PRO-C221

VIDEO CHAT APP-SMTP

What is our GOAL for this CLASS?

In this class, we have learned about **SMTP** and how to send emails with the Gmail account using **nodemailer**

What did we ACHIEVE in the class TODAY?

- Learned about **SMTP**
- Learned to send emails through **Gmail** using **nodemailer**

Which CONCEPTS/ CODING BLOCKS did we cover today?

- We used the **Simple Mail Transfer Protocol**
- We used **the nodemailer Module**

Understanding concepts:

Simple Mail Transfer Protocol:

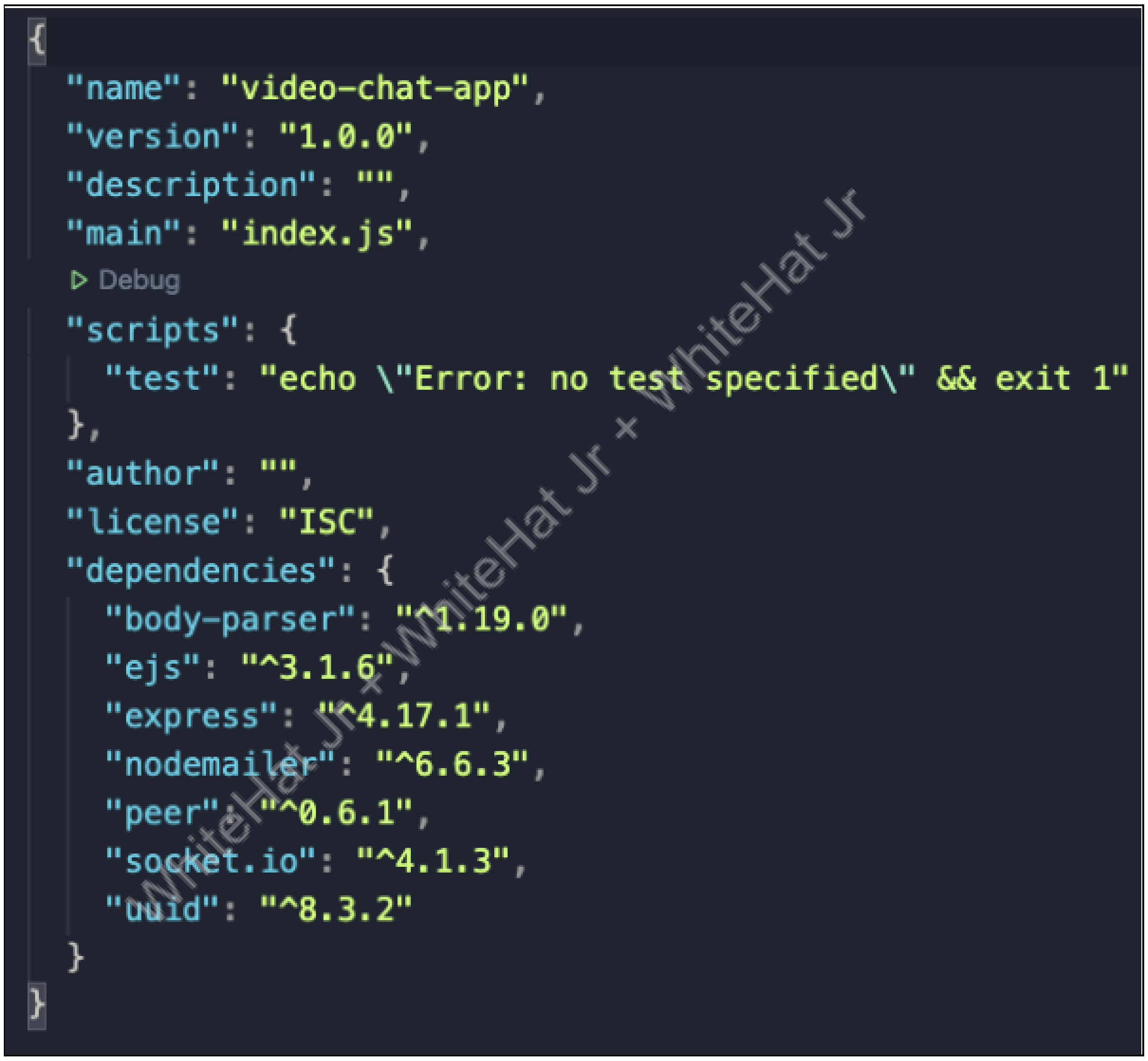
Simple Mail Transfer Protocol is an internet-based standard communication protocol for electronic mail transmissions. This protocol is used by all email services out there, including Gmail.

