ASSEMBLY CODE VISUALISER

A-Level Computer Science NEA

Centre No. 74615 | Centre Name: Dubai College | Candidate Name: Raghav Awasthi | Candidate No: 4010 | Component Code: 7517/C

Dubai College
Raghav Awasthi

Chapter 1 Table of Contents

Chapter 2 Colour and Style Scheme *9*

Chapter 3 Analysis *10*

Background **10**

Problem Overview **10**

Client **10**

Interview 1 with Client **11**
- Conversation 11
- Summary 13
- Initial Client Requirements 14

Existing Solutions **15**
- Peter Higginson's Little Man Computer 15
  - Pros 17
  - Cons 18
  - Summary 19
- Paul Hankin's Little Man Computer 20
  - Pros 22
  - Cons 22
  - Summary 24

Interview with Potential User 1 ..... **24**
- Conversation 24
- Summary 27
- Additional Requirements 27
- Updated Overall Requirements 28

Document Analysis **30**
- GCSE Specification 30
- A-Level Specification 32
- Summary 35
- Additional Requirements 35
- Updated Overall Requirements 36

Interview with Potential User 2 ..... **37**
- Conversation 37
- Summary 39
- Additional Requirements 40
- Updated Overall Requirements 40

Interview 2 with Client **41**
- Conversation 41
- Summary 44
- Additional Requirements 44
- Updated Overall Requirements 44
- Basic Idea and Understanding 46
- Analysis of Existing System 46

Prototypes 48
Entity-Relationship Diagram 49

Scope 51
Teacher-Created Challenges 51
Overview 51
Discussion and Conclusion 51
Student-Created Challenges 51
Overview 51
Discussion and Conclusion 52
"Publishing" Students' Solution 52
Overview 52
Discussion and Conclusion 52
Creating a Translator 52
Overview 52
Discussion and Conclusion 52
Summary 53

Exploring Technologies 54
Frontend 54
Overview 54
General Necessary Technologies 54
Additional Technologies 54
Backend 54
PHP 54
Python – Flask 55
Python – Django 55
NodeJS 56
Database 57
MongoDB vs MySQL 57
Translator 58

Interview 3 with Client 59
Requirements So Far 59
Summary 61

Final Requirements 63

Modelling 65

Chapter 4 Documented Design 67

Overview 67

Technologies 67
Link to Requirements 68
Express JS 68
Middleware 68
Template Engine 69
Overview 69
Pug vs EJS 69
Express Explanation 70
Routing 70

File Structure 72
Http Errors 73
Path 73
Cookie Parser 73
Morgan 74
Express Flash 74
Express-Session 74
Redis (Redis Store, Connect Redis, and Redis Client) 75
Dotenv 76
Passport (and Passport Local) 76
MySQL 77
Bcrypt 77
Lodash 78
Moment 78
Chosen 79
Leader Line 79

User Experience 82

Landing Page 82

Navigation Menu 82
No Account 82
Student Account 83
Teacher Account 84
Prototype 84

Little Man Computer Visualisation 88
Link to Requirements 88
Basic Template for Layout 90
Prototyped Code 91
Main Visualisation Layout 91
Div Highlighting Example 96
Instruction Set Page 96
How To Use This Resource Page 99

Assembly Code and Editor 102
Link to Requirements 102
Editor Layout and Translator 103
Instruction Set Prototypes 103
HLT 103
INP 103
OUT 104
ADD 105
SUB 106
LDA 107
STA 108
BRA 109
BRZ 110
BRP 111
Uploading Text File and Converting to Assembly Code 112
NodeJS, Express, and File Upload Template 112
Single File Upload Prototyped Code 113
Pre-Created Assembly Codes 114

Little Man Computer **115**
UML Diagram 116
Scanning the Assembly Code Text Editor and Loading into RAM 116
Running Instructions 121
Dealing with Errors 123

Login and Sign-up **124**
Link to Requirements 124
Recalling requirement 10: 124
Obviously, it is very important that the actual sign-up and login processes are designed, so these will be done first, followed by the challenges part of this requirement. 124
Flowcharts 125
Sign-up 125
Login 125
Login and Sign-up Combined 126
Entity-Relationship Diagram 128
Prototypes 129

Cookies and Sessions **133**
Persistent Cookies 133
First-Party Cookies 133
Issuing Session Tokens 133
Authenticating Users via Cookies 135
Refreshing Session Tokens 137
Implementing User Logout 138

Challenges **140**
Link to Requirements 140
Final Database 140
Teacher Creating and Assigning Challenges 141
Creating a New Challenge 141
Editing a Challenge 143
Deleting a Challenge 143
Assigning a Challenge 143
Student Viewing and Completing Assigned Challenges 145
Teacher Viewing Students' Assigned Challenges 147

Chapter 5 Technical Solution ***149***

Technical Skills **149**
Technical A Skills 149
Technical B Skills 149
Technical C Skills 150

Coding Practices/Styles **150**
Coding Styles 150
My Coding Practices 150

Working File Directory **152**

Database Table Organisation **153**

NodeJS and EJS Default Files **153**
~/node_modules/ 153

~/views/ 154
~/views/error.ejs 154
~/views/index.ejs 154
~/views/partials/ 155
~/routes/ 155
~/routes/index.js 155
~/app.js 158
~/lib/ 160
~/bin/ 160
~/bin/www 160
~/.env 162
~/public/ 163
~/public/stylesheets/ 163
~/public/plugins/ 163

Configurations **164**
~/lib/db.js 164
~/lib/redis.js 165

Main Visualisation / Home Page **166**
~/public/javascripts/lmc.js 167
~/views/partials/main_visualisation.ejs 204
~/views/partials/lmc_script.ejs 210

Instruction Set and How to Use the Resource **211**

Login and Sign-Up **212**
~/views/partials/menu.ejs 212
~/public/javascripts/menu.js 215
~/public/images/ 215
Sign-up 215
~/views/login_sign_up/sign_up.ejs 215
~/routes/login_sign_up/sign_up.js 220
~/public/javascripts/form_toggler.js 226
~/views/login_sign_up/partials/script.ejs 227
Login 227
~/views/login_sign_up/login.ejs 227
~/routes/login_sign_up/login.js 231
~/lib/auth.js 232
~/lib/passport-config.js 233

Challenges **238**
Creating, Editing, and Assigning Challenges as a Teacher 238
~/views/teacher_challenges/create_challenges.ejs 238
~/public/javascripts/active_challenges_display_toggler.js 243
~/views/teacher_challenges/new_challenge.ejs 245
~/views/teacher_challenges/edit_challenge.ejs 247
~/routes/teacher_challenges/create_challenges.ejs 249
Viewing and Submitting Assigning Challenges as a Student 265
~/views/student_challenges/view_set_challenges.ejs 265
~/public/javascripts/student_toggles.js 270
~/views/student_challenges/complete_challenge.ejs 271

Viewing Assigning Challenges' Solutions as a Teacher 282
~/views/teacher_challenges/view_student_challenges.ejs 282
~/views/teacher_challenges/more_information.ejs 285
~/views/teacher_challenges/view_answer.ejs 290
~/routes/teacher_challenges/view_student_challenges.js 291

Chapter 6 Testing ***300***

Overview **300**

Test Plan/Table **300**

User Testing **339**
Overview 339
Rating System 339
Teacher User Tests 340
Teacher 1 340
Teacher 2 341
Summary 343
Student User Tests 343
Student 1 343
Student 2 344
Summary 346

Failed Tests and Refinements **347**
Test 17, 106, and 129 347
Test 23 and 112 348
Test 25, 96, 114, 119, and 137 348
Test 54 349
Test 87, 88, and 89 350
Test 24, 113, and 136 350
Test 22, 111, and 134 351
Test 148 351
Test 164 and 166 352
Test 172 352

Chapter 7 Evaluation ***354***

Achievement of Objectives **354**
Client Requirement 1 354
Client Requirement 2 355
Client Requirement 3 355
Client Requirement 4 356
Client Requirement 5 357
Client Requirement 6 358
Client Requirement 7 358
Client Requirement 8 359
Client Requirement 9 359
Client Requirement 10 360

Overall Effectiveness of Solution **363**

Client's (Mr. Wood's) Feedback Letter **365**

Future Improvements **367**
Syntax Highlighting 367

Check Format of .txt File Uploaded 367
Force Instructions Page on First Login 368
Colour Code Text in Verbose Output 368
Display Students' Challenges by Class 369
Forgot My Password 369
Confirmation Message 369
Button to Copy Assembly code and Redirect to Visualisation Page 370

Chapter 8 Appendix I: All Project Code 370

~/bin/www 370
~/lib/auth.js 372
~/lib/db.js 373
~/lib/passport-config.js 373
~/lib/redis.js 377
~/public/javascripts/active_challenges_display_toggler.js 378
~/public/javascripts/form_toggler.js 379
~/public/javascripts/lmc.js 380
~/public/javascripts/menu.js 416
~/public/javascripts/student_toggles.js 416
~/public/stylesheets/main_visualisation_style.css 417
~/public/stylesheets/style.css 422
~/routes/login_sign_up/login.js 426
~/routes/login_sign_up/sign_up.js 427
~/routes/student_challenges/view_set_challenges.js 433
~/routes/teacher_challenges/create_challenges.js 441
~/routes/teacher_challenges/view_student_challenges.js 456
~/routes/index.js 463
~/views/instructions/how_to_use.ejs 465
~/views/instructions/instruction_set.ejs 467
~/views/login_sign_up/partials/script.ejs 470
~/views/login_sign_up/login.ejs 470
~/views/login_sign_up/sign_up.ejs 474
~/views/partials/head.ejs 479
~/views/partials/lmc_script.ejs 479
~/views/partials/main_visualisation_head.ejs 479
~/views/partials/main_visualisation.ejs 479

~/views/partials/menu.ejs **485**
~/views/partials/script.ejs **487**
~/views/student_challenges/partials/script_toggle.ejs **487**
~/views/student_challenges/complete_challenges.ejs **487**
~/views/student_challenges/view_completed_challenge.ejs **489**
~/views/student_challenges/view_set_challenges.ejs **490**
~/views/teacher_challenges/partials/head.ejs **494**
~/views/teacher_challenges/script_toggle.ejs **494**
~/views/teacher_challenges/script.ejs **494**
~/views/teacher_challenges/style.ejs **497**
~/views/teacher_challenges/create_challenges.ejs **497**
~/views/teacher_challenges/edit_challenge.ejs **502**
~/views/teacher_challenges/more_information.ejs **504**
~/views/teacher_challenges/new_challenge.ejs **509**
~/views/teacher_challenges/view_answer.ejs **510**
~/views/teacher_challenges/view_student_challenges.ejs **511**
~/views/error.ejs **514**
~/views/index.ejs **515**
~/.env **515**
~/app.js **516**
Chapter 9 Bibliography ***518***

Chapter 2 Colour and Style Scheme

For this entire document, I will be using the following colour scheme and coding system:

- ⇒ This colour will be used for regular text.
- ⇒ This colour will be used for figures, tables and any other references that are made (these can be clicked on to jump to the place being referenced in the document, if being viewed digitally).
- ⇒ *Any italicised text during interviews/conversations will be used for comments and/or additional information about the interview/conversation.*
- ⇒ This highlight will be used to signify that the proposed idea is tentative and needs to be discussed in further detail, which is generally done in the Scope or Technologies section within the Analysis, or in the Documented Design section.
- ⇒ This colour will be used to highlight ideas that cannot be done in the Analysis section, as well as failed tests in the Testing section.
- ⇒ This colour will be used to highlight any tests that passed but had additional comments in the Testing section, as well as any comments that were added after the testing video was completed within that section.

Chapter 3 Analysis

Background

Dubai College, located in Dubai, United Arab Emirates, is one of the leading British curriculum schools overseas. The Computer Science department at the school is one of the newer ones; it started offering the AQA GCSE and A-Level Computer Science qualification in the autumn of 2016. Although it is a relatively new addition to the school, it has quickly become a very popular subject, and many students choose to take it. I am one of the nine students who has chosen to take Computer Science as one of their A-Levels from 2021 to 2023.

Problem Overview

I have been informed by my client (see below) that teaching the concept of Computer Architecture and how the Assembly Language works is a very challenging task, especially to students who are more visual in their learning. From an abstract level, learning about CPUs can be quite simple: it simply takes some input, processes it, and produces an output. However, as you delve deeper into the workings of the Fetch-Decode-Execute cycle, it can quickly become a very complex process. Teaching this to a varied range of students in a simple way so that they can understand what is happening within the computer is said to be challenging (see interview below).

There are very few online resources that allow people to write their own assembly code, while at the same time viewing what is happening inside the CPU – to allow people to see how the code they have written affects the CPU and thus, learn about the concepts of Computer Architecture. Furthermore, there are no resources or platforms that are available online for students to test their assembly coding skills and improve upon them, or even learn how to write in the assembly language.

Client

The Computer Science department is composed of four members of staff. I have been approached by Mr. Wood, the Head of Subject for Computer Science, about creating a platform to simplify and make the process of teaching Computer Architecture and how the Assembly Language works easier.

I have set up a meeting with Mr. Wood to discuss the initial ideas and the basic overview of the project, as well as clearly outline the preliminary requirements for it. The interview is documented below.

Interview 1 with Client

Conversation
Date: 08/02/2022
Attendees: Raghav Awasthi and Mark Wood (Client)
Abbreviations used for this interview: RA = Raghav Awasthi; MW = Mark Wood; LMC = Little Man Computer.

Greetings are not included.

RA: Would you mind expanding a bit more about the basic idea that we initially discussed?
MW: Yes. So, I'd like for there to be a teaching resource that teaches students about computer architecture and the assembly language because, currently, these two topics are quite hard to teach, especially to students who understand things better when they can visualise what is going on. I'd like this resource to be easily accessible to students, in order to help them learn about these topics, and improve their skills when it comes to writing code in assembly. Now, one thing that you may want to keep in mind is that these topics are on the Year 11 GCSE spec, as I'm sure you remember, having done the AQA Computer Science GCSE yourself, as well as the A-Level specification.
RA: Yeah.
MW: So, I think getting an idea of how you balance this as a teaching resource will be quite important.
RA: Right, that makes sense.

