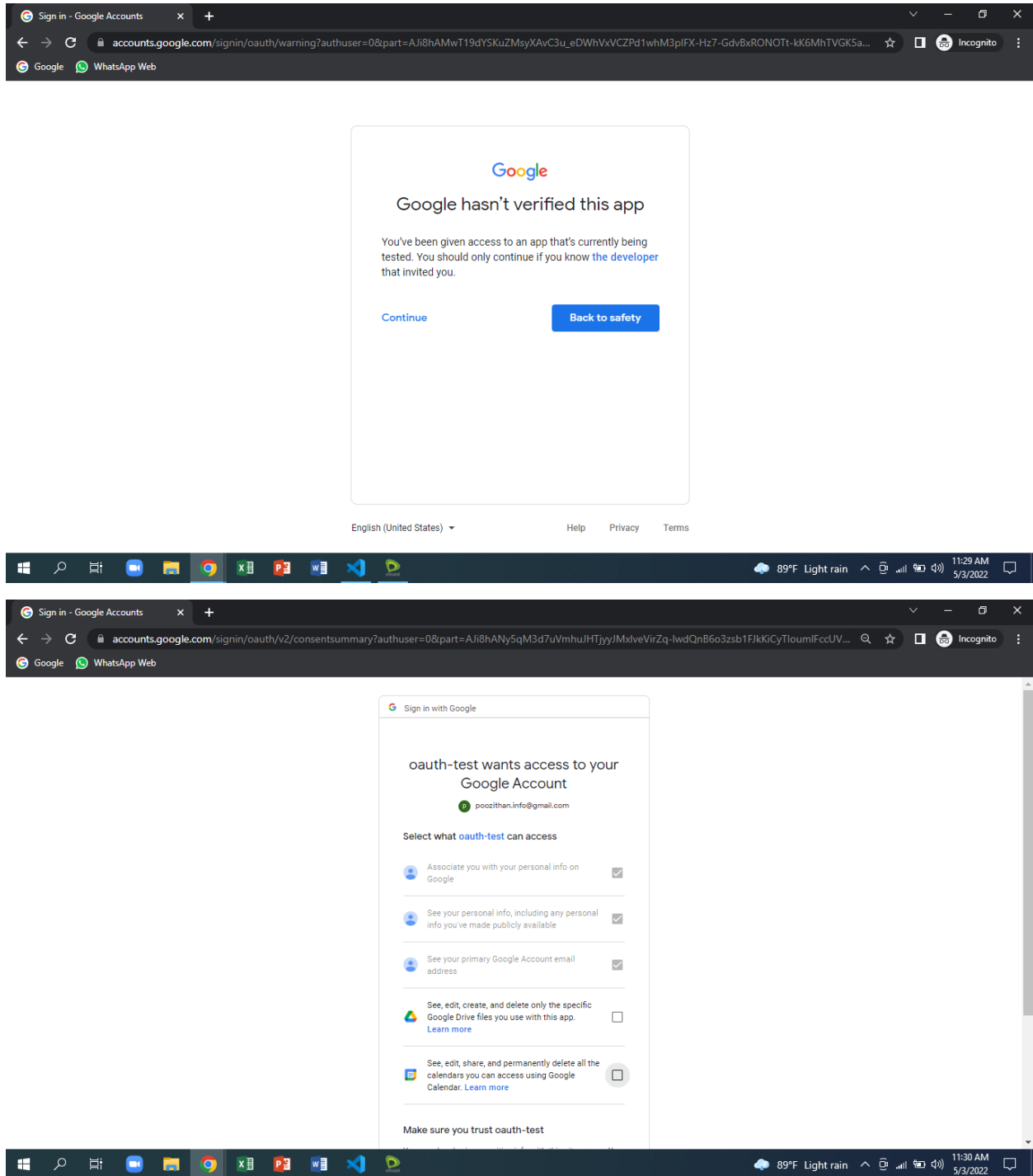
- Click the “Signin with Google” Button the below page will be loaded.

