COM519 Advanced Database Systems Assessment

Lecturer: Joe Appleton

Student: Jacob Dale

Hand in Date: 15/Jan/2021

word count: 2238

Hosted Web app can be found at: https://nameless-savannah-04348.herokuapp.com/

Git Hub repository: https://github.com/JSDale/AdvancedDbAssessment

Abstract

I was tasked to create a ***proof of concept*** web application that is attached to a database on the backend to solve a real world problem. The real world problem would have preferably been one experienced in work, unfortunately my work place currently has no need for a web application so I used a real world problem from my personal life. The application and server should be hosted on an external server, such as Heroku and Atlas MongoDB, it should show I have an understanding of current work place practices, it should also be saleable and maintainable.

Mission Statement - Introduction

I am a leader of a small youth group (ages 10 - 16) belonging to the local Church of England church in my village and currently we must rely on our memories for the medical requirements of the children. We also only use a paper register for each of our meetings, this is problematic as that is our only copy and if we loose it we have no record of who attended when. This can also cause an issue in an event of a disaster, such as fire. If we can't get that to that register to ensure everyone is out and safe it might result in injury or death.

We have a duty of care to those children and their parents trust us with said care. Having an online resource to keep track of attendance and medical requirements would be a great help, as sometimes we get the kids to bring in food for end of year parties etc... so if they could see what the dietary requirements of other kids are there is less risk of allergic reactions. According to the NHS *https://digital.nhs.uk/data-and-information/find-data-and-publications/supplementary-information/2018-supplementary-information-files/hospital-admissions-for-allergies-and-anaphylactic-shock*, the number of Finished Admission Episode (FAEs) with primary diagnosis of allergies for the year **2013-14** for ages **10 and under** were **4,869** and ages 11 to 18 were **2,873**. For the year 2017-18 **10 and under** were **6,167** and 11 to 18 were **4,743**.

I would also like to implement a forum for the kids to use to talk to each other, as finding children who are open about their beliefs can be hard and can feel isolating. This may not be implemented in the proof of concept as the register and information on the children are the top priorities.

System Overview

Functionality

The system will be comprised of two main functions, a service that keeps track of attendance and a service that allows users to view all profiles, create and edit their own.

Datastores

I will be using a single non-relational MongoDB database stored in a cloud service called MongoDB Atlas. I will use the model view controller (MVC) method to implement my application.

Key Views and Interfaces

The key views are:

- The Home Page.
- The log in page.
- The create user page.
- The edit profile page.
- The view all profile page.
- The attendance tracker page.

The key interface(s) are:

- youth.js

The key views are the web pages that will be used the most and the interfaces are the JavaScript files that interface between the webpage, the data model and the database.

Key System Components

the above image depicts the key components within my application. The blue arrows represent a web page linking to another web page, the purple arrows show what calls on and gets information from the controllers, the red/brown arrows show what implements the data models and the green arrows show what interacts with the database. I didn't use a framework for the creation of my diagram and I added the color scheme for easier reading as I, personally, find it hard to understand all black diagrams with a lot of arrows leading to different places.

Key Design Decisions

NoSQL

In the system overview I said I will be using a NoSQL approach in the implementation of the database, this is because the data I am using doesn't require high data security and the validation that a SQL approach grants can be done through the backend. I am developing this application in a agile manner, which allows me to not have all the documentation final before development can start, this means that any changes made to the database during development will need to be updated in the databases schema and entity relationship diagram (ERD). The database should also be scaled easier, although I don't expect a rush of new applicants for the youth group, some of the other leaders have ties to brownies and scouts and if they liked how the app worked they could suggest that the those groups look into using it. Those youth groups have a much wider turn around on children than we do so, the more scalable the application, the better.

There are two collections within my database and they don't relate in any way. I could remove certain fields from the "youths" collection and put them in their own collection. For example, I could extract the "interests" fields, that way they won't have to be repeated. However, they are only strings and there are no size constraints that would require me to not repeat data.

Full Stack Java Script

Another design decision was to use a full java script stack, this decision was straight forward. The two main driving forces were:

- It's the language I am most confident in using for this type of application.
- Setting up a development environment is easy.
- Java script can be run universally on most, if not all operating systems (OS).
- It is becoming more popular among the tech giants, such as google, amazon and Netflix.

Model - View - Controller (MVC)

Using the MVC method to implement a web application is popular choice as it abstracts and loosely couples the code. This is great for reusability and reduces the amount of duplicated code, which allows for greater maintainability. This also results in the development time being reduced. It improves the code security because one modification shouldn't affect the entire model the views of web applications get changed regularly, from a font change to a complete overhaul of the UI, these changes won't affect the business logic of the application, which won't slow down the business logic either.

