Submitted for the Degree of B.Sc. in Computer Science, 2015-2016

DBLS

Database Learning System

Registration Number: 201240204
Name: Adam McGhie

"Except where explicitly stated all the work in this report, including appendices, is my own and was carried out during my final year. It has not been submitted for assessment in any other context."

_____ _____

Signature *Date*

"I agree to this material being made available in whole or in part to benefit the education of future students."

_____ _____

Signature *Date*

CONTENTS

TABLE OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this chance to thank the following;

- I am indebted to Clemens Kupke, for his advice & guidance as project supervisor, and acting as a sounding board for the ideas I had for the project
- My mother for her advice on teaching methods and techniques she has found useful in her experience.
- My family for supporting me through the thick and the thin.
- For Craig for being there when I didn't want to drink alone

ABSTRACT

Introduction to Relational Databases can be daunting to first time users and those who are not secure in their understanding of the usage of Relational Databases. The current methodology in use for assigning exercises to students has the major bottleneck that each question for each exercise must be individually marked on a per student basis. As class sizes grow to meet demand, the amount of resources needed to provide the same level of education to students grows exponentially.

The software that has been developed as part of this project is designed to remove the bottleneck by removing the need for manual marking of student exercises. The application was built with the following requirements

- Ability to mark student submitted queries
- Ability to execute arbitrary SQL statements against Lecturer provided databases
- Ability to track student progress

1 INTRODUCTION

Databases have always been a key cornerstone of computer science and information technology since their inception in the early 1960's. With their usage only predicted to grow in the coming years having graduates trained in their use is a necessity in the current employment environment. While the current method of lectures and lab demonstrators marking each student individually has worked well for many years it is close to being unsustainable.

As such this projects software was developed to remove the need for manual marking.

1.1 OBJECTIVES

The primary goal of this project was to develop an environment where lecturers can assign students database related exercises and have them automatically marked by said system, and where students can execute arbitrary queries without being able to destroy the environment in which they are working, either intentionally or unintentionally as has previously happened by students dropping tables when they we not supposed to. Additionally, a staff administration tool for lecturers to oversee student progress per exercise, and being able to add new exercises was requested.

1.2 OUTCOME

Overall the application has developed into a fully functional piece of software built to fulfil both primary and secondary objectives, including handling arbitrary query execution in a secure manner. Although response rates during user evaluation were low many of those surveyed felt that that the instant feedback was the most advantageous feature.

2 RELATED WORK

In the early stage of the project, research was done on current methods of teaching the current Relational Database class at the University of Strathclyde, methodologies currently employed at other institutions for running massive open online courses (MOOCs) on relational databases, and relevant academic material on teaching usage of relational databases.

2.1 CURRENT METHODS USED

Before I started this project I was coincidentally a lab demonstrator for the Relational Database class (University of Strathclyde, 2016) currently run at the University of Strathclyde, combined with the fact that I had sat the same class previously meant I was in a unique position of both having sat the exercises for the class and marked them.

In my experience as a student taking the Relational Database class, one major issue I and many others experienced was long wait times up to 1 hour 30 minutes. This is excessive and detrimental to students by having them wait but still be unmarked by the end of the laboratory period. In my own experience I had to wait usually 30 to 45 minutes after finishing the exercise before it was marked.

Another issue with the current method of manual marking is it requires students have to learn or have extensive experience with Unix-like terminal interfaces. This is due to it being exclusively taught with the SQL++ command line client. This means that for the first few weeks' students are running into constant issues logging into the remote Oracle SQL server and trying work out how to navigate the Ubuntu Unity user interface (Vaughan-Nichols, 2014).

2.2 MASSIVE OPEN ONLINE COURSES

Massive Open Online Courses are the latest fad that many large universities are providing. They are open courses with allow unlimited number of participants to remotely take classes in an array of subjects (MOOC List, 2016) (Dakkak, 2013).

There are two main categories MOOC's fall into xMOOC's and cMOOC's (Siemens, 2012). xMOOC style classes emphasise mass scaling of the class, from an existing syllabus from a University. cMOOC style classes emphasise knowledge generation and creation, with participants encouraged to contribute to the learning network to shape the direction of the class.

The first xMOOC's such as "Introduction to Artificial Intelligence" run by Stanford University (Udacity, Inc., 2016) and "Circuits and Electronics" by edX (A joint project by Harvard & MIT) (Breslow, et al., 2013) are in the xMOOC classification since they follow the same structure as the in-attendance classes of the same name.

cMOOC's are not usually run by a higher education institution and are usually organized by individuals with a passion on a specific topic. Organizers and participants work together to build the framework. They are more open and flexible to changes from the participants, and can provide a more tailored learning experience.

2.3 EXISTING WEB APPLICATION SOLUTIONS

As part of the research into MOOC's and other available online resources for demonstrating, teaching, or learning relational databases. No solution was available which met the primary objectives of the project, namely automated marking of student solutions.

The closest available solution to the project is SQLCourse.com and SQLCourse2.com (Prior, 2003). While the SQLCourse sites do allow arbitrary submission of SQL statements they do not

store any tested query but merely run the provided SQL query through a test system to check it is a valid command and that it has no semantic errors.

3 SYSTEM DESIGN

This chapter will give an overview of the finalized system architecture, and the previous design iterations.

3.1 SOFTWARE ARCHITECTURE

The finalised system currently uses a fairly standard 3 tier layout (Figure 1); frontend, application, and backend. Not shown is an NGINX sever to offload SSL processing to simplify the application code so SSL, gzip encoding, and securing the backend against malicious actors, is done by NGINX (Arg! Team, 2012), in the specific case of the Appendix A it has also been setup to serve the static content files that are used by the Database Learning System, this does not affect the application if it is run directly.

The frontend which is sent to the user is HTML rendered using the EmbeddedJS template engine (4.2.6), and all static code such as CSS, fonts, client side JavaScript, and HTML. The application layer consists of the ExpressJS (4.2.4) routing requests to the exercise processing system. Backend is used to store all the data used in the application, it consists of a MongoDB document databased for storing user records and oAuth keys and tokens. The second part of the backend is a relational database used for storing exercises and successful attempts by users at the aforementioned exercises.

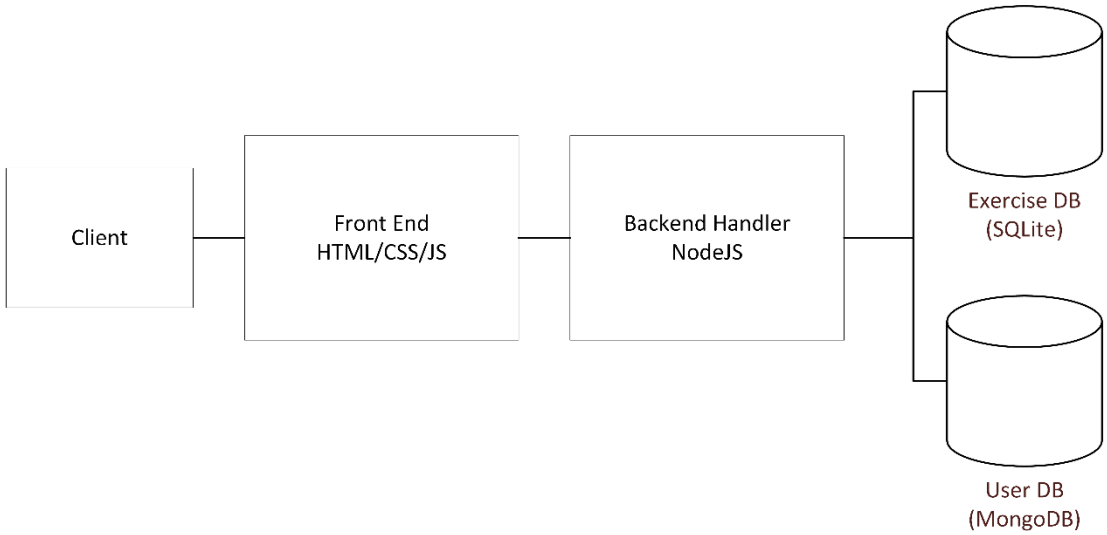

Figure 1Finalized Project Design

The original proposed system architecture used a 4 tier system (Figure 2), the same tiers as the finalised design but with continuous integration (CI) style runners being used to execute submitted exercises in an air gapped-like environment.

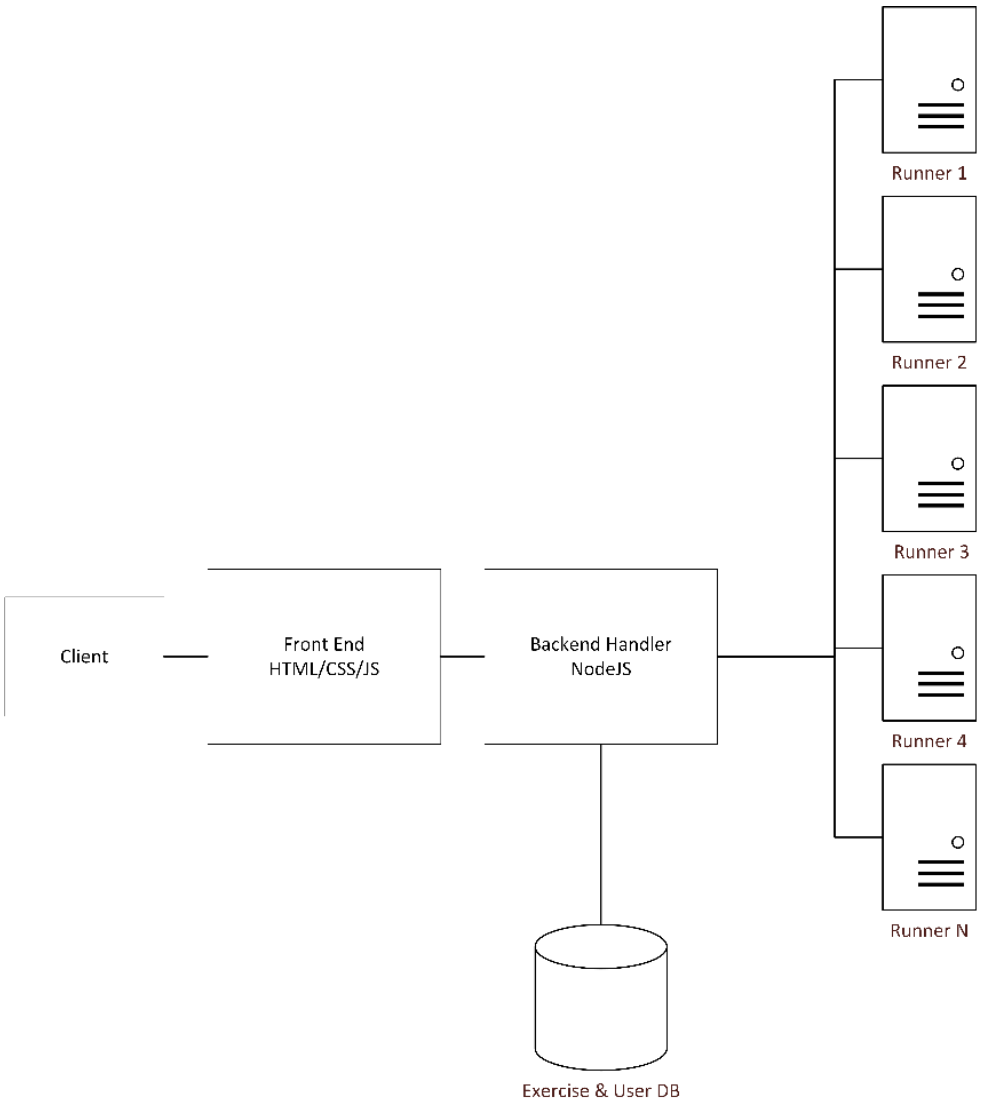

Figure 2 Initial Project Design

This CI type structure was designed to allow massive horizontal scaling of the backend processing system across many separate systems. The original design also used MongoDB to store all information, but for quick development using SQLite for storing exercise data.

The initial design also specified a unique markdown processor for marking questions. It was built to be column and row ambiguous, so data can be in any column order and sorted in any order. This would have allowed users to enter the query however they wished but still mark it as correct as long as the expected data was in the data returned by their query.

3.2 DATABASE DESIGN

As the Database Learning System is a highly database centric system, as such careful design of the databases was required.

As MongoDB is a NoSQL document-oriented database it has no defined structure, and thus cannot have a schema design like a normal relational database. The passport library (4.2.2) used for this project does require a schema be defined so it can store the authorization details for multiple login types. This can be seen in Appendix G which defined the types, and names used by passport to build the documents it stores in the MongoDB. As is shown each authentication method requires its own sub-object which is then used to store the necessary tokens for oAuth authentication.

The SQLite exercise database operates exactly like a standard relational database. It contains two tables, one to store exercises and the other to store correct user responses (See below).

Name	***Type***	***Primary Key***	***Not Null***	***Default***
id	Integer	✓	✓	
question	Text	×	✓	
Answer	Text	×	×	null
testQuery	Text	×	✓	
hint	Text	×	×	null
testDB	Blob	×	✓	
type	Integer	×	✓	0

Figure 3 Exercise Table

Name	***Type***	***Primary Key***	***Not Null***	***Default***
userID	Text	✓	✓	
exID	Integer	✓ (Foreign Key)	✓	
answer	Text	×	×	
date	Datetime	×	✓	current_timestamp

Figure 4 User responses table

As shown in Figure 4 the tables are relational on the exID which must relate to the id shown in Figure 3. Originally these were to be stored as a collection in the MongoDB but it was found that it was easier programmatically to store exercise content in SQLite along with user responses.

3.3 USER INTERFACE DESIGN

As with the database design the user interface went through several design changes. The initial design was originally done on paper which was similar to the 2nd iteration design shown in Figure 5, Figure 6, Figure 7, and Figure 8 which can be found in Appendix I.

A primary requirement of the user interface is responsive that it must have responsive design for to support multiple varying screen sizes. In previous projects by the Author made extensive use Twitter Bootstrap version 3. As such original proposed design was proposed to be based on Twitter Bootstrap.

During the course of the project additional frameworks were trialled and bootstrap was dropped in support of a more modern design based on Google's material design principles which is covered in more detail in 4.2.3 below.

The original design was going to use a tabbed structure for user profile and staff overview pages to allow multiple classes to be supported by the system, and possibly cross-year comparisons to be made inside the application. This feature was dropped from the requirements because the university has only ever run a single database class per year, and if excluded it would greatly simplify the user and exercise tables shown above.

