COMP 2537 网络开发 2

作业 1

介绍：

对于此作业，您将使用 Node.js 创建一个简单的网站。您还将创建并连接到 MongoDB 数据库，以存储用户及其姓名、电子邮件和密码。您将使用 Atlas 创建一个可公开访问的 MongoDB 数据库（网址：https:// [www.mongodb.com/atlas）。](http://www.mongodb.com/atlas))

如果用户成功进行身份验证（提供有效的电子邮件和密码），将创建一个会话来存储用户的信息。如果用户具有有效的会话，则将其视为已登录。您将加密的 会话信息存储在 **MongoDB 会话数据库**中。会话必须在 1 小时后过期。如果用户未登录（没有有效的会话），他们一定无法查看会员的页面——毕竟这是会员的唯一页面！

我们将听取用户的意见，以获取他们的帐户信息并让他们登录。我们需要正确处理MongoDB数据库中的NoSQL注入攻击，以便用户无法故意攻击我们的数据库。

我们将使用 .env 文件来存储加密机密和您的 MongoDB 数据库凭据。密码 **不得** 存储在 git 存储库中。如果是这样，并且您的存储库公开或意外泄露，则攻击者现在可以访问您的数据库，以对它执行任何操作！让我们将密码放在黑客无法触及的地方。

您的网站需要托管在 Render 等网站上。Render 是一个完全托管的 Web 应用程序托管平台 （<https://render.com/>）。Render 可以提供许多功能，包括 1） 让我们离开本地主机，以便我们可以与其他人共享我们的网站;2）创建一个可公开访问的域名;3）安装和运行我们的node.js服务器;4）当我们提交时自动部署我们的代码，并将我们的代码推送到GitHub上的主分支。

过去，我使用过 Heroku 和 Qoddi（类似的托管服务），但是，那是他们提供免费开发托管的时候。现在，即使是最简单的网站，他们也会收费。Render 免费提供类似的服务。

用户故事：

如您所知，用户故事是敏捷开发方法的关键部分，因为它们有助于从最终用户的角度定义有价值的软件功能。

以下是在此分配中实现的用户情景的一些示例：

̶ 作为用户，我希望能够 **登记** 创建一个新帐户，以便我可以访问仅限会员的区域。

̶ 作为用户，我希望能够 **登录** 添加到我现有的帐户，以便我可以访问仅限会员的区域。

̶ 作为用户，我希望能够 **登出** 我的帐户，以便其他人无法访问它。

̶ 作为用户，我希望看到一个 **主页** 如果我是，可以选择注册或登录 **当前未登录**.

̶ 作为用户，我希望看到一个 **主页** 欢迎我，并向我展示转到会员专区并注销（如果我当前在）的选项 **已登录**.

̶ 作为用户，我希望能够看到一个 **随机图像** 当我访问会员专区时，它在视觉上更有趣。

̶ 作为用户，我希望被告知如果 **用户名或密码** 我输入了 **不正确** 登录时，以便我可以纠正错误。

推荐安装的软件：

**源树**： <https://www.sourcetreeapp.com/>



Sourcetree 是一个很棒的 GUI 和可视化工具，用于查看和管理 Git 存储库。在我看来，它比 github.com、命令行上的 git、VS Code 甚至 GitHub Desktop 中的版本控制更能查看存储库的当前状态。

Sourcetree 是跨平台的，适用于 Windows 和 Mac，并且是免费的！

**Studio 3T**： <https://studio3t.com/>



Studio 3T 是查看和编辑所有 MongoDB 集合和数据的好方法。可在 Windows、Mac 和 Linux 上免费使用。

**VS 代码**： <https://code.visualstudio.com/download>



一个伟大的、多功能的IDE，适用于所有类型的编码项目，包括用Node.js编写的网站。

Chrome 扩展程序：

还有一些适用于 Google Chrome Web 浏览器的扩展可能会有所帮助。

**“编辑此cookie**” by [editthiscookie.com](http://editthiscookie.com/)

让我们看看会话 cookie，编辑它们或删除它们。 <https://chrome.google.com/webstore/detail/editthiscookie/fngmhnnpilhplaeedifhccceomclgfbg?hl=en> 注意：不需要使用此扩展。 您可以通过开发工具访问 cookie。但是，这提供了一种快速简便的方法来查看和删除 cookie。