The Nodemailer Module

The Nodemailer module makes it easy to send emails from your computer. The Nodemailer module can be downloaded and installed using npm.

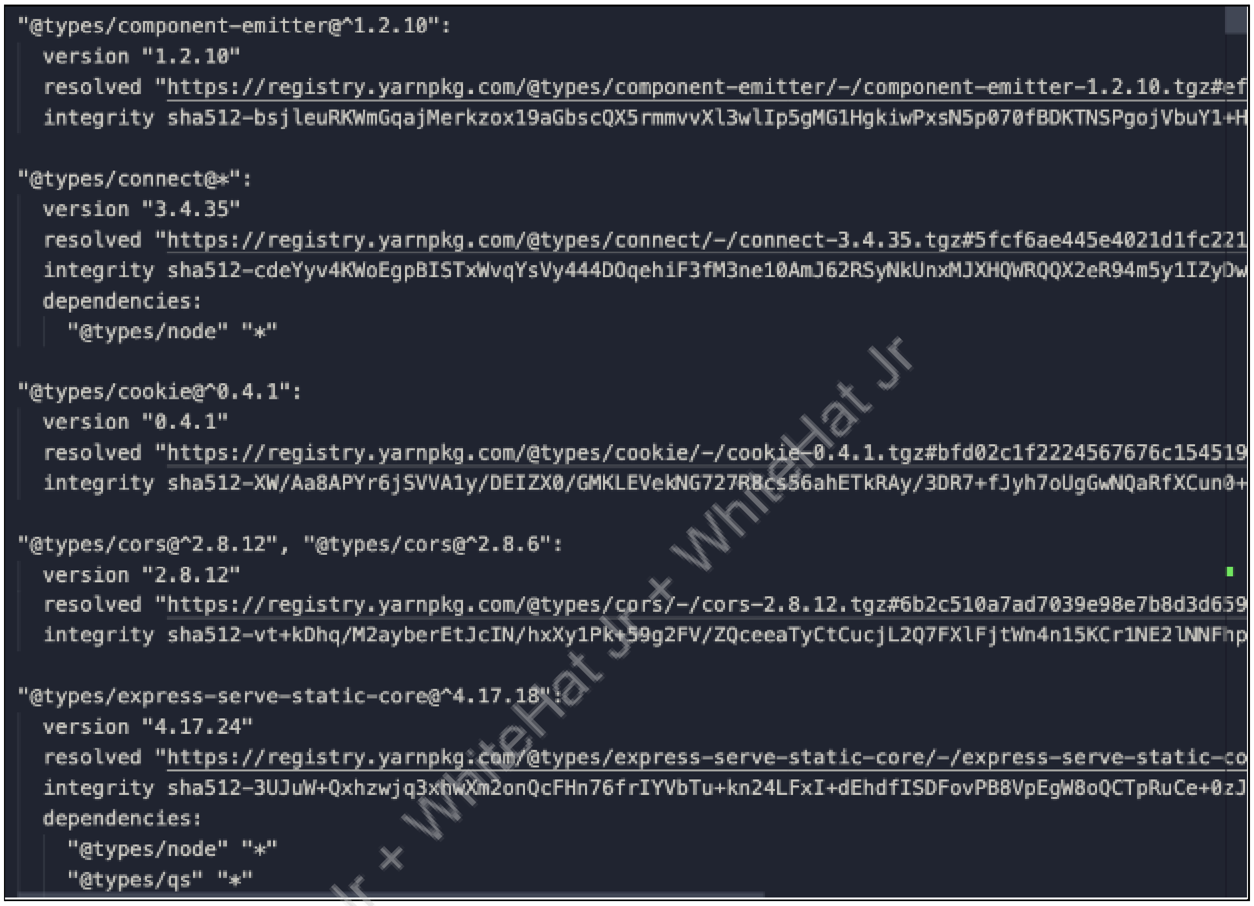
How did we DO the activities?