MVC also aids search engine optimization (SEO) as it provides ease to develop SEO friendly URLs. This is because MVC can take a URL like https://www.mywebapp/join and send the user to the creation page instead of the URL needing to be https://www.mywebapp/create-user.ejs?name=bob etc...

Security and Scalability

Scalability

Throughout this document scalability has been referenced and the decisions made to improve it. By choosing the MVC method this makes the application scalable in terms of development time as any additions to the application should be just that, an addition, the applications pre-existing business logic shouldn't need to be changed on mass unless the functionality changes drastically.

The NoSQL approach allows for quick scalability as there aren't as many dependencies to consider when modifying the database compared to the SQL approach.

Security

The database itself is stored on MongoDB Atlas, so I will have to rely on their security measures as well as implementing some within the application.

The application requires users to log in to utilise its functionality therefore their passwords will need to be stored in the database which is a security risk. I have mitigated that risk by using bcrypt to hash the passwords before they are stored in the database. There is also data which some might consider personal being stored on the database in plain text, like their date of birth and medical requirements, so in order to store said data a user agreement would need to be written which every user would need to agree to on user creation. As this is a proof of concept only test data will be stored in the database. The test data will be created by myself or an application such as *Informatica Test Data Management* therefore a user agreement is not required, yet.

Conclusion and Reflection

What Went Well

In order to speed up production of the application and help me focus more on the backend, I implemented a responsive design library called Materialize. I looked up using twitter bootstrap, however there seemed to be a steep learning curve with bootstrap and materialize have easy to use web interface documents. This reduced the development time dramatically as it also implements mobile views. I did have to add a hamburger menu for mobile users, but the documentation walked me through it.

The MVC method helped a lot during development as it loosely coupled the code, which meant when I edited a function and broke it, the other functionality was unaltered.

The first functionality I implemented was creating a user, which was really simple to implement as I had reference work from previous lectures that I could alter and implement. The same goes for the login page.

What Could be Improved

As this is a proof of concept application, many improvements can be made. The first improvement could be moving the functionality of adding attendance away from the individual user as you can't always rely on kids to fill it in. The only way around ensuring they add their attendance is to get them to do it during the session. An extension of that is to create admin users that can access certain areas that normal users can't. At the moment there some webpages can only be accessed by logging in, but anyone who knows the URL can create a user and find out information on the children. This is a security risk so some kind of user authentication (where an admin accepts a user) should be implemented before release.

Currently there is no pagination in the attendance tracker or view all profile pages, this should be added as it would make the web pages more user friendly, especially for mobile.

There is a search function implemented in the view all profiles web page which searches all the fields of the users. This could be improved by adding a filter option so, the user can select the specific field to search. For example if someone entered "can't swim" in the other notes section and someone entered swimming is their interest it could return both those users. This wouldn't be the case at the moment as the search isn't advanced enough to match words that aren't the exact search. The search parameters also need to be an exact match at the moment, including case sensitivity, which is something I would like to improve.

I would like to implement a search function within the view attendance page as it would make looking up who attended which weeks a lot easier.

Within the ejs pages, specifically attendance-tracker.ejs, there is embedded JavaScript code, which doesn't follow having the code loosely coupled for improved scalability and maintainability. However, I am unsure if this can be abstracted into a .js file as it is part of rendering the HTML for seeing all the kids attendance. During troubleshooting some issues with reading and writing to the attendance array in the attendance controller, I read that instantiating an array with

```
var arr = new Array(1,2,3);
```

is now considered bad practice and it should be instantiated with

```
var arr = [1,2,3];
```

I plan to read into this more before release as I am not sure what the differences are. By using one method over another could cause unexpected bugs.

I implemented a basic profanity filter when creating a user account, which does catch some words. The filter can easily miss words if they have a number or a character implemented into it. The check only ensures the word by itself isn't input, if they were to input multiple words the filter wouldn't be able to catch that. So this would need to be improved in a future sprint.

During development I would have liked to debug the backend JS with breakpoints as it would have helped slowing down the execution and knowing if variables were being populated with data, however I got around this by writing the variables to the console and when exceptions were thrown their messages were descriptive enough for me to lookup what I need to change to get it working.

Conclusion

In conclusion the web application fulfils the requirements laid out in this document as it allows users to be created and a register to be taken. This application is, however, still a proof of concept as there are improvements I would like to make and functionality I would like to add.

I had difficulties throughout the development process, especially when working with the attendance array, but I managed to overcome them and create an application that does as intended. I learnt a lot from this project as I have never created a full stack JavaScript application before and wasn't sure if I would like JavaScript as a language. I tend to prefer Object-Oriented Development with languages that implement classes, like C# and Java. However, I have come out this project with a different view on JavaScript and NoSQL development. I can see that both have a place in the development world.