在线工具：

一些 **GUID/UUID 联机生成器**：<https://guidgenerator.com/> <https://www.uuidgenerator.net/guid> <https://www.guidgen.com/>

当想要为会话加密创建唯一密钥和密钥时，这些是有用的站点。

您将需要的节点模块：

在 Node 中，我们需要一些外部库来使我们的网站正常工作：

̶ **快递**

根据 Express 的官方网站 （<https://expressjs.com/>）：，“Express 是一个最小而灵活的Node.js Web 应用程序框架，为 Web 和移动应用程序提供了一组强大的功能”，并提供了无数用于创建网站的实用方法和中间件。

# ̶ 快速会话 和 连接-mongo

express-session 模块用于创建一个会话，该会话可以确保只有已登录的用户才能看到仅限会员的页面。connect-mongo 模块用于将会话数据存储在加密的 Mongodb 数据库中。这样，如果需要重新启动服务器，您就不会丢失服务器端会话信息。

̶ **bcrypt 的**

BCrypt 是一个哈希库，用于在将密码存储在数据库中之前对密码进行加盐和哈希处理。这是此分配的要求，并且是良好的行业惯例，因为纯文本（未加盐、未哈希的密码）不得存储在数据库中。如果数据库通过攻击被破坏，BCrypt 创建的加盐和哈希密码对黑客来说仍然是未知和无用的。

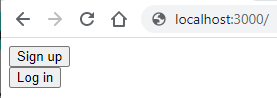
请记住，为了使用这些模块，您需要使用以下命令安装它们：

npm i <module\_name> 安装 express 的示例：npm i express

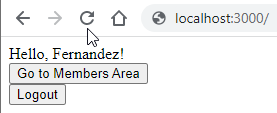
目标：

您的Node.js网站必须包含以下页面和功能：

1. 主页 – 站点： / 方法： GET

如果用户未 **登录**，则此主页将包含以下链接：

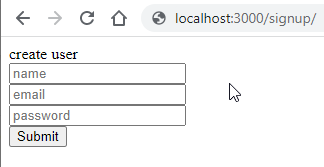
* 1. 注册 （/signup）
  2. 登录 （/login）

否则，如果用户 **已登录**，则此主页将：

1. 打个招呼并提供用户名，例如，Hello， Fernadez。

您必须在登录/注册期间将用户名存储在会话中，并在此处显示。

1. 提供指向成员区域 （/members） 的链接。
2. 有一个链接，该链接将注销用户 （/logout）。这将结束会话并重定向回 （/）。
3. 注册页面 – 站点： /注册 方法： GET

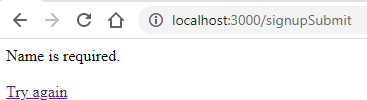


表单将收集以下信息以进行注册：

* 1. 名字
  2. 电子邮件
  3. 密码

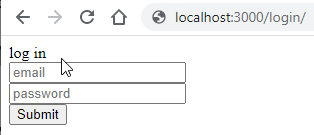
注册表单将 发布 表单字段。

您将验证所有三个输入，以确保它们不为空。如果用户忘记输入三个输入中的一个或多个，您将发回一条相应的消息，说明缺少哪个字段。例如： *请提供电子邮件地址。* 您还将提供返回注册页面的链接，以便用户可以重试：



如果这三个字段不为空，请将用户添加到 MongoDB 数据库。确保您使用 Joi 来验证输入（名称、电子邮件和密码），以便无法进行 NoSQL 注入攻击。将 **名称**、 **电子邮件** 和 **BCrypted 哈希 密码** 作为用户添加到数据库。然后，创建一个会话并将用户重定向到 /members 页面。

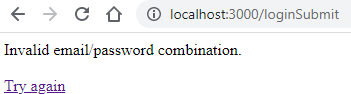
1. 登录页面 – 站点： /登录 方法： GET



表单将收集以下信息以进行注册：

1. 电子邮件
2. 密码

登录表单将 发布 表单字段。根据 MongoDB 数据库检查用户。确保您使用 Joi 来验证输入（电子邮件和密码），以便无法进行 NoSQL 注入攻击。如果找到具有该电子邮件的用户，请检查密码是否与 BCrypted 密码匹配。如果电子邮件和密码匹配，则将用户名存储在会话中，然后用户登录（重定向到 /members）。