1. Update **package.json** to add **nodemailer**



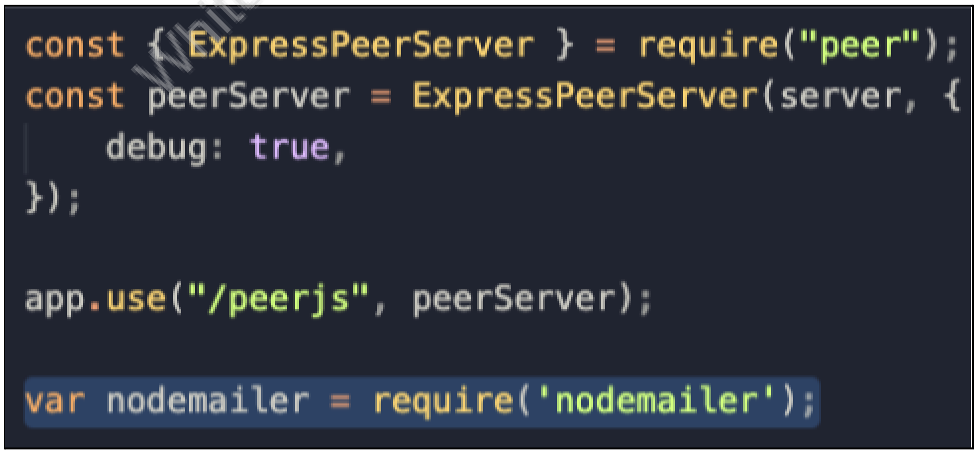
```
{
  "name": "video-chat-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "ejs": "^3.1.6",
    "express": "^4.17.1",
    "nodemailer": "^6.6.3",
    "peer": "^0.6.1",
    "socket.io": "^4.1.3",
    "uuid": "^8.3.2"
  }
}
```

2. Update **yarn.lock** to add **nodemailer**



```
"@types/component-emitter@^1.2.10":
  version "1.2.10"
  resolved "https://registry.yarnpkg.com/@types/component-emitter/-/component-emitter-1.2.10.tgz#ef
  integrity sha512-bsjleuRKWmGqajMerkzox19aGbscQX5rmmvvXl3wlIp5gMG1HgkiwPxsN5p070fBDKTNSPgojVbuY1+H

"@types/connect@*":
  version "3.4.35"
  resolved "https://registry.yarnpkg.com/@types/connect/-/connect-3.4.35.tgz#5fcf6ae445e4021d1fc221
  integrity sha512-cdeYyv4KWoEgpBISTxWvqYsVy444DOqehiF3fM3ne10AmJ62RSyNkUnxMJXHQWRQQX2eR94m5y1IZyDw
  dependencies:
    "@types/node" "*"

"@types/cookie@^0.4.1":
  version "0.4.1"
  resolved "https://registry.yarnpkg.com/@types/cookie/-/cookie-0.4.1.tgz#bfd02c1f2224567676c154519
  integrity sha512-XW/Aa8APYr6jSVVA1y/DEIZX0/GMKLEVekNG727R8cs56ahETkRAy/3DR7+fJyh7oUgGwNQaRfXCun0+

"@types/cors@^2.8.12", "@types/cors@^2.8.6":
  version "2.8.12"
  resolved "https://registry.yarnpkg.com/@types/cors/-/cors-2.8.12.tgz#6b2c510a7ad7039e98e7b8d3d659
  integrity sha512-vt+kDhq/M2ayberEtJcIN/hxXy1Pk+59g2FV/ZQceeaTyCtCucjL2Q7FXlFjtWn4n15KCr1NE2lNNFhp

"@types/express-serve-static-core@^4.17.18":
  version "4.17.24"
  resolved "https://registry.yarnpkg.com/@types/express-serve-static-core/-/express-serve-static-co
  integrity sha512-3UJuW+Qxhzwjq3xhwXm2onQcFHn76frIYVbTu+kn24LFxI+dEhdfISDFovPB8VpEgW8oQCTpRuCe+0zJ
  dependencies:
    "@types/node" "*"
    "@types/qs" "*"
```