RA: About the teaching resource itself, is there a particular way that you would like it to be accessible, for example, as an application or a website? Also, are there any features or anything in particular that you think absolutely needs to be in there?
MW: Let me think... If you've seen the *LMC*, I do like that a lot. I think one of the things that we don't elaborate much on with the *LMC*, which even if we did, could be a little bit confusing, is the instruction set. So, when you are writing the instructions or interpreting the instructions on the *LMC*, you can click on the little button that says help, and it'll go to the Little Man Computer's instruction saying, something like: a 9 that is an input if it is followed by a 01, or it's an output it's followed by a 02 – I can't remember exactly what it is off the top of my head, but I think being able to look at that a little bit more and maybe making that a little bit more real might be a good idea. I don't know if that was your vision for this.
RA: Yeah, I could give that a little more thought.
MW: Yeah, that's something I'd look at a little bit more. What is the instruction set? How can I see the instruction set? Because that's the other thing as well, on the *LMC*. When you click help, it takes you to another web browser page where you have to read the instructions, then you have to go back and you go, oh look, what's this? It's a 9? Oh, what's a 9? Go back into the other page – oh, it's this. And yeah, I could pull them up, side-by-side, but why don't they just have that instruction set outlined on the interface so I can see what's what.

RA: Yeah, because, if memory serves, even the help page is quite verbose and has a lot of unnecessary text. So essentially, I need to simplify the process of looking up the instructions from the instruction set.
MW: Yes.

MW: Just in terms of the *LMC* and so we're on the same page, you know what I'm talking about, right?
RA: Yeah.
MW: So, you know, the hundred memory locations that are in there? And then you've got this simulated bus.
RA: Yeah.
MW: And then you've got the CPU; you've got the control unit; you've got the arithmetic logic unit; you've got the input; you've got the output. And then on the left-hand side, you've got some more things.
RA: Yeah, like the text field.
MW: Yeah. Oh, but you've also got the bit where you can write the assembly code.
RA: Right.
MW: Visually, what are your thoughts on that? You're a student yourself, so I'd just like to see, what are your thoughts on that?
RA: I was thinking that, obviously it's a good idea to keep the field for the basic CPU architecture, and, although there should be something like a field to write the code, perhaps that could include some sort of syntax highlighting somehow?
MW: Yeah, because it currently isn't like that. I think that'd be very cool, and I really like the sound of that.

RA: Again, as you said, the instruction set pop-up, and perhaps also showing the actual Fetch-Decode-Execute cycle?
MW: Oh, yes! I think that'd fit in really well. That's a great idea. Fundamentally, I just want a visualisation tool, and an application tool, in which I can apply what we're learning, that allows me to show more complexity than what the *LMC* does now. Now, that's what I want. I think reaching out to students that have used it, and been taught it, and saying to them, you know, how did you sort of contextualize these topics? What's the stuff that you would've wanted? That's not the research I've personally done. I think that I would like you to go and find out, more importantly, what the students want. How they've visualised it. What sorts of things would they have wanted? Like you said, I would never have thought of syntax highlighting for the assembly. Could never even thought of that, but you're a student who is doing that unit and you probably would have liked that.
RA: Right.
MW: I suggest you go and speak to other students, see what they want, then come back to me, and say, this is what the students want.

RA: What about the debugging when it comes to writing the assembly code?
MW: Yeah, that's a good point, because that's not the case. You know, again, I'll keep referring to the *LMC* because that's what we currently use – which, again, is a great resource. It really is, and it is very abstract. I think what we'd probably be looking for – again, what I said earlier about the GCSE and A-Level – is that we use the *LMC* for GCSE, and it's great.
RA: Yeah.
MW: But it does not cover enough detail for what's needed at A-Level, and even some smaller parts of the GCSE. And again, like you said, well, what if I want to write some slightly more complex assembly code? If you've written it wrong in the *LMC*, it's wrong. It just won't work. There's no debugging, and if you were going to introduce more complex features, and then it doesn't work? I think you would certainly need some sort of debugging to show things like where it's gone wrong and why it's gone wrong.

RA: Also, what do you think about including assembly language challenges to better students' assembly language programming skills?
MW: Definitely! That would be cool again! Another great, great idea. Because then it wouldn't just be a tool, whereby, for example, I could teach it in the lesson and people apply it in the lesson and then we leave it at that. It could mean taking it outside the lesson and having a go at these challenges. Hmm… I'm going to throw one in here… could be an idea. What about if they could self-mark the challenges and I can see them.
RA: Oh, that sounds really interesting.
MW: So, as a teacher, not necessarily set assignments…
RA: Right.
MW: But I can be a teacher with a class, and the class can complete challenges. And then I, as a teacher, can go and see that class, and what challenges they've completed.
RA: Right. But each student would only be able to see their own completed challenges – sort of like their own log.
MW: Yeah, that would be really cool. But again, making sure that, to use it, they obviously don't have to do that, but that would be a nice little feature.

Summary
The interview with Mr. Wood gave me quite a good idea about the basic overview and most important requirements that will be required for the teaching resource. We discussed things such as challenges to convert high-level code to assembly language, the feature to include students as part of a class, syntax highlighting, and being able to view the instruction set in an easier way, just to name a few.

From our conversation and the numerous points that we discussed; I have outlined the basic initial client requirements below.

Initial Client Requirements

1. Allow users to write code in assembly language
2. Allow users to view visualisations of assembly code they have written (CPU and RAM)
3. There should be automatic syntax highlighting for any code that is written
4. There should be verbose debugging for erroneous code
    a. This could mean creating a translator for converting assembly code written in the text field into a language which can display the errors appropriately
5. Should be able to view the instruction set on the same page
6. Show the Fetch-Decode-Execute cycle and what is currently being run
7. Create a list of coding challenges that can be solved by converting code from a High-Level Language into assembly code
8. Accounts
    a. Student account
        i. Able to view and complete challenges
        ii. Able to then self-mark challenges and submit
        iii. Able to view a log of their previously completed challenges
        iv. Able to view a log of previously saved code (including non-challenges)
    b. Teacher account
        i. Teachers have classes which are groups of students
        ii. Can view students' submitted challenges
        iii. Can view students' challenges even if not submitted
    c. No account
        i. Can still view visualisations and code in assembly
        ii. Able to view and complete and self-mark challenges, but don't have a log of previously completed ones
        iii. Cannot submit challenges to a teacher – no linked account
        iv. Cannot see a log of previously saved code

The interview also included a lot of discussion about Peter Higginson's Little Man Computer, the current tool that is used to teach students about Computer Architecture and the Assembly Language. While I also used this tool in order to learn about these topics, I decided to revisit the website and take a deeper look into its features, as well as its advantages and where features could be added or improved upon (see below).

Existing Solutions

Peter Higginson's Little Man Computer

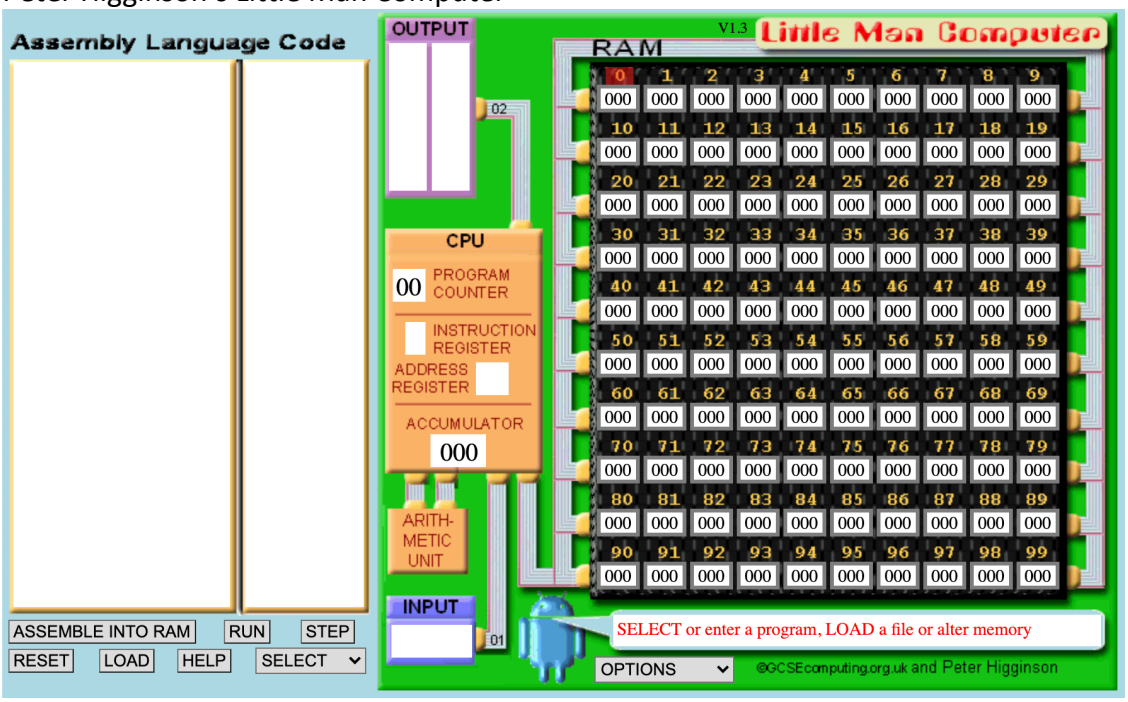

Figure 3.1 – basic Peter Higginson's LMC screen (Little Man Computer - CPU Simulator [Screenshot], 2014)

