

Media-Management-Platform

WT WS24/25

Goals:

Authorization

Goals

- Require user to authenticate
- Show different streams based on user

Camera Management

a

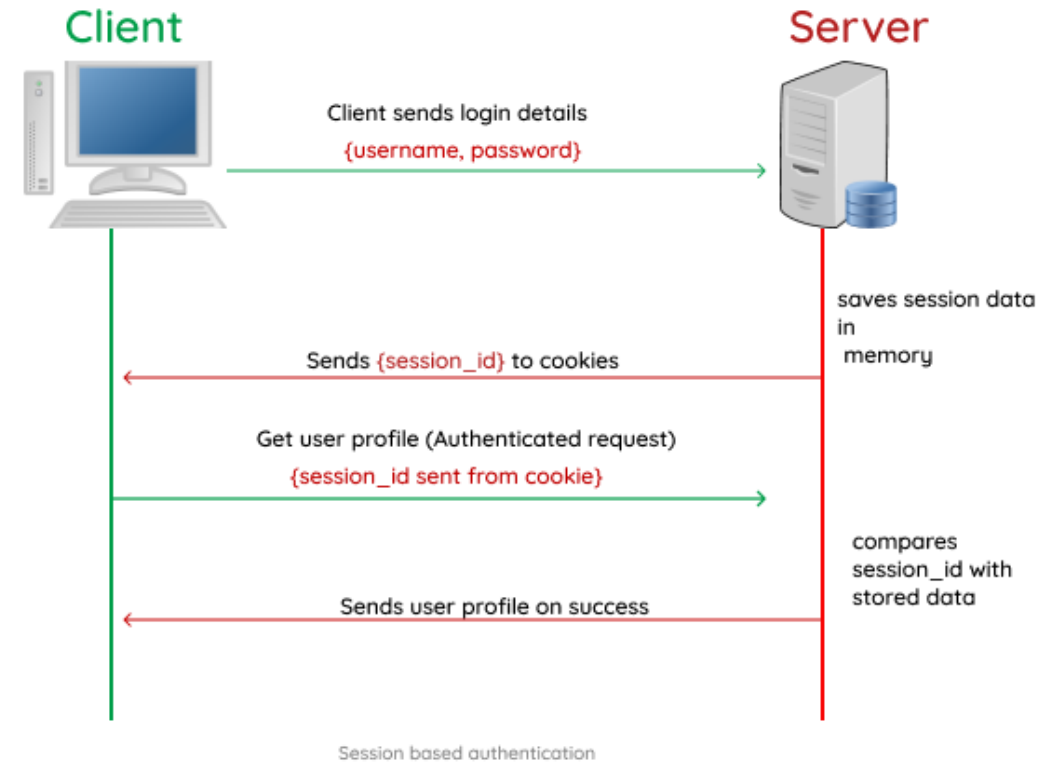
▶ 0:00



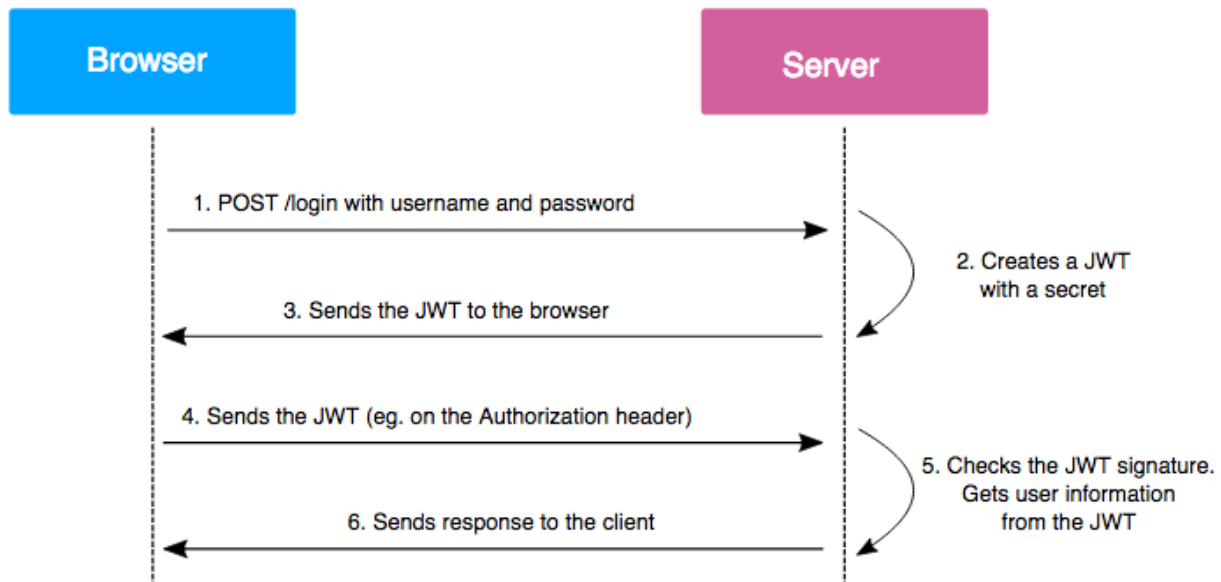
Backend

Authentication

- At first: Session based
- Too complicated with frontend
➔ JWT



JsonWebToken



Encoded

PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7InVzZXJuYW11IjoiaSIsInVzZXJJRCI6ImUwNDFhYTM1LWE3NWItNGY0Zi04YTQ0LWE2Nzc4Y2NmZTk5YSJ9LCJpYXQiOjE3MzcwMzI2ODMsImV4cCI6MTczNzIxOTA4M30.BiVxrS7tSwiX0GJifF94L0uBP7Nc2vJzZ36oGoGV0Ik

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "data": {    "username": "1",    "userID": "e041aa35-a75b-4f4f-8a44-a6778ccfe92a"  },  "iat": 1737132683,  "exp": 1737219083}
```

VERIFY SIGNATURE

HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
) ☐ secret base64 encoded

JWT Example: /login

```
app.post("/login", async (req, res) => {
  // Check if user exists
  const user = await findUser(req.body.username);
  if (!user) {
    return res.status(401).json({ message: 'Invalid credentials.' });
  }
  try {
    // Check if password is correct
    if (await bcrypt.compare(req.body.password, user.password)) {
      logger.info("Authentication OK");
      const token = jwt.sign(
        {
          data: {username: user.username, userID: user.uuid},
        },
        jwtSecret,
        {
          expiresIn: "24h",
        },
      );
      // Send token to client
      res.status(200).json({ token });
    } else {
      res.status(401).json({ message: 'Invalid credentials.' });
    }
  }
});
```

Request body:

```
{
  "username": "exampleUser",
  "password": "examplePassword"
}
```

Response body:

```
{  
| eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7InVzZXJuYW1lbnQiOiJ0bm9keSIsImNpdGUiOiJkaW8ifX0.  
}
```


Passport-JWT

- Strategy for PassportJS
- Authentication middleware
- Returns 401 if JWT is wrong

```
const jwtDecodeOptions = {  
  jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),  
  secretOrKey: jwtSecret,  
};
```

```
passport.use(  
  new JwtStrategy(jwtDecodeOptions, (payload, done) => {  
    const user = findUser(payload.data.username);  
    if (user) {  
      return done(null, user);  
    } else {  
      return done(null, false);  
    }  
  })  
);
```

```
app.get(  
  "/self",  
  passport.authenticate("jwt", { session: false }),  
  (req, res) => {
```

Backend api functionality

Explain frontend

Explain frontend

Explain frontend=>backend