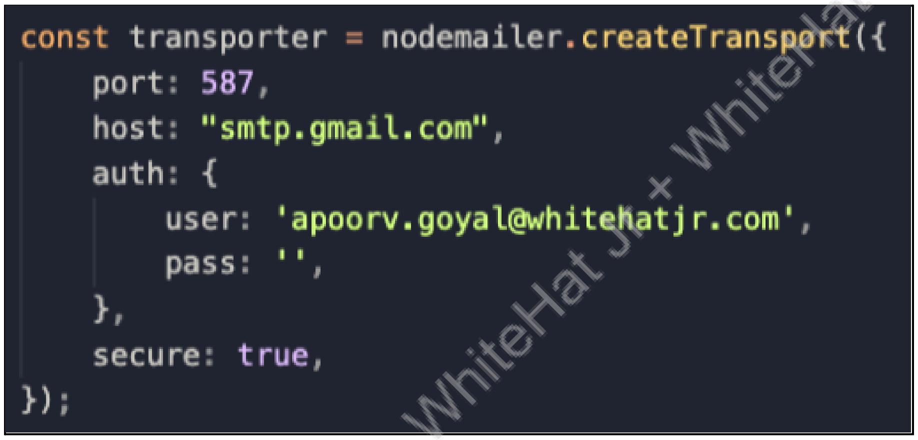
3. In **server.js** add **require('nodemailer')**



```
const { ExpressPeerServer } = require("peer");
const peerServer = ExpressPeerServer(server, {
    debug: true,
});

app.use("/peerjs", peerServer);

var nodemailer = require('nodemailer');
```

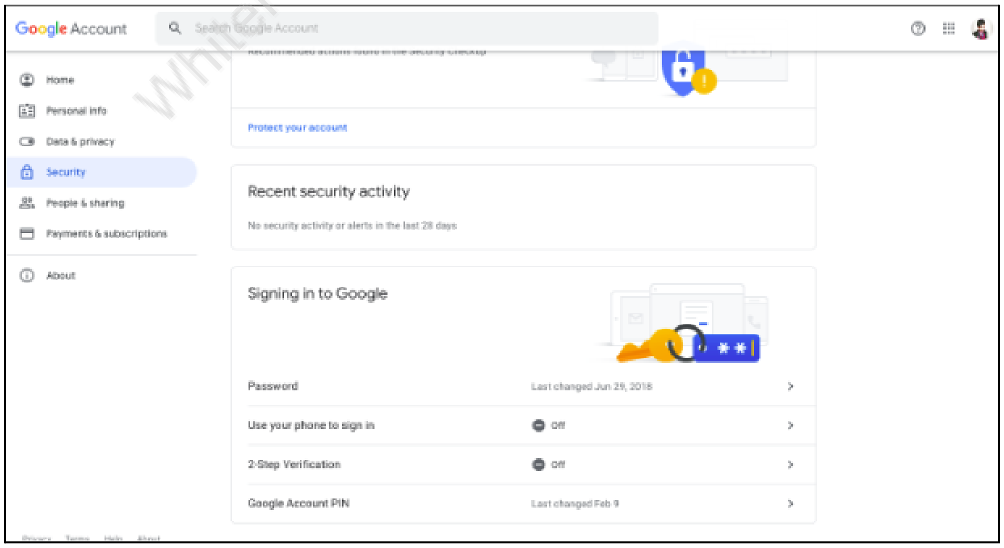
4. Create a **transporter** for our mailer in which we are calling the **nodemailer.createTransport()** function.

   Parameters for the function:
   - **port** - Port number that it should use - **587**
   - **host** - **smtp.gmail.com** for Gmail's **SMTP**
   - **auth** - That contains a
     - **user: email ID**
     - **pass: password** through which it should send an email
   - **secure** - **true**, to send encrypted emails



```
const transporter = nodemailer.createTransport({
    port: 587,
    host: "smtp.gmail.com",
    auth: {
        user: 'apoorv.goyal@whitehatjr.com',
        pass: '',
    },
    secure: true,
});
```