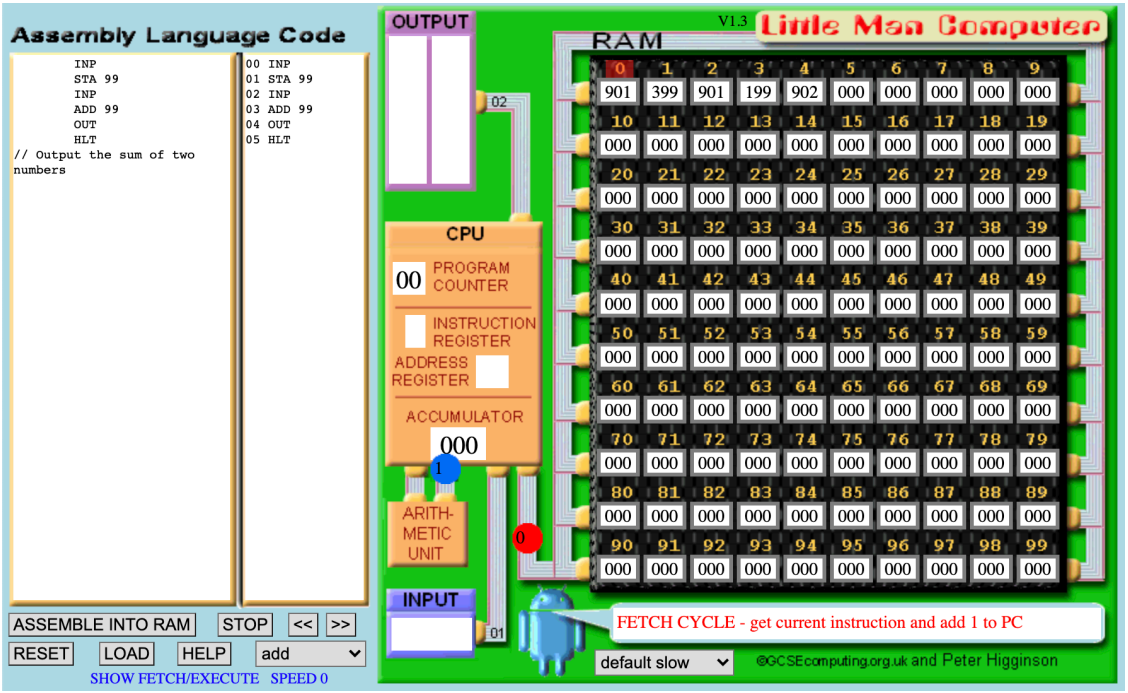

Figure 3.2 – code being run

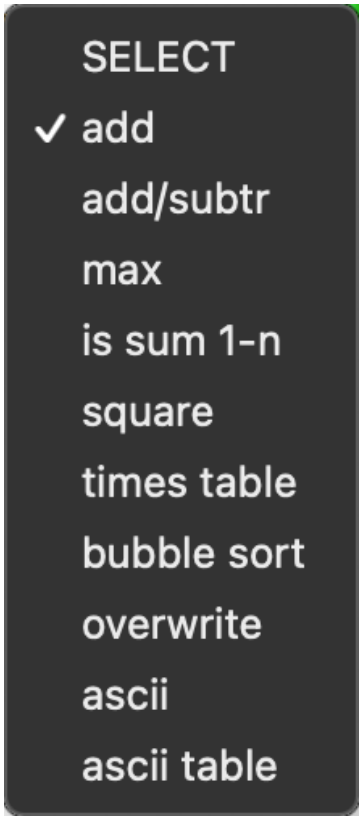

Figure 3.4 – examples of simple programs/algorithms

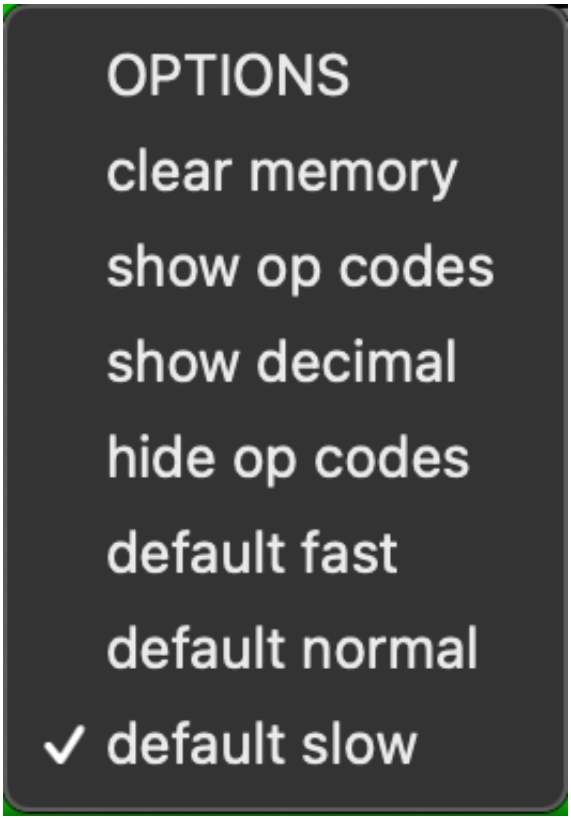

Figure 3.3 – options to view data

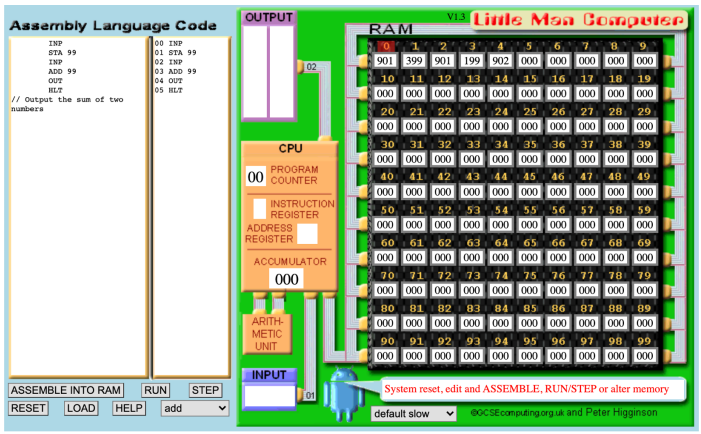

Figure 3.5 – entire webpage as shown on browser

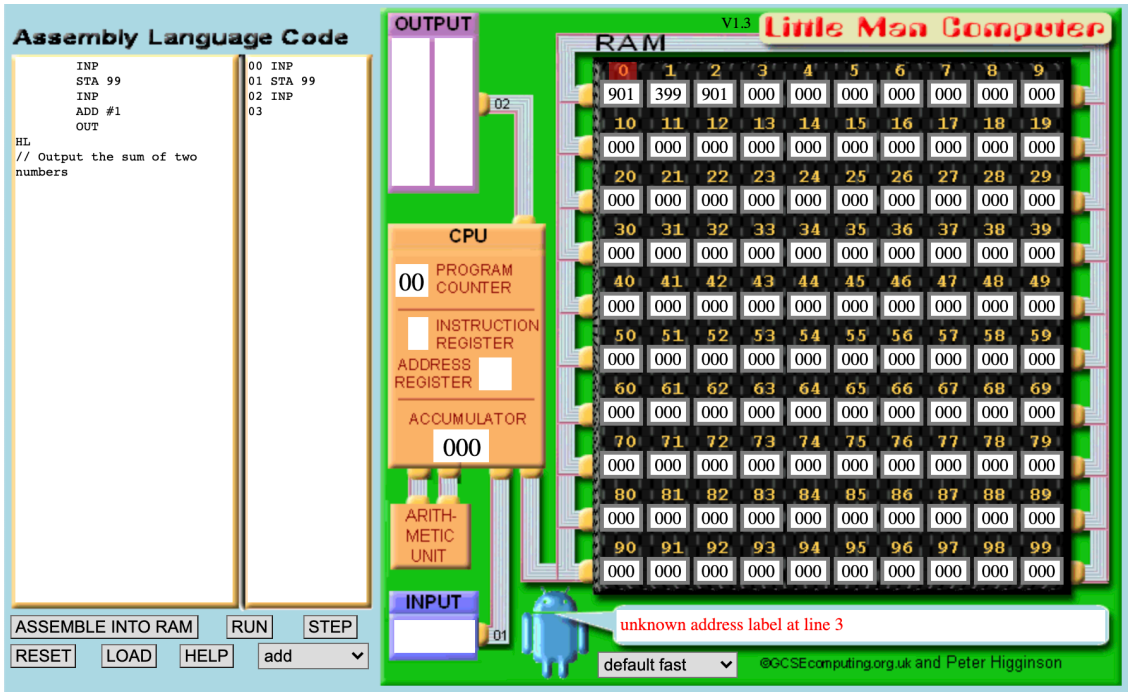

Figure 3.6 – small error display when erroneous code

Note that the LMC computer cannot tell whether memory locations contain instructions or data. Most modern computers can segment their memory but there has to be a way for the output from one program the become the instructions of another. DAT is a fake instruction to tell the assembler that this location contains data.

This implementation allows you to output both numbers (OUT) and characters (OTC). On a real computer you would only have the OTC instruction and the program would have to convert numbers to characters before printing. (Standard routines are provided.) The same is true for input but we have provided INP to input numbers for simplicity.

The assembler also allows us to give names (called labels) to addresses. Any word that is not a recognised instruction is assumed to be a label. If the label is before the instruction (or DAT) then it defines the label as representing that address; if it is after the instruction (or DAT) then the address is substituted for the label. The two boxes to the left of the LMC are the input to the assemble function and the intermediate output with the labels converted to the correct numbers.

You can also put comments into the program (with this implementation). Any line starting with // is ignored by the assembler. (I have put mine at the end so the intermediate output lines up with the input but you can put them anywhere in the program.) The assemble function also looks only for an optional initial label, an instruction code and an address for the instruction if it has one, anything after that is ignored. So you can also put comments after instructions. (The // is optional in this case but good practice.) Beware that if you omit an address you will normally get 0, but if you miss out the address and add a comment you will get an error.

The assembler accepts both lower and upper case and for instructions will accept either or a mixture. Note that the case is important for labels, Loop and loop are different labels for example (but not good practice).

In addition to using SELECT (see next paragraph), the CPU can be programmed by entering numbers representing the instructions directly into memory, by typing into or modifying the assembly language area or by preparing a file containing symbolic instructions which can be loaded into the assembly language area. A program called the assembler is then run to convert the assembly language to numbers in memory. (Using a file is similar to how I prepared programs on paper tape in 1970 - except it is easier to correct mistakes in Notepad.) To help you, ASSEMBLE is automatically done whenever you change the assembly language area.

There is a SELECT button which will allow you to select from the following supplied programs:

add	Output the sum of two numbers. Requests input twice.
add/subtr	Input three numbers. Output the sum of the first two and the third minus the first.
max	Input two numbers and output the higher.
is sum 1-n	Test if the input is a triangular number, that is one that is the sum of 1 to a number n. If the number input is the sum of 1 to n, output n otherwise output 0.
square	Output the square of a number input.
times table	Input one number and output the times table for that number as far a the number times itself.
bubble sort	Input upto 20 numbers followed by a zero to start the sort. (I added restart capability to Mike Coley's original.)
overwrite	Memory overwrite illustrates one of the issues with memory being used for both instructions and data. As the program runs, numbers are INPUT and stored in memory locations 0 onwards. When memory location 10 is reached the instructions in the main program loop start to be overwritten with data, causing the program to stop running correctly as the LMC tries to interpret the data as an instruction. (Mike Coley)
ascii	ASCII stands for American Standard Code for Information Interchange and is the way of encoding characters so they can be stored and manipulated by computers. The original code used the numbers 33 to 126 to represent printable characters and 0-31 and 127 to represent control functions. 32 is used for space and is normally considered a printable character (even though you cannot print it). The only useful control function on the LMC is 10 for newline. This program outputs the numbers 32 to 126 as characters.
ascii table	Prints a number, a space and then the corresponding character for numbers 32 to 96. Above 99 the 4 characters per line output limit becomes a problem but you could remove the space and change max to go further.

Figure 3.7 – snippet of information from help page for instruction set

Pros

- ⇒ Visualisations of the various parts of the CPU and its links via buses are done well – most parts (e.g., Program Counter, Accumulator) are shown, and the visualisations of the data moving between various parts on the computer (see Figure 3.2) are quite smooth.

- ⇒ The website includes pre-programmed assembly code as examples for different types of algorithms to run (see **Error! Reference source not found.**). For example, the assembly c ode to add 2 numbers together when taken from input (see Figure 3.2).
- ⇒ The visualisation itself also has different options such as the speed to view it, and the different ways that one can view the data (see Figure 3.3).
- ⇒ There is also a tiny and very basic system debugger near the bottom right that explains when an input needs to be entered or if there is an unknown value entered in the code (see Figure 3.6).
- ⇒ The information on the help page is extremely detailed and is done in a way that all the different functions of Peter Higginson's Little Man Computer are explained in complete detail, so even the niche features can be used, without any need to contact the original creator (see Figure 3.7).

Cons

- ⇒ Although the design of the visualisation is done well, the website's front-end itself leaves a little to be desired e.g., the main part of the website (see Figure 3.1) doesn't automatically rescale to fit the web browser's entire screen on different devices (see Figure 3.5).
- ⇒ Although there is a debugger (see Figure 3.6), it doesn't necessarily show what sort of error is shown, e.g., Syntax error, Runtime error, etc. Furthermore, although the debugger tells you which line a possible problem is on, it can only show 1 error on 1 line at a single point in time. This could perhaps be improved by having a function or something similar that completely understands the assembly code, knows all the errors, and thus, can display all the errors in another field.
- ⇒ While line numbers are shown (to the right of the assembly coding text area) once a user has typed in their code (see Figure 3.6), it would be a lot better if the line numbers were in the same text field that the user is typing in. This could also be used in conjunction with the point mentioned above to show some sort of syntax highlighting with the use of the debugger.
- ⇒ Although the information on the help page is quite helpful, it is extremely verbose, and seems daunting to anyone who has not used the resource before. Furthermore, due to the fact that the instruction set is on a different page to the original page with the visualisations and the text input field, users need to constantly switch back and forth between the tabs in order to write the correct code for each operation (see Figure 3.7).

Summary

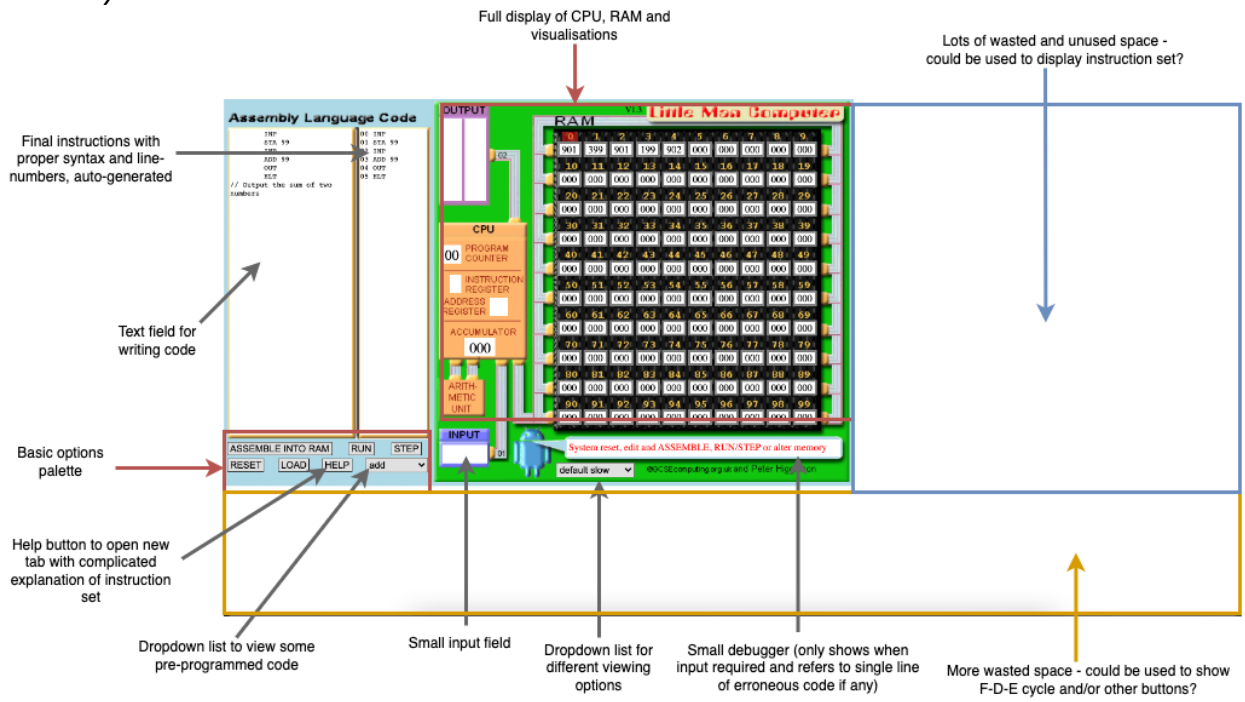

Figure 3.8 – annotations and evaluation of Peter Higginson's LMC

Although the teachers at Dubai College only use Peter Higginson's Little Man Computer to teach students about computer architecture and the assembly language, I decided to look into a few other existing solutions in order to gain a better understanding of various features that could potentially be implemented in my own solution. I also decided to do this because I thought that it could help me avoid accidentally making a solution too similar to something that already exists, and it could perhaps aid me in thinking about the layout that my solution will take, and how various features will be organised.

Paul Hankin's Little Man Computer

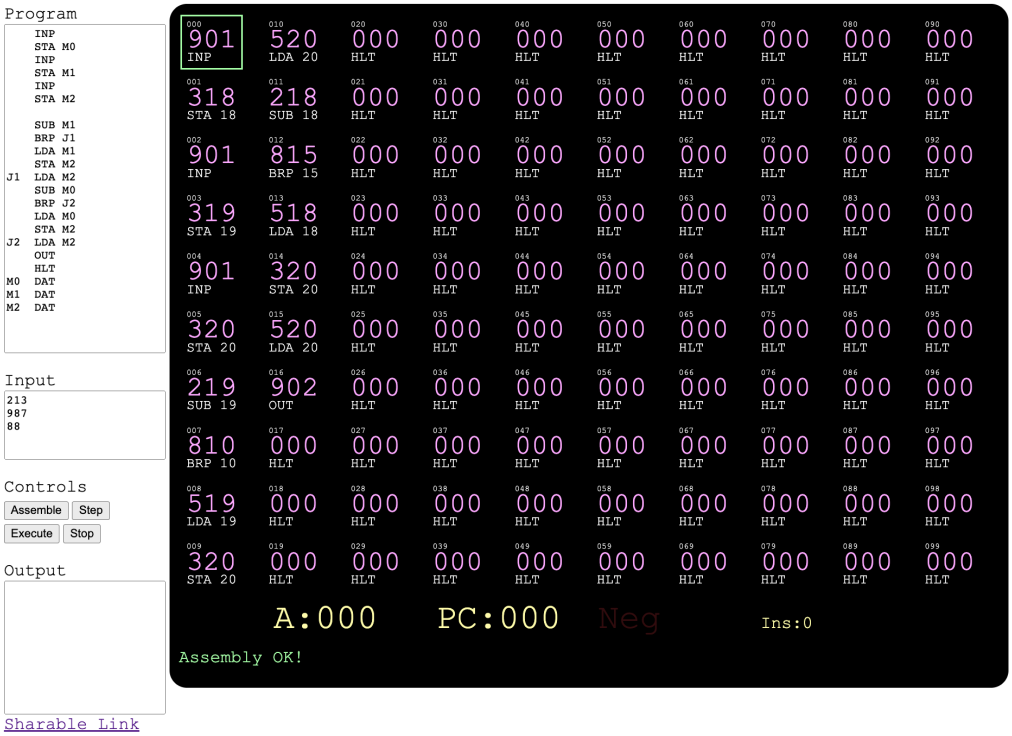

Figure 3.9 – basic Paul Hankin's LMC (Little Man Computer Emulator [Screenshot], 2016)