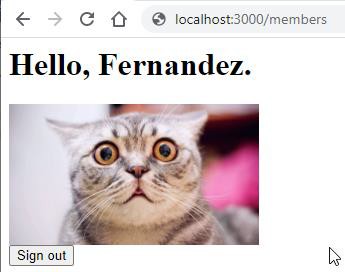
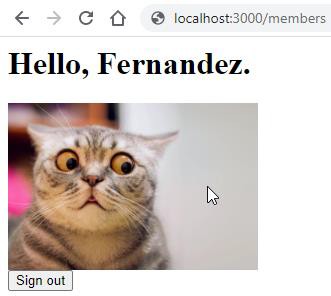
如果登录失败，请向用户发送相应的消息。例如： *未找到用户和密码。* 此外，请提供返回登录页面的链接，以便用户可以重试：

1. 仅限会员的页面 – site： /members 方法： GET

如果用户具有 **有效的会话** （如果已登录）：

* 1. 打个招呼并提供用户名，例如，Hello， Fernandez。（使用存储在会话中的 name 值）
  2. 有一个链接，该链接将注销用户（结束会话并重定向回 /）。
  3. 从 3 张图像中随机显示一张图像。

图像必须存储在服务器上的 /public 文件夹中。



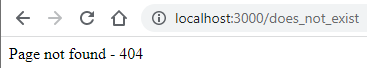
如果用户 **没有会话** （或会话无效，即未登录）：

一个。 重定向至主页（/）。

1. 注销页面 – site： /logout method： GET

此页面没有任何内容，但必须销毁用户的会话 （req.session.destroy（）;） 并重定向回主页 （/）。

1. 404 页面 – 站点：任何未分配的 URLs 方法： GET



任何与上述站点不匹配的页面都将发送 HTML 状态代码 **404** 并显示未找到的页面 – 例如： /does\_not\_exist

您的 .env 文件：

确保用于连接到MongoDB数据库的用户名和密码存储在 .env 文件中。此外，将会话的加密密钥存储在 .env 文件中。确保您的

.env 将添加到 .gitignore 中，以便 .env 文件不会添加到您的 git 存储库中。.env 文件绝不能 出现在您的 git 存储库中！

创建一个名为 .env 的文件。在这里，您将定义所有用户、密码和密钥。

MONGODB\_HOST=YourMongoDBClusterHere MONGODB\_USER=YourUsernameHere MONGODB\_PASSWORD=YourPasswordHere MONGODB\_DATABASE=YourMongoDBDatabaseHere MONGODB\_SESSION\_SECRET=Some\_Session\_Key\_Here1 NODE\_SESSION\_SECRET=Some\_Session\_Key\_Here2

笔记：

等号前后没有空格 – 例如：

MONGODB\_USER=您的用户名这里

不

MONGODB\_USER = 您的用户名这里

虽然可以 将变量值括在单引号或双引号中，但通常会省略它们。

MONGODB\_USER=您的用户名这里

或

MONGODB\_USER=“您的用户名在这里”

两者都用于定义变量。

更多信息请见： <https://www.npmjs.com/package/dotenv> <https://hexdocs.pm/dotenvy/dotenv-file-format.html>

为了在我们的代码中使用这些变量，我们需要使用以下命令安装 dotenv Node 模块：

npm install dotenv

我们需要像这样对 dotenv Node 模块：

require（'dotenv'）.config（）;

我们可以像这样使用 process.env 访问这些变量：

process.env.MONGODB\_USER

和

process.env.MONGODB\_PASSWORD

**专业提示：**

许多其他数据库提供程序（MongoDB、MySQL 等）将为我们提供如下连接 URL （MongoDB）：

mongoDB+SRV：//my\_user：xhov21ul@cluster0.fuu9a.mongodb.net/用户

或者这个（MySQL）：

[mysql://a0wj08yp3h60iax9:xhov21ul59ilv8e5@jtb9ia3h1pgevwb1.cbetxkdyhws](mailto:xhov21ul59ilv8e5@jtb9ia3h1pgevwb1.cbetxkdyhws) b.us-east-1.rds.amazonaws.com:3306/kjm3rtal36l1mlue