5. Use **app passwords** to generate **passwords** that can be used in our apps to access Google services such as Gmail:



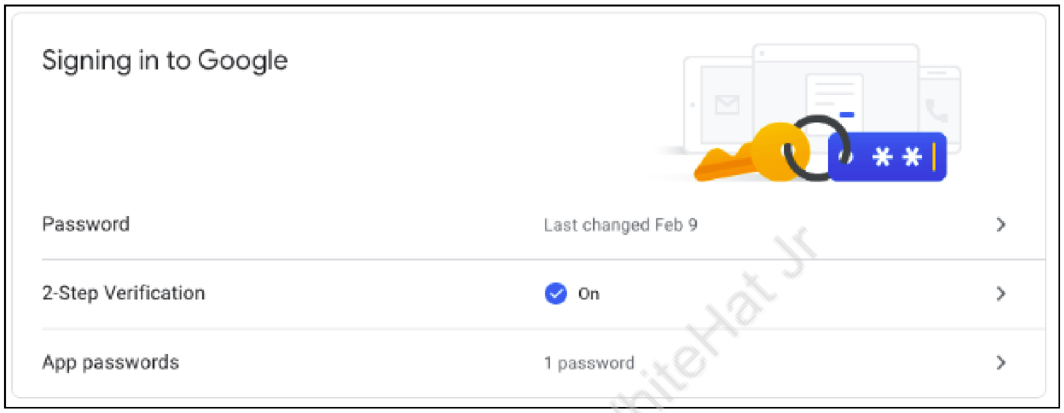
Recent security activity

No security activity or alerts in the last 28 days

Signing in to Google

Password	Last changed Jun 29, 2018	>
Use your phone to sign in	Off	>
2-Step Verification	Off	>
Google Account PIN	Last changed Feb 9	>

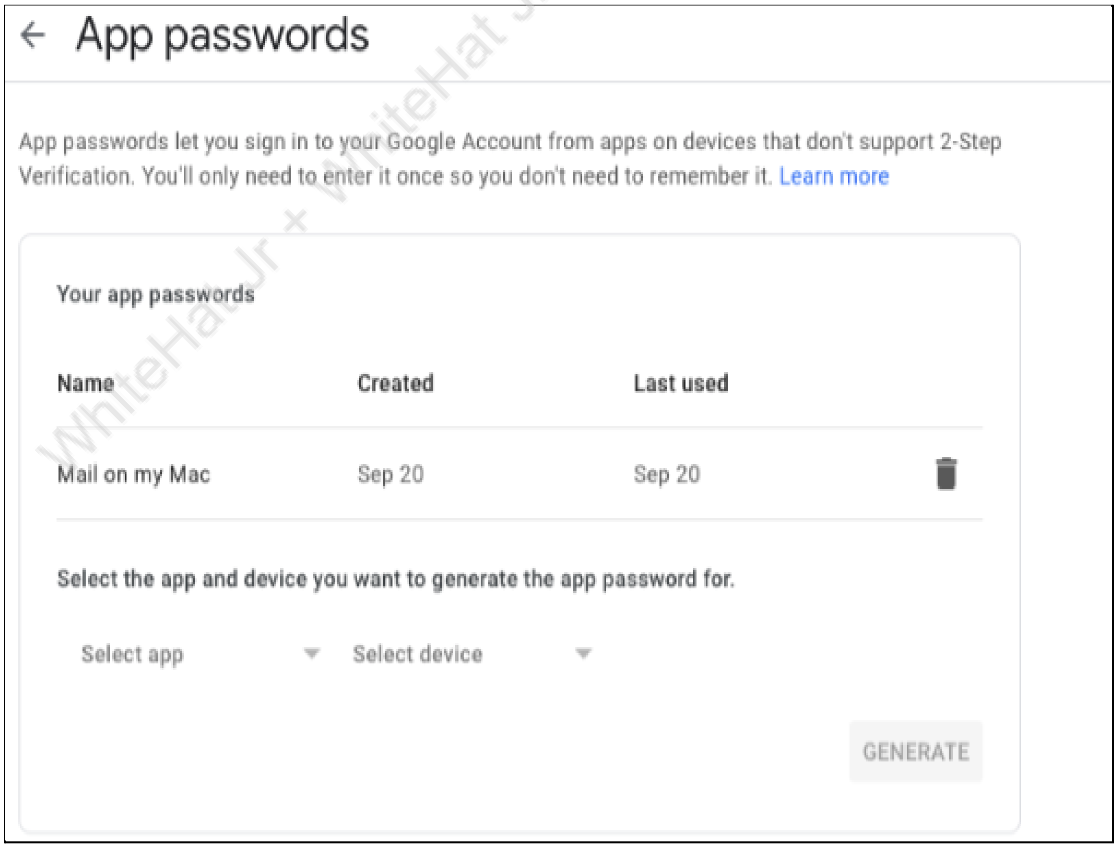
- Ensure that your **2-Step Verification** is turned on in the Signing into Google section



