Certainly! I will provide a detailed step-by-step guide for creating a login system for your employee time sheet webpage. We'll use Node.js, Express, MongoDB, and Passport for this project.

**Step 1: Install Node.js**

First, you need to have Node.js installed on your computer. Node.js allows you to run JavaScript on the server-side. Visit the [official Node.js website](https://nodejs.org/) to download and install the latest LTS version.

**Step 2: Create a new project folder**

Create a new folder for your project (e.g., **login-system**). This folder will contain all the files for your application.

**Step 3: Initialize the project**

Open a terminal or command prompt, navigate to your project folder, and run the following command to initialize the project with a **package.json** file:

bashCopy code

npm init -y

The **package.json** file is used to manage your project's dependencies and settings.

**Step 4: Install required packages**

In this project, we will use several npm packages to create a login system. Run the following command to install them:

bashCopy code

npm install express mongoose bcryptjs body-parser ejs express-session passport passport-local passport-local-mongoose

Here's an overview of these packages:

* **express**: A web framework for Node.js.
* **mongoose**: An Object Data Modeling (ODM) library for MongoDB.
* **bcryptjs**: A library for hashing passwords.
* **body-parser**: A middleware for parsing request bodies.
* **ejs**: A template engine for creating dynamic HTML pages.
* **express-session**: A middleware for managing sessions.
* **passport**: An authentication middleware for Node.js.
* **passport-local**: A Passport strategy for local authentication (username and password).
* **passport-local-mongoose**: A Mongoose plugin for simplifying the use of Passport with MongoDB.

**Step 5: Create the server.js file**

Create a new file named **server.js** in your project folder. This file will contain the main code for your Express app and the login system.

Refer to the complete and updated **server.js** file I provided in a previous response, and copy the code into your newly created **server.js** file.

This code sets up the Express app, connects to a MongoDB database, defines the User schema, configures Passport for authentication, and sets up routes for login, registration, and other actions.

**Step 6: Create EJS templates for the views**

In this step, we'll create three EJS (Embedded JavaScript) templates for the views (HTML pages) of our web application. EJS is a template engine that allows you to embed JavaScript code within HTML, making it easy to create dynamic web pages.

Create a new folder named **views** inside your project folder, and create three new files inside: **index.ejs**, **login.ejs**, and **register.ejs**. Add the code from my previous response for each of these files.

These templates represent the home page, login page, and registration page of your application.

**Step 7: Run the server**

In this step, we'll start the Node.js server to run our web application.

1. Open the terminal or command prompt, and make sure you're inside your project folder.
2. Type the following command to start the server:

bashCopy code

node server.js

1. You should see the following message in the terminal or command prompt:

arduinoCopy code

Server running on port 3000

This indicates that your server is up and running successfully.

1. Open your preferred web browser and go to **http://localhost:3000**. You should see the home page of your web application, which displays links to the login

Top of Form