4 DETAILED DESIGN AND IMPLEMENTATION

This chapter will delve into the details of the finalized implemented system, library dependences, and major feature overview.

4.1 IMPLEMENTATION LANGUAGES

The contents of this section describe the major languages and systems used in the implementation of the software application for this project.

4.1.1 Node JS

NodeJS is an open source runtime environment based off Google Chromium's V8 JavaScript Engine. It is commonly used to build server side web applications and developer JavaScript Object Notation Application Programme Interfaces (JSON API's) (Tilkov & Vinoski, 2010). NodeJS' choice of running as an event driven loop architecture with the ability to do Asynchronous I/O make it idea for building high throughput and mass web-scale applications and those which require real-time interactions (Teixeira, 2013) (Orsini, 2013).

Originally the project was going to be developed in PHP and several tests were run to find easy to use PHP web application frameworks. Larval, CakePHP and Code Igniter were all trialled and simple example applications were developed. The resulting development process was found to be clunky and not easy to grasp by the developer (Embry, 2016) (Munroe, 2012).

As such using a NodeJS framework was suggested and ExpressJS (4.2.4 below) was found to be an excellent fit for this project.

4.1.2 SQLite

SQLite is a relational database which unlike most relational databases does not run as a client-server but as a library for use by a program. This means that SQLite has almost no dependences since all database files are stored to disk, and no persistent server needs to run in the background for the application to function.

Its syntax is almost identical to PostgreSQL and implements almost all of the SQL-92 standard. It is limited slightly as it does not enforce type checking on non-primary key columns in tables, it

cannot write to views (this can be negated using triggers), and it has no internal permissions system (ergo if you can read any part of the database you can read any table, and delete, insert or update data in the tables).

In the project all exercise information and user results are stored using SQLite. This is so students can execute SQL queries on actual SQL databases without fear of ruining anything irreparably, as has been known to happen with current methods (see 2.1).

4.1.3 MongoDB

MongoDB is a document-oriented database, NoSQL database. It forgoes the normal table-based structure to use a JSON-like document notation with a dynamic schema. This JSON-like structure makes it especially easy to use MongoDB with NodeJS applications (Wilson, 2013). MongoDB is excellent for large scale applications, since it can replicate to the Nth server with automatic fail-over in the event of server failure (MongoDB Inc., 2016).

United Software Associates published a report on MongoDB comparing it to similar NoSQL Document databases, Cassandra and Couchbase, and found that in some test cases was found to outperform the others by up to 25x (United Software Associates, 2015).

In the project it is used to store all user related information, such as hashed passwords and oAuth user tokens.

4.1.4 HTML5

HTML5 is the 5th generation of HTML standard published by the World Wide Web Consortium (W3C) (Hickson, et al., 2014).

It adds new elements which are used within the project such as the nav tag (which is used to define navigation sections), footer tag, and a new file API for access.

4.2 THIRD PARTY TOOLS

Since this project is specifically about building an auto-marking relational database teaching system several libraries (or node modules) were used to make the project possible.

4.2.1 Npm

NPM, or node package manager, is the built in package manager for NodeJS. It is used to install necessary packages and their dependences (npm, Inc, 2016). It takes inspiration from PEAR and CPAN, package managers for PHP and Perl respectively, to serve almost every public node module and easier deployment by using a package.json (Appendix E) which can specify either specific versions or minimum required versions for modules used in the project. Additionally, it can be used to specify aliased command, such as ‘start’, to launch the program via npm.

4.2.2 Passport

Passport, and its associated libraries, are the de facto authentication middleware for use with NodeJS web applications based on expressJS. It is easy extendible using modules to authenticate with a multitude of services without needing to code specific stratagems for authenticating.

In this project passport is used to provide all major authentication for the project since it is a well-tested library that deals with a lot of the necessary validation of user input. The project also uses the following passport modules

- passport-twitter – Twitter specific oAuth integration
- passport-google – Google specific user authentication, for
- passport-local – Uses mongo DB to store authentication details from other passport modules
- passport-GitLab – GitLab specific oAuth integration
- passport-saml – Used for the attempted integration with the CIS SimpleSAMLphp system

4.2.3 Materialize CSS

Material Design is a specification (Google, Inc, 2016) from Google, based on the recent redesign they have done for their products and the Android operating system. It makes use of depth effects, grid layouts, responsive UI, and animations & transitions to provide meaning as if it was a physical.

There are several Material Design compliant CSS frameworks available. For the project Material Design Lite, Polymer, and Materialize CSS

Polymer (Polymer Project, 2016) is an extensive library built using Web-Components, a new method of component based web design. While it has an excellent choice of elements for material design, it's use of Web-Components which has not been fully standardised and thus requires the use of a shiv to be supported completely by all major browsers except for chrome, with Microsoft Edge and Safari only supporting one of the 4 major sections of Web Components. For these reasons it was not chosen to be used in the project since it was decided that compatibility must be a key factor of any user interface designed.

Material Design Lite (Google, Inc, 2015) is the official public release of a Material Design CSS framework by Google. It has complete support by all four major browsers and has no JavaScript requirements with all animations, such as menu bar side-in being done with CSS animation. It does lack some features such as banners and pop-up dialogue boxes.

Materialize CSS (Wang, et al., 2015) is an open source CSS framework built to the Material Design principles by a group of students at Carnegie Mellon University. It uses a combination of CSS and client side JavaScript to provide an excellent selection of elements to use. It supports all 4 major browsers and has a similar set of standard elements as Material Design Lite, but since it provides a client side JavaScript library it can support some more complex elements such as

collapsible lists (used on the staff exercise details page) and modals (used for the support chat link).

4.2.4 Express-generator & Express

Express.js (ExpressJS, 2016) is a framework for NodeJS designed to build fast and scalable web applications. It is considered to be de facto standard when building NodeJS web applications (Serby, 2012).

The Express generator is an add-on tool which is strongly suggested when developing a web application from scratch which is going to be based around Express.js. It allows the user to specify basic information about the application they are building such as which render engine they wish to use; it selects Jade by default but for this project EmbeddedJS was selected.

Express generator will then use the information provided to create the skeleton of the application and the necessary modules for essential functions such as cookie parsing, http processing, error logging, static file serving, etc.

4.2.5 Pm2

PM2 (also known as production process manager) is a node application built to automatically maintain node web applications. It supports automatic load balancing, automatic clustering, log aggregation, automated deployment, and system monitoring as key features.

The main feature used in the project is PM2's cluster mode. By default, node applications run as a single non-blocking event loop, but as was found when programming usage of SQLite, some operations can be severely impacted by its single threaded nature. To counter this support was added for using cluster mode to allow multiple instances of the application to run.

This allows requests to be served by multiple different instances of the application depending on current load levels. For deploying updates, it also simplifies the work flow by using an auto-pull module which checks periodically for updates to the project git repository and does a hot reload of the running clustered instances so that during the reload no open connections to clients are broken and no downtime is registered even for a split second.

4.2.6 EmbeddedJS (EJS)

EmbeddedJS is a html output rendering engine for either in browsers or, as used in this project, server side html template rendering.

As mentioned previously (4.2.4) express by default uses the Jade engine for HTML output. Jade uses a completely different style to what is commonly used for template rendering, using a unique tag-less indentation heavy markdown (Gorbatchev, 2014).

Due to EJS using standard HTML to build the page, but supporting using specialised delimiters to use logic, includes, and variables. It was chosen to be the HTML output renderer for the project.

4.2.7 Piwik

Piwik is an open source application to track user interaction on a web site. It tracks users approximate geographic location, ISP, operating system, browser, and more.

It's feature set is similar to Google Analytics, which was also considered for the project. Google Analytics and Piwik both use JavaScript embedded into a webpage, with a tracking cookie set to allow cross-page tracking of users to monitor the user flow of the website and how that could be optimised.

Piwik was chosen over google analytics since it is able to be self-hosted which allows user data to be stored securely and erased at the end of the project for ensured data security.

4.2.8 KeyMetrics.io & PMX

KeyMetrics.io, and its interaction module PMX, are a monitoring service for NodeJS applications. It works at a base level with PM2 to display currently running processes, but with PMX it allows any node application to run in a cluster mode.

Cluster mode allows a node application to scale across all available CPU's, load balance network traffic based on process CPU utilization, and use hot reloading for 0-second downtime. Hot reloading works by restarting each running process of the application one-by-one ensuring that it has been restarted successfully before restarting any other process (KeyMetrics, Inc, 2016).

Cluster mode requires the application must be stateless. Thanks to the way session handling works in express, the project application is stateless since they read the session encryption keys from the same configuration file.

4.2.9 jQuery

jQuery, also known as the "write less, do more" library, is used by the Materialize CSS framework (4.2.3 above). It is designed to make document navigation/manipulation, animation, event handling and more trivially easy, by abstracting away low level interactions.

The project uses jQuery since it is a required component of the JavaScript functions, such as the modal used by the support chat pop-up panel.

4.2.10 Bower

Bower is a package manager for client-side libraries (

Appendix F), in a similar method to NPM. The project uses it to manage fonts, CSS, and JavaScript dependences, and can also automatically update libraries for deployment.

4.2.11 Helmet & CSURF

Helmet is an ExpressJS middleware which is designed to add HTTP security headers. In its current configuration helmet will hide the powered-by header, block iframe access, block sniffing of MIME type, protect against cross site scripting attacks

Csurf is an ExpressJS middleware built to issue and validate CSRF tokens which for processing user logins (De Ryck, et al., 2011). CSRF tokens work by providing a unique token which are checked on submission to stop malicious actors from forcing users unwanted actions on the site.

4.3 DEVELOPMENT ENVIRONMENT

The initial development environment was done on Windows 10, using NodeJS 5.7.1 and MongoDB 3.2.3. For writing code originally Notepad++ was used but it was changed to use Sublime Text due to its superior code development features. The primary test browser for the Database Learning System was Google Chrome but the final version was also tested in Mozilla Firefox.

Version control was used for this application. The version control used was git with the repository hosted on the StrathTech GitLab server. The full log of all git commits can be found in

Appendix B.

4.3.1 Cloud9

After developing the skeleton of the application using Notepad++ and Sublime Text, it was brought to the attention of the developer that Cloud9 may be a better development environment.

Cloud9 is a web based integrated development environment, which uses google to host a VPS which it interacts with to run your code on a live Linux system. This allows development of NodeJS applications by forwarding certain ports from the google hosted VPS to behind a reverse proxy.

4.3.2 Production Server Environment

To simulate a production environment a server was setup as if the Database Learning System was going to be run there as a service. The server was deployed on a Scale way C2S with Debian 8.2 as the base operating system. The system was then hardened following the Security Checklist guide (Anon., 2016).

A Nginx web server was installed and configured for HTTPS (Appendix A). Valid SSL certificates were obtained from the let's encrypt and setup to auto-renew ad infinitum (Croome, 2015). Pm2 was then setup with an auto pull module also installed. This allowed any additions to the master branch of the git repository to be automatically deployed.

The initial setup was done using the build script which is can be found as build.sh in the root of the project code repository.

4.4 DBLS Features

This section will explain some of the features within the Database Learning System application, along with the methods and technological process which facilitate them.

4.4.1 Automatic Marking

Automated marking of submissions was the core reason this project was suggested. The original proposed design contained a preliminary specification for a unique markdown language which would be row order, and column order ambiguous.

This would have allowed lecturers to define the expected output data and then done a comparison against the student's output. After discussions with the current lecturer for the relational database class and the project supervisor, it was deemed too complex for the project and a simpler method was required.

Once the decision was made to not use a standard client server relational database, and instead use SQLite. A program was found which could differentiate SQLite databases and report any differences between them. A native NodeJS module with this functionality was not able to be found, and in testing the command line SQLdiff client it was found to only be suitable for locating differences in table structure and table contents, not select queries which make up the majority of current class exercise questions.

The current software uses a JSON string comparison method. When a student submits an answer their query is executed on an air gapped database and the output JavaScript object recorded, the lecturer test query is also executed on the air gapped database, as with the student query, and the output object is recorded.

These database objects are converted into JavaScript Object Notation and a string comparison is done between them. If they match, then the student is deemed to have done the exercise correctly and that is recorded into the database for staff review.

The results of all student submitted queries are sent back to the browser as human readable "pretty" JSON. Which is then code highlighted client side by highlight.js in the same scheme as GitHub so students can read the returned code even easier.

4.4.2 Exercise Overview

After discussions with the project supervisor a staff section which would display currently available exercises, student answers, and allow the addition of new exercises into the live system.

The primary staff overview page displays the current number of student viewable exercises, and how many have been completed. This will allow staff an over view on the answers given, and allow them to review student answers.

The staff exercise overview page displays information about the selected exercise, such as question, answer, and the test query. Additionally, all student answers are shown as a collapsible list. This allow a higher number of responses to be shown in a smaller area, with a detailed view showing submission time and submitted student query viewable via a clicking on the student's identifier in the list.

Finally, there is an add exercise page which allows staff to add newly assigned questions for students to attempt. It currently takes three text fields; answer (currently not used but is included for future expansion. See 0 below), the test query which is used to compare against student submissions, and the question. The question field is displayed by the application as un-escaped HTML; this allows the staff submitting the question to include images, highlights, and text formatting.

The add exercise page also requires the uploading of the test database which uses a file picker. This can be uploaded either by clicking the large file button to open a file selector, or by dragging and dropping the file onto the page. A successful submission will cause a success pop-up to appear.

4.4.3 User Profile

A user profile page was created to display all currently available exercises, as well as currently completed exercises. As an additional feature it also allows users to link and unlink their accounts to the integrated external login systems described 4.4.4 below. As a touch for personalization if the users email is available their gravatar will be requested for display on the page.

Previous versions of the user profile page put a tick or a cross beside each exercise to display if they had been completed, but this system continually had trouble which would either display no exercises or if all exercises we're completed a duplicated list of exercises.

4.4.4 External Login Integration

One of the requested features that was to be developed was allowing external login from third party services, this was to allow students to authenticate with their University of Strathclyde DS accounts. Thanks to the passport library used for handling user authentication, supporting account linking from multiple sources was easily doable.

4.4.4.1 SAML, LDAP, and University Authentication

The initial idea for supporting integration with the university was to utilise the Computer and Information Sciences SimpleSAML gateway service. This is designed to allow web applications to send authentication requests to the gateway, and allow it to process the authentication and return the authenticated user profile.