Signing in to Google

Password	Last changed Feb 9	>
2-Step Verification	On	>
App passwords	1 password	>

- Click on **App Passwords**



App passwords

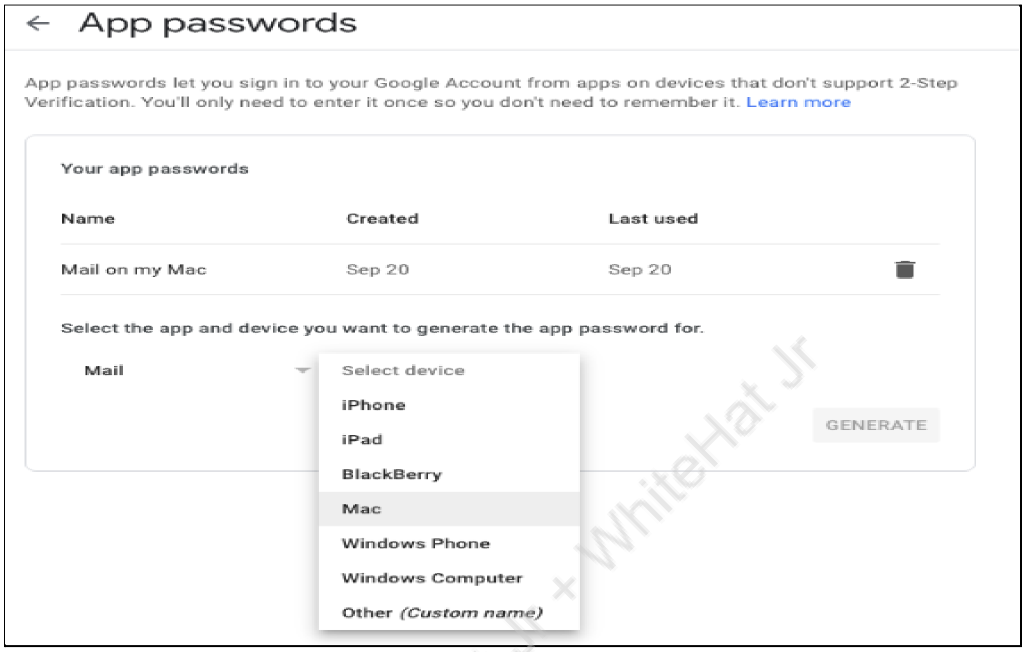
App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. Learn more

Your app passwords

Name	Created	Last used
Mail on my Mac	Sep 20	Sep 20

Select the app and device you want to generate the app password for.

- Select **Mail** in the app and **Mac (or Windows)** for your device.



App passwords

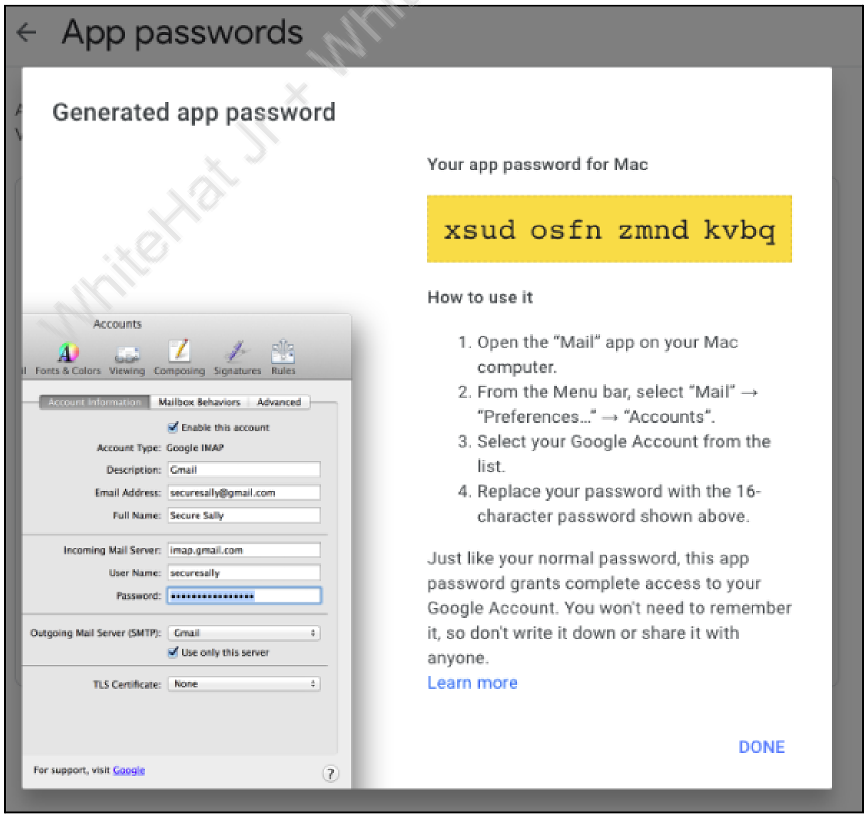
App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. Learn more

