COM3504/COM6504 智能网络

2022-2023年的任务

最后期限：2023年5月19日，星期五，下午3点

交付：通过黑板提交压缩文件（见第8节--提交）。

1. 简介

这项作业将测试你使用本模块中所教授的方法和技术创建一个网络应用程序的能力。

它将测试你对客户端-服务器架构的各个组成部分的理解，以及你在Node.JS中的编码技能。

这是一个小组项目，小组由模块负责人分配。你提交的作品必须是你自己的作品，不能抄袭。你**不能**使用任何在讲座中没有明确推荐的图书馆，因为我们会将其视为**使用不正当手段**。

2. 场景

你将开发一个 "观鸟 "的渐进式网络应用程序，为用户（通常是观鸟者）提供记录和查看鸟类踪迹的方法，并帮助进行识别。你需要把它开发成一个网络应用程序，并配备支持性的服务器基础设施。使用该网站，用户可以添加新的观察结果，查看观察结果，还可以对识别进行评论。

该应用程序应允许访问一个分类的**鸟类目击**列表。该系统将允许：

- 按日期/时间分类（至少）查看看到的情况，以及是否完成了识别。按距离分类是一个 "延伸 "的目标（也就是说，很高兴能拥有）。
- 增加了一个新的目击者
  - 请注意，一旦添加，大部分的视线将无法修改。
  - 每个目击者都包含（至少）以下内容
    - 看到的日期和时间
    - 定位
    - 对目击事件的（简短）描述
    - 识别（可能是未知的或不确定的）--**这可以更新：**
      - 观察鸟类的原始用户需要对建议感到满意，并可以更新观察结果以显示识别。
      - 该标识应与从DBPedia知识图谱中获得的信息相联系。这些信息包括该鸟类的通用名称/科学名称、英文描述和URI（应链接到描述该鸟类的

DBPedia页面）。

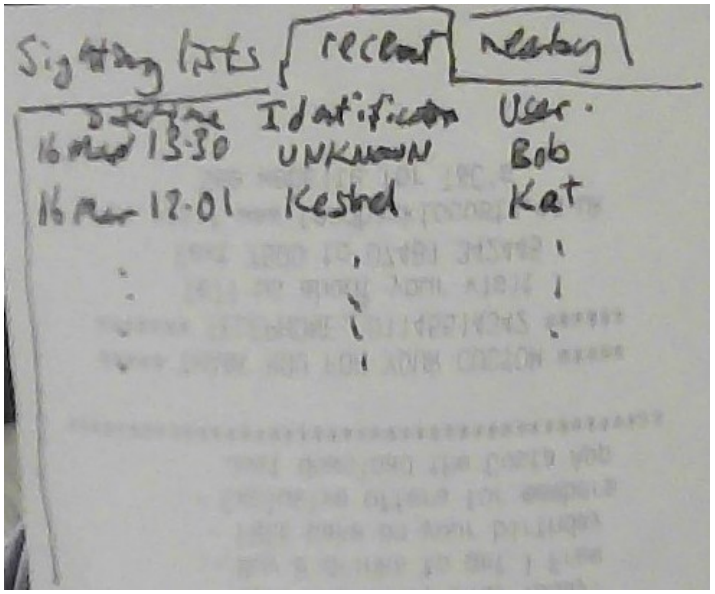
- 一张照片(通常)