了解如何将这些连接 URL 解析为它们的各个部分非常重要。对于 MongoDB：

下面是相同的连接 URL，经过颜色编码和标记，以区分 URL 的关键部分。

# mongoDB+SRV：//my\_user：xhov21ul@cluster0.fuu9a.mongodb.net/users

协议： **mongodb+srv**

用户： **my\_user**

密码： **xhov21ul**

# 主持人： cluster0.fuu9a.mongodb.net

端口：（未指定）数据库： **用户**

对于MySQL：

# mysql：//a0wj08yp3h60iax9：xhov21ul59ilv8e5@ jtb9ia3h1pgevwb1.cbetxkdyhwsb.us-east-1.rds.amazonaws.com：3306/ kjm3rtal36l1mlue

协议： **MySQL**

用户： **a0wj08yp3h60iax9**

密码： **xhov21ul59ilv8e5**

# 主持人： jtb9ia3h1pgevwb1.cbetxkdyhwsb.us-east-1.rds.amazonaws.com

端口： **3306**

数据库： **kjm3rtal36l1mlue**

创建 Git 存储库：

为您的网站创建 git 存储库。确保添加所有文件，提交并将它们**推送到** github.com 上。称为 main（或 master）的单个分支就可以了。

如前所述，请确保已从 git 存储库中排除 .env 文件。创建一个包含以下内容的 .gitignore 文件：

/node\_modules package-lock.json

.env

在 Render（或其他托管服务）上托管：

使用 Render（或其他托管服务）创建一个帐户。使用你的帐户，创建一个链接到 git 存储库的应用。

有关使用 Render 的详细信息，请参阅内容 ->资源部分中的学习中心。



**重要！** 请记住将环境变量添加到 Render 应用程序，否则，Render 将无法访问您的所有机密（密码和登录信息）。有关添加 .env 文件的说明，请参阅学习中心的“创建呈现应用程序”文档。

防范 NoSQL 注入攻击：

每当我们查询数据库并包含来自用户的输入时，我们都需要确保用户没有有意（或无意）添加一些会改变我们查询的 NoSQL 命令。我们经常将用户输入作为查询的一部分，例如，当我们要求用户提供用户名并在数据库中查看它们是否存在时，添加用户在表单中输入的用户名、电子邮件和密码，或者当用户请求更改其密码时，我们会在数据库中更新其密码哈希。

URL 参数

|  |  |
| --- | --- |
| Form Fields | URL parameters |
|  |  |

如果我们要从表单字段直接包含用户输入（无需先验证），例如

req.body.username 或来自 MongoDB 查询中的 req.params.id 等 URL 参数，如下所示：

等待 userCollection.find（{**用户名： req.params.id**}）.toArray（）;

用户可能会添加一些 MongoDB 代码，这些代码可能会更改我们的查询结果。这是 **BAD**。 **这是非常糟糕的！**

我们需要验证用户提供的数据是否不包含特殊字符或 MongoDB 命令。我们需要一个像 Joi（另一个有用的节点模块）这样的验证器。

要安装 Joi：

npm i joi

使用 Joi，我们可以确保用户输入只是一个字符串，而不是 MongoDB 解释为 MongoDB 命令的数组或 JSON 对象。

若要在代码中使用 Joi，需要满足以下要求：

const Joi = require（“joi”）;

在下面的示例中，我们确保 req.body.username 是一个字符串 （.string（）），不超过 20 个字符 （.max（20）） 并且不为空 （.required（））。架构定义了一个变量，我们根据架构验证 req.body.username。如果出现错误

（潜在的 NoSQL 攻击），我们将重定向到不同的页面，防止用户的输入作为 userCollection.find（） 中查询的一部分执行。

const schema = Joi.string().max(20).required();

const validationResult = schema.validate(**req.body.username**); if (validationResult.error != null) {

console.log(validationResult.error);

**res.redirect("/login"); return;**

}

const result = await userCollection.find({username: username}).toArray();