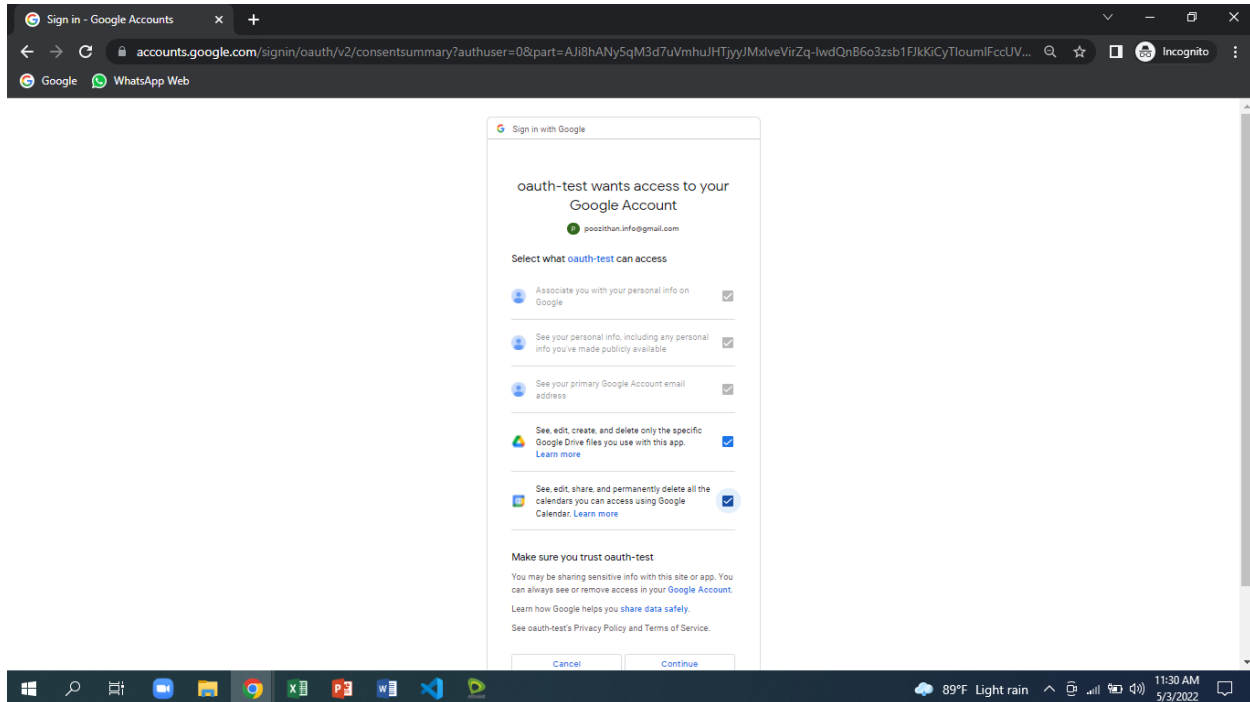


- Enter User name and password, the page will be redirect to the authentication via Google authentication

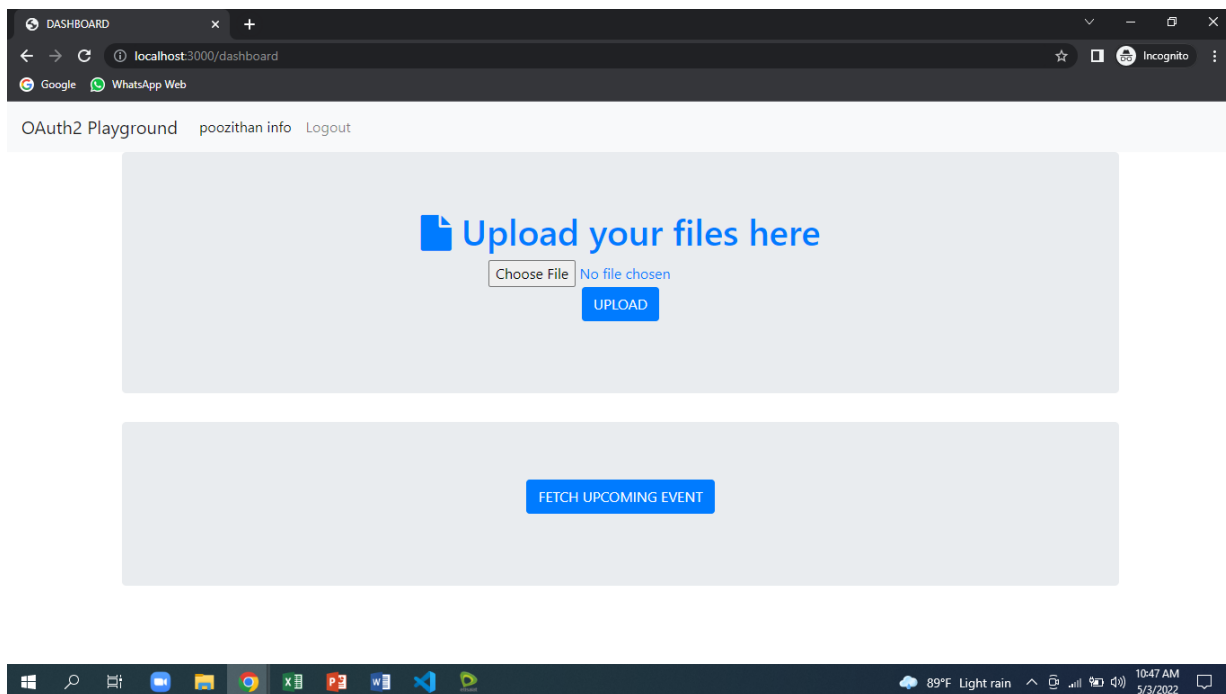


- Once user enter the user credentials, it redirect to the authentication process and user allow to access the google account