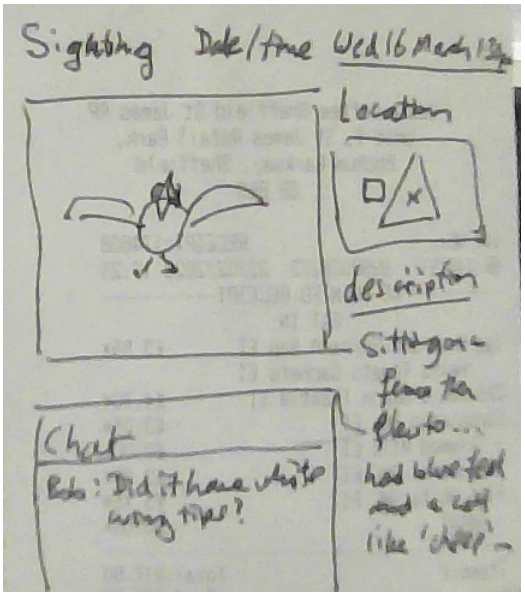
- 用户的 "昵称"；作为一个字符串--没有必要实现一个登录系统 o一个 与每一个景象相关的聊天/评论，其中包含：
  - 对观察结果的评论，即关于该鸟是什么的讨论和建议

当一个新的景象被添加时，所有用户都可以访问它（即你不需要实施一个登录系统或一套隐私规则--当用户进入网站时，他们将看到所有的景象）。

3. 设计

在下面的草图中，你可以看到一些初步的 "信封背面 "草图，包括目击者名单（左）和个别目击者细节（右）。设计是留给你的--请在下面的草图上加以改进。请注意，你需要在这些草图上增加/修改细节，而且你还需要为（至少）增加一个新的目击点和更新识别状态增加新的草图。一个好的方法是让目击细节界面也能用于添加新的目击和更新识别。





当点击列表中的一个目击事件时，将显示个人细节。对于报告目击事件的原始用户，这将允许他们更新 "识别 "细节。其他细节在创建后被固定下来，但必须能够（实时地）就该目击事件进行聊天。这种聊天对所有查看该目击事件的人都是公开的。原始用户也可以使用聊天来添加更多的细节。

4. 组成部分

以下是必须包括的组件清单。

4.1 网络应用

- 该网络应用程序**必须：**
  - 循序渐进
  - 支持线上和线下的互动
  - 使用EJS和JavaScript在node.JS框架中实现。
- 该网络应用程序**可以**
  - 是多模式的（即通过多种设备提供访问，如手机、笔记本电脑等）。

功能

- 该网络应用程序必须允许创建新的目击事件，并提供相关细节--描述、日期/时间、用户昵称、位置（应来自地理定位或可从地图上选择）和相关图像（照片）。
- 该网络应用程序必须允许查看所有目击事件的相关细节--描述、日期/时间、用户昵称、位置（应来自地理定位或可从地图上选择）和相关图像（照片）。
- 观察结果可以按最近的和（扩展目标）按距离（如 "乌鸦飞"）进行排序。
- 通过点击，可以选择一个观察点，以了解更多细节/对话。
- **在线**时，必须将详细信息发送到数据库中
  - 图片很可能需要被转换为64进制。图像被上传到服务器上，然后存储在MongoDB数据库中。图片可以从文件系统上传，也可以使用URL进行引用。
- 当用户处于**离线状态**时，你必须允许将更改存储在本地（以便以后上传）、包括：
  - 必须能够在浏览器（indexedDB）中创建（至少）一个新的观测点
  - 强烈建议在浏览器中离线持有一个以上的（新）视力。
  - 必须能够将聊天信息添加到用户新发现的和以前发现的事物上。创建的
  - 注意：你**不能**离线**保存所有**目击细节，因为只有一部分与每个用户相关。你应该保留一份**观察清单的**本地副本，但**不是所有细节**。
- 当设备再次在线时，你需要：
  - **首先将**本地修改上传到服务器 - 这应该是一个安全的操作，不应该需要考虑任何服务器的变化
    注意，在这之后，你应该能够安全地从服务器重新加载
  - 检索视线列表的更新（即自上次同步以来）（从头检索所有内容是一个较弱的解决方案）。这应该是新的目击事件和（仅）对目击事件的识别的变化。
  - 你也应该重新加载与该用户创建的视线相关的任何聊天信息
  - 作为一个延伸目标，你应该通知用户他们聊天记录中的新消息（即为他们创造的目击事件）。