Initially at the start of the project the SimpleSAML gateway was setup as a service provider. This will mean it will only return the username and confirmation of authentication of said user. When using passport, it is not possible to use it in this format due to passport requiring a user profile be returned.

Passport-saml, the passport module which processes SAML based authentication, expects the SAML gateway to act as an identity provider. Which the Computer and Information Sciences SimpleSAML gateway was not setup to provide. A SimpleSAML gateway was eventually reconfigured for use as an identity provider. For unknown reasons this continually returned request errors from the SimpleSAML gateway, and it was unable to be traced by the Computer and Information Sciences Support Team.

An alternative which was proposed was using Lightweight Active Directory Protocol (LDAP) for authenticating users. In researching this option, it was found that the Computer and Information Sciences department runs its own federated LDAP server which allows anonymous binding.

The federated LDAP server however only stores the authentication information for Computer and Information Sciences department students only, so no student outside of the department would be able to authenticate. When information was queried about using the master LDAP server run by University IT, it was found to require an authorized distinguished name to bind to.

When Computer and Information Sciences System Support was asked for information on how they would wish an LDAP based user authentication system be implemented they strongly discouraged implementing this route since they did not with a student project to take-in valid student authentication details.

4.4.4.2 oAuth; GitLab, Twitter, & Google

With the inability to authenticate against one of the normal methods an alternative was sought. A GitLab server is currently being phased into the current class structure so students can learn git before they go into industry. It was found that a GitLab server has the ability to act as an oAuth

server, which would allow a user to authenticate from the GitLab service and be returned with a user profile to the application.

The Strathclyde Technology Society (StrathTech) allowed the application to be tested on their GitLab server, and this proved that it was a possible method to authenticate users against. The issue with the Computer and Information Science Department GitLab server is that the StrathTech server authenticates users against their LDAP backend, but the Computer and Information Sciences' GitLab is setup to only use local accounts with each account being manually created and thus is not suitable as an authentication backend.

Since GitLab oAuth was implemented it was trivial to also implement supporting both Google and Twitter as authentication mechanisms.

4.5 SECURITY FEATURES

As with all modern web applications security is a key element of the Database Learning System software. As mentioned in 4.2.11 above, the application has basic Cross Site Scripting (XSS) (OWASP, 2016) protection provided by a X-XSS-Protection http header which is set to block mode. While most browsers have XSS protection enable by default this header will override the local user setting if they have disabled it.

The application also sends a X-Frame-Options header (OWASP, 2015). This is used to stop clickjacking attacks caused where an attacker uses iframes to cause a user to click elements on the page which they did not know they were interacting with due to it being obscured by transparent or opaque layers.

Additionally, CSRF tokens are used at login and signup. This uses both a cookie and a random token combination to stop unauthorized commands being sent by a malicious site. CSRF tokens are not issued for SQL submissions since it was decided that having the ability to replay a command via reload was a beneficial feature, but it can be enabled at a later date if deemed necessary.

The Database Learning System was also penetration tested by Detectify, a Software as a Service web application which will analyse and attempt to exploit a website to find security bugs including the current OWASP top 10 list. The full summary report from Detectify can be seen Appendix C, but Detectify was not able to find any critical security bugs in the application, and only one application related bug rated at medium risk. The medium rated bug is the CSRF token is not set as HttpOnly, this is by design of the CSURF middleware currently employed for this task since many applications use AJAX or similar to submit CSRF tokens.

5 EVALUATION AND TESTING

This section will discuss the evaluation tests and results that were carried out on the Database Learning System.

5.1 SOFTWARE TESTING

As shown in Appendix L verification tests were carried out after major changes to the application to maintain functionality. This essential in this application since it was able to show several critical, application ruining bugs during development.

An example of one of the critical bugs found was when developing the functionality to add exercises via the staff panel. Since this page requires a file to be uploaded a middleware was used to handle file verification, and validate input. When using the recommended file handler for expressJS based applications it was found that user login, and student exercise submission tests. The tests resulted in page timeouts when sending post requests. With further testing it was found that unless the form's type was explicitly specified the file handling middleware would block further processing until the client timed out throwing an error 400 internally to the application.

During development the "All Exercise Completion" test continually caused issues with the exercise list showing duplicate tests when all exercises as completed. Eventually as the project deadline drew nearer the exercise list was changed to not display the current user's completion status per exercise as this issue was unable to be patched in time

5.2 USER TESTS

Since the Database Learning System was built with end users in mind, getting user feedback for finished first version was essential. The application was trialled on a number of users who had all completed the current class on relational databases.

The test was split into three main components; First users were asked to login to the system, secondly users were asked to attempt three sample SQL questions, finally users were asked to complete a short questionnaire on their experience using the system (Appendix K).

All users found the signup system every easy to complete, with 75% of students using the local signup method and the other 25% using a third party signup method. 75% of students surveyed rated the signup experience "Very satisfactory" or above. In one instance the twitter login did not return successfully to the application but the issue was fixed with a page reload.

When asking about the ease of use 50% of students rated it as "Somewhat satisfied" or lower. The main issue those students raised, as mentioned in their comments, is the flow between different exercise is not intuitive since a user needs to go back to their profile page before proceeding. This was stated very plainly by Participant 4 in their response on any additional thoughts on the software "allow proceeding from one task to the next one without having to go back to the profile".

During user testing a participant managed to find a major bug in the application, in which a user can cause their progress to date to be lost. They found that if you have only one account login method, you can unlink said login method which will cause all their data to be stored but unlinked from any currently stored account.

Overall when asked to compare it to the current relational database class, with 5 being the same as the current class and higher number being better, overall the application was rated as 7.5 out of

a possible 10. Users also requested that having helpers would be a nice additional feature for those that are stuck on specific questions. While hints are in the database schema they remain unimplemented to allow this version of the software to be finished in a reasonable timeframe.

5.3 STANDARDS COMPLIANCE

As with all web applications standards compliance is key to have as much cross browser support as possible. To provide support for the four major browsers as mentioned in 4.2.3 above, the Materialize CSS framework has support for all major browsers. With support for some older browsers implemented using html5shiv. This was tested using the Browser Shots service to request screenshots of the Database Learning System in 168 browsers/operating systems combinations, including Internet Explorer 5.1 – 11.0, Firefox v3.6 – v47.0, and Chrome v9 – v49.

As is recommended the rendered HTML output was put through a HTML validator (World Wide Web Consortium, 2016). Validation is recommended because it is very useful for debugging styling errors (such as missing closing tags), makes the application future proof since the current stylings do not reply on specific quirks caused by browser parsing errors, and it makes maintenance of the interface easier if future developers know that the current output is valid.

When first run this revealed several errors in the templates used by EmbeddedJS to supply the rendered HTM, such as missing div tags in the header, and miss matched paragraph tags.

5.3.1 Colour Blindness

Since the University of Strathclyde has a wide variety of students, some of which may have visual difficulties. It was requested that the user interface be accessible to as wide variety of students as possible.

A plugin for chrome was found which is able to overlay a filter onto a webpage to simulate viewing it with a variety of the different kinds of colour blindness (Lvivski, 2014). These are shown in Appendix J: Figures 10 – 16, with Figure 9 being a control image with no filter applied.

As shown in Appendix J all content is clearly readable no matter what kind of colour blindness a student may have. A practical test was carried out as one of the persons who carried out a user test on the application is blue-green colour blind and had no difficulty using the application.

6 SUMMARY AND CONCLUSION

In this section a summary for the overall success of the project, Database Learning System. Points discussed will include overview about the good and bad points of the project, possible improvements for future versions, and a general summary of the project.

6.1 SUMMARY

Overall this project has been successful in meeting the primary objectives set out, and will provide a robust solution managing class exercises in future years. While the project is by no means perfect, it's generally high approval rating from test users show to the high quality of the

application produced. The application has room for further enhancement of its feature-set, since some features have room for more functionality and there is room for new features to be added to the application.

6.2 FUTURE WORK

Some ideas of future enhancements for the project which were not fully developed in the current version of the application or new features.

6.2.1 Feature Enhancements

Some features in the Database Learning System were not able to be developed fully to be ready for the final submitted version. Some ideas on enhancing these are:

- Re-design of the user flow and a better study of the human/computer interface. As was mentioned in the user testing many found that once the exercise was completed having a "next exercise" button would have been a great addition.
- Add clearer information to the user about what exercises they have completed
- Move user answers from being stored in the current SQLite database, to be stored as part of the user object generated by passport. This would make it easier to match user information records to their exercise results.
- Store exercises as a MongoDB collection. While this wasn't able to be developed for this version it would make the software easier to scale using MongoDB's built in replication management.
- Fix the account linking, so as mentioned in 5.2 above, users cannot unlink their last account and thus accidentally erase their current account in MongoDB.
- Add fine grained permissions. The permissions model is currently a Boolean for Admin or not, adding a demonstrator level may be added to allow read only review.

6.2.2 New Features

Some features which were considered but not included in the Database Learning System due to various reasons are as follows:

- Using a client-server SQL server for exercise execution. This was deemed not to be feasible with the allotted timeframe, but would allow students to run commands in an environment closer to what they will find in industry.
- Function to export student answers in various formats. This could be for a student to export their own answers for review, or if a lecturer all student replies.
- Add feedback functionality, so feedback from lecturers and/or demonstrators can be sent to students regarding their exercise submissions.
- While the current system can store exercise hints, the display feature is not currently implemented. Adding this so after a number of unique attempts a hint could become available or a time after the exercise become available.
- Currently exercises are available as soon as the lecturer adds the exercise, adding a scheduled release option would be a good feature so a weekly exercise can be made available.

6.3 CONCLUSION

In conclusion the Database Learning System has satisfied the constraints placed on it, and arrived at a successful outcome. Since many of the features which were originally suggested have been implemented and the high user rating given shows that it is an improvement over the current methodology used when teaching the Relational Database class. There is room for improvement of both current and new features of the system in future versions, which makes the future of the Database Learning System intriguing and full of potential.

REFERENCES

Anon., 2016. *securitychecklist.org.* [Online]
Available at: https://securitychecklist.org/
[Accessed 27 March 2016].

Arg! Team, 2012. *HARDENING NODE.JS FOR PRODUCTION PART 2: USING NGINX TO AVOID NODE.JS LOAD.* [Online]
Available at: http://blog.argteam.com/coding/hardening-node-js-for-production-part-2-using-nginx-to-avoid-node-js-load/

Breslow, L. et al., 2013. Studying Learning in the Worldwide Classroom: Research into edX's First MOOC. *Research & Practice in Assessment,* Volume 8.

Croome, C., 2015. *Using the webroot domain verification method - Server - Let's Encrypt Community Support.* [Online]
Available at: https://community.letsencrypt.org/t/using-the-webroot-domain-verification-method/1445/46
[Accessed 2 March 2016].

Dakkak, N., 2013. *What are MOOCs, and how can you benefit from them?.* [Online]
Available at: http://www.wamda.com/2013/05/what-are-moocs-what-mean-for-middle-east
[Accessed 21 March 2016].

De Ryck, P., Desmet, L., Joosen, W. & Piessens, F., 2011. Automatic and Precise Client-Side Protection against CSRF Attacks. *Computer Security – ESORICS 2011,* 12-14 September, pp. 100-116.

Embry, D., 2016. *Why PHP Sucks.* [Online]
Available at: http://webonastick.com/php.html
[Accessed 27 March 2016].

ExpressJS, 2016. *Express - Node.js web application framework.* [Online]
Available at: http://expressjs.com/
[Accessed 26 March 2016].

Google, Inc, 2015. *Material design Lite.* [Online]
Available at: https://www.getmdl.io/
[Accessed 26 March 2016].

Google, Inc, 2016. *Introduction - Material design - Google design guidelines.* [Online]
Available at: https://www.google.com/design/spec/material-design/introduction.html
[Accessed 26 March 2016].

Gorbatchev, A., 2014. *Comparing JavaScript Templating Engines: Jade, Mustache, Dust and More.* [Online]
Available at: https://strongloop.com/strongblog/compare-javascript-templates-jade-mustache-dust/

Hickson, I. et al., 2014. *A vocabulary and associated APIs for HTML and XHTML.* [Online]
Available at: https://www.w3.org/TR/html5/
[Accessed 22 March 2016].

KeyMetrics, Inc, 2016. *Automatic load balancing.* [Online]
Available at: http://pm2.keymetrics.io/docs/usage/cluster-mode/

Lvivski, Y., 2014. *Spectrum - Chrome Web Store.* [Online]
Available at:
https://chrome.google.com/webstore/detail/spectrum/ofclemegkcmilinpcimpjkfhjfgmhieb?utm_source=chrome-app-launcher-info-dialog
[Accessed 16 March 2016].

MongoDB Inc., 2016. *Replication Introduction.* [Online]
Available at: https://docs.mongodb.org/manual/core/replication-introduction/
[Accessed 22 March 2016].

MOOC List, 2016. *MIT | MOOC List.* [Online]
Available at: https://www.mooc-list.com/university-entity/mit?static=true
[Accessed 21 March 2016].

Munroe, L., 2012. *PHP: a fractal of bad design.* [Online]
Available at: https://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/
[Accessed 27 March 2016].

npm, Inc, 2016. *npm.* [Online]
Available at: https://www.npmjs.com/

Orsini, L., 2013. *What You Need To Know About Node.js.* [Online]
Available at: http://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs/
[Accessed 22 March 2016].

OWASP, 2015. *Clickjacking.* [Online]
Available at: https://www.owasp.org/index.php/Clickjacking
[Accessed 25 March 2016].

OWASP, 2016. *Cross-site Scripting (XSS).* [Online]
Available at: https://www.owasp.org/index.php/Cross-site_scripting
[Accessed 25 March 2016].

Polymer Project, 2016. *Welcome - Polymer 1.0.* [Online]
Available at: https://www.polymer-project.org/1.0/
[Accessed 26 March 2016].

Prior, J. C., 2003. Online Assessment of SQL Query Formulation Skills. *Proceedings of the Fifth Australasian Conference on Computing Education*, pp. 247-256.

Serby, P., 2012. *Case study: How & why to build a consumer app with Node.js.* [Online]
Available at: http://venturebeat.com/2012/01/07/building-consumer-apps-with-node/

Siemens, G., 2012. *MOOCs are really a platform.* [Online]
Available at: http://www.elearnspace.org/blog/2012/07/25/moocs-are-really-a-platform/
[Accessed 21 March 2016].