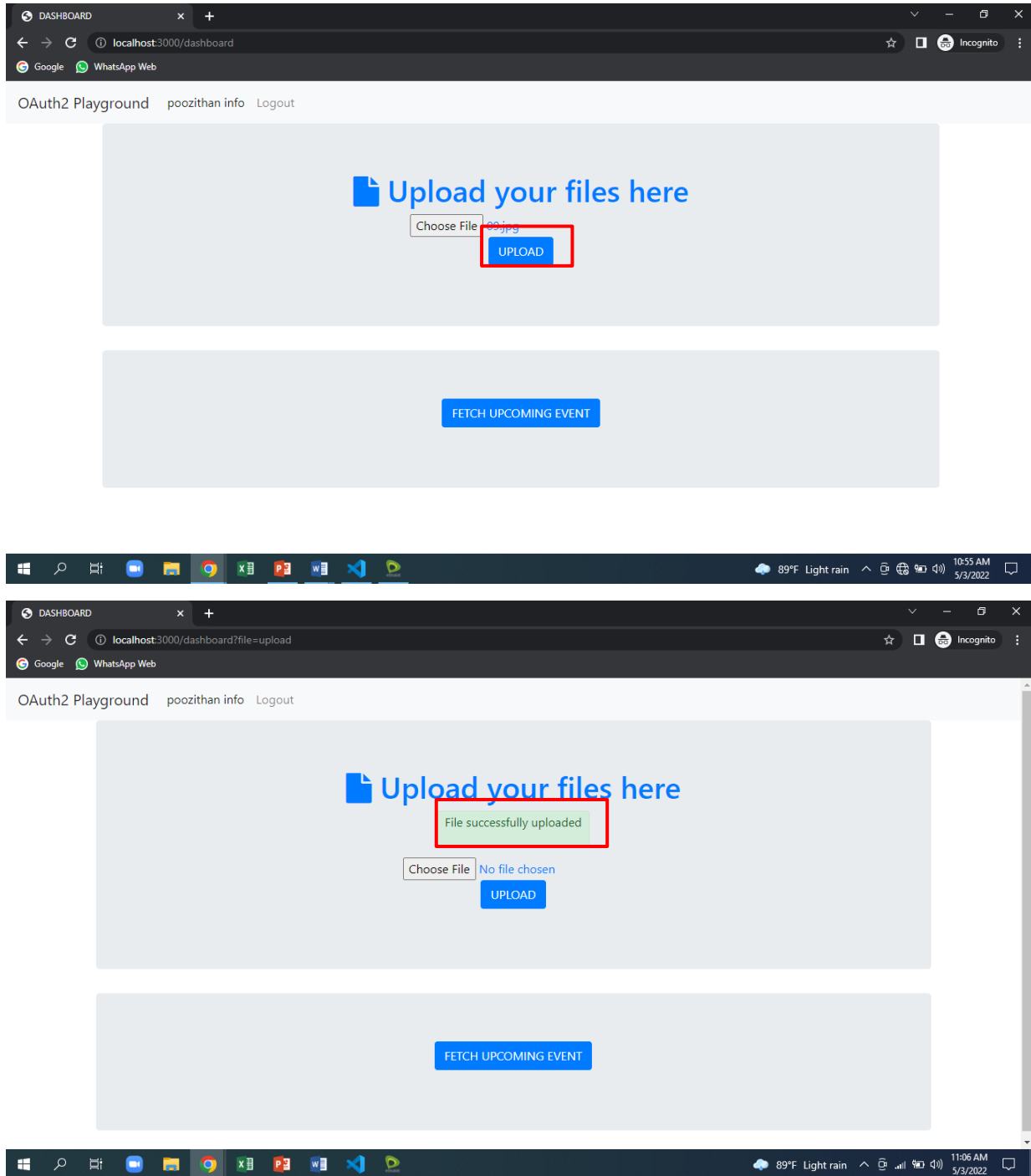


- After the authentication process, the main page will be loaded.