Figure 3.10 – stepped program being run

Figure 3.11 – small 1-line error identifier

Instructions

Numeric code	Mnemonic code	Instruction	Description
1xx	ADD	ADD	Add the value stored in mailbox xx to whatever value is currently on the accumulator (calculator). Note: the contents of the mailbox are not changed, and the actions of the accumulator (calculator) are not defined for add instructions that cause sums larger than 3 digits. Similarly to SUBTRACT, one could set the negative flag on overflow.
2xx	SUB	SUBTRACT	Subtract the value stored in mailbox xx from whatever value is currently on the accumulator (calculator). Note: the contents of the mailbox are not changed, and the actions of the accumulator are not defined for subtract instructions that cause negative results - however, a negative flag will be set so that **7xx (BRZ)** and **8xx (BRP)** can be used properly.
3xx	STA	STORE	Store the contents of the accumulator in mailbox xx (destructive). Note: the contents of the accumulator (calculator) are not changed (non-destructive), but contents of mailbox are replaced regardless of what was in there (destructive)
5xx	LDA	LOAD	Load the value from mailbox xx (non-destructive) and enter it in the accumulator (destructive).
6xx	BRA	BRANCH (unconditional)	Set the program counter to the given address (value xx). That is, the value in mailbox xx will be the next instruction executed.
7xx	BRZ	BRANCH IF ZERO (conditional)	If the accumulator (calculator) contains the value 000, set the program counter to the value xx. Otherwise, do nothing. Whether the negative flag is taken into account is undefined. When a SUBTRACT underflows the accumulator, this flag is set, after which the accumulator is undefined, potentially zero, causing behavior of BRZ to be undefined on underflow. Suggested behavior would be to branch if accumulator is zero and negative flag is not set. Note: since the program is stored in memory, data and program instructions all have the same address/location format.
8xx	BRP	BRANCH IF POSITIVE (conditional)	If the accumulator (calculator) is 0 or positive, set the program counter to the value xx. Otherwise, do nothing. As LMC memory cells can only hold values between 0 and 999, this instruction depends solely on the negative flag set by an underflow on SUBTRACT and potentially on an overflow on ADD (undefined). Note: since the program is stored in memory, data and program instructions all have the same address/location format.
901	INP	INPUT	Go to the INBOX, fetch the value from the user, and put it in the accumulator (calculator) Note: this will overwrite whatever value was in the accumulator (destructive)
902	OUT	OUTPUT	Copy the value from the accumulator (calculator) to the OUTBOX. Note: the contents of the accumulator are not changed (non-destructive).
000	HLT/COB	HALT/COFFEE BREAK	Stop working/end the program.
	DAT	DATA	This is an assembler instruction which simply loads the value into the next available mailbox. DAT can also be used in conjunction with labels to declare variables. For example, DAT 984 will store the value 984 into a mailbox at the address of the DAT instruction.

Figure 3.12 – Wikipedia LMC Instruction Set (Wikipedia - Little man computer [Screenshot], 2022)

Figure 3.13 – entire webpage as shown on browser

Pros

- ⇒ The step counter is extremely helpful and shows what is happening in a step-by-step manner, which is very easy to follow (see Figure 3.10). The fact that the memory location (which is currently being used in the step) is highlighted by a box around it draws attention to the current process happening.
- ⇒ The output, input, program, and control boxes are placed neatly on the side, in order to give the entire web page a nice, minimalist style and scheme, which is quite visually appealing (see Figure 3.9).
- ⇒ There are very few colours used, adding to the minimalistic style, and doesn't distract the user from the important features of the webpage, and what the resource is actually supposed to be used for.
- ⇒ By using the instruction set displayed on Wikipedia (see Figure 3.12), it is highly likely that the users will be familiar with this exact instruction set, if they have looked into computer architecture and assembly language before because Wikipedia is a very widely used website, and most people recognise it.
- ⇒ The Wikipedia page also has very lengthy and extremely in-depth explanations about each instruction and the instruction set as whole, as well as what the Little Man Computer is, how it can be used, and what it's useful for.

Cons

- ⇒ Although there is 1 line to display errors, as well as which line it is located on, it cannot locate – or doesn't alert the user – of multiple errors (there are 2 errors that should be identified in the Program input field in Figure 3.11).

- ⇒ There is no direct link or pop-up from the main Little Man Computer visualisation resource to the instruction set, so one needs to go to the parent page to access and find out that the resource uses the same instruction set as shown in Wikipedia (see Figure 3.12).
- ⇒ There are no line numbers in the text field where assembly code is entered – this makes it extremely hard to keep track of each line and where certain parts of the code or algorithm are.
- ⇒ There is an input box which has to have predefined inputs, so a user cannot enter inputs while the program is running, when an input is required. This also means having to know exactly which inputs you want to take in, as well as how many inputs are required, and which order they need to be inputted in.
- ⇒ There aren't any options to adjust the visualisation, such as the speed, or quickly clearing the data stored in the RAM (memory), so that a new algorithm or piece of code can be written, and the old one will not affect it.
- ⇒ There aren't any predefined or pre-programmed algorithms which can be used in order to view the layout of the assembly code that is required, as well as aid users in understanding how basic operations, such as adding two numbers, can be computed using assembly language.
- ⇒ The website's contents don't automatically resize to the full screen of the page on the browser, and the space could be utilised much for efficiently and effectively, perhaps including the instruction set in the unused space, or some other information.

Summary

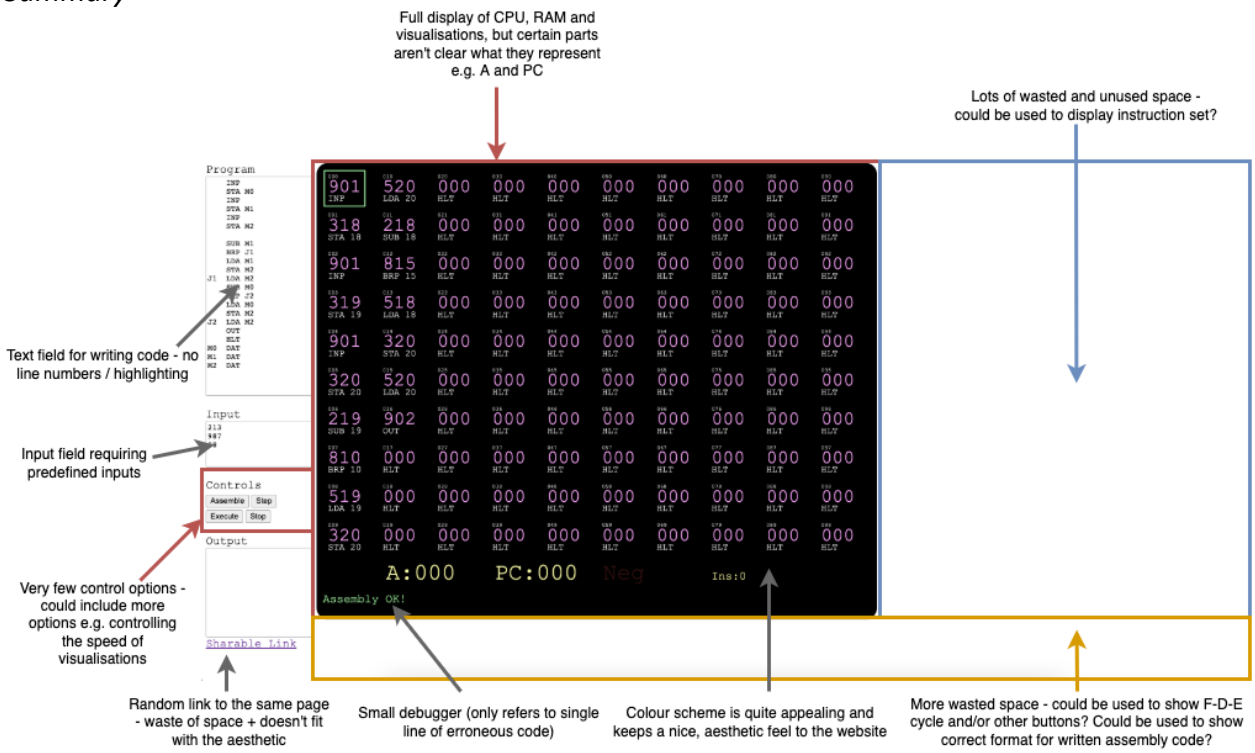

Figure 3.14 – annotations and evaluation of Paul Hankin's LMC

During the Interview with Mr. Wood (see above), we realised that a lot of features of the resource should be tailored to be most helpful to the students, along with a few features for teachers. Therefore, I need to interview students who have learned about these topics and understand what the most useful features would be (see below & below).

Having researched previous solutions, including the current resource used at the school to teach these topics, and gaining a more detailed understanding of each of them, I thought that this would be the perfect time to interview a student about the resource, showing them the annotations and notes above (Figure 3.8 & Figure 3.14), getting their feedback on each of them, and questioning them about the best features that they would like implemented in the solution, as well as any other suggestions they may have.

Interview with Potential User 1

Conversation

Date: 12/02/2022

Attendees: Raghav Awasthi and Nakul Pillai (Year 12 student)

Abbreviations used for this interview: RA = Raghav Awasthi; NP = Nakul Pillai; LMC = Little Man Computer.

Greetings are not included.

RA: Firstly, when you're writing assembly code, what do you think about syntax highlighting and debugging?
NP: Yeah. I feel like in the current *LMC*, it's a bit difficult to really understand what you were writing and what is actually there and what it does, so, syntax highlighting, it just makes things simpler to understand. I think the current solution doesn't have that, so I think it would be definitely a good thing to implement.
RA: Right.

RA: Are there any other particular features you think that would be better, or that would help in contextualizing and understanding the assembly language?
NP: I feel like the *LMC* solution currently, it's just a lot of information on the screen. A lot of stuff is happening, and you don't know what a lot of it does. It doesn't actually detail it. So, not a short description, but just a better way of understanding what each aspect of the *LMC* does would be good. And maybe descriptions for individual bits, at least some sort of introduction tutorial, maybe, as to what it does, instead of just dumping a bunch of information on you.
RA: Right, okay. Yeah, that makes sense. So, would you think that creating some sort of link from the main screen into a new page that just has really detailed explanations would be good?
NP: Yeah, I think that's exactly what needs to be done.

RA: Another thing was having your own login as a student and then students can be put into classes for teachers to see. And essentially what'll happen is, for each individual student, there'll be a bunch of programming challenges that you can try and convert from a given language into assembly code. So essentially what I want to do is have challenges written in high level languages, and then you try to convert it into assembly code.
NP: Right.
RA: What would you say is the best high-level language or languages for that purpose – so students can understand that code and attempt to convert it?
NP: Probably Python, because, at least at Dubai College, our main language that we learn is Python, so it's easier for students to transfer the skill there. And because students know Python better, it's probably easier for them to understand what the code or the high-level language does and then be able to interpret what the lower-level language can do.
RA: Right, that makes a lot of sense, because this resource is for this school specifically, and the only high-level language that students are taught is Python.

RA: Also, as a student, every time you log in, would you want to be able to see all your previously completed challenges? Like a log almost?

NP: Yes. That would be helpful. Cause say sometimes you're not too sure what you're currently doing. It's easier to look back at the previous ones and see how those work, especially with if you're trying a new challenge, you're not too sure about it, you can go back and kind of read over the old one and see what it does, and it gives you better insight.

RA: And like I said, the teacher will probably be able to see all of your completed challenges, but do you think it would be helpful to to let teachers set challenges? Cause during my earlier discussion, it was sort of like: there are challenges, and students can go home and do them if they want to; not necessarily, but only if they want to. Do you think it would be better if teachers had the ability to set assignments or challenges?
NP: I mean, one thing you could do, is allow teachers to create challenges and there can be either a list of challenges that students could do, but if a teacher needs someone to do it, then they can assign the challenge. But keep it so that a student can go home and do whatever they want. You can have a list of all the existing challenges, and when a teacher assigns or creates one that needs to be done, it can then be sent to them.
RA: Right.

RA: Also, how would you feel about being able to see other students' completed challenges? Would you rather just see your own because the teacher will probably be able to see all, but would you, as a student, feel uncomfortable with other students seeing your challenges?
NP: Yeah, potentially. Because if I'm slightly shy, or don't want to show my work, other people – apart from the teacher – shouldn't just be able to, say, go into my profile, and see my solutions. What you could do is, if a teacher thinks that a student's solution is really good, they could highlight it, or publish it, so that other students would be able to see the solution – perhaps even who wrote the solution, but maybe not, for the sake of anonymity.
RA: Okay.

RA: One more thing. The differences between the GCSE and A-level spec. I need to know how far, or how much information you think I should go into, for this. So, doing just A-Level or just the GCSE spec, or combined?
NP: The A-Level is probably better because that goes a bit further into the actual workings. So, showing the FDE cycle would be helpful. Also, I'm looking through it, and I think that the A-Level contains everything the GCSE contains, and more.
RA: Yeah, I think it does.
NP: Yeah. So then when you do it for the A-Level spec, you would have already covered the GCSE students as well.
RA: Okay. So, you think that including the FDE cycle would also be fine, because you would like this as an A-Level student?
NP: Yeah. But because it'd be helpful at GCSE too. And if you're a GCSE student, you could focus on just the aspects that you need. Perhaps one thing you could do is give the user an option when they log in, whether they want to view the GCSE version, or view the A-Level version.

RA: Right, and obviously the A-Level version will have more content.
NP: Yeah. Or maybe you can just toggle a button that says “Show FDE Cycle”, so it’s just 1 page, but more information is added if you want the “A-Level version”.
RA: Yeah. Cause I don't think there are many differences between the A-Level and GCSE, so it might be better to just have one instead.

Summary
During this interview with Nakul (an ex-GCSE Computer Science student and current A-Level Computer Science student), I was able to reaffirm some of the conclusions that I made from the first client interview and added to the Initial Client Requirements (see above).

One specific thing about the layout and view of the resource was that there shouldn’t be too much information thrown at the user – as if throwing them into the deep end. Instead, having quite a basic and simple way of understanding each component or part of the resource or page – such as with an introductory tutorial of some sort to the platform – would be extremely helpful. He also mentioned that perhaps linking the main resource to some other page with more verbose, detailed, and in-depth explanations of each part of the resource could be provided for the students who want more information.