Teixeira, P., 2013. *Professional Node.js: Building Javascript Based Scalable Software.* 1st ed. Indianapolis: John Wiley & Sons, Inc.

Tilkov, S. & Vinoski, S., 2010. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing,* 14(6), pp. 80-83.

Udacity, Inc., 2016. *Intro to Artificial Intelligence | Udacity.* [Online]
Available at: https://www.udacity.com/course/intro-to-artificial-intelligence--cs271
[Accessed 21 March 2016].

United Software Associates, 2015. *High Performance Benchmarking,* Pleasanton, CA: United Software Associates Inc..

University of Strathclyde, 2016. *Class Catalogue - Class Details.* [Online]
Available at:
http://ben.mis.strath.ac.uk/classcatalogue/control/classdets?uiocode=123958&show=basics&displayCalendar=

Vaughan-Nichols, S. J., 2014. *Hands on with Ubuntu 14.04: The best Ubuntu desktop ever.* [Online]
Available at: http://www.zdnet.com/article/hands-on-with-ubuntu-14-04-the-best-ubuntu-desktop-ever/
[Accessed 21 March 2016].

Wang, A., Chang, A., Mark, A. & Louie, K., 2015. *Documentation - Materialize.* [Online]
Available at: http://materializecss.com/
[Accessed 26 March 2016].

Wilson, M., 2013. *Building Node Applications with MongoDB and Backbone.* 1st ed. Sebastopol: O'Reilly Media, Inc.

World Wide Web Consortium, 2016. *Showing results for https://dbls.ovh/ - Nu Html Checker.* [Online]
Available at: https://validator.w3.org/nu/?doc=https%3A%2F%2Fdbls.ovh%2F
[Accessed 26 March 2016].

APPENDIX A

NGINX server configuration

```
server {
   listen       80;
   server_name dbls.ovh localhost;
   if ($msie) {
      return 301 https://$host/no-ie.html;
   }
   location '/.well-known/acme-challenge' {
      default_type "text/plain";
      root        /tmp/letsencrypt-auto;
   }
  location / {
     return 307 https://$host$request_uri;
  }
}
server {
   listen 443 ssl http2;
   ssl_certificate /etc/letsencrypt/live/dbls.ovh/fullchain.pem;
   ssl_certificate_key /etc/letsencrypt/live/dbls.ovh/privkey.pem;
   gzip on;
   gzip_vary on;
   gzip_proxied any;
   location /bower/ {
      root /root/dbls/;
      try_files $uri $uri/ =404;
   }
   location /static/ {
      root /root/dbls/;
      try_files $uri $uri/ =404;
   }
   location = /no-ie.html {
      root   /usr/share/nginx/html;
   }
   location / {
       if ($msie) {
          return 301 https://$host/no-ie.html;
       }
       proxy_pass http://127.0.0.1:3000;
   }
   error_page   500 502 503 504  /50x.html;
   location = /50x.html {
      root   /usr/share/nginx/html;
   }
}
```

APPENDIX B

Git Log

* b299bc9 - fix many thing * Enable sweet alerts * fix login error flashes * fix exercise addtions breaking login and post requests * add error handling to add exercise page (3 minutes ago) <Adam McGhie>
* 0f3e26e - serialize breaks stuff mkay (3 days ago) <Adam McGhie>
* f3adbe6 - fix the last fix (3 days ago) <Adam McGhie>
* 989a16f - fixes and CSRF stuff (3 days ago) <Adam McGhie>
* 9f031f1 - serialize main db queries on staff page (7 days ago) <Adam McGhie>
* cc619cc - cache db on staff page (7 days ago) <Adam McGhie>
* 053fafd - add network and port monit (7 days ago) <Adam McGhie>
* bc52fb5 - serialize db connections so we can actually close it (7 days ago) <Adam McGhie>
* 1b5c302 - add pmx to required mod (8 days ago) <Adam McGhie>
* cc0382f - update exercise table (8 days ago) <Adam McGhie>
* d12cd01 - add pmx to monitor http traffic via pm2 (8 days ago) <Adam McGhie>
* 6ca3f7e - add staff ex page (8 days ago) <Adam McGhie>
* 4031228 - add ability to add exercises from web page (8 days ago) <Adam McGhie>
* 4e08d17 - derp (10 days ago) <Adam McGhie>
* 6886a6e - fix staff display of exercises (10 days ago) <Adam McGhie>
* 71b2d48 - fix duplication of exercises on all ex completion (10 days ago) <Adam McGhie>
* 13ab5b6 - remove db close because it breaks important stuff like submitting exercises (11 days ago) <Adam McGhie>
* 6b7bd38 - close memory leak caused by open db connection and fix #4 (12 days ago) <Adam McGhie>
* 29b8dd0 - use material icons on android and web. (2 weeks ago) <Adam McGhie>
* ea6f5d0 - add material icons for android (2 weeks ago) <Adam McGhie>
* 7f7ca96 - fix display of side bar when logged out (2 weeks ago) <Adam McGhie>
* 46ef4f2 - fix not displaying exercises correctly when none have been answered (2 weeks ago) <Adam McGhie>
* 0bc142f - revoke last commit. Programmer is caused an ID-10T error (2 weeks ago) <Adam McGhie>
* 9577334 - add fix for manifest since side bar doesn't work in standalone mode (2 weeks ago) <Adam McGhie>
* d5d9fd1 - add manifest and fix icon locations (2 weeks ago) <Adam McGhie>
* 216e616 - fix icon color (2 weeks ago) <Adam McGhie>
* 33e9769 - add logo generated in icon form with http://www.gieson.com/Library/projects/utilities/icon_slayer/ (2 weeks ago) <Adam McGhie>
* db54a40 - add staff stuff (2 weeks ago) <Adam McGhie>
* df14ef9 - fix user page issues (2 weeks ago) <Adam McGhie>
* d53ca48 - add exercise overview page (2 weeks ago) <Adam McGhie>
* 9569b26 - Add staff page (2 weeks ago) <Adam McGhie>
* 348c2c8 - add html5shiv to bower list (3 weeks ago) <Adam McGhie>

* 1bfdde6 - fix minor UI bugs (3 weeks ago) <Adam McGhie>
* fc2cb16 - add demo of HTML encoding in exercise questions (3 weeks ago) <Adam McGhie>
* de06e9a - add respond for IE people, fix highlightJS not rendering (3 weeks ago) <Adam McGhie>
* 8f2f213 - fix issue of login page throwing EJS render error (3 weeks ago) <Adam McGhie>
* 709bc84 - make progress bar indeterminate if 0% completed (3 weeks ago) <Adam McGhie>
* 19b7eff - add custom error 500 page (3 weeks ago) <Adam McGhie>
* 8d6af78 - center text on 404 (3 weeks ago) <Adam McGhie>
* a067490 - remove config (3 weeks ago) <Adam McGhie>
* 1e35141 - add environment option to config (3 weeks ago) <Adam McGhie>
* af74c34 - do config stuff (3 weeks ago) <Adam McGhie>
* 0585e97 - make it pretty (3 weeks ago) <Adam McGhie>
* c962220 - use custom 404 page & increase cache delay on static hosting (3 weeks ago) <Adam McGhie>
* 083e734 - add 404 page (3 weeks ago) <Adam McGhie>
* d6c8165 - Change output on profile page render (3 weeks ago) <Adam McGhie>
* c02c318 - clear staff output (3 weeks ago) <Adam McGhie>
* c8b3937 - add isAdmin flag, remove CIS auth section (3 weeks ago) <Adam McGhie>
* 19eef01 - move some CSS from head to stylesheet (3 weeks ago) <Adam McGhie>
* 73fba97 - add sweetalerts to html output (3 weeks ago) <Adam McGhie>
* 2e080f4 - add SweetAlerts (3 weeks ago) <Adam McGhie>
* 73b7962 - fix profile button not being on the right side of profile card (3 weeks ago) <Adam McGhie>
*   e819749 - Merge branch 'compression' into 'master' (3 weeks ago) <Adam McGhie>
|\
| * 701a897 - (origin/compression, compression) async & defer js and dns-prefetch all the things (3 weeks ago) <Adam McGhie>
| * e8f9363 - add support for sending http content with compression (3 weeks ago) <Adam McGhie>
|/
* ae5bef4 - async and defer blocking JS (3 weeks ago) <Adam McGhie>
* 756a639 - add link to SQLite Language docs (3 weeks ago) <Adam McGhie>
* 08b6714 - fix duplicate display when all exercises completed (3 weeks ago) <Adam McGhie>
* 8b1051a - clean up profile view (3 weeks ago) <Adam McGhie>
* 5b1c002 - fix duplicate display of assigned exercises (4 weeks ago) <Adam McGhie>
* a2b9e69 - fix profile card prematurely ending container (4 weeks ago) <Adam McGhie>
* 57701f7 - exclude exercise.db and require copy of default.db (4 weeks ago) <Adam McGhie>
* 681fe73 - add blank staff route (4 weeks ago) <Adam McGhie>
* d7b5882 - fix adding completed exercises to user’s records (4 weeks ago) <Adam McGhie>
* e95cfbf - Fix user profile to correctly display attempted exercises and progress stats (4 weeks ago) <Adam McGhie>
* 006b2a0 - add auto-marking of user submissions (4 weeks ago) <Adam McGhie>
* e70934f - fix db (4 weeks ago) <Adam McGhie>

```
*   89fefbf - Merge branch 'master' of https://gitlab.strathtech.co.uk/amcghie/dbls (4 weeks ago)
<Adam McGhie>
|\
| * 838e260 - fix build script issues (4 weeks ago) <Adam McGhie>
| * 4d140ab - fix public folder (4 weeks ago) <Adam McGhie>
| * 09d6cbe - add consolidate profile card and use bower provided CSS, JS, & Fonts (4 weeks
ago) <Adam McGhie>
* | c10c117 - add answers table to exercise to store answers from users (4 weeks ago) <Adam
McGhie>
|/
* a5a6498 - change setup to build script because $BULLSHIT_REASONS (4 weeks ago) <Adam
McGhie>
* 6d3ed37 - use local GitHub style for code highlight (4 weeks ago) <Adam McGhie>
* ebe5781 - change highlight.js to use local instead of CDN (4 weeks ago) <Adam McGhie>
* 7f15442 - add highlightjs (4 weeks ago) <Adam McGhie>
* 181a3f2 - add folder creation for bower components (4 weeks ago) <Adam McGhie>
* d264c6e - add font awesome to bower requirements (4 weeks ago) <Adam McGhie>
* 5fd278d - recurse copy for bower components (4 weeks ago) <Adam McGhie>
* 2cf8408 - allow bower to execute as root (4 weeks ago) <Adam McGhie>
* 14472a0 - you already have the code why clone it again? (4 weeks ago) <Adam McGhie>
* 0154cf7 - fix typo (4 weeks ago) <Adam McGhie>
* 07117f7 - add setup script (4 weeks ago) <Adam McGhie>
* 78da166 - remove submodules and just use a setup script in future (4 weeks ago) <Adam
McGhie>
* 6df7933 - Use bower to install components (4 weeks ago) <Adam McGhie>
* 9a32441 - fix modules (4 weeks ago) <Adam McGhie>
* be9dd0c - remove stuff (4 weeks ago) <Adam McGhie>
* 3648d46 - change submodule location (4 weeks ago) <Adam McGhie>
* f92b47f - change path for serving static files (4 weeks ago) <Adam McGhie>
* a617a7e - add materialize css theme to submodules (4 weeks ago) <Adam McGhie>
* a48bf69 - that should have properly removed it now (4 weeks ago) <Adam McGhie>
* b9cfc42 - remove bootstrap submodule (4 weeks ago) <Adam McGhie>
*   0e7c578 - Merge branch 'new-theme' (4 weeks ago) <Adam McGhie>
|\
| * 5cbab96 - (origin/new-theme) update exercise db with different question 2 (4 weeks ago)
<Adam McGhie>
* | cd1a9cf - add some debug flags (4 weeks ago) <Adam McGhie>
* | 0857bab - fill out GitLab schema (4 weeks ago) <Adam McGhie>
* | f918674 - fix callback (4 weeks ago) <Adam McGhie>
* | 323f675 - fix call back path on account link (4 weeks ago) <Adam McGhie>
* | 206a654 - fix GitLab auth (4 weeks ago) <Adam McGhie>
* | 3aa393b - add icons and squashin bugs (4 weeks ago) <Adam McGhie>
* | 0e86e79 - add fontawesome icons (4 weeks ago) <Adam McGhie>
* | f5b70fe - change login button spacing (4 weeks ago) <Adam McGhie>
```

```
* | 60b1036 - remove disquis thread trigger (4 weeks ago) <Adam McGhie>
* | eb9394e - fix UI overlap on index page (4 weeks ago) <Adam McGhie>
* |   67be06e - Merge branch 'master' of https://gitlab.strathtech.co.uk/amcghie/dbls (4 weeks ago)
<Adam McGhie>
|\ \
| * | b5dbdf0 - Fix twitter oauth details (4 weeks ago) <Adam McGhie>
* | | 8162d99 - fix minor issue when first loading exercise page via get request (4 weeks ago)
<Adam McGhie>
|/ /
* |   096e5d5 - Merge branch 'new-theme' into 'master' (4 weeks ago) <Adam McGhie>
|\ \
| |/
| * 2651621 - (new-theme) update link to use custom kiwi client (4 weeks ago) <Adam McGhie>
* |   bf69c84 - Merge branch 'new-theme' into 'master' (4 weeks ago) <Adam McGhie>
|\ \
| |/
| * 9870434 - add comparison of lecturer vs. student answer and reply based on it (4 weeks ago)
<Adam McGhie>
| * 3615fe0 - fix view to look FABULOUS! (4 weeks ago) <Adam McGhie>
| * 22dd7d9 - fixes for material theme (4 weeks ago) <Adam McGhie>
| * 2da8cb8 - add materialize public files (4 weeks ago) <Adam McGhie>
| * eaa0ba6 - Try using materialize CSS (4 weeks ago) <Adam McGhie>
* | f576723 - yeah that don't work bro (4 weeks ago) <Adam McGhie>
|/
*   c7f74ff - Merge branch 'master' of https://gitlab.strathtech.co.uk/amcghie/dbls (4 weeks ago)
<Adam McGhie>
|\
| * 0aa95f8 - Add license (4 weeks ago) <Adam McGhie>
* | d6feacd - add pm2 support (4 weeks ago) <Adam McGhie>
|/
* f5f7536 - add emoji.rodeo auth validation script (4 weeks ago) <Adam McGhie>
* 22227f3 - fix stuff (4 weeks ago) <Adam McGhie>
* 2ae94ce - update exercise db with ex2 database (4 weeks ago) <Adam McGhie>
* e4d552e - remove api auth (4 weeks ago) <Adam McGhie>
* 66c04e1 - api all the things (4 weeks ago) <Adam McGhie>
* a894f18 - change highlight theme (4 weeks ago) <Adam McGhie>
* 34b7b3f - make the JSON supah mega pretty. Just like me (4 weeks ago) <Adam McGhie>
* 0dd3796 - pretty JSON is pretty (4 weeks ago) <Adam McGhie>
* 4a894f5 - try all (4 weeks ago) <Adam McGhie>
* e54bb30 - it ain't run its exec (4 weeks ago) <Adam McGhie>
* 39bb5b3 - Update exercise to run instead of get SQL, bootstrap stuff, hide tmp dir with Git
ignore (4 weeks ago) <Adam McGhie>
* 282622e - remove database when usage is over (4 weeks ago) <Adam McGhie>
* 23954d8 - stringify userRow (4 weeks ago) <Adam McGhie>
```