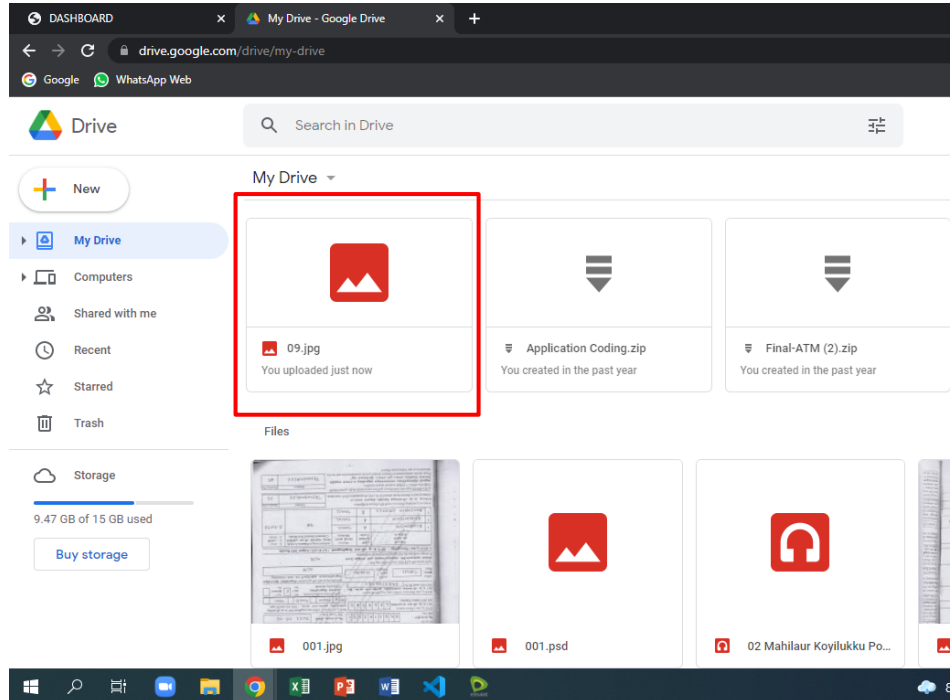


Service 2 : Upload the files to the google drive

- If you want to upload the files, click the “Choose file” button icon and select the files from your device. After the selection of files, click the upload button. Once the uploading completed, ou will receive the message “File successful uploaded”

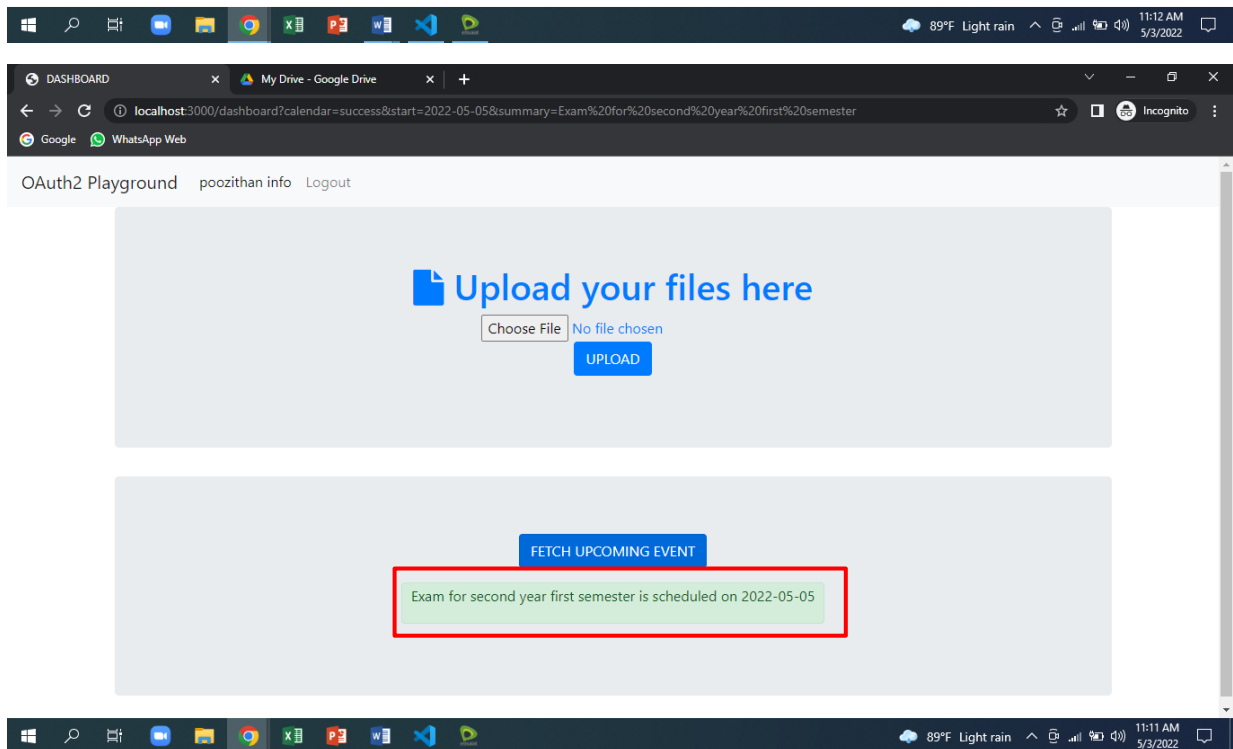
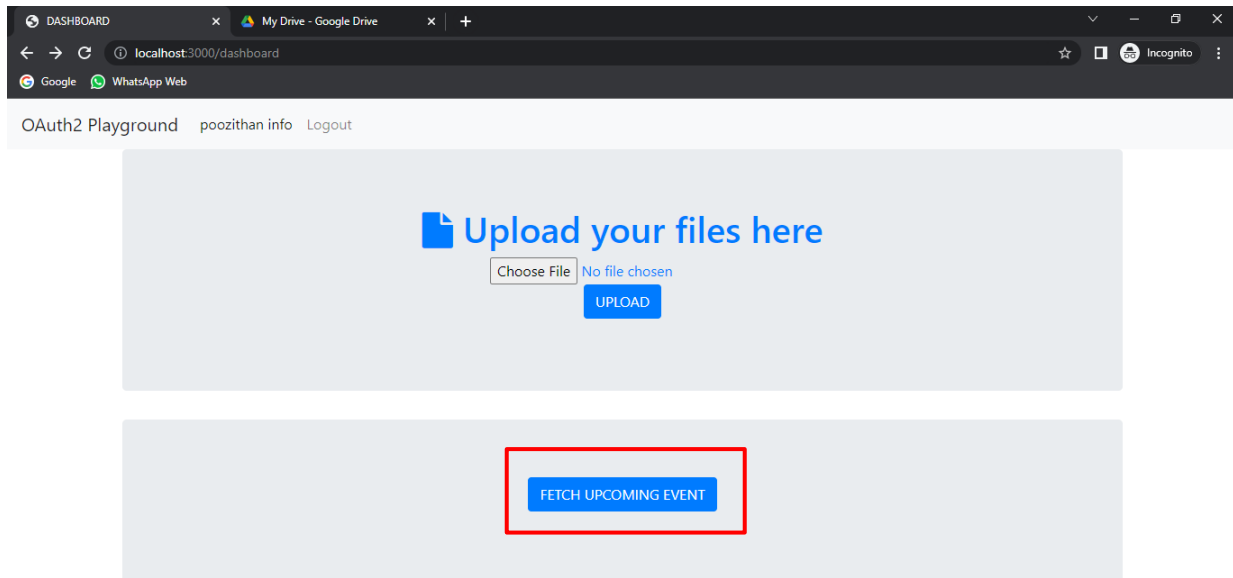