在一个稍微复杂一点的例子中，我们想要同时验证多个变量，我们可以在 Joi 模式中指定哪些变量和哪些类型是预期的。本示例验证表单提交中的用户名和密码。

const schema = Joi.object(

{

username: Joi.string().alphanum().max(20).required(), password: Joi.string().max(20).required()

});

const validationResult = schema.validate({**req.body**});

50 分

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| A home page links to signup and login if not logged in;  and links to members and sign if logged in. | Five marks |
| A members-only page that displays 1 of 3 random images stored on the server. | Five marks |
| The members-only page will redirect to the home page if no valid session is found. | Five marks |
| The signout buttons end the session. | Five marks |
| A 404 page that "catches" all invalid page hits and sets the status code to **404**. | Five marks |
| Session information is stored in an encrypted MongoDB session database.  Sessions expire after 1 hour. | Five marks |
| Password is BCrypted in the MongoDB database. | Five marks |
| Your site is hosted in a hosting service like Render. | Five marks |
| All secrets, encryption keys, and database passwords are stored in a .env file. | Five marks |
| The .env file is NOT in your git repo. | Five marks |
| Note: You will receive a **mark of 0** if your code doesn't run  (i.e. too many errors to run properly, cannot connect to the database, etc.)! I will be running the code on your Render app; please make sure it is available until after you receive your marks! |  |
| **Total:** | **50 marks** |

指向托管站点的链接（在 Render 上）。

|  |
| --- |
| **Submission:** |
| Here is a link to your YouTube video demo.  ex: https://youtu.be/c8nPuL5phG1 |
| A link to your GitHub Repo with your Node.js application. (REMEMBER: Commit **AND PUSH** your code to GitHub!)  ex: https://github.com/comp2537-s24/assignment1-studentname |
| A link to your hosted site (on Render).  ex: <http://myawesome-2537-a1.onRender.com/> |

自评分清单：

找到一个自评分作业清单模板，对于每个项目， **要么将其留空**（如果您没有完全完成任务），要么在它旁边放一个 **x**（表示您完成了任务）。

（完整）

|  |  |  |
| --- | --- | --- |
| [ ] | The signout buttons end the session. | (incomplete) |
| [**x**] | The .env file is NOT in your git repo. | (complete) |

14. 显示您的

|  |  |
| --- | --- |
| **Your video needs to include the following:** | **Example:** |
| Your name | Patrick Guichon |
| Your set | 1A |
| Which assignment are you demoing? | Assignment 1 |

|  |
| --- |
| **Notes about your video:** |
| If we don't see all the functionality and features of your website demonstrated in the video,  you will lose marks for not having completed this part of the website. |
| You do NOT need to show your face while recording the video; you must be narrating the  video. Tell us what you are doing as you demo your site. |
| Make sure we can hear your voice clearly and at an adequate volume. |
| Max video length: 4m:00s.  Your video must **NOT be longer than 4 minutes**. |

|  |
| --- |
| **Demo the following functionality in this order:** |
| 1. Show the home page (/) running on Render (or other hosting service) |
| 2. Show the user's database in Studio 3T. |
| 3. Show the session database in Studio 3T. |
| 4. Click on the Signup link and create a new user.  After the user is created, the user is logged in and must be sent to the /members page. |
| 5. Show the user database in Studio 3T – the new user must be added  The password must be hashed (and not in plaintext). |
| 6. Show the session database in Studio 3T – the session must be encrypted, and the ID must  match your cookie in the browser. |
| 7. Click on Sign out. |
| 8. Show the session database in Studio 3T – the session must be deleted. |
| 9. Delete your cookie and attempt to go directly to /members without a valid session. |
| 10. From the login page, enter a valid email address but an invalid password.  You must be provided with an error message saying invalid password. |
| 11. From the login page, enter a valid email and password to show you are properly logged in. |
| 12. Go to a page that doesn't exist, for example,/does not exist, to show the 404 page. |
| 1. Show the code for your site in your editor:    * Show that you are using variables from your .env file.    * Show that your sessions are set to expire after 1 hour.    * Show that you are validating user inputs using Joi. |
| 14. Show that your .env is added to your .gitignore and your .env is **NOT** in your git repo. |