* 6e2d415 - rename row2 to userRow to clarify usage (4 weeks ago) <Adam McGhie>
* 7947548 - add type row to db, change saveOut location for ex db's (4 weeks ago) <Adam McGhie>
* d5de399 - remove data buffer from console.log (4 weeks ago) <Adam McGhie>
* 507350b - test (4 weeks ago) <Adam McGhie>
* 783733d - add some debug infos (4 weeks ago) <Adam McGhie>
* 9ee12d6 - fix writeout to make sure folder exists (4 weeks ago) <Adam McGhie>
* 407de5f - use fileout to prove database survives the db trip (4 weeks ago) <Adam McGhie>
* 7446e20 - attempt fix (4 weeks ago) <Adam McGhie>
* c1da73e - forgot a fuckin'); fml (4 weeks ago) <Adam McGhie>
* ae364d0 - add db echo back (4 weeks ago) <Adam McGhie>
* 255b51e - update exercise.db with a proper question database (4 weeks ago) <Adam McGhie>
* 8eecd80 - change hilight.js to alternate style (4 weeks ago) <Adam McGhie>
* f29be62 - add hilight.js support to SQL output (4 weeks ago) <Adam McGhie>
* 41b3f9d - change emoji-rodeo theme (4 weeks ago) <Adam McGhie>
* 4b4b659 - add shared footer to all pages (4 weeks ago) <Adam McGhie>
* 58dd5d6 - add shared footer with emoji-rodeo for reaction (4 weeks ago) <Adam McGhie>
* 4a4f71e - update shit (4 weeks ago) <Adam McGhie>
* 44b61df - attempt to fix this fucking thing (5 weeks ago) <Adam McGhie>
* 6a872ac - add debug output for StrathTech GitLab auth profile (5 weeks ago) <Adam McGhie>
* 444f0d0 - simplify GitLab storage (5 weeks ago) <Adam McGhie>
* 9405459 - profiles! (5 weeks ago) <Adam McGhie>
* 533c819 - try fixing stuff (5 weeks ago) <Adam McGhie>
* d1c66ce - change host url (5 weeks ago) <Adam McGhie>
* 9422ad1 - fix stuff (5 weeks ago) <Adam McGhie>
* 14f5586 - add StrathTech auth (5 weeks ago) <Adam McGhie>
* af0e6ad - add stuff (5 weeks ago) <Adam McGhie>
* ee3c975 - JQuery first before the rest of the useless bull shit (5 weeks ago) <Adam McGhie>
* 4957aeb - add foot... may have forgotten the damn thing (5 weeks ago) <Adam McGhie>
* 33bbeaf - add shared foot with JS (5 weeks ago) <Adam McGhie>
* bacb3c1 - add control meta's to head (5 weeks ago) <Adam McGhie>
* a3473b5 - update mini icons (5 weeks ago) <Adam McGhie>
* 3ec3cf2 - change icon used on exercise page (5 weeks ago) <Adam McGhie>
* 42c7c6e - add titles to ejs renders (5 weeks ago) <Adam McGhie>
* c0cb8a6 - suffix html :titles (5 weeks ago) <Adam McGhie>
* 682a42e - add common head section (5 weeks ago) <Adam McGhie>
* d97c8c3 - reformat exercise so it looks better on small screens (5 weeks ago) <Adam McGhie>
* 8022dc4 - add exercise well (5 weeks ago) <Adam McGhie>
*   bba4ca9 - Merge branch 'master' of https://gitlab.strathtech.co.uk/amcghie/dbls (5 weeks ago) <Adam McGhie>
|\
| * 5034a66 - update domain (5 weeks ago) <Adam McGhie>
* | fb3457c - change some links (5 weeks ago) <Adam McGhie>
|/

* 52f94d7 - stuff happened (5 weeks ago) <Adam McGhie>
* 22b8f06 - fix exercise lookups (5 weeks ago) <Adam McGhie>
* b50bb33 - add generic view for future use (6 weeks ago) <Adam McGhie>
* a473b40 - add routes for all the things remove Facebook from index login options update packages necessary (6 weeks ago) <Adam McGhie>
* 3382957 - Start on exercise processor (6 weeks ago) <Adam McGhie>
* 780858e - Add authentication details (6 weeks ago) <Adam McGhie>
* 5987b87 - update app.js Add exercise route Load correct configs fix session cookies (6 weeks ago) <Adam McGhie>
* fbb7ec9 - add basic system sans exercise processor (6 weeks ago) <Adam McGhie>

APPENDIX C

Detectify Output

dbls.ovh

Scan time

Scan started
2016-03-26 16:11

Scan finished
2016-03-26 16:18

Finding summary

- Name Servers Distributed on Single Autonomous System
- Cookie is not set to be HttpOnly
- Execution After Redirect (EAR)
- HTTP Server Version
- Lacking DMARC Policy
- OPTIONS Disclosure
- Service Providers
- WHOIS

Name Servers Distributed on Single Autonomous System

What does this mean?

RFC 2182 (https://www.ietf.org/rfc/rfc2182.txt)).

What can happen?

If the ASN were to be rerouted in an invalid manner, the entire domain zone would fail and that would in turn make the domain unreachable.

Summary

	Found at	
1	ns18.ovh.net	5.4

Cookie is not set to be HttpOnly

What does this mean?

One or more cookies lack the flag HttpOnly-flag.

our knowledge base (http://support.detectify.com/customer/portal/articles/1969826-missing-httponly-flag-on-cookies).

What can happen?

If an attacker discovers an XSS he may use it to steal cookies which haven't got the HttpOnly-flag.

Summary

Entry	Fount at	CVSS
1	http://dbls.ovh/login	4.3

Execution After Redirect (EAR)

What does this mean?

The server sends data after the redirect.

What can happen?

An attacker can get hold of information intended just for logged in users, which in turn could lead to a complete compromise of the system, although this varies from case to case.

Summary

Entry	Fount at	CVSS
1	http://dbls.ovh/dokuwiki	0
2	https://dbls.ovh/login	0

HTTP Server Version

What does this mean?

The web server publicly discloses which web server software it's and possibly which specific version.

What can happen?

An attacker can use that information to look up known vulnerabilities for the specific software and then use them against the website.

Summary

Entry	Fount at	CVSS
1	http://dbls.ovh	0
2	https://dbls.ovh	0

Lacking DMARC Policy

What does this mean?

The domain lacks a DMARC policy.

What can happen?

An attacker will be able to spoof emails originating from any subdomain having either an A, AAAA or MX record. In most clients this is possible regardless if SPF policies are put in place.

Summary

Entry	Fount at	CVSS
1	Dbls.ovh	0

OPTIONS Disclosure

What does this mean?

Your webserver discloses its supported HTTP methods. This poses no threat by itself. It may however aid an attacker in finding unusual configuration.

Summary

Entry	Fount at	CVSS
1	Dbls.ovh	0

Service Providers

What does this mean?

The listed providers are authorized to host different parts of your infrastructure.

What can happen?

Anyone can retrieve this data. It's only here to serve as an indicator of what vendors have access to.

Summary

Entry	Fount at	CVSS
1	Dbls.ovh	0

WHOIS

What does this mean?

External parties may look up contact information and other data related to your server environment and employees by querying a whois server. These types of lookup services could be used by an attacker to gather intelligence about you and your domain. However, there should always be whois records available. It's a fundamental part of being a domain owner.