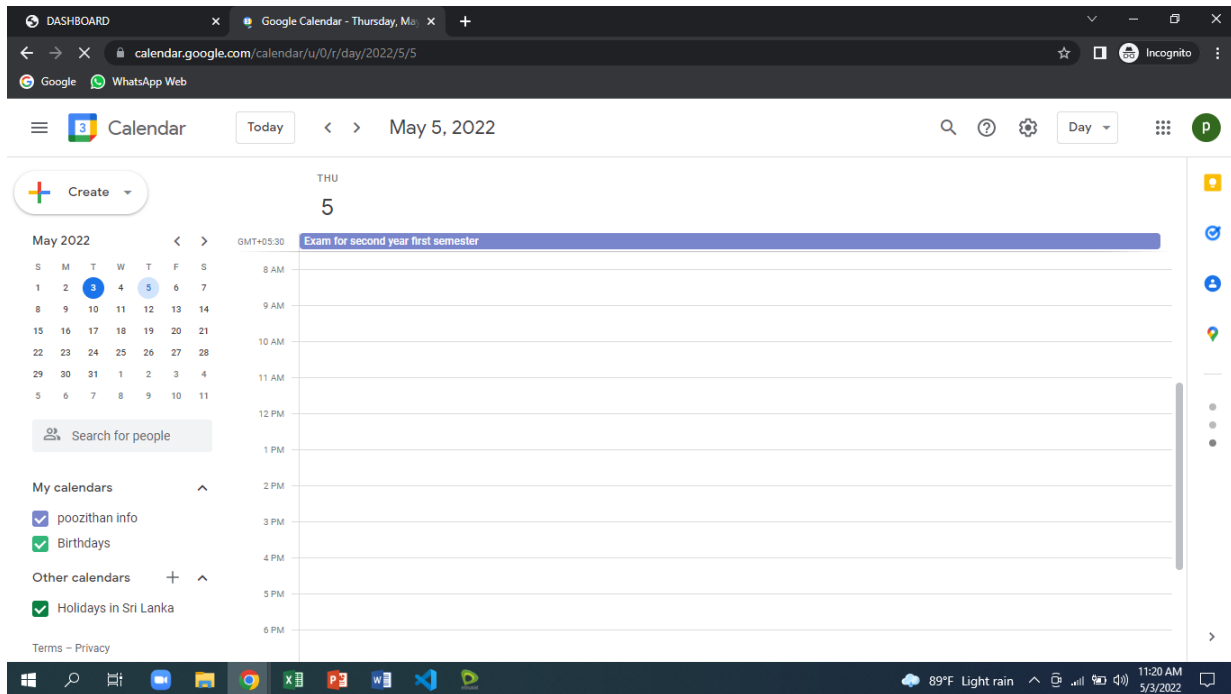
- After completed you will see your files has been successfully uploaded into google Drive.