We also briefly mentioned the possibility of teachers being able to create challenges but had a much more in-depth discussion about the possibility of teachers being able to “publish” students’ solutions to the rest of the class, so that they can all see a model solution. I have added this to the updated requirements; however, I am still quite unsure about whether this would be required, so I have discussed this in further detail in the Scope section (see below).

From the information and ideas discussed above, I have drawn out the necessary additional requirements that need to be made and appended to the overall requirements.

Additional Requirements

8. Create an introductory tutorial for new users of the resource
    a. This means showing the basic features of it and how to use each
8. Create a link on the main resource that opens a new page with full descriptions and explanations of every part of the resource including the Instruction Set
8. Coding challenges should be written in Python for students to attempt to convert this into Assembly code
8. Students cannot view other students’ solutions
    a. Teachers can “publish” students’ solutions for other students to see them
8. Students can attempt challenges without teachers having to set them
    a. Teachers can set challenges which are required to be completed by students
8. Have options to toggle particular components or sections of the resource off/on

Updated Overall Requirements

1. Allow users to write code in assembly language
2. Allow users to view visualisations of assembly code they have written (CPU and RAM)
3. There should be automatic syntax highlighting for any code that is written
4. There should be verbose debugging for erroneous code
    a. This could mean creating a translator for converting assembly code written in the text field into a language which can display the errors appropriately
5. Should be able to view the instruction set on the same main page
    a. Add a link on the main page that goes to a page with full descriptions and explanations of each component of the resource and the Instruction Set
6. Show the Fetch-Decode-Execute cycle and what is currently being run
7. Create a list of coding challenges that can be solved by converting code (in Python) into assembly code
8. Create an introductory tutorial for new users of the resource
    a. This means showing the basic features of each component with a short description and how to use each
9. Have an option for users to toggle particular components on the main page off or on
10. Accounts
    a. Student account
        i. Able to view and complete challenges (without a teacher setting them)
        ii. Able to then self-mark challenges and submit
        iii. Able to view a log of their previously completed challenges
        iv. Able to view a log of previously saved code (including non-challenges)
        v. Cannot view other students' solutions to challenges unless "published"
    b. Teacher account
        i. Teachers have classes which are groups of students
        ii. Can set challenges to be completed by all students in a class
        iii. Can view students' submitted challenges
        iv. Can view students' challenges even if not submitted
        v. Can "publish" students' solutions to allow other students to view them
    c. No account
        i. Can still view visualisations and code in assembly
        ii. Able to view and complete and self-mark challenges, but don't have a log of previously completed ones
        iii. Cannot submit challenges to a teacher – no linked account
        iv. Cannot see a log of previously saved code

One thing that was heavily discussed during the interview was the difference between the GCSE and A-Level specification for Computer Science, and how that would affect the resource. We weren't too sure about the exact differences between the two, so we concluded that an option to toggle certain components of the resource off would be useful, if a GCSE student feels too

overwhelmed by the amount of information presented on the resource. However, I think that I could do further research into this and actually identify the exact differences between the two specifications, in order to determine whether a toggling feature is actually required or not. Doing this could also help me identify a definite list of the various components that need to be represented in the visualisation. Hence, I decided to conduct document analysis (see below) on both specifications, but specifically the sections that talk about Computer Architecture and the Assembly Language.

Document Analysis

In this section, I have included the most recent AQA Computer Science GCSE and AQA Computer Science A-Level specifications available to the public, online, as of March 2022. Please note that I have only included the specification points that are relevant to what needs to be analysed.

GCSE Specification

3.4.5 Systems architecture

Content	Additional information
Explain the role and operation of main memory and the following major components of a central processing unit (CPU) within the Von Neumann architecture: • arithmetic logic unit • control unit • clock • register • bus.	A bus is a collection of wires through which data/signals are transmitted from one component to another. Knowledge of specific registers is not required.
Explain the effect of the following on the performance of the CPU: • clock speed • number of processor cores • cache size.	
Understand and explain the Fetch-Execute cycle.	The CPU continuously reads instructions stored in main memory and executes them as required: • fetch: the next instruction is fetched to the CPU from main memory • decode: the instruction is decoded to work out what it is • execute: the instruction is executed (carried out). This may include reading/ writing from/to main memory.

Figure 3.15 – AQA Computer Science GCSE Specification extract for Computer Architecture (GCSE Computer Science Specification, 2020)

Figure 3.15 shows what the students taking the AQA GCSE Computer Science course need to learn about computer architecture. It shows that they need to know about the Arithmetic Logic

Unit (ALU), the control unit, the clock, registers, and finally, buses. The students need to know about their role and operation (the way they work) within the Central Processing Unit (CPU) and hence, these five components form the absolute bare-minimum requirements that should be shown in the visualisation part of this resource. The students need to understand the fact that a bus is a collection of wires through which data is transmitted, so the resource should represent a bus as multiple lines, and all data that is being moved from and to various components of the CPU should go via buses.

The students also need to be able to understand the Fetch-Decode-Execute (FDE) cycle, and so, my initial thoughts are that including some sort of output for what is happening with regards to the FDE cycle could aid students understand this process – even if this goes into a little bit more complexity. Although the specification only requires the aforementioned five components and an explanation of the FDE cycle, the memory (RAM) is also essential to understand how data is actually stored, retrieved, and used within the CPU. Thus, some memory locations should also be included within the visualisation.

I now need to analyse the A-Level Specification in order to find the major differences between the two specifications, and eventually come to a conclusion about the final requirements of what the visualisation is and needs to include.

A-Level Specification

4.7.3.1 The processor and its components

Content	Additional information

Explain the role and operation of a processor and its major components:
• arithmetic logic unit
• control unit
• clock
• general-purpose registers
• dedicated registers, including:
  • program counter
  • current instruction register
  • memory address register
  • memory buffer register
  • status register.	

4.7.3.2 The Fetch-Execute cycle and the role of registers within it

Content	Additional information
Explain how the Fetch-Execute cycle is used to execute machine code programs including the stages in the cycle (fetch, decode, execute) and details of registers used.	

Figure 3.16 – AQA Computer Science A-Level Specification extract for Computer Architecture (AS and A-level Computer Science Specification [Screenshot], 2019)

4.7.3.5 Machine-code/assembly language operations

Content	Additional information

Understand and apply the basic machine-code operations of:
• load
• add
• subtract
• store
• branching (conditional and unconditional)
• compare
• logical bitwise operators (AND, OR, NOT, XOR)
• logical
  • shift right
  • shift left
• halt.
Use the basic machine-code operations above when machine-code instructions are expressed in mnemonic form- assembly language, using immediate and direct addressing.	

Figure 3.17 – AQA Computer Science A-Level Specification extract for Assembly Language

Instruction set

LDR Rd, <memory ref>	Load the value stored in the memory location specified by <memory ref> into register d.
STR Rd, <memory ref>	Store the value that is in register d into the memory location specified by <memory ref>.
ADD Rd, Rn, <operand2>	Add the value specified in <operand2> to the value in register n and store the result in register d.
SUB Rd, Rn, <operand2>	Subtract the value specified by <operand2> from the value in register n and store the result in register d.
MOV Rd, <operand2>	Copy the value specified by <operand2> into register d.
CMP Rn, <operand2>	Compare the value stored in register n with the value specified by <operand2>.
B <label>	Always branch to the instruction at position <label> in the program.
B<condition> <label>	Branch to the instruction at position <label> if the last comparison met the criterion specified by <condition>. Possible values for <condition> and their meanings are: EQ: equal to NE: not equal to GT: greater than LT: less than
AND Rd, Rn, <operand2>	Perform a bitwise logical AND operation between the value in register n and the value specified by <operand2> and store the result in register d.
ORR Rd, Rn, <operand2>	Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d.
EOR Rd, Rn, <operand2>	Perform a bitwise logical XOR (exclusive or) operation between the value in register n and the value specified by <operand2> and store the result in register d.
MVN Rd, <operand2>	Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d.
LSL Rd, Rn, <operand2>	Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d.
LSR Rd, Rn, <operand2>	Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.
HALT	Stops the execution of the program.

Figure 3.18 – AQA Computer Science A-Level Assembly Language Instruction Set (AS and A-level Computer Science Assembly Language Instruction Set [Screenshot], 2019)

Figure 3.16 shows us that students at A-Level are required to know about 5 more components of the CPU than they did at GCSE – they need to know about 5 dedicated registers: The Program Counter, Current Instruction Register, Memory Address Register, Memory Buffer Register, and finally, the Status Register. Along with this, as before, students need to be able to understand the FDE cycle, however the only difference this time is that they need to understand the use of the dedicated registers within the cycle as well.

Figure 3.17 highlights the fact that A-Level students need to be able to understand and use a variety of basic machine code operations. This means that they also need to be able to write machine-code instructions in mnemonic form- assembly language. All of the instructions that students need to be able to know are listed in Figure 3.18, and so, these are all of the instructions that the resource should be able to process and thus show a visualisation of.

As mentioned before, the RAM is essential in understanding the processes of the CPU and how it works – data is fetched from the RAM and stored there. Hence, the RAM must be included within the visualisation.

Having now analysed both the GCSE and A-Level specifications, I need to discuss the explicit differences in greater detail and come to a decision as to what to include within the resource. I have done this below.

Summary

The main difference between the requirements for the visualisation between the A-Level and GCSE is the fact that the A-Level students need to be able to understand the processes of the dedicated registers and how they're used in the FDE cycle, whereas GCSE students do not need to be able to do this. Although there are 5 new components (the dedicated registers) that are added, I feel as though they don't add much more complexity to the processes of the CPU and what happens within – this view was also backed by Nakul in the Interview with Potential User 1 section (see above), where he stated that he didn't feel much of a difference between the two qualifications. Even within the processes of the FDE cycle, the dedicated registers do not add much more complexity, especially since the GCSE specification covers what a register is. Therefore, I believe that there is no reason to exclude the dedicated registers – or add an option to exclude them – from the visualisation.

In addition, another difference is the fact that A-Level students are formally required to be able to code in the Assembly Language, using the instruction set provided by AQA (see Figure 3.18), whereas the GCSE specification doesn't explicitly state this. However, in order to gain an understanding of how the visualisation works and to write one's own code to view a visualisation, the Assembly Language is a required component for the resource, so it must be included.

Overall, although there are a few differences between the two specifications, these variances are not that great. In fact, including the additional parts from the A-Level specification may even make it easier to understand the content on a GCSE level, thus there need not be any differences in the visualisation of the computer architecture between the GCSE and the A-Level students. Therefore, I will be including everything from the A-Level specification in the resource, and this includes everything from the GCSE specification.

I have formally detailed the additional requirements for the visualisation below.

Additional Requirements

1. Allow users to write code in assembly language (Figure 3.18 contains all the instructions and operations that the resource should be able to run)
2. Allow users to view visualisations of assembly code they have written, with the components defined in Figure 3.15 & Figure 3.16 included in the visualisations
3. Show the Fetch-Decode-Execute cycle and what is currently being run – including the processes with the 5 aforementioned Dedicated Registers

4. The Instruction Set displayed to the user should be the A-Level AQA Computer Science Instruction Set

Updated Overall Requirements

1. Allow users to write code in assembly language (Figure 3.18 contains all the instructions and operations that the resource should be able to run)
2. Allow users to view visualisations of assembly code they have written, with the components defined in Figure 3.15 & Figure 3.16 included in the visualisations
3. There should be automatic syntax highlighting for any code that is written
4. There should be verbose debugging for erroneous code
    a. This could mean creating a translator for converting assembly code written in the text field into a language which can display the errors appropriately
5. Should be able to view the instruction set on the same main page
    a. Add a link on the main page that goes to a page with full descriptions and explanations of each component of the resource and the A-Level Instruction Set
6. Show the Fetch-Decode-Execute cycle and what is currently being run – including the processes with the 5 aforementioned Dedicated Registers
7. Create a list of coding challenges that can be solved by converting code (in Python) into assembly code
8. Create an introductory tutorial for new users of the resource
    a. This means showing the basic features of each component with a short description and how to use each
9. Accounts
    a. Student account
        i. Able to view and complete challenges (without a teacher setting them)
        ii. Able to then self-mark challenges and submit
        iii. Able to view a log of their previously completed challenges
        iv. Able to view a log of previously saved code (including non-challenges)
        v. Cannot view other students' solutions to challenges unless "published"
    b. Teacher account
        i. Teachers have classes which are groups of students
        ii. Can set challenges to be completed by all students in a class
        iii. Can view students' submitted challenges
        iv. Can view students' challenges even if not submitted
        v. Can "publish" students' solutions to allow other students to view them
    c. No account
        i. Can still view visualisations and code in assembly
        ii. Able to view and complete and self-mark challenges, but don't have a log of previously completed ones
        iii. Cannot submit challenges to a teacher – no linked account
        iv. Cannot see a log of previously saved code

After having analysed the GCSE and A-Level specifications and arriving at the conclusion, I still feel as though there is still some more information that could be taken from a student about a few other features, and as though some points from the first interview with a potential user could be expanded upon. Getting a second opinion would also help me in attaining a better general understanding of the necessities for the resource from a student's perspective, so I have decided to interview another potential user.

Interview with Potential User 2

Conversation
Date: 18/02/2022
Attendees: Raghav Awasthi and Pranav Sangwan (Year 12 student)
Abbreviations used for this interview: RA = Raghav Awasthi; PS = Pranav Sangwan; LMC = Little Man Computer.

Greetings are not included.

RA: Well, my first question for you is, since it's you who will be using it, are there any ways that you can think the teaching of Computer Architecture and Assembly Language could be improved on, at Dubai College?
PS: Yeah. So, since we are first taught about the assembly language and then learn more about it from the *LMC*, I think we do understand what the functions do, but it doesn't really help us visualise what's happening because it's really complicated to see things like what the numbers [instructions] are. I think it could be improved if it were more accessible to students trying to learn about what happens when functions are called in the assembly language.
RA: Right. Sort of going through it line by line and actually seeing the visualisations more clearly?
PS: Yeah. And the steps actually being explained in the visualisation.
RA: Oh yeah. That seems like a really good feature. Would you like another field for verbose output, just to explain what's happening?
PS: Probably yeah, something like subtitles for each action.

RA: As a student, would you prefer this as an online resource or an application or… how do you think it would best work or be the most accessible for you?
PS: Probably an online resource because we'll only use this a few times a year maximum, so keeping that as an application may not be the best idea. Like for me, I only download an application if it's something I regularly use, otherwise it takes up unnecessary space on my computer. So, if I do want to access it, I think it would be fine if I just access it online.
RA: Right.
PS: And also, we use the school computers and that might not be the best thing to do, like try to install it on each computer.