Summary

Entry	Fount at	CVSS
1	Dbls.ovh	0

APPENDIX D

Full node module list with dependences listed to depth N

```
dbls@1.0.0 /home/dbls/dbls
├── bcrypt-nodejs@0.0.3
├─┬ body-parser@1.13.3
│ ├── bytes@2.1.0
│ ├── content-type@1.0.1
│ ├── depd@1.0.1
│ ├─┬ http-errors@1.3.1
│ │ ├── inherits@2.0.1
│ │ └── statuses@1.2.1
│ ├── iconv-lite@0.4.11
│ ├─┬ on-finished@2.3.0
│ │ └── ee-first@1.1.1
│ ├── qs@4.0.0
│ ├─┬ raw-body@2.1.6
│ │ ├── bytes@2.3.0
│ │ ├── iconv-lite@0.4.13
│ │ └── unpipe@1.0.0
│ └─┬ type-is@1.6.12
│   ├── media-typer@0.3.0
│   └── mime-types@2.1.10
├─┬ compression@1.6.1
│ ├─┬ accepts@1.3.2
│ │ └── negotiator@0.6.0
│ ├── bytes@2.2.0
│ ├─┬ compressible@2.0.7
│ │ └── mime-db@1.22.0
│ ├── on-headers@1.0.1
│ └── vary@1.1.0
├── connect-flash@0.1.1
├─┬ cookie-parser@1.3.5
│ ├── cookie@0.1.3
│ └── cookie-signature@1.0.6
├─┬ cookie-session@1.2.0
│ └─┬ cookies@0.5.0
│   └── keygrip@1.0.1
├─┬ csurf@1.8.3
```

```
| └─┬ csrf@3.0.1
|   ├── base64-url@1.2.1
|   ├── rndm@1.2.0
|   ├── scmp@1.0.0
|   └─┬ uid-safe@2.1.0
|     └── random-bytes@1.0.0
├─┬ debug@2.2.0
| └── ms@0.7.1
├── ejs@2.3.4
├─┬ express@4.13.4
| ├─┬ accepts@1.2.13
| | └── negotiator@0.5.3
| ├── array-flatten@1.1.1
| ├── content-disposition@0.5.1
| ├── cookie@0.1.5
| ├── depd@1.1.0
| ├── escape-html@1.0.3
| ├── etag@1.7.0
| ├── finalhandler@0.4.1
| ├── fresh@0.3.0
| ├── merge-descriptors@1.0.1
| ├── methods@1.1.2
| ├── parseurl@1.3.1
| ├── path-to-regexp@0.1.7
| ├─┬ proxy-addr@1.0.10
| | ├── forwarded@0.1.0
| | └── ipaddr.js@1.0.5
| ├── range-parser@1.0.3
| ├─┬ send@0.13.1
| | ├── depd@1.1.0
| | ├── destroy@1.0.4
| | └── mime@1.3.4
| ├── serve-static@1.10.2
| ├── utils-merge@1.0.0
| └── vary@1.0.1
├─┬ express-fileupload@0.0.4
| ├── connect-busboy@0.0.2
| └─┬ fs-extra@0.22.1
|   ├── graceful-fs@4.1.3
|   ├── jsonfile@2.2.3
```

```
|   └─┬ rimraf@2.5.2
|     └─┬ glob@7.0.3
|       ├─┬ inflight@1.0.4
|       | └── wrappy@1.0.1
|       ├─┬ minimatch@3.0.0
|       | └─┬ brace-expansion@1.1.3
|       |   ├── balanced-match@0.3.0
|       |   └── concat-map@0.0.1
|       ├── once@1.3.3
|       └── path-is-absolute@1.0.0
├─┬ express-session@1.13.0
| ├── cookie@0.2.3
| ├── crc@3.4.0
| ├── depd@1.1.0
| └── uid-safe@2.0.0
├── fs@0.0.2
├─┬ helmet@1.3.0
| ├── connect@3.4.1
| ├── dns-prefetch-control@0.1.0
| ├── dont-sniff-mimetype@1.0.0
| ├─┬ frameguard@1.1.0
| | └── lodash.isstring@4.0.1
| ├─┬ helmet-csp@1.1.0
| | ├── camelize@1.0.0
| | ├─┬ content-security-policy-builder@1.0.0
| | | └── dashify@0.2.1
| | ├─┬ lodash.assign@4.0.4
| | | ├── lodash.keys@4.0.5
| | | └── lodash.rest@4.0.1
| | ├── lodash.isfunction@3.0.8
| | ├─┬ lodash.reduce@4.2.0
| | | ├── lodash._baseeach@4.1.1
| | | ├── lodash._baseiteratee@4.5.2
| | | └── lodash._basereduce@3.0.2
| | ├── lodash.some@4.2.0
| | └── platform@1.3.1
| ├── hide-powered-by@1.0.0
| ├── hpkp@1.1.0
| ├─┬ hsts@1.0.0
| | └── core-util-is@1.0.2
```

```
│ ├── ienoopen@1.0.0
│ ├── nocache@1.0.0
│ └── x-xss-protection@1.0.0
├── JSON@1.0.0
├─┬ mongoose@4.4.8
│ ├── async@1.5.2
│ ├── bson@0.4.21
│ ├── hooks-fixed@1.1.0
│ ├── kareem@1.0.1
│ ├─┬ mongodb@2.1.8
│ │ ├── es6-promise@3.0.2
│ │ ├─┬ mongodb-core@1.3.5
│ │ │ └─┬ require_optional@1.0.0
│ │ │   ├── resolve-from@2.0.0
│ │ │   └── semver@5.1.0
│ │ └─┬ readable-stream@1.0.31
│ │   ├── isarray@0.0.1
│ │   └── string_decoder@0.10.31
│ ├── mpath@0.2.1
│ ├── mpromise@0.5.5
│ ├─┬ mquery@1.10.0
│ │ ├── bluebird@2.10.2
│ │ └── sliced@0.0.5
│ ├── muri@1.1.0
│ ├── regexp-clone@0.0.1
│ └── sliced@1.0.1
├─┬ morgan@1.6.1
│ └── basic-auth@1.0.3
├─┬ multer@1.1.0
│ ├── append-field@0.1.0
│ ├─┬ busboy@0.2.12
│ │ ├─┬ dicer@0.2.5
│ │ │ └── streamsearch@0.1.2
│ │ └── readable-stream@1.1.13
│ ├─┬ concat-stream@1.5.1
│ │ ├─┬ readable-stream@2.0.6
│ │ │ ├── isarray@1.0.0
│ │ │ ├── process-nextick-args@1.0.6
│ │ │ └── util-deprecate@1.0.2
│ │ └── typedarray@0.0.6
```

```
│ ├─┬ mkdirp@0.5.1
│ │ └── minimist@0.0.8
│ ├── object-assign@3.0.0
│ └── xtend@4.0.1
├─┬ passport@0.3.2
│ ├── passport-strategy@1.0.0
│ └── pause@0.0.1
├─┬ passport-gitlab@0.5.0
│ ├─┬ passport-oauth@1.0.0
│ │ └─┬ passport-oauth2@1.2.0
│ │   └── uid2@0.0.3
│ └── util@0.10.3
├─┬ passport-google@0.3.0
│ ├─┬ passport-openid@0.3.1
│ │ ├── openid@0.5.13
│ │ └── passport@0.1.18
│ └── pkginfo@0.2.3
├─┬ passport-google-oauth@1.0.0
│ ├── passport-google-oauth1@1.0.0
│ └── passport-google-oauth20@1.0.0
├── passport-local@1.0.0
├─┬ passport-saml@0.15.0
│ ├── q@1.1.2
│ ├─┬ xml-crypto@0.8.4
│ │ ├── xmldom@0.1.19
│ │ └── xpath.js@1.0.6
│ ├─┬ xml-encryption@0.7.4
│ │ ├── async@0.2.10
│ │ ├── ejs@0.8.8
│ │ ├── node-forge@0.2.24
│ │ └── xpath@0.0.5
│ ├─┬ xml2js@0.4.16
│ │ ├── sax@1.1.6
│ │ └─┬ xmlbuilder@4.2.1
│ │   └── lodash@4.6.1
│ ├─┬ xmlbuilder@2.5.2
│ │ └── lodash@3.2.0
│ └── xmldom@0.1.22
├─┬ passport-twitter@1.0.4
│ ├─┬ passport-oauth1@1.1.0
```

```
│ │ └── oauth@0.9.14
│ └── xtraverse@0.1.0
├─┬ pmx@0.6.1
│ └── json-stringify-safe@5.0.1
├── serve-favicon@2.3.0
└─┬ sqlite3@3.1.1
  ├── nan@2.1.0
  └─┬ node-pre-gyp@0.6.14
    ├─┬ mkdirp@0.5.1
    │ └── minimist@0.0.8
    ├─┬ nopt@3.0.4
    │ └── abbrev@1.0.7
    ├─┬ npmlog@1.2.1
    │ ├── ansi@0.3.0
    │ ├─┬ are-we-there-yet@1.0.4
    │ │ ├── delegates@0.1.0
    │ │ └─┬ readable-stream@1.1.13
    │ │   ├── core-util-is@1.0.1
    │ │   ├── inherits@2.0.1
    │ │   ├── isarray@0.0.1
    │ │   └── string_decoder@0.10.31
    │ └─┬ gauge@1.2.2
    │   ├── has-unicode@1.0.0
    │   ├─┬ lodash.pad@3.1.1
    │   │ ├── lodash._basetostring@3.0.1
    │   │ └─┬ lodash._createpadding@3.6.1
    │   │   └── lodash.repeat@3.0.1
    │   ├─┬ lodash.padleft@3.1.1
    │   │ ├── lodash._basetostring@3.0.1
    │   │ └─┬ lodash._createpadding@3.6.1
    │   │   └── lodash.repeat@3.0.1
    │   └─┬ lodash.padright@3.1.1
    │     ├── lodash._basetostring@3.0.1
    │     └─┬ lodash._createpadding@3.6.1
    │       └── lodash.repeat@3.0.1
    ├─┬ rc@1.1.2
    │ ├── deep-extend@0.2.11
    │ ├── ini@1.3.4
    │ ├── minimist@1.2.0
    │ └── strip-json-comments@0.1.3
```

```
├─┬ request@2.64.0
│ ├── aws-sign2@0.5.0
│ ├─┬ bl@1.0.0
│ │ └─┬ readable-stream@2.0.2
│ │   ├── core-util-is@1.0.1
│ │   ├── inherits@2.0.1
│ │   ├── isarray@0.0.1
│ │   ├── process-nextick-args@1.0.3
│ │   ├── string_decoder@0.10.31
│ │   └── util-deprecate@1.0.1
│ ├── caseless@0.11.0
│ ├─┬ combined-stream@1.0.5
│ │ └── delayed-stream@1.0.0
│ ├── extend@3.0.0
│ ├── forever-agent@0.6.1
│ ├─┬ form-data@1.0.0-rc3
│ │ └── async@1.4.2
│ ├─┬ har-validator@1.8.0
│ │ ├── bluebird@2.10.2
│ │ ├─┬ chalk@1.1.1
│ │ │ ├── ansi-styles@2.1.0
│ │ │ ├── escape-string-regexp@1.0.3
│ │ │ ├─┬ has-ansi@2.0.0
│ │ │ │ └── ansi-regex@2.0.0
│ │ │ ├─┬ strip-ansi@3.0.0
│ │ │ │ └── ansi-regex@2.0.0
│ │ │ └── supports-color@2.0.0
│ │ ├─┬ commander@2.8.1
│ │ │ └── graceful-readlink@1.0.1
│ │ └─┬ is-my-json-valid@2.12.2
│ │   ├── generate-function@2.0.0
│ │   ├─┬ generate-object-property@1.2.0
│ │   │ └── is-property@1.0.2
│ │   ├── jsonpointer@2.0.0
│ │   └── xtend@4.0.0
│ ├─┬ hawk@3.1.0
│ │ ├── boom@2.9.0
│ │ ├── cryptiles@2.0.5
│ │ ├── hoek@2.16.3
│ │ └── sntp@1.0.9
```

```
│ ├─┬ http-signature@0.11.0
│ │ ├── asn1@0.1.11
│ │ ├── assert-plus@0.1.5
│ │ └── ctype@0.5.3
│ ├── isstream@0.1.2
│ ├── json-stringify-safe@5.0.1
│ ├─┬ mime-types@2.1.7
│ │ └── mime-db@1.19.0
│ ├── node-uuid@1.4.3
│ ├── oauth-sign@0.8.0
│ ├── qs@5.1.0
│ ├── stringstream@0.0.4
│ ├── tough-cookie@2.1.0
│ └── tunnel-agent@0.4.1
├─┬ rimraf@2.4.3
│ └─┬ glob@5.0.15
│   ├─┬ inflight@1.0.4
│   │ └── wrappy@1.0.1
│   ├── inherits@2.0.1
│   ├─┬ minimatch@3.0.0
│   │ └─┬ brace-expansion@1.1.1
│   │   ├── balanced-match@0.2.0
│   │   └── concat-map@0.0.1
│   ├─┬ once@1.3.2
│   │ └── wrappy@1.0.1
│   └── path-is-absolute@1.0.0
├── semver@5.0.3
├─┬ tar@2.2.1
│ ├── block-stream@0.0.8
│ ├─┬ fstream@1.0.8
│ │ └── graceful-fs@4.1.2
│ └── inherits@2.0.1
└─┬ tar-pack@2.0.0
  ├── debug@0.7.4
  ├─┬ fstream@0.1.31
  │ ├── graceful-fs@3.0.8
  │ └── inherits@2.0.1
  ├─┬ fstream-ignore@0.0.7
  │ ├── inherits@2.0.1
  │ └─┬ minimatch@0.2.14
```

```
│   ├── lru-cache@2.7.0
│   └── sigmund@1.0.1
├── graceful-fs@1.2.3
├── once@1.1.1
├─┬ readable-stream@1.0.33
│ ├── core-util-is@1.0.1
│ ├── inherits@2.0.1
│ ├── isarray@0.0.1
│ └── string_decoder@0.10.31
├── rimraf@2.2.8
├─┬ tar@0.1.20
│ ├── block-stream@0.0.8
│ └── inherits@2.0.1
└── uid-number@0.0.3
```

APPENDIX E

Package.json for database learning system

```
{
  "name": "dbls",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "JSON": "^1.0.0",
    "bcrypt-nodejs": "0.0.3",
    "body-parser": "~1.13.2",
    "compression": "^1.6.1",
    "connect-flash": "^0.1.1",
    "cookie-parser": "~1.3.5",
    "cookie-session": "^1.2.0",
    "csurf": "^1.8.3",
    "debug": "~2.2.0",
    "ejs": "~2.3.3",
    "express": "~4.13.1",
    "express-fileupload": "0.0.4",
    "express-session": "^1.13.0",
    "fs": "0.0.2",
    "helmet": "^1.1.0",
    "mongoose": "^4.4.3",
    "morgan": "~1.6.1",
    "multer": "^1.1.0",
    "passport": "^0.3.2",
    "passport-gitlab": "^0.5.0",
    "passport-google": "^0.3.0",
    "passport-google-oauth": "^1.0.0",
    "passport-local": "^1.0.0",
    "passport-saml": "^0.15.0",
```

```
    "passport-twitter": "^1.0.4",
    "pmx": "^0.6.1",
    "serve-favicon": "~2.3.0",
    "sqlite3": "^3.1.1"
  }
}
```

APPENDIX F

Bower.json for database learning system

```
{
  "name": "dbls",
  "description": "Database Learning System",
  "main": "",
  "license": "MIT",
  "moduleType": [],
  "homepage": "https://dbls.ovh",
  "private": true,
  "ignore": [
    "**/.*",
    "node_modules",
    "bower_components",
    "test",
    "tests"
  ],
  "dependencies": {
    "Materialize": "materialize#^0.97.5",
    "font-awesome": "fontawesome#^4.5.0",
    "highlightjs": "^9.1.0",
    "sweetalert": "^1.1.3",
    "respond": "^1.4.2",
    "html5shiv": "^3.7.3"
  }
}
```

APPENDIX G

MongoDB User Schema used by Passport

```
var userSchema = mongoose.Schema({
    local: {
        email: String,
        password: String,
    },
    twitter: {
        id: String,
        token: String,
        displayName: String,
        username: String
    },
    google: {
        id: String,
        token: String,
        email: String,
        name: String
    },
    GitLab: {
        id: String,
        displayName: String,
        token: String,
        email: String
    },
    exercise: {
        id: String,
        mark: Number
    },
    isAdmin: Boolean
});
```

APPENDIX H

SSL Report: dbls.ovh (212.47.229.180)

Assessed on: Fri, 25 Mar 2016 23:16:29 UTC

Summary

Overall Rating: A+

Server Key and Certificate #1

Subject	dbls.ovh
Fingerprint SHA1:	0d3e5b6a929508d2fd5ce18b1c810543cf35b084
Pin SHA256:	SbnUj4UfvvLgzQTpJUws3ud3lqiUG4RPPIQP/f01V48=
Common names:	dbls.ovh
Valid from	Fri, 18 Mar 2016 23:58:00 UTC
Valid until	Thu, 16 Jun 2016 23:58:00 UTC (expires in 2 months and 22 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	Let's Encrypt Authority X1
AIA:	http://cert.int-x1.letsencrypt.org/
Revocation status	Good (not revoked)
Trusted	Yes

Certification Paths

Path #1: Trusted

1 Sent by server dbls.ovh

Fingerprint SHA1: 0d3e5b6a929508d2fd5ce18b1c810543cf35b084

Pin SHA256: SbnUj4UfvvLgzQTpJUws3ud3lqiUG4RPPIQP/f01V48=

RSA 2048 bits (e 65537) / SHA256withRSA

2 Sent by server Let's Encrypt Authority X1

Fingerprint SHA1: 3eae91937ec85d74483ff4b77b07b43e2af36bf4

Pin SHA256: YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=

RSA 2048 bits (e 65537) / SHA256withRSA

3 In trust store DST Root CA X3 Self-signed

Fingerprint SHA1: dac9024f54d8f6df94935fb1732638ca6ad77c13

Pin SHA256: Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys=

RSA 2048 bits (e 65537) / SHA1withRSA

Protocols

TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

Protocol Details

DROWN (experimental)	No, server keys and hostname not seen elsewhere with SSLv2
Secure Renegotiation	Supported
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
POODLE (SSLv3)	No, SSL 3 not supported (more info)
POODLE (TLS)	No (more info)
Downgrade attack prevention	Yes, TLS_FALLBACK_SCSV supported (more info)
SSL/TLS compression	No
RC4	No
Heartbeat (extension)	Yes
Heartbleed (vulnerability)	No (more info)
OpenSSL CCS vuln. (CVE-2014-0224)	No (more info)
Forward Secrecy	Yes (with most browsers) ROBUST (more info)
ALPN	No
NPN	Yes h2 http/1.1
Session resumption (caching)	Yes

Session resumption (tickets)	Yes
OCSP stapling	Yes
Strict Transport Security (HSTS)	Yes
HSTS Preloading	Not in: Chrome, Edge, Firefox, IE, or Tor
Public Key Pinning (HPKP)	No
Public Key Pinning Report-Only	No
Long handshake intolerance	No
TLS extension intolerance	No
TLS version intolerance	No
Incorrect SNI alerts	No
Uses common DH primes	No
DH public server param (Ys) reuse	No
SSL 2 handshake compatibility	Yes

Miscellaneous

Test date	Fri, 25 Mar 2016 23:14:30 UTC
Test duration	118.873 seconds
HTTP status code	200
HTTP server signature	nginx/1.9.12
Server hostname	dbls.ovh