Your app passwords

Name	Created	Last used
Mail on my Mac	Sep 20	Sep 20

Select the app and device you want to generate the app password for.

- Click on **Generate** button and you will see your **password**



App passwords

Generated app password

Your app password for Mac

xsud osfn zmnd kvbq

How to use it

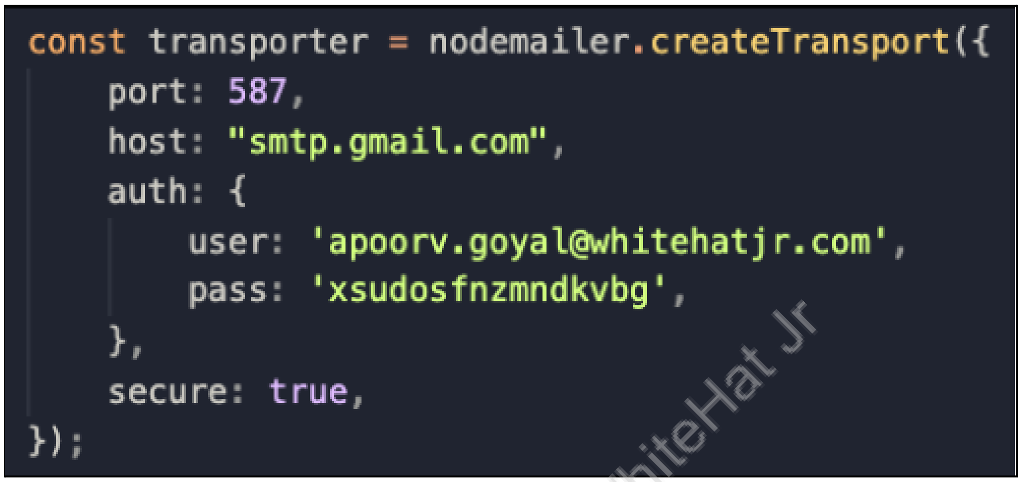
1. Open the "Mail" app on your Mac computer.
2. From the Menu bar, select "Mail" → "Preferences..." → "Accounts".
3. Select your Google Account from the list.
4. Replace your password with the 16-character password shown above.

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.
Learn more

Accounts

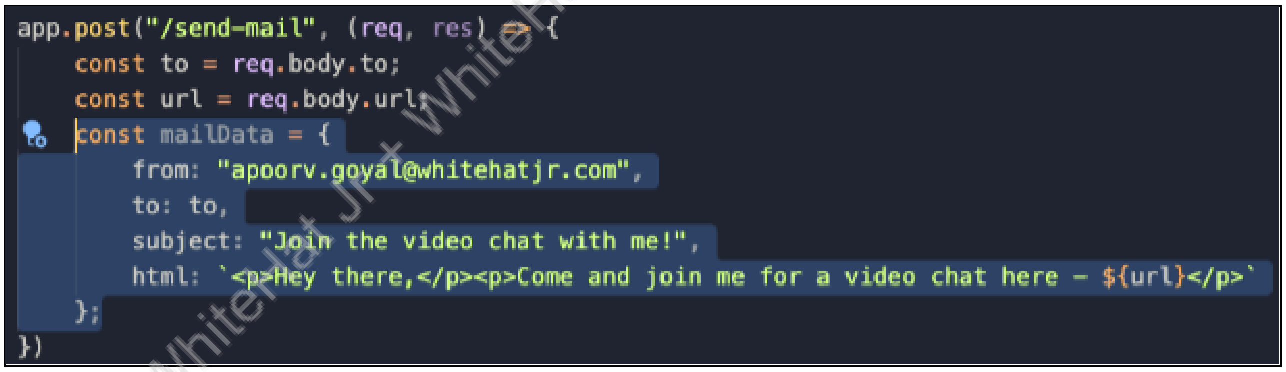
Enable this account
Account Type: Google IMAP
Description: Gmail
Email Address: securesally@gmail.com
Full Name: Secure Sally
Incoming Mail Server: imap.gmail.com
User Name: securesally
Password: ••••••••••••••••
Outgoing Mail Server (SMTP): Gmail
Use only this server
TLS Certificate: None
For support, visit Google