你可能如何做到这一点？

- 观察结果的创建可以以表格的形式实现，你可以提供文字、照片等。照片应该（至少）通过HTML5表格上传，或者（更好）通过网络摄像头--或手机摄像头（如果你选择这个选项）拍摄照片。

> 注意：创建一个移动网络应用并不是必须的，但你可能希望这样做，因为地理定位、相机和测试离线使用在移动设备上会更好。请注意，Chrome浏览器允许你选择以移动设备的方式查看网页--选择开发者工具，然后选择图标
>
> 在此强调：
>
> 还要注意的是，Chrome浏览器（默认情况下）坚持使用HTTPS进行地理位置定位，除了本

4.2 聊天信息

- 应用程序的聊天部分**必须：**
  - 循序渐进
  - 支持线上和线下的互动
  - 在node.JS框架中使用socket.io实现。
  - 不堵车
- 聊天应用程序**可以**
  - 是多模式的（即通过多种设备提供访问，如手机、笔记本电脑等）。

功能

- 当用户选择视线时，必须出现（实时）聊天（可能在底部）。
- 任何新的信息都必须是实时可视化的，因为它是可用的
- 用户必须能够添加新的信息，这些信息将实时出现在其他用户的聊天记录中。(当他们打开它时）。作为一个扩展目标，如果有人在他们创建的视线中添加了一个信息，用户应该得到通知。
- 当在线时，这些应该实时更新。
- 离线时，可将消息添加到用户创建的视线中，并且聊天会 "同步"。当在线时，新的信息会被显示出来，并使每个人都能看到观察到的细节
- 由于信息是与特定的景象相联系的（最终），你可能会选择将聊天记录储存在在MongoDB中的观察文件/对象内
- 请注意，目前不能编辑或删除信息。

你可能如何做到这一点?

您可能会（从服务器）"广播 "任何带有 "视线ID "的新聊天信息，因此网络应用可以选择它需要更新的信息--即用户有视线开放的地方--和/或在他们创建视线后应该得到通知。

4.3 服务器

- 服务器**必须**：
  - 使用NodeJS+Express实现
  - 必须支持聊天系统
  - 必须连接到MongoDB。

4.4 数据存储和检索

- 数据存储**必须**使用
  - MongoDB作为网络数据库
  - indexedDB作为浏览器存储
    - 注意：使用cookies或本地存储不适合本模块。
- 对于从DBPedia链接的信息
  - 使用fetch-sparql-endpoint模块
- 从DBPedia SPARQL端点检索正确的注释。

6. 所提供的材料和外部图书馆

注意：在作业中不能使用第三方代码，除了讲座或实验课中**明确**提供的内容。例如，你可以使用讲座幻灯片或实验课中给出的任何代码，但你不允许从网上下载任何代码或使用任何其他软件来完成作业的相当一部分。**未经授权重新使用第三方软件将被视为抄袭。**如有疑问，请在使用任何第三方代码前向讲师申请许可。尽管在讲义中没有提到，但允许使用的库是用于改善任何界面的**外观和感觉的**css/js库（例如Bootstrap）。对于其他库，请在使用前询问。

允许的图书馆列表：

- CSS/Javascript：靴带（Bootstrap
- NPM库：Passport, Express, node static, body parser, fetch, socket.io等。
- 所有的代码和在实验和讲座中使用的任何库（如socket.io，Axios等）。不

允许使用的库的例子：

- Angular, React, React Native, Vue - 注意，不允许使用服务器/浏览器框架
- 任何建立在Javascript之上的语言，例如，需要编译的语言。