APPENDIX I

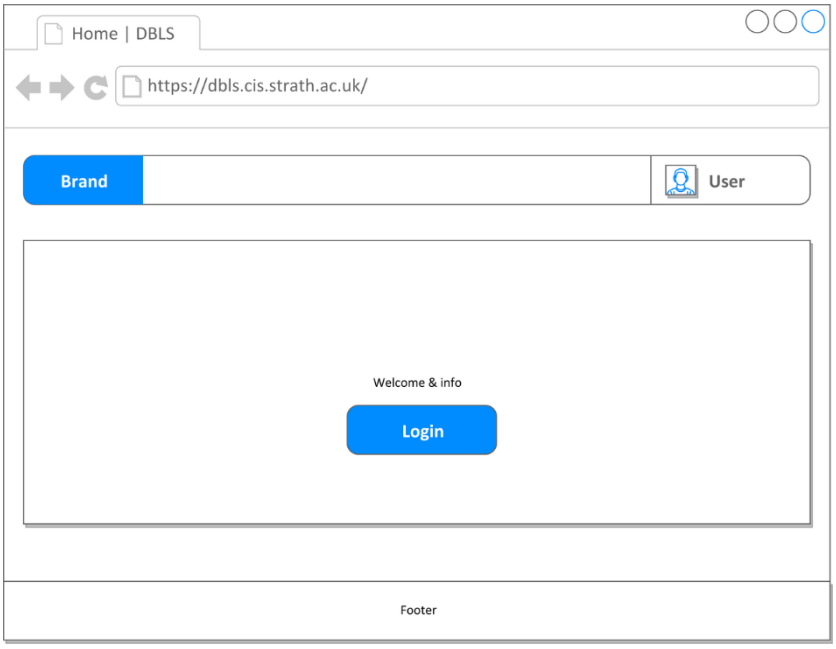

Figure 5 2nd iteration design for user landing page (wireframe)

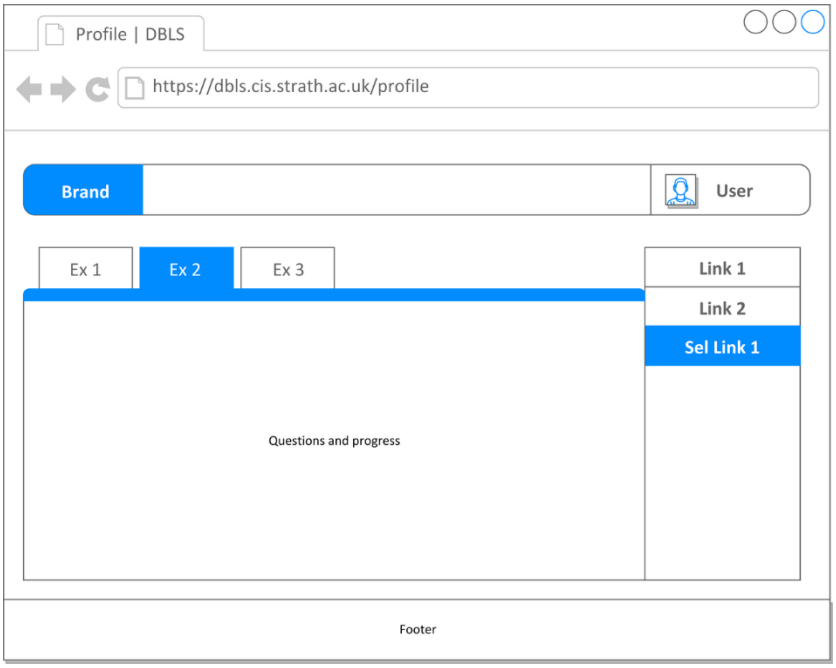

Figure 6 2nd iteration design for user profile (wireframe)

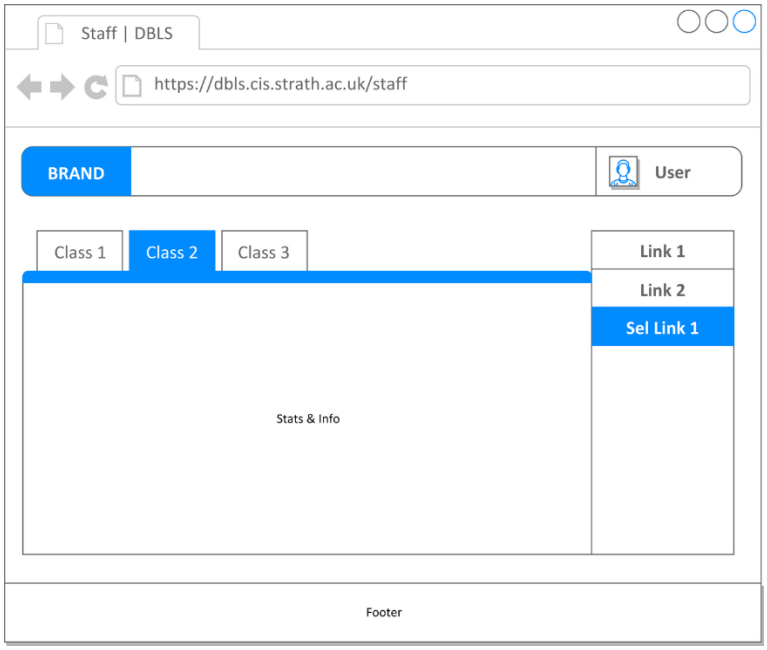

Figure 7 2nd iteration design for staff page (wireframe)

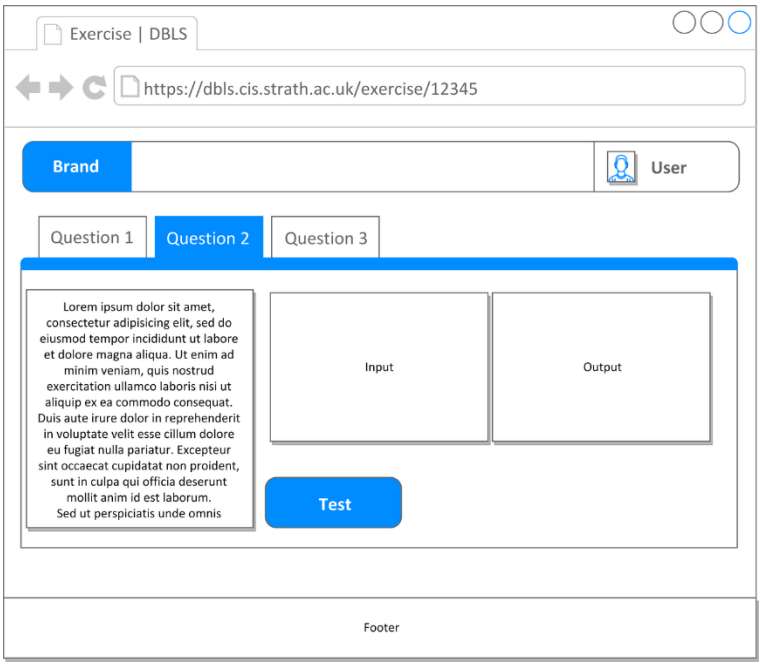

Figure 8 2nd iteration design for exercise page

APPENDIX J

DBLS Profile page colour blindness tests.

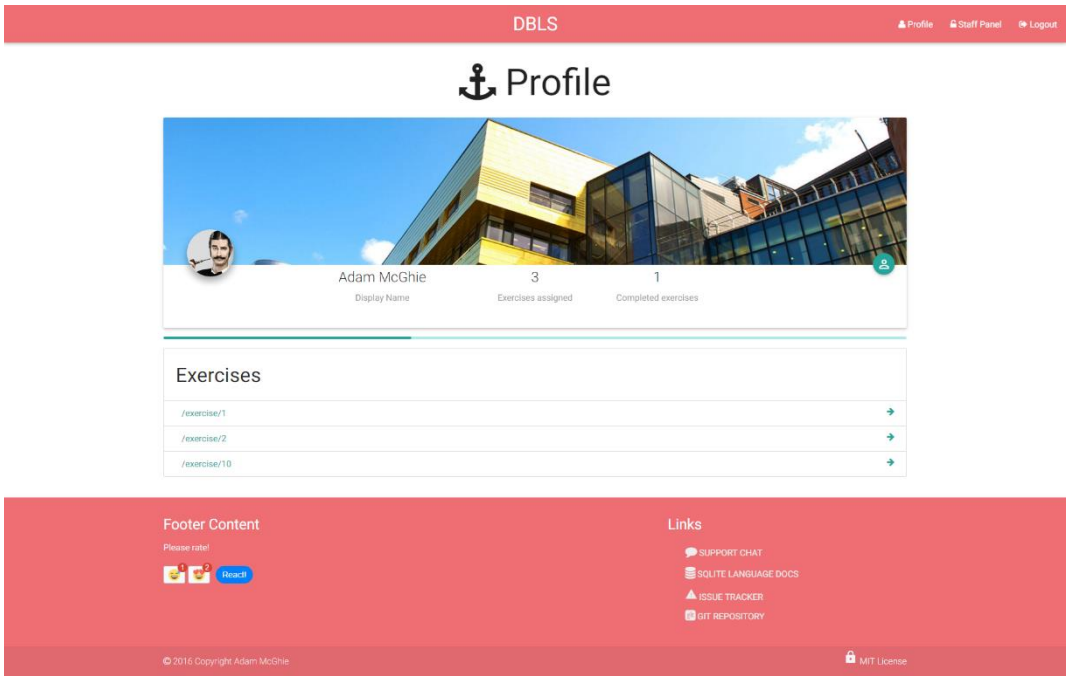

Figure 9 DBLS Profile page (Normal)

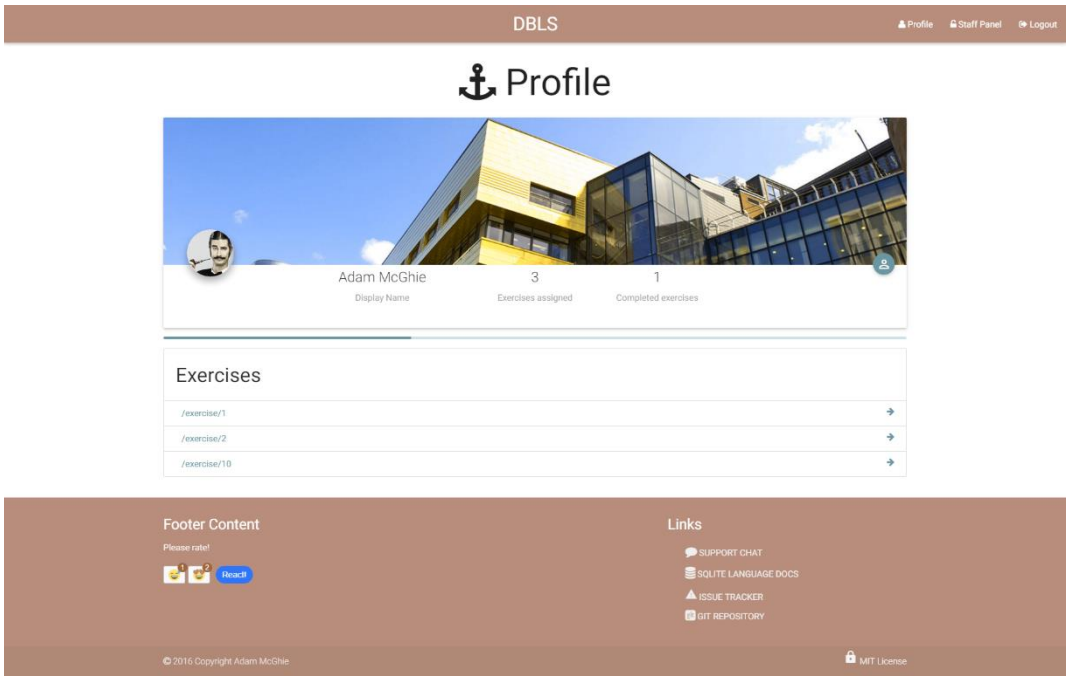

Figure 10 DBLS Profile page (Deuteranomaly)

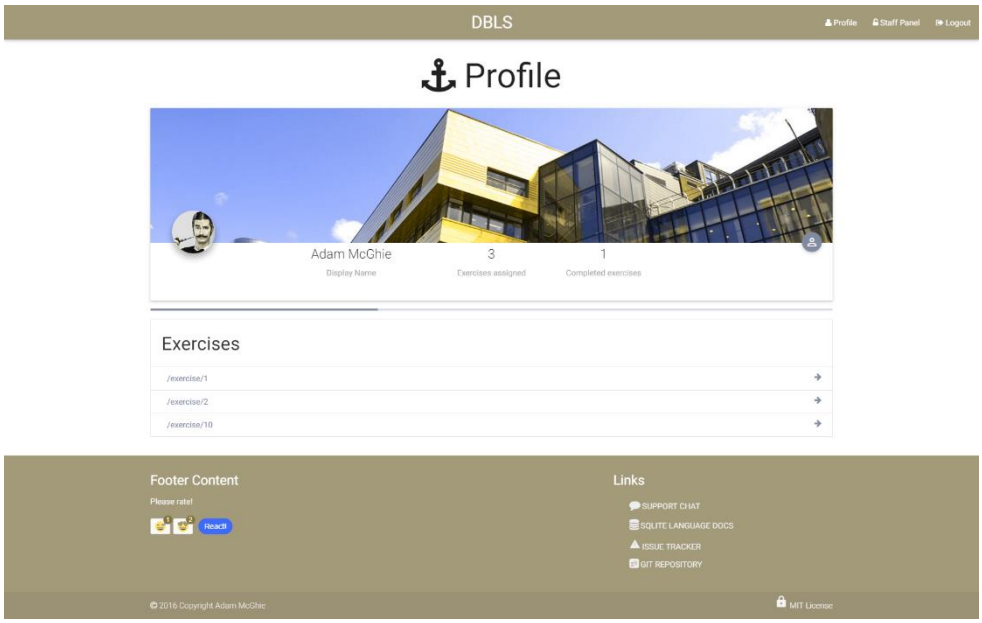

Figure 11 DBLS Profile page (Deuteranopia)