RA: Right, of course. Since the Computer Science department has desktops in each classroom, it may not be the most ideal thing for the teachers to have to install an application into each and every device.
PS: Exactly.

RA: When it comes to challenges, which I've explained previously, what sorts of challenges do you think would be best? Obviously, there'll be basic things, like adding two numbers, subtracting two numbers, multiplying and stuff, but there could also be slightly more complex stuff like a higher or lower game. What do you think about that? How complex would you want these sorts of challenges to go?
PS: Personally, I think you could get a whole range of challenges and then anyone can contribute to making challenges, and once those challenges have been verified by like at least like five people who have coded in it or something, then they can then classify, yes, this challenge is this. So that kind of builds up a database of challenges for everyone.
RA: Yeah, that's pretty cool – I hadn't thought about that. So, just for clarification, what you're saying is that people should be able to create their own challenges and share that with the rest of their class.
PS: Yeah.

RA: Obviously apart from the challenges, you'd want to be able to write your own code, just like do whatever you want, right?
PS: Yeah.
RA: And obviously you'd want to be able to keep a log of that for when you sign in?
PS: Yeah. I think having an account could certainly be useful because a lot of the times there's like a niche thing that you also code and you forget how to do that, but obviously that's not something you have to keep in your memory as you may not use it but having that as a log that you can fall back on, that would certainly be useful.

RA: Also, what are your thoughts on uploading and downloading completed solutions? So, for example, let's say you found out how to add two numbers in assembly. You wrote the assembly code; you saw the visualisations. Would you want to be able to like download that as sort of a text file or something to your local machine?
PS: Personally, I wouldn't want to that much because I wouldn't really need some random piece of assembly code, sitting on my computer, when there's no real way to run it on my own machine, offline. I'm just kind of going to be using the resource as more of a learning experience to understand, and once it's done, I'm kind of finished with that.

PS: On the other hand, uploading would be a lot more useful.
RA: Why do you think that?
PS: Uploading would allow you to – say you're not connected to the Wi-Fi – you'd be able to type it locally and then view the visualisations later.

Summary

During this interview with Pranav (also an ex-GCSE Computer Science student and current A-Level Computer Science student), I was able to confirm quite a few points that I'd discussed previously in the other interviews and expand on some new ideas brought into light by Pranav.

He mentioned the idea of having some sort of field for verbose output, that fully describes and explains the step-by-step processes going on in the processor. I think that this could be paired really well with the requirement for showing the FDE cycle, so this could be combined into a single field which lists every single operation being carried out and has some sort of indicator to show which part of the FDE cycle is currently being executed.

We also discussed what sort of platform this should be – an application or a website. After quite a bit of discussion, we concluded that a website would be the best approach, particularly because an application would require the resource to be installed on every single desktop computer in every single computer science room, which would be extremely time consuming, inefficient, and just not optimal. A website, on the other hand, would make it much easier for students to access the resource, as they would also easily be able to access it on their own personal devices without taking additional storage space. Making the resource a website also reduces the possibilities of failures caused as a result of creating an application such as being able to run the resource in the same way on multiple different Operating Systems.

Something else that was suggested was the idea for students to be able to create their own challenges, with the answers. Then these challenges could be published to the large list of all challenges if 5 or more people "verified" these – by testing the challenge and making sure that it works. This seemed like a really cool and interesting idea, but may not necessarily be necessary for this project, so I have discussed this further in the Scope section (see below).

We also discussed uploading and downloading text files. The basic idea that I proposed to him was that you would be able to write the assembly language code in a text file, and then this could be uploaded into the resource, and automatically added into the text field, ready to run the visualisation. For downloading, the idea was similar, but instead, a user would be able to click the download button, which would save the current text in the text field as a text file onto the user's local system. After a bit of thinking, we established that downloading may not be very useful, as there's no way to actually run that text file on a local machine, and instead, there would just be a random text file lying around on a device. However, the upload feature is very good, because it would allow a user (who temporarily doesn't have Internet access) to write the assembly code in a text file, and just have to upload it to the website when they have Internet access, in order to view the actual visualisation, or submit the challenge.

I have compiled all of the ideas above into very basic overviews of any things that have not been previously discussed during interviews, and I have detailed the new information below.

Additional Requirements

8. Verbose output with step-by-step details of each process being run
    a. In the same component, there should be an indicator of which stage of the FDE cycle is currently being run
8. The resource should be a Web Application (not a website – interactive)
8. Students can create their own challenges (tentative)
    a. These can be "verified" if 5 or more people test them
8. There should be an option to upload text files, automatically convert them into the right format, and put them into the text field where the assembly code goes

Updated Overall Requirements

1. The resource should be a Web Application
2. Allow users to write code in assembly language (Figure 3.18 contains all the instructions and operations that the resource should be able to run)
3. Allow users to view visualisations of assembly code they have written, with the components defined in Figure 3.15 & Figure 3.16 included in the visualisations
4. There should be automatic syntax highlighting for any code that is written
5. There should be verbose debugging for erroneous code
    a. This could mean creating a translator for converting assembly code written in the text field into a language which can display the errors appropriately
6. Have an option for users to upload a text file
    a. This should automatically be converted into the appropriate format and put into the text field where the assembly code should go
7. Should be able to view the instruction set on the same main page
    a. Add a link on the main page that goes to a page with full descriptions and explanations of each component of the resource and the A-Level Instruction Set
8. Show a verbose output of step-by-step details of each process being run
    a. This should include an indicator of which part of the FDE cycle is being run – including the processes with the 5 aforementioned Dedicated Registers
9. Create a list of coding challenges that can be solved by converting code (in Python) into assembly code
10. Create an introductory tutorial for new users of the resource
    a. This means showing the basic features of each component with a short description and how to use each
11. Accounts
    a. Student account
        i. Able to view and complete challenges (without a teacher setting them)
        ii. Able to then self-mark challenges and submit

iii. Able to view a log of their previously completed challenges
iv. Able to view a log of previously saved code (including non-challenges)
v. Cannot view other students' solutions to challenges unless "published"
vi. Can create their own challenges
    1. Type the instructions
    2. Display an example of the algorithm in Python
    3. Define shown and hidden test-cases
    4. Able to "upvote" student-created challenges after testing them
    5. 5 or more "upvotes" to be pushed to the global list of challenges

b. Teacher account
    i. Teachers have classes which are groups of students
    ii. Can set challenges to be completed by all students in a class
    iii. Can view students' submitted challenges
    iv. Can view students' challenges even if not submitted
    v. Can "publish" students' solutions to allow other students to view them

c. No account
    i. Can still view visualisations and code in assembly
    ii. Able to view and complete and self-mark challenges, but don't have a log of previously completed ones
    iii. Cannot submit challenges to a teacher – no linked account
    iv. Cannot see a log of previously saved code

Having done the above, I now have quite a good understanding of what the student users of the system require and want as part of the resource. With this information, the next thing that I needed to do was discuss these new ideas with my client and see if he agrees with the suggestions. Speaking to the client with my knowledge of current online resources, as well as what features seem to be in high demand will aid me in articulating the current proposed idea for the overall solution and getting feedback from him will help me in finalising the requirements of the project. Thus, I set up another interview with my client.

Interview 2 with Client

Conversation
Date: 22/02/2022
Attendees: Raghav Awasthi and Mark Wood (Client)
Abbreviations used for this interview: RA = Raghav Awasthi; MW = Mark Wood; LMC = Little Man Computer.

Greetings are not included.

RA: One thing that was suggested was, with the challenges, if students could "push" their own challenges – write them up themselves.

MW: So, are you saying someone puts a challenge out there, and then maybe people can vote that as a good challenge? So, a ranking system like Reddit, or Code Wars?
RA: Yeah, basically.
MW: Yeah. I think that idea is brilliant.
RA: The only thing is, because it's a teaching resource, is it necessary?
MW: Sorry, let me just clarify what you're saying. Are you saying that when students have solved the problem, people can rank their solutions? Or are you saying students can create challenges for other students to solve?
RA: Yeah, the second one.
MW: I think if you're going to do that, there needs to be for some very clear indicator that this is a student's challenge, and this is a teacher's challenge. So, it might be for example, a green problem is a teacher problem, and an amber problem is a student problem. If there's going to be some sort of leaderboard or way to check what the solutions are – of students – I, as a teacher, need to be able to see, these are this student's green solutions, which means I want to see their solutions to challenges a teacher has set.
RA: Okay.
MW: But I might also want to see the student's amber solutions, which are the solutions he's created for the students.

RA: Okay, so are you saying that teachers should also be able to create some challenges?
MW: Yeah, I think so.
RA: Because there will probably be predefined challenges already. Just things like adding, multiplying, etc.
MW: Do you know what, for this project, since it's a teaching resource… I'm not quite sure.
RA: Yeah, so it basically depends on whether you think that the feature would actually help you teach students.
MW: I'm just thinking, because that might just make my life slightly harder if it's included, if you know what I mean. Because as a teacher, then, I would have to say, yeah… here are the exact outputs of this program that I'm looking for. So, I would have to code the algorithm in a language first. Then I would have to try to convert it to Assembly myself, and see what outputs are actually produced. Then I would have to say, here are the outputs that are expected, and no other outputs will be acceptable.
RA: Right, and that language would probably be Python?
MW: Yeah, because again, that's what we teach at this school.
RA: Yeah.
MW: But the thing is, essentially, what you're trying to mark as a teacher, is the logic. So that means your style and the design, yeah?
RA: Right, because you want to see their thought process; the abstraction; the decomposition; the actual logic, like you said.
MW: Yeah, so I'm not really quite sure how you would implement that – because it would definitely be a massive task to accomplish. So, I'm leaving that feature down to you, as the

developer of this resource, because I don't think it's completely necessary, but I could also see the few potential benefits of it. I'm leaving it to you so you can do it based on whether you feel it's in scope, or whether it's out of scope.
RA: Right.

RA: OK, so, from the students' perspectives, when I spoke to them in the interviews, they were saying that they don't necessarily want the teacher to be able to see all of their code. Even if it's not their challenges that they've solved, but their own code just to see the visualisations because they still want to keep a log of their code.
MW: Yeah, that's fine. So basically, like a submit button or a private/public button, and it says these ones I'm happy for people to see, and these ones I'm not.
RA: Right, because students still want to use this resource as a learning resource individually as well.

MW: But I don't think they should have the option to not let me see challenges that I set.
RA: Yeah.
MW: So, they should only be able to set the ones that they build independently to private. Do you know what I mean? Like, if I sent an assignment, I want to be able to see it. So yeah, I'm more than happy for them to be able to have sort of private code that they've written. I think it'll definitely encourage them to work on it on their own, rather than being pressured by, oh… my teacher might see this, so I don't want to use this resource in case he sees my code and it doesn't work. Yes, I'm more than happy for students to have their own private code.
RA: Great, that makes a lot of sense.

RA: I was also thinking about the format, or what sort of resource this should be because, during one of the earlier interviews, I was suggested by a student to do a website.
MW: Yeah, a web-based project seems the most appropriate because there's a lot of school computers, and it would be a real hassle to try and install the software in every single device.
RA: Yeah, that was exactly my thought process.

RA: The other thing was download and upload. So, uploading files seemed to be useful, because, say you didn't have internet access, you'd want to be able to write a text file that contains assembly code, and when you upload that file, it just automatically copy-pastes the assembly code to text field. Right? But then download might not be as useful because you can't really see visualisations with random assembly code, saved as a text file on your local machine.
MW: Yeah, I see what you're saying. I see what you mean. I guess it would be cool. Download… I can't really see any reason for wanting that – the only thing I could think of is, what if I wanted to print out the code for students to have on a paper to annotate and analyse stuff? But in that case, I'd just grab the code, copy, and paste it into Word, and I can't see a need to press a button and download.
RA: Yeah.

MW: Yeah, I can't see that being necessary… unless I see that down the line, I do not need that as a requirement.

Summary

During this second interview with my client, we went over the additions that could be made to the resources that the students had suggested and discussed whether they would help the teacher use the resource as a teaching resource.

We discussed the idea of teachers being able to create their own challenges as well. This was briefly mentioned in the first interview with a potential user (see above), but I went through this in much more detail with Mr. Wood. We also had further discussions about the idea of the student-created challenges. Eventually, we arrived at the conclusion that these two features may not be necessary but could potentially enhance the resource by giving it a few additional features. As these two things need to be discussed in even further detail, I have spoken about them in the Scope section (see below & below).

Another thing that we discussed was whether teachers should be able to see students' challenges, even if they haven't completed them, and whether teachers should be able to see students' personal code which they merely use to see the visualisation, rather than as a challenge (their own personal log). After some discussion, we concluded that teachers should only be able to see students' submitted challenges, as well as any challenges that have been set by the teacher, regardless of whether they've been completed or not – these can be combined into a single requirement below.

Finally, we confirmed that the resource would be most suitable as a web-based project, as that would be easily accessible on every device. As a result of deciding on creating a web-based resource that is reliant on Internet, we discussed about file upload and download, expanding on some ideas that I discussed during the second interview with a potential user (see above). We came to the same conclusion that only file upload would be useful.

After completing this interview and writing a basic overview of each point that was discussed, I have added any new ideas or points below, and amended the overall requirements.

Additional Requirements

1. Teachers can create their own challenges
2. Teachers can only view challenges for students (both completed and unfinished)

Updated Overall Requirements

1. The resource should be a Web Application
2. Allow users to write code in assembly language (Figure 3.18 contains all the instructions and operations that the resource should be able to run)

3. Allow users to view visualisations of assembly code they have written, with the components defined in Figure 3.15 & Figure 3.16 included in the visualisations
4. There should be automatic syntax highlighting for any code that is written
5. There should be verbose debugging for erroneous code
    a. This could mean creating a translator for converting assembly code written in the text field into a language which can display the errors appropriately
6. Have an option for users to upload a text file
    a. This should automatically be converted into the appropriate format and put into the text field where the assembly code should go
7. Should be able to view the instruction set on the same main page
    a. Add a link on the main page that goes to a page with full descriptions and explanations of each component of the resource and the A-Level Instruction Set
8. Show a verbose output of step-by-step details of each process being run
    a. This should include an indicator of which part of the FDE cycle is being run – including the processes with the 5 aforementioned Dedicated Registers
9. Create a list of coding challenges that can be solved by converting code (in Python) into assembly code
10. Create an introductory tutorial for new users of the resource
    a. This means showing the basic features of each component with a short description and how to use each
11. Accounts
    a. Student account
        i. Able to view and complete challenges (without a teacher setting them)
        ii. Able to then self-mark challenges and submit
        iii. Able to view a log of their previously completed challenges
        iv. Able to view a log of previously saved code (including non-challenges)
        v. Cannot view other students' solutions to challenges unless "published"
        vi. Can create their own challenges
            1. Type the instructions
            2. Display an example of the algorithm in Python
            3. Define shown and hidden test-cases
            4. Able to "upvote" student-created challenges after testing them
            5. 5 or more "upvotes" to be pushed to the global list of challenges
    b. Teacher account
        i. Teachers have classes which are groups of students
        ii. Can set challenges to be completed by all students in a class
        iii. Can view students' submitted challenges
        iv. Can view students' challenges even if not submitted
        v. Can "publish" students' solutions to allow other students to view them
        vi. Cannot view students' personal saved code (non-challenges)
        vii. Can create their own challenges

1. Type the instructions
2. Display an example of the algorithm in Python
3. Define shown and hidden test-cases

c. No account
   i. Can still view visualisations and code in assembly
   ii. Able to view and complete and self-mark challenges, but don't have a log of previously completed ones
   iii. Cannot submit challenges to a teacher – no linked account
   iv. Cannot see a log of previously saved code

At this point, looking back on the latest overall requirements, I realised that I had forgotten to discuss one crucial aspect of the system in detail (it has only been briefly considered). That is the actual sign-up and log-in for the students and teachers. Consequently, I have gone into further details about these two things below.