7. 标识模式

不同部分的分值：

1. 网络应用和聊天应用：35%
    - 在线下和线上都能工作
    - 实施网络工作者
    - 实现一个有索引的数据库
2. 服务器：20%
    - 正确地组织服务器和路线
    - 正确的Swagger文档
    - 非阻塞性组织（使用承诺和回调，多个服务器等）。
3. 服务器 数据存储15%：
    - MongoDB：
        - 正确地使用MongoDB，包括正确地组织到模型中、控制器，等等。
4. 异步和双向的客户/服务器通信 15%的比例
    - 正确使用HTTP请求和socket.io
5. 与知识图谱的联系 15%

你为作业的每一部分提交的代码的质量将占到分数的相当大一部分。我们通常会检查：

- 代码功能：代码如何满足任务描述中设定的要求。
- 代码文件。这包括
    - 行内注释，使你的代码意图对阅读的人来说更清晰。
    - 文档：关于代码及其使用的更高层次的评论。关于更多的信息
      代码级文档的准则见
      http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html
    - Gitlab/Github的历史和来源
- 代码执行：我们必须能够使用正常的计算机毫无问题地运行你的代码。
- 代码质量：你的代码应该遵循关于类和变量命名以及缩进的一致格式，这可以通过使用IDE（WebStorm）轻松实现。它必须包含对异常情况的正确使用和处理。你还应该包括适当的命名和代码文档，即注释。
- 代码组织（例如，可读性、node.js中模块的使用、Javascript文件等）。

8. 提交资料

你的解决方案必须包含在一个独立的目录中，**以你的小组命名，**并压缩成一个压缩文件通过黑板提交。

所有的代码应该在<MainDirectory>/solution目录下。请注意，我们将同时检查和运行这些代码。

(a) 所有代码必须用HTML5/Javascript/CSS/Node.JS开发。我们必须能够在标准机器上顺利运行你的解决方案。除了运行npm install之外，不应该需要特别安装任何外部库。

(b) 外部节点模块不能包含在你提交的文件中，所以避免提交非常大的压缩文件，这将导致问题，特别是在接近截止日期时。请确保创建一个完整的package.json文件，以便我们可以运行npm install来安装所有的模块。一些css和js库可以通过HTML/EJS文件的外部链接提供。

请注意，代码的质量占相关分数的一部分，所以一定要写好代码。

特别是，我们希望你能按照实验中使用的相同思路来组织你的代码。例如，Mongo DB代码的组织必须遵循讲座中提供的组织（控制器、模块等）。

此外，你还应提交：

1. 一个README.md文件，阐明<主目录>中的任何安装/运行指令。N.B. 安装和运行应该仍然是容易/简单的。
2. 代码文档
3. 几张屏幕截图或一段显示应用程序的视频，这样我们就可以了解当我们运行你的解决方案时应该发生什么 <MainDirectory>/Screenshots

9. 团队贡献

你将使用buddycheck对自己和你的团队成员按以下标准进行评分：

1. 出勤率和准时性（对主管/自己的团队会议等）。
2. 有能力与其他团队成员有效合作。
3. 对项目交付物的内容和组织做出贡献。
4. 贡献的质量。
5. 捐款的及时性。

你的最终个人分数将由一个适用于你的团队总分的比例系数形成。该比例系数由你的同行评估和你的团队成员返回的评估决定。注意：系里的团队评估政策要求有证据支持个人分数与团队分数相差超过+/-15%；考虑的证据包括导师的会议出席/笔记以及slack/github互动。

10. 反馈信息

最后期限是固定的。因此，你应该计划好你的工作，争取至少在截止日期前几天交出作业--不要留到截止日期，以防出错，然后发现你迟到了。

请注意，计算机科学系对迟交课业的学生实行相当严厉的处罚。如果你想看一下细节，你可以在大学的网站上找到它们。

如果你想得到形成性反馈，可以在实验课上随时提供。在第9周和第10周，我们将为你们提供预约时间，你们可以预约（每组1个）来演示项目的当前版本并接受口头反馈（不计分）。

11. 对这项任务的疑问?

如果你对作业有任何疑问，请随时通过讨论区或电子邮件与我们联系。