- Copy the **password**
- Paste it in the "**pass'** in our **transporter's** code.



```
const transporter = nodemailer.createTransport({
    port: 587,
    host: "smtp.gmail.com",
    auth: {
        user: 'apoorv.goyal@whitehatjr.com',
        pass: 'xsudosfnzmndkvbg',
    },
    secure: true,
});
```

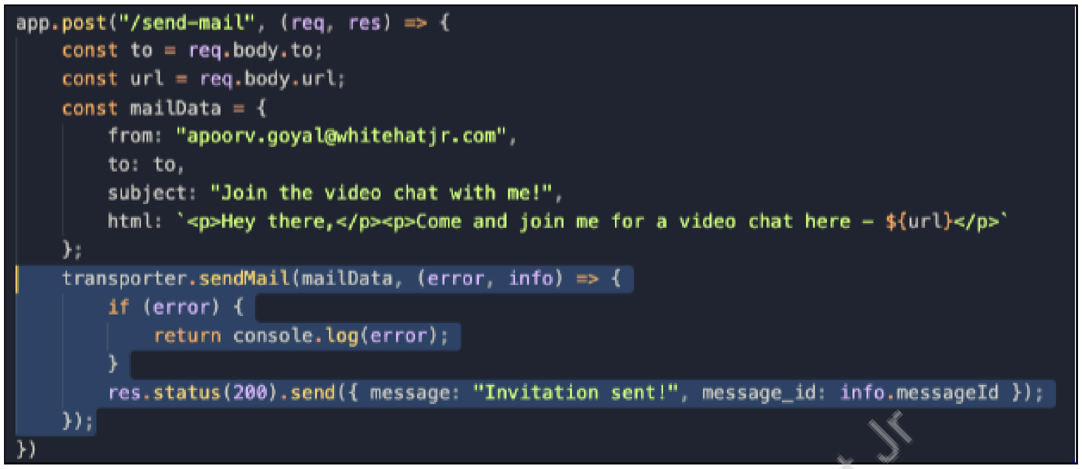
6. Create an **API**, that can receive a **POST** request from the client with the room's **URL** and **To** (the email id of the person we want to send the invite to) data. Create a **POST** request with **app.post()** function.



```
app.post("/send-mail", (req, res) => {
    const to = req.body.to;
    const url = req.body.url;
    const mailData = {
        from: "apoorv.goyal@whitehatjr.com",
        to: to,
        subject: "Join the video chat with me!",
        html: `<p>Hey there,</p><p>Come and join me for a video chat here - ${url}</p>`
    };
})
```

7. Use the **transporter.sendMail()** to send the email with our mailData as it's first argument, and we have an arrow function to handle success and errors.

   Inside the function,
   - **If** we have an **error**, we are logging the **error** to the console.
   - **else** we are sending a **response** with **res**, whose status code will be **200**, letting the client know that the Invitation is sent!



```
app.post("/send-mail", (req, res) => {
    const to = req.body.to;
    const url = req.body.url;
    const mailData = {
        from: "apoorv.goyal@whitehatjr.com",
        to: to,
        subject: "Join the video chat with me!",
        html: `<p>Hey there,</p><p>Come and join me for a video chat here - ${url}</p>`
    };
    transporter.sendMail(mailData, (error, info) => {
        if (error) {
            return console.log(error);
        }
        res.status(200).send({ message: "Invitation sent!", message_id: info.messageId });
    });
})
```

We have successfully learned about **SMTP** and how to send emails with the Gmail account using **nodemailer**

What's NEXT?

In the next class, we will be completing this app by connecting our **API** with our **client-side** and testing the functionality.

Expand Your Knowledge