Sign-up and Login

Basic Idea and Understanding

As the resource stands currently, there are two kinds of accounts: a student account and a teacher account, as well as one more option, which is no account. For the purposes of this section, I will only be discussing the student and teacher accounts, as having no account doesn't really affect or impact sign-ups and/or logins.

Analysis of Existing System

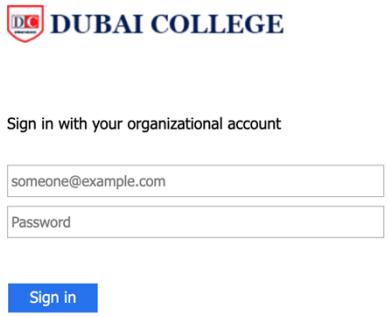

Figure 3.19 – Dubai College email login page

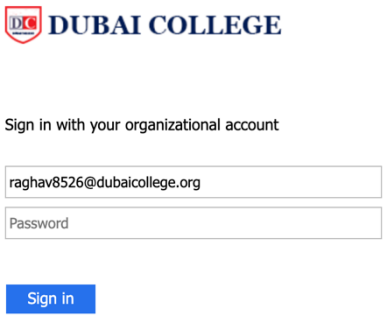

Figure 3.20 – Example of a Dubai College student email address

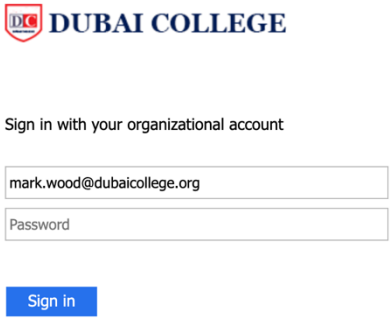

Figure 3.21 – Example of a Dubai College teacher email address

Figure 3.19, Figure 3.20, and Figure 3.21 show the basic inputs that are required to login into a teacher or student's school email, as well as their respective email addresses. We can see that the common feature in both email addresses is the "@dubaicollege.org", which is a required field to identify the person as part of the Dubai College community. As this resource is being specifically developed for Dubai College, this means that when a person tries to sign-up for the website, their email ID needs to end in "@dubaicollege.org".

Additionally, a student's email is their forename, followed by their student ID (also known as their student number), in the format "forenamesudentID@dubaicollege.org". As shown in the example above, my name is "Raghav Awasthi", and my student ID is "8526", so the first part of my email is "raghav8526", and altogether forms my email ID "raghav8526@dubaicollege.org".

For a member of staff, their email is their forename, followed by a ".", followed by their surname, in the format "forename.surname@dubaicollege.org". As shown in the example above, my client's name (who is also a teacher at the school) is "Mark Wood", so the first part of his email is "mark.wood", and altogether forms his email ID "mark.wood@dubaicollege.org".

Another thing that should be mentioned is that the email IDs are case-insensitive, so "raghav8526@dubaicollege.org" is the same email ID as "Raghav8526@dubaicollege.org". Using the formatting of emails mentioned in the previous paragraphs, the website for sign-ups will do some validation – on the teacher and student pages respectively – and check if the email entered is a valid possibility. If it is, then and only then will the website allow the user to sign-up to the page. Of course, all of the other information has to be entered in order to sign-up, so these will all be required fields.

Due to the fact that there are two types of accounts that can be created and used, the sign-up sheet should have the option to select either a "teacher" or "student" account option. Furthermore, for logging into the website, there should be two similar login pages – one for students, and one for teachers. These should also have an option to switch to the other type of login. Moreover, there should be a "Forgot Password" option, in case a teacher or student has forgotten their login password.

When a student is signing up for the resource, they should also be able to enter their year group, and because they will be assigned to classes, they should also be able to enter their teacher's name, so that the website can automatically assign students who have signed up into their respective classes. In order to make this robust and easy for students during the sign-up, the year group and teacher option should be a drop-down list. For the teacher option, the only choices that will be displayed are the teachers who have already signed up, so this will be drawn from the database of teachers.

One more thing is that neither the student nor the teacher needs to enter in their full name – a student can be uniquely identified using the first part of their email ID – their first name and student number are used for this, and this should be easy to do in the backend because I doubt that anyone's name will include a number in it, so it should be easy to break up everything before the "@" into "first name" and "student number". This process should be much easier for a teacher because their email ID contains their first name and last name before the "@", and they are separated by a ".".

Using all of the information mentioned above, I decided that it would be best to prototype the sign-up and login pages and draw an entity-relationship diagram in order to get a better understanding of how everything should look and link together in the front end and the back end. These are shown below.

Prototypes

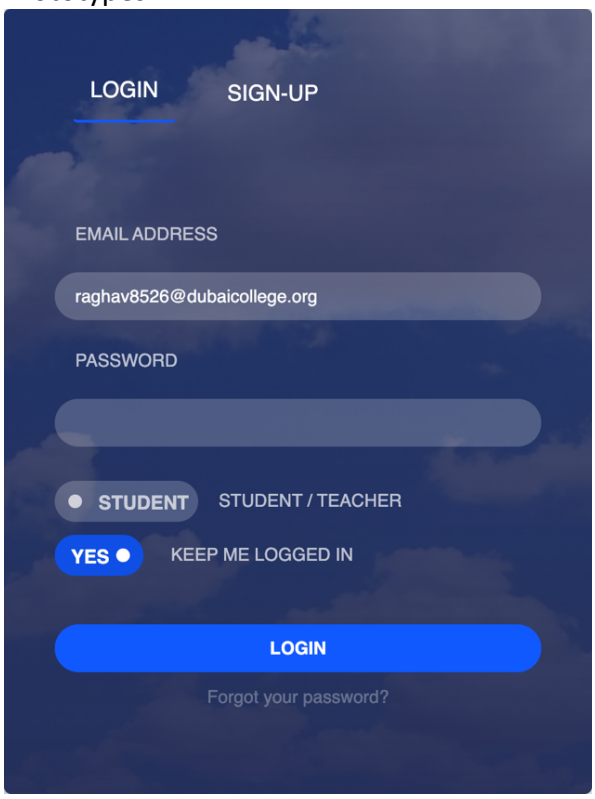

Figure 3.22 – Login page prototype

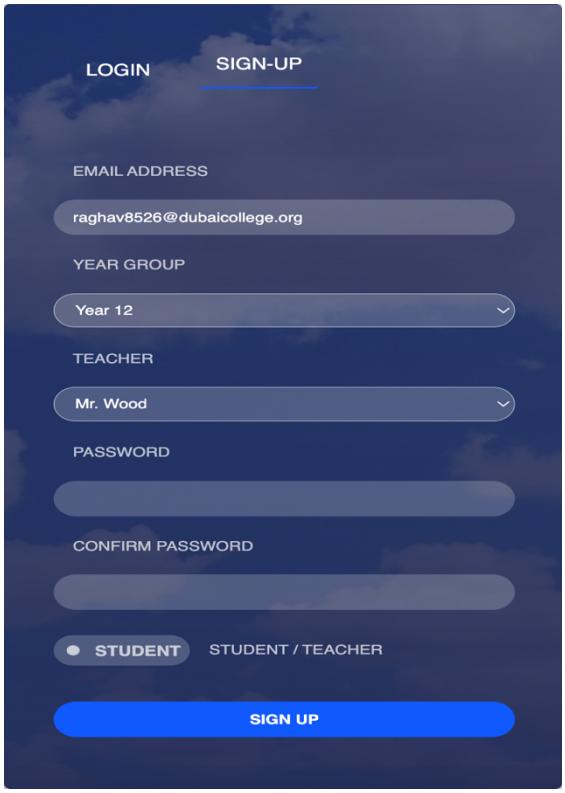

Figure 3.23 – Sign-up page prototype

Entity-Relationship Diagram

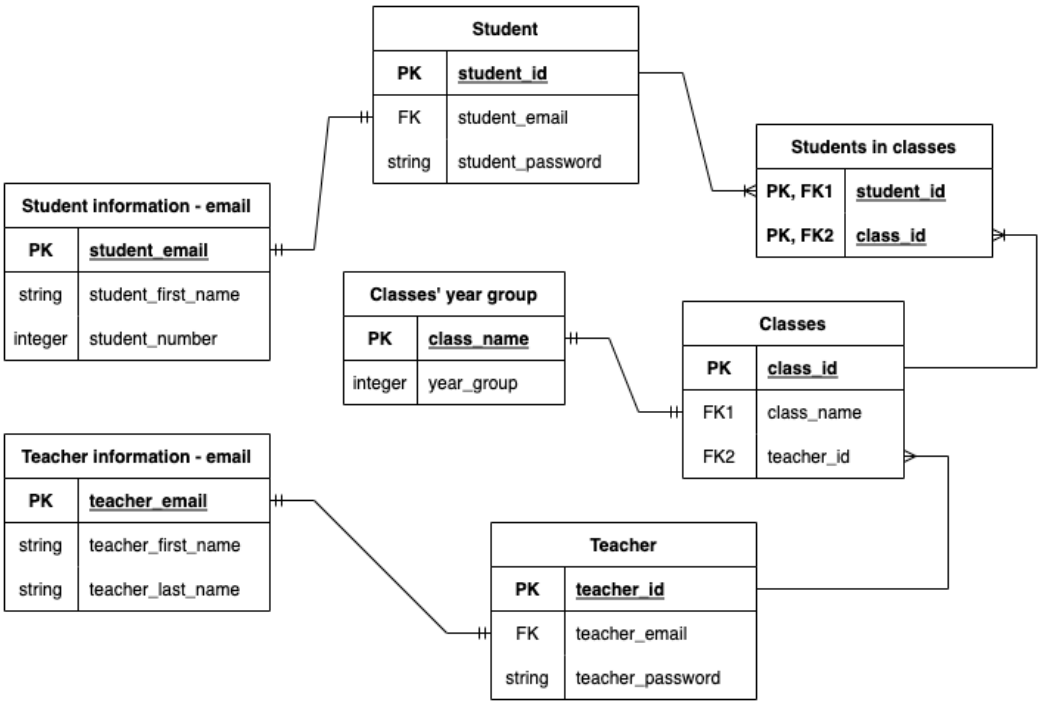

Figure 3.24 – Entity-relationship diagram for students and teachers, after sign-up

Figure 3.24 shows a basic entity-relationship diagram of the tables in the database that should be created and updated after a user signs up for the website. I have outlined the reasons for splitting up the tables the way that I did and the reasons for choosing certain primary and foreign keys below.

The reason that student number was not used as the primary key to uniquely identify students is because the school reuses these numbers after students leave the school, so there could be a case where a student with the same first name is assigned the same student number, hence resulting in a duplicate that may already potentially be in the database. There is only one teacher per class, and there is also only one class per teacher per year, so, for example, Mr. Wood cannot teach class 10B and 10C, they must have different teachers, and they will only have one teacher. In this example, that means that Mr. Wood may be the teacher for 10B, while Mrs. Chandler is the teacher for 10C. The reason that there is a “classes’ year group” table is because, when the user enters a year group and a teacher to sign up, the table “classes” will contain all the necessary information, and this can be backtracked to the “classes’ year group” table in order to find the name of the class.

In order to get the database into 3rd Normal Form, each field in a table must be fully dependent on only the primary key, and the year group is dependent on the class name, as the integer part of the class name is the year group (i.e., class 10C is year 10). The same goes for the other

tables. There must also be an intermediate table “Students in classes”, because otherwise, there would be a many-to-many relationship between the “Student” and the “Classes” tables, which is not allowed, and that also unnecessarily repeats data.

Having prototyped the login and sign-up pages, as well as having drawn an entity-relationship diagram, I thought that it would be best to revisit any other details or features that were mentioned previously, and explore them in more detail, so that I could determine whether those features are reasonable or not. Hence, the next section is the Scope section, where I have discussed everything that I was still unsure about in further detail.

Scope

Teacher-Created Challenges

Overview

To recap, this is the idea that teachers are able to define their own challenges, which means being able to write the code in Python and in Assembly, typing out the instructions, and defining test-values and hidden test-cases. Once these teacher-created challenges are created, they would be stored in the database, and a teacher would always be able to access this. If a teacher now wants to assign this challenge that they've created, they would be able to simply click an "assign" button somewhere, which shows a list of the classes that the teacher teaches, and they would then be able to simply select the classes that they want to assign the challenge to.

Discussion and Conclusion

The main point of contention with this idea is that teachers are actually supposed to be marking the "logic" of a student's solution, rather than just simply if their solution works or not. The basic overview defined above doesn't allow a teacher to actually mark the logic, as the teacher would simply see if the challenge was completed or not. This idea could therefore perhaps be altered so that, along with whether the student has passed the test-cases or not, the teacher can also actually see the code. This would mean that the teacher would be able to look at the way that the student has solved the problem in the Assembly Language and determine whether the "logic" and processes are correct or not. Then perhaps there could also be a feature to "return" or "mark" the submitted challenges.

However, as I'm writing about this, I am realising that for just the simple idea of a teacher-created challenge, there would be many additional features that would need to be created. This could take a lot of time to make, and this feature anyways isn't an essential part of the resource. Moreover, I think that teachers may just not end up using this feature, as it would take quite a bit of work. One would need to first determine what the challenge is; then write it up with an explanation; then write it in Python; then write it in Assembly; then define the test-cases and so on. This would take way too much time for teachers to actually use this feature. Therefore, I believe that this is out of scope.