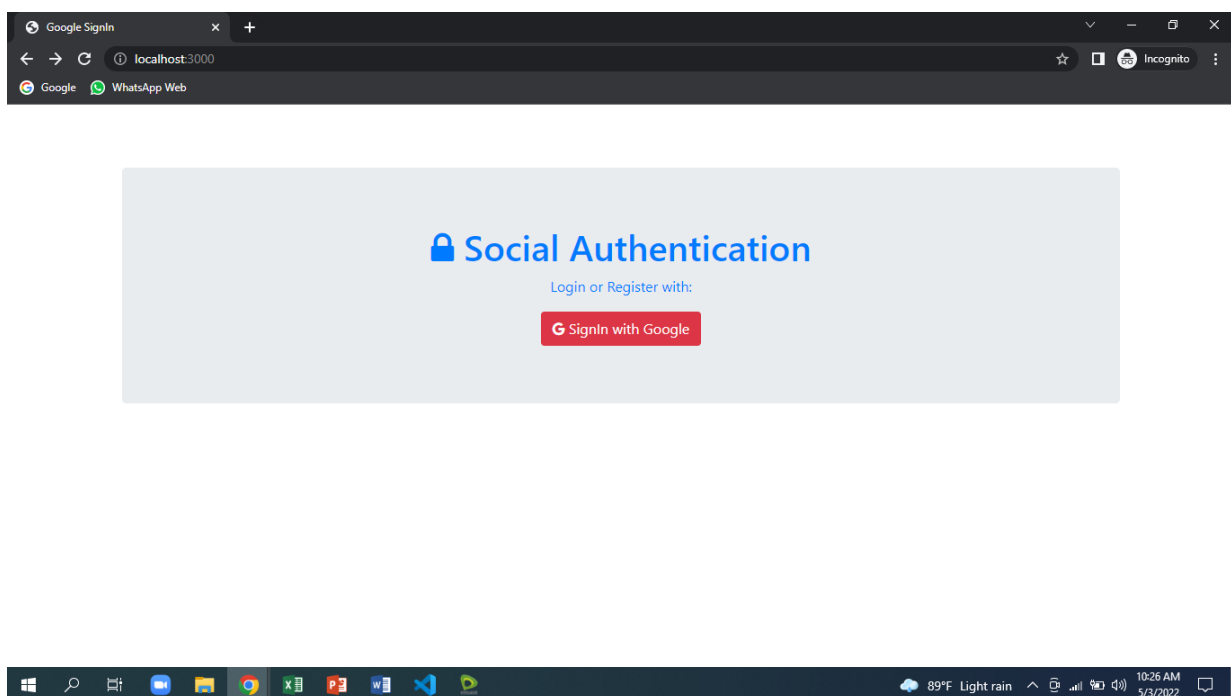
Service 3 : View the upcoming event to schedule the google calendar

- If you want to see the upcoming event that schedule in google calendar click the “FECTH UPCOMING EVENT” button. Then day to day events message will be shown.





- Then click the logout button the page will be redirect to the below page.



Appendix

1. Index.js

```
const express = require('express')
const homeRouter = require('./routes/home')
const authRouter = require('./routes/auth')
const __ = require('./middleware/passport')
const passport = require('passport')
const session = require('express-session')
const nunjucks = require('nunjucks')
const fileUpload = require('express-fileupload')

// init app
let app = express()
const port = process.env.PORT
app.listen(process.env.PORT, '0.0.0.0', () => console.log(`server is running on
${process.env.PORT}`));

// init view
nunjucks.configure('views', {
  autoescape: true,
  express: app
});

// init static
app.use('/static', express.static('public'))

// init session
app.use(session({
  secret: process.env.SECRET_KEY,    //decode or encode session
  resave: false,
  saveUninitialized: false,
  cookie: {
    maxAge: 2 * 60 * 1000
  }
}));

// init passport
app.use(passport.initialize())
app.use(passport.session())
```

```
//current User
app.use(function (req, res, next) {
  res.locals.currentUser = req.user;
  next();
})
```

```
// file upload
app.use(fileUpload());
```

```
// init routes
app.use("/", homeRouter) // home
app.use('/auth', authRouter) // auth
```