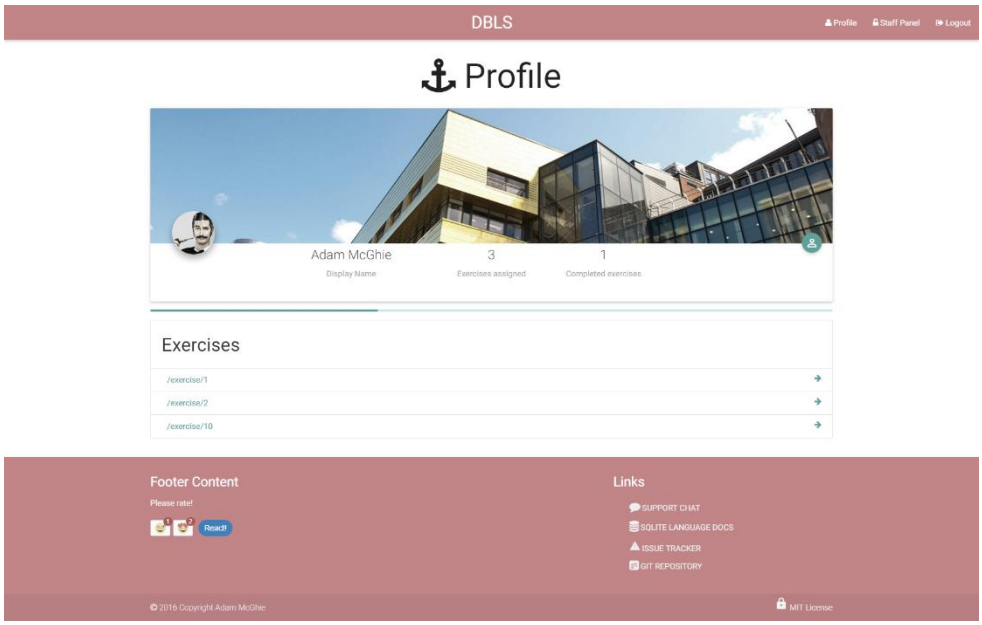

Figure 12 DBLS Profile page (Achromatically)

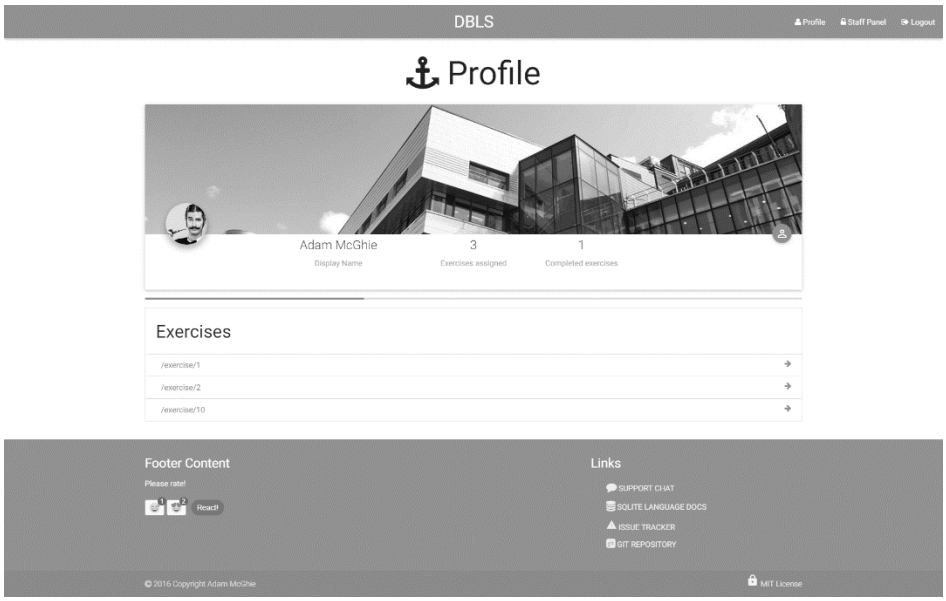

Figure 13 DBLS Profile page (Achromatopsia)

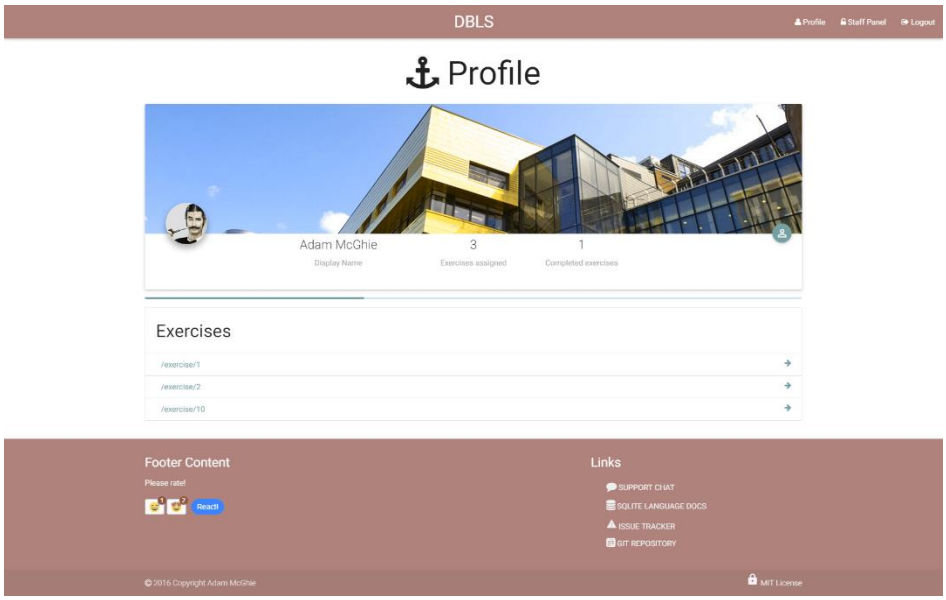

Figure 14 DBLS Profile page (Protanomaly)

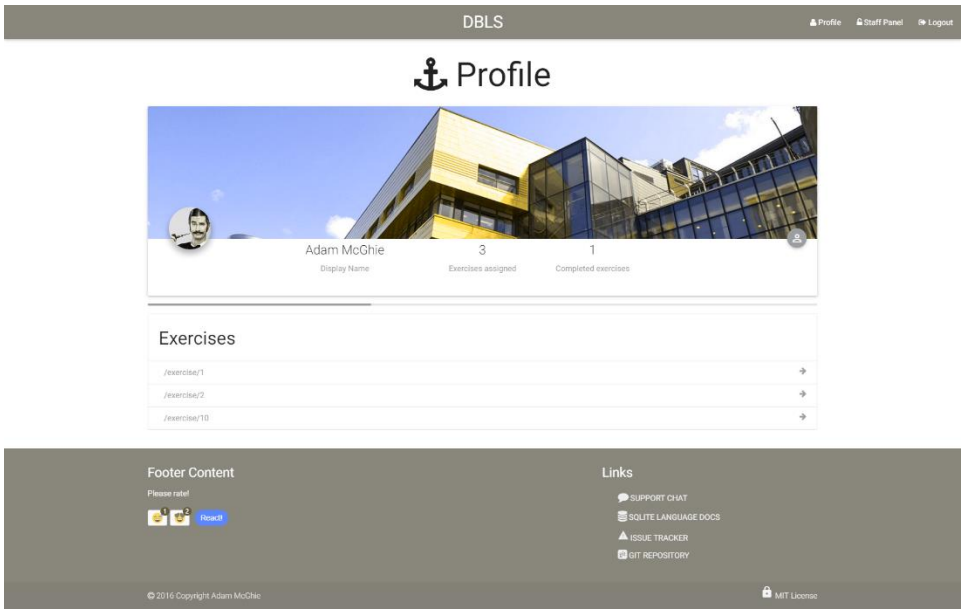

Figure 15 DBLS Profile page (Protanopia)

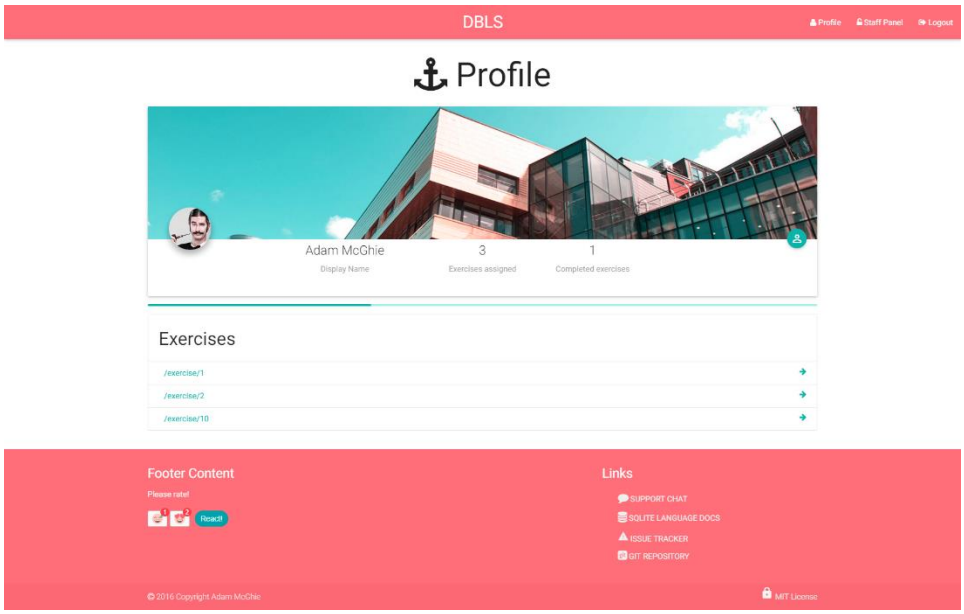

Figure 16 DBLS Profile page (Tritanopia)

APPENDIX K

How likely is it that you would recommend this software compared to the current class?	How satisfied are you with the reliability of this software?	How satisfied are you with the security of this software?	How satisfied are you with this software's ease of use?	How satisfied are you with the look and feel of this software?	How satisfied are you with the account setup experience of this software?	How satisfied are you with the value for the money of this software?	Any additional thoughts?
7	Somewhat satisfied	Somewhat satisfied	Not so satisfied	Somewhat satisfied	Very satisfied	Extremely satisfied	UI could be better
8	Somewhat satisfied		Not so satisfied	Very satisfied	Somewhat satisfied	Extremely satisfied	HCI User flow could be better
6	Extremely satisfied	Somewhat satisfied	Very satisfied	Very satisfied	Extremely satisfied	Somewhat satisfied	I can't SQL maybe teach HQL instead?
7	Somewhat satisfied	Not at all satisfied	Not so satisfied	Very satisfied	Somewhat satisfied	Somewhat satisfied	allow proceeding from one task to the next one without having to go back to the profile
7	Not so satisfied	Not so satisfied	Somewhat satisfied	Very satisfied	Very satisfied	Very satisfied	If the bugs were fixed I would be very satisfied
8	Extremely satisfied	Somewhat satisfied	Very satisfied	Very satisfied	Extremely satisfied	Extremely satisfied	helpers when stuck
7	Very satisfied	Very satisfied	Very satisfied	Very satisfied	Somewhat satisfied	Not at all satisfied	Next button would be nice
10	Extremely satisfied	Very satisfied	Extremely satisfied	Extremely satisfied	Extremely satisfied	Extremely satisfied	Next button would be nice

APPENDIX L

During the development several basic tests were done to verify that updates did not break functionality in the application.

Signing up
Description:

- Test to ensure that new accounts can be created

Pre-condition

- User is not logged in
- User is on the application index page
- User chooses to sign up using the local method

Test Steps

1. Click “Local Signup”
2. Enter an email address and password
3. Click submit
4. If valid wait to be redirected to user profile page

Result

- New user account is created in the MongoDB
- User logged into the new account

Student Exercise Submission
Description:

- Test to ensure that users can submit answers to exercises

Pre-condition

- User is logged in
- User has not previously attempted the question
- User is currently viewing their profile page

Test Steps

1. Click unattempted exercise
2. Enter wrong answer and submit
3. Verify that system responds with wrong answer text
4. Enter correct answer and submit
5. Verify that system responds with correct answer banner

Result

- After Step 2 page banner should display “answer not correct”
- After Step 4 page banner should display “answer correct”

Add Exercise
Description:

- Test to ensure that staff can add new exercises

Pre-condition

- User is logged in
- User has permission “isAdmin” set to true
- User is viewing the staff overview page

Test Steps

1. Click “Add Exercise” button
2. Enter test values for text fields on the page
3. Click “file” button on page
4. Select pre-built example SQLite database
5. Click Submit
6. Wait for “Success” popup to appear
7. Navigate to Staff overview page
8. Verify new exercise has appeared

Result

- New exercise appears in exercise list

Account Linking
Description:

- Verify that linking external login methods to an account

Pre-condition

- User is logged in
- User account currently only has a local account

Test Steps

1. Open advanced user profile by clicking on profile header image
2. Click “Link” under the Twitter header
3. Follow instructions from Twitter authorization page until redirected back
4. Redo step 1
5. Click “Link” under Google header
6. Follow instructions from Google authorization page until redirected back
7. Logout
8. Login using Twitter method from index page
9. Verify account ID is the same from the advanced user profile panel
10. Logout
11. Login using Google method
12. Verify account ID is the same from the advanced user profile panel

Result

- In both login steps user account ID remains the same

All Exercise Completion
Description:

- Answer all exercises to verify information displayed to the user

Pre-condition

- User is logged in
- User has not completed all exercise

Test Steps

1. Click unattempted exercise
2. Successfully complete exercise
3. Repeat steps 1 & 2 until all exercises are completed

Result

- Verify that the “Exercises Assigned” and “Completed Exercises” counters both match
- Verify Progress bar is full
- Verify that Exercise list shows all exercises as completed

APPENDIX M

Installation and Setup guide

QUICK START

This guide will help you quickly get a copy of the DBLS up and running

Requirements

- Must run nodeJS v4.7 or later
- Have greater than 512MB ram

Install

1. Install nodeJS if you haven't already
2. Install MongoDB and ensure it's running
3. Enter the following into a a terminal in the folder you have the DBLS code
    a. `bash build.sh`
4. Wait for it to get all its required libraries
    a. if this is a first time setup you may be asked for your sudo password to install some features
5. Modify the configuration files found in `./config/`
6. run `npm start` to start the application on port 3000

PRODUCTION INSTALLATION

This will detail how to run a clustered instance of the Database Learning System to scale

Requirements

- Must run nodeJS v4.7 or later
- Have greater than 1024MB of RAM
- Minimum 2 CPU cores

Install

1. Install nodeJS if you haven't already
2. Install MongoDB and ensure it's running
3. Enter the following into a a terminal in the folder you have the DBLS code
    a. `bash build.sh`
4. Wait for it to get all its required libraries
    a. if this is a first time setup you may be asked for your sudo password to install some features
5. Modify the configuration files found in `./config/`
6. Install PM2 with the following command `sudo npm -g install pm2`

Setup

1. Modify the configuration files found in `./config/`
2. Start DBLS in cluster mode with the following command
    a. `pm2 start bin/www -i 0 –name "DBLS"`

3. The DBLS will now start as in a cluster with up to the number of instances started being the number of CPU cores available.

Note: It is advised that DBLS be run behind an NGINX reverse proxy. An example configuration for this can be found in Appendix A.