Student-Created Challenges

Overview

This is similar to the previous one in the sense that it allows a user to create their own challenge and state the correct answers for given and hidden test-cases. However, this differs in the fact that student-created challenges would have to be "verified", unlike teacher-created challenges. For student-created challenges, at least 5 or more people would be required to "upvote" the challenge, which means that they have completely tested the challenge to make

sure that it works without any bugs. If a challenge is “verified”, then it can be pushed to the database of all challenges.

Discussion and Conclusion

As this is quite similar to the teacher-created solutions, I foresee the same complications as before. Furthermore, I would also need to create a system to allow for “upvotes”, which in itself adds many more complexities, and could also require a lot more data to be stored within the database. Having mentioned databases, another thing is that each additional challenge would take more space in the database, and, considering the limited resources I have available to me for database space, I think it may be better to also omit this feature from the resource.

“Publishing” Students’ Solution

Overview

This is the feature where a teacher is able to click on a “publish” button, which pushes a student’s completed challenge code to the “public” space, where any other student in the same class can view the student’s completed challenge, along with the code they wrote.

Discussion and Conclusion

Similar to the teacher-created challenges, this again only helps students see how to write the Assembly code, rather than understand the logic behind what was written. Although this feature could potentially help students in finding better ways of actually coding in the Assembly Language, which is also one use for this resource, I think that, due to the limited time restrictions of this project, this doesn’t seem like a reasonable feature to add because again, many new complexities would have to be introduced for this, such as creating a “dashboard” of sorts or a “public” space for each class, where each student belonging to the class could communicate. In addition, sharing code should not be a priority, as students should be trying to understand the logic and reasoning behind the code, rather than just copy-pasting the code, in order to see the visualisations. Hence, I believe that this feature is also outside of the scope of this project.

Creating a Translator

Overview

In order to have some sort of verbose debugging for the visualisations and for the Assembly code entered into the text field, it I needed to find a way of converting the entered code into the same code as my backend, and then using that backend code to debug the new code in the language at the backend, and finally convert any bugs or errors, back into output that can be displayed to the users.

Discussion and Conclusion

A translator seems to be the most appropriate way of converting entered code from the website’s frontend into code that can be processed at the backend, and then redirected back to

the output of the visualisation, in order to show verbose output of any errors. The fact that the two existing solutions that I looked at did not have this feature just adds to the fact that this is a necessary feature for the resource that I am building. Furthermore, by creating a translator, it should also be a lot easier to code the visualisations, as well as the syntax highlighting. Therefore, building a translator has to be within the scope of this project.

Having explored all of the ideas that were previously tentative, I now know exactly which features will and will not be – or need to be – included in the resource. Hence, I have compiled all of the features discussed in this Scope section (as well as previously discussed features) into a table which delineates whether a feature is in-scope or out-of-scope. This has been done below.

Summary

IN SCOPE	OUT OF SCOPE
Account system for students / teachers	Teacher-created challenges
Creating a Translator	Student-created challenges
Writing Assembly code to view visualisations	“Publishing” students’ solutions
Automatic syntax highlighting	Native application support
Verbose debugging / error logging	
Uploading text files	
Introductory tutorial	
Challenges	

Table 3.1 – Features and whether they’re in or out of scope

Having now formalised what is to be included and what is not to be included for the resource in Table 3.1, I need to actually investigate how I am going to code everything. Hence, I have explored various technologies below, in an attempt to determine which technologies would be most beneficial and the best fit for my project.

Exploring Technologies

Frontend

Overview

Due to the fact that the resource has to be web-based and cannot be a native application on different Operating Systems, it cannot be created with Swift or Objective-C for iOS, macOS or other Apple Operating Systems' development, and it cannot be developed using Java for Android devices.

General Necessary Technologies

As a result of requiring the resource to be a website, it is a given that HTML will be used for the basic layout of the webpages. Furthermore, CSS will have to be used in order to style the pages, and JavaScript must be used in order to make the pages interactive.

Additional Technologies

The intended end-users are school students and teachers, so, the user interface should be responsive and dynamic, but at the same time, it should also look relatively aesthetically pleasing. This can be accomplished using Bootstrap, a library that contains several stylesheets of predefined CSS options, which can be used to create a smooth and simple design for the web resource. Also, jQuery is a very useful JavaScript library in the sense that it makes it extremely easy and simple for users to manipulate HTML documents in the browser. jQuery is commonly used for professional websites in order to create responsive pages and greatly simplify a lot of the JavaScript that is required for the interactive interface of the website.

Backend

PHP

PHP is a very widely used option for web development, and almost all websites incorporate some form of PHP, whether that be to manipulate cookies and how they're stored, or whether that is for other purposes. PHP also has a massive online community, with a really solid overall documentation, so choosing PHP as the backend option means that I will always have some sort of support online in order to fix bugs, or research other features. PHP also incorporates really well with SQL databases, and, as this project requires a lot of data to be stored in and processed from the database, PHP would definitely be a sensible choice for the backend development of this web application.

One downside to PHP is that it is said to have a relatively difficult learning curve at the beginning, when compared to other backend frameworks that are built on top of languages such as Python. However, I have already had a little bit of experience coding in PHP as I have created a mini project using it, therefore, I do not believe that using PHP for this project will be too difficult. On the other hand, PHP is also not very secure due to its open-source nature, because the ASCII-text fields are readily available online. Moreover, PHP transmits data via the

GET Method, and even passwords that are entered are visible via this request method, which is not very secure. It is additionally widely believed that PHP offers very little when it comes to error handling, and it lacks debugging tools, which are needed to search for errors and/or warnings.

I think that having a backend that can easily identify errors and handle them well is an essential requirement for this resource, specifically because I will have to be able to display verbose output for any errors in the Assembly code entered by the user, which will require having a backend that can run that sort of code. Therefore, although PHP is advantageous when it comes to having well-written documentation, the fact that it doesn't have the best error-handling, and is not very secure, leads me to believe that it may not be the most suitable language to use for the backend of this resource.

Python – Flask

Flask is often described using the words "lightweight" and "minimal". This is because Flask was originally created to allow people with little experience to try and get a start in full-stack web development. It is widely documented, and being an open-source project, there are always people who have found fixes to any errors that one may encounter during the development stage. Being lightweight means that Flask has less dependencies than Django, and so will take up less storage – which is a limiting factor while developing this resource, as a lot of storage will be taken up by many other things such as the challenges. Furthermore, Flask also performs faster, and processes requests quicker than Django. Another major advantage of using Flask is the fact that it is a framework built on top of Python. Python is a very easy-to-use language, and I have a lot of experience with it, which makes Flask seem like an ideal framework to use.

However, a disadvantage with being lightweight is the fact that there aren't that many native features, and so, quite a few libraries and external sources and modules will have to be used in tandem with Flask. This means having to research how to integrate these libraries, and the libraries may have outdated dependencies, thus creating the possibility of vulnerabilities within the resource, as outdated versions may not be patched, and therefore will be exploitable. In addition, Flask doesn't have a very wide range of available libraries for different features, such as visualisations, for example. This being the case, I can foresee a plethora of problems when it comes to coding various features such as an interactive user interface, integrated with JavaScript at the frontend.

Python – Django

Like Flask, Django is a framework built on top of Python, which means that it will be relatively easy to learn. It is also has particularly well-written documentation, and, due to its large base of contributors (even larger than the number of contributors for Flask), it is extremely easy to ask for help for any specific errors or difficulty that one may run into. It therefore has more APIs in general than Flask. The Django framework is also scalable – it is based on loosely coupled

architecture, which provides it the functionality to run on almost any device's web browser. Django is also extremely secure – it was made by the world's best web developers and thus has a very small possibility of security loopholes in Django, even in the user authentication system. On top of that, any sensitive data such as passwords are automatically encrypted with a long security key.

Being such a heavy framework, however, Django is quite a bit slower than Flask when it experiences higher traffic. Furthermore, the framework is infamous for its monolithic nature – it takes an approach which is known as "convention over configuration", making the lives of developers extremely difficult when it comes to customisations and modifications to templates. Learning Django is also a much larger undertaking than it is for Flask as the former has got quite strict conventions, and all of them need to be followed.

Some additional information about Django is the fact that it is often considered an enterprise solution, which is good for larger companies that need to accommodate high traffic. It also includes a template system, built-in object-relational mapping, as well as a custom admin panel. It is a very powerful framework, and I get the feeling that it does way more than is needed for this project.

Overall, due to the monolithic nature, the steeper learning curve of Django than most of the other explored technologies, and the fact that it is much more powerful than what is required for this project, I will not be using Django for the backend of this resource.

NodeJS

NodeJS is a framework built on top of JavaScript, which is immediately a positive sign, as I will have to use JavaScript for the frontend anyways, and NodeJS integrates with this really well – it is more widely thought of as a full-stack framework, rather than just a backend framework. This also means that it has all of the pros of full-stack JavaScript development – better efficiency and overall developer productivity, as well as high speed and performance. Another great advantage of NodeJS is the fact that it supports myriad libraries, which are all extremely diverse and these libraries cover everything that I could possibly need for this resource, making this framework tremendously flexible. Furthermore, this doesn't have the same problem as Flask when it comes to outdated libraries and modules – NodeJS comes with an amazing packet manager (Node Packet Manager, often abbreviated to npm) which automatically manages all dependencies.

This technology is also faster than the other Python-based frameworks because it uses a Just-In-Time compiler, rather than a separate Python interpreter. Although NodeJS is relatively new, it has a very high quality of maintenance. Furthermore, this framework is ideal for real-time applications such as this visualisation resource because it is inherently asynchronous – it has an event-driven architecture and streaming modules. NodeJS is also a lightweight technology tool,

which makes it scalable as it uses the microservices architecture, rather than the monolithic architecture.

Although using NodeJS has many advantages, it also has its fair share of disadvantages. The first thing is that performance bottlenecks with heavy computational tasks. Luckily, the visualisations and other parts of this resource will not require extremely heavy computational tasks, so there is little reason to believe that this disadvantage will affect this resource if this framework is used. Another con is that debugging requires more effort – JavaScript doesn't have its own debugger without any dependencies. This could potentially be a problem while creating the translator in order to display verbose debugging output for the Assembly code for the user, however, due to the widely available range of libraries, I believe that there will be a relatively simple way of overcoming this obstacle. Though there are a multitude of available libraries, there are also several npm registries and libraries that are either of poor quality, or incomplete and not appropriately documented, hence I will have to be very careful about the libraries that I decide to use.

One more thing to note is that, although NodeJS is very useful for the processing aspect of the resource and the majority of the development process for a full-stack application, it is often used alongside Express JS or Angular JS and so, I will probably have to learn one of those two at some point as well. However, I will not know which one to use until I actually start the development process, and so, I have not included a discussion or overview of those two technologies in the section, because it will be much easier to see which one suits the requirements the best, when it comes to the point where I am required to choose between them.

Taking everything above into account, I believe that NodeJS is the best framework to use for this application, due to the wide variety of libraries, and the fact that it can be really easily integrated with the JavaScript at the frontend.

Database

MongoDB vs MySQL

Although I've had previous experience with MySQL, as I have created a mini project with it, integrated with PHP, various sources have led me to believe that MongoDB is also an extremely good database for projects and is one of the databases that – with drivers or interfaces – supports NodeJS and combines really well with it. MySQL can also be used in tandem with NodeJS. Both MongoDB and MySQL are also open-source projects, which means that there are lots of tutorials and well-written documentation for each. These are pretty much the only similarities between the two databases.

When it comes to how the database is actually structured, MongoDB is a NoSQL database, which means that, if required, different documents in a collection can have different structures

but also, a pre-defined structure for the incoming data can be defined and adhered to – it has a dynamic schema. MySQL, on the other hand, uses Structured Query Language (SQL) to access a database, and the schema cannot be changed – the inputs must follow the schema that is set out. For my project, having a structured database would probably be the best because there are quite a few places where a similar sort of structure is followed, and having pre-defined data types and data models means that it is slightly more secure. Therefore, MySQL may be better in this case.

Another difference is that MongoDB, originally designed with high availability and scalability in mind, includes features for replicating data, whereas MySQL doesn't allow efficient replication, but instead requires a user to access associated data using joins – and this minimises duplication, also saving storage in the process. Storage is a limiting factor for this project and so, I believe that MySQL will be better as I will be able to store less data more efficiently.

Overall, even though MongoDB comes with much more flexibility and is perhaps even slightly easier to use, I will be using MySQL for this project, because it is more efficient in terms of storing data using less storage space, and I also have some previous experience using it.

Translator

As mentioned during the cons list of Peter Higginson's Little Man Computer (above), I need to create a translator that can be used to convert the Assembly code into JavaScript (as I'm using NodeJS for the backend of the web application), which could easily help me identify errors within the assembly code that the user has written. I need to decide on whether to create a compiler or an interpreter.

An interpreter takes a single line of code and very little run-time to analyse it. On the other hand, a compiler takes an entire program and takes a comparatively long time to analyse it. However, once compiled, the compiled code can be run quickly, whereas interpreted code is rather slow to run. A compiler displays all the errors after compilation, and if the code has any mistakes, it won't compile. An interpreter, however, displays each error line by line – this may be more useful for this project as syntax highlighting will be made a lot easier if an interpreter is used, because each line can be analysed one at a time, thus making the verbose output a lot easier to code as well.

Noting the above details, I suppose that coding an interpreter may be the most advantageous, however, a Just-In-Time compiler is also a possible option. Nevertheless, I will continue the project assuming that I will be coding an interpreter, because I will not know if I need to change to a Just-In-Time compiler until I actually start coding and testing the resource.

Having explored all technologies and considerations, I think it would be best to have another meeting with my client to inform him of everything and so the final requirements can be finalised, and I can ensure that I am on the right track.

Interview 3 with Client

Before anything else, I think it is a good idea to list what the requirements look like up to this point.

Requirements So Far

1. The resource should be a Web Application which is coded using NodeJS as the framework and MySQL as the database.
2. Users should be allowed to write code in the Assembly Language and be able to run the machine-code operations which do the following, as defined in the AQA A-Level specification:
    a. Load
    b. Add
    c. Subtract
    d. Branching (conditional & unconditional)
    e. Compare
    f. Logical Bitwise Operators:
        i. AND
        ii. OR
        iii. NOT
        iv. XOR
    g. Logical:
        i. Shift Right
        ii. Shift Left
    h. Halt
3. Users should be able to view visualisations of the assembly code that they have written, with the following components included in the visualisation (being smoothly