2. Passport.js

```
const passport = require('passport')
const GoogleStrategy = require('passport-google-oauth20')

let strategy = new GoogleStrategy(
  {
    clientID: process.env.CLIENT_ID,
    clientSecret: process.env.CLIENT_SECRET,
    callbackURL: process.env.LOGIN_REDIRECT_URL,
    passReqToCallback: true
  },
  (__request, accessToken, __refreshToken, profile, done) => {
    //save data in user session
    user = {
      "accesstoken": accessToken, // this token will be used to request google resources
      'googleID': profile.id,
      'name': profile.displayName,
      'email': profile._json.email
    }
    done(null, user)
  }
)

let serializerFunction = function (user, done) {
  let sessionUser = {
    _id: user.googleID,
    accessToken: user.accesstoken,
    name: user.name,
    email: user.email
  }
```

```

    }
    done(null, sessionUser)
  }

```

```

let deserializerFunction = function (user, done) {
  done(null, user)
}

```

```

passport.serializeUser(serializerFunction) // middleware to save user info in the session
passport.deserializeUser(deserializerFunction) // middleware to retrieve user info in the session
passport.use(strategy)

```

3. Route

- auth.js

```

const { Router } = require('express')
const passport = require('passport')

```

```

let router = Router()

```

```

router.get('/login', (req, res) => {
  console.log("login", req.user)
  if (req.user) {
    console.log("user is already loggedin")
    res.redirect('/dashboard') // user already authenticated hence redirect the user to dashboard
  }
  else {
    console.log("redirecting user to google signin")
    res.render('auth.ejs', { 'title': 'Login' }) // user is not authenticated hence render the login page
  }
}

```

```

}) // handle get requests on /auth/login endpoint

```

```

router.get('/login/google', passport.authenticate("google", {
  scope: ['profile', "https://www.googleapis.com/auth/drive.file",
  "https://www.googleapis.com/auth/calendar", "email"]
})) // handle get request on /auth/login/google endpoint

```

```
router.get('/google/callback', passport.authenticate('google'), (__req, res) => {
  res.redirect('/dashboard') // user successfully logged in hence redirect the user to
  dashboard
}) // handle get requests on /auth/google/callback endpoint

router.get('/logout', (req, res) => {
  req.logout(); // passport function to handle logout
  res.redirect('/') // redirect the user to home page
}) // handle get requests on /auth/logout endpoint

module.exports = router
•   home.js
const { Router } = require('express')
const { google } = require('googleapis')

const router = Router()

// middleware to check authentication
function isLoggedIn(req, res, next) {
  if (req.isAuthenticated()) {
    return next();
  }
  res.redirect("/auth/login")
}

router.get("/", (req, res) => {
  if (typeof req.user == "undefined") {
    res.redirect('/dashboard')
  } else {
    res.redirect('/auth/login')
  }
})

router.get('/dashboard', isLoggedIn, (req, res) => {

  let dashboardArgs = {
    title: 'Dashboard',
    googleid: req.user._id, // google user id
    name: req.user.name,    // google username
    email: req.user.email   // gmail id
  }
})
```

```

    if (req.query.file !== undefined) { // will have valid value upon file upload completion.
    This is populated by
      // file upload middleware
      if (req.query.file === "upload") // upload successful
        dashboardArgs.file = "uploaded"
      else if (req.query.file === "notupload") // upload failed
        dashboardArgs.file = "notuploaded"
    }

    if (req.query.calendar !== undefined) { // will have a valid value upon a calendar event
    retrieval request
      if (req.query.calendar === "success") { // successfully retrieved the event
        dashboardArgs.calendar = "success"
        dashboardArgs.start = req.query.start // event starttime
        dashboardArgs.summary = req.query.summary // event summary
      } else if (req.query.calendar === "failed") { // failed to retrieve the event
        dashboardArgs.calendar = "failed"
      } else if (req.query.calendar === "empty") { // no events in the calendar
        dashboardArgs.calendar = "empty"
      }
    }
    res.render('dashboard.ejs', dashboardArgs) // render the dashboard
  })

  router.post('/upload', isLoggedIn, async (req, res) => { // handler for google drive file
  upload
    try {

      // configure google drive connection with client token
      const oauth2Client = new google.auth.OAuth2()
      oauth2Client.setCredentials({
        'access_token': req.user.accessToken // accessToken from the google login
      });

      const drive = google.drive({
        version: 'v3',
        auth: oauth2Client
      });

      let { name: filename, mimetype, data } = req.files.file_upload // deserialize the file

```

object

```

const driveResponse = await drive.files.create({
  requestBody: {
    name: filename,
    mimeType: mimetype
  },
  media: {
    mimeType: mimetype,
    body: Buffer.from(data).toString()
  }
});

if (driveResponse.status === 200)
  res.redirect('/dashboard?file=upload') // successfully uploading the file
else
  res.redirect('/dashboard?file=notupload') // file upload failed
} catch (error) {
  console.log(error)
  res.redirect('/dashboard?file=notupload') // file upload failed
}
})

router.get('/calendar', isLoggedIn, async (req, res) => {
  try {
    // configure google calendar connection with client token
    const oauth2Client = new google.auth.OAuth2()
    oauth2Client.setCredentials({
      'access_token': req.user.accessToken // accessToken from the google login
    });

    const calendar = google.calendar({
      version: 'v3',
      auth: oauth2Client
    });

    const calendar_list_response = await calendar.calendarList.list(maxResults = 1) //
    select the first calendar
    console.log(calendar_list_response.data.items[0].id)

    const eventList = await calendar.events.list({ // retrieve the event list
      calendarId: calendar_list_response.data.items[0].id,

```



```

        timeMin: (new Date()).toISOString(),
        maxResults: 1,
        singleEvents: true,
        orderBy: 'startTime',
    })
    console.log(eventList.data.items)
    if (eventList.data.items.length > 0) {
        nextEventObj = eventList.data.items[0] // select the upcoming event
        console.log(nextEventObj)
        upcomingEvent = {
            start: nextEventObj.start.dateTime || nextEventObj.start.date,
            summary: nextEventObj.summary
        }
        console.log("upcoming event", upcomingEvent)

res.redirect(`/dashboard?calendar=success&start=${upcomingEvent.start}&summary=${
upcomingEvent.summary}`) // succefully retrieved hence redirect

    } else {
        res.redirect(`/dashboard?calendar=empty`) // no calendar events
    }
} catch (error) {
    console.log(error)
    res.redirect(`/dashboard?calendar=failed`) // file upload failed
}

})

module.exports = router

```

4. GUI Design

- **auth.ejs**

```
<!-- views/pages/auth.ejs -->
```

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<title>{{ title }}</title>
```

```
<link rel="stylesheet"
```

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" />
```

```

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<style>
  body {
    padding-top: 70px;
  }
</style>
</head>

<body>
  <div class="container">
    <div class="jumbotron text-center text-primary">
      <h1><span class="fa fa-lock"></span> Social Authentication</h1>
      <p>Login or Register with:</p>
      <a href="/auth/login/google" class="btn btn-danger"><span class="fa fa-
google"></span> SignIn with
        Google</a>
    </div>
  </div>
</body>

</html>
•      dashboard.ejs
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>{ {title} }</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" />
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
</head>

<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">OAuth2 Playground</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav"

```

```

        aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            <li class="nav-item active">
                <a class="nav-link" href="#">{{ name }} <span class="sr-
only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/auth/logout">Logout</a>
            </li>
        </ul>
    </div>
</nav>

<div class="container">
    <div class="jumbotron text-center text-primary">
        <h1><span class="fa fa-file"></span> Upload your files here</h1>
        <div>
            {% if (file == "uploaded") %}
            <div class="alert alert-success" style="padding: 2px 10px; display: inline-
block; width: fit-content">
                <p>File successfully uploaded</p>
            </div>
            {% elif (file == "notuploaded") %}
            <div class="alert alert-danger" style="padding: 2px 10px; display: inline-block;
width: fit-content">
                <p>File failed to upload please try again</p>
            </div>
            {% endif %}
        </div>
        <form action="/upload" method="post" enctype="multipart/form-data">
            <div class="form-group">
                <input style="display: inline-block; width: fit-content" type="file"
class="form-control-file"
                required name="file_upload" id="file_upload">
                <button style="display: block; margin: 0 auto" class="btn btn-primary mb-2"
type="submit">UPLOAD</button>
            </div>
        </form>
    </div>
</div>

```

```

</div>
<div class="jumbotron text-center text-primary">
  <form action="/calendar" method="get" enctype="multipart/form-data">
    <div class="form-group">
      <button style="display: block; margin: 0 auto" class="btn btn-primary mb-2"
type="submit">FETCH
      UPCOMING EVENT</button>
    </div>
  </form>
  <div>
    { % if (calendar == "success") % }
    <div class="alert alert-success" style="padding: 2px 10px; display: inline-
block; width: fit-content">
      <p> {{ summary }} is scheduled on {{ start }} </p>

    </div>
    { % elif (calendar == "failed") % }
    <div class="alert alert-danger" style="padding: 2px 10px; display: inline-block;
width: fit-content">
      <p>Failed to fetch upcoming event</p>
    </div>
    { % elif (calendar == "empty") % }
    <div class="alert alert-success" style="padding: 2px 10px; display: inline-
block; width: fit-content">
      <p>No events found</p>
    </div>
    { % endif % }
  </div>
</div>
</div>
</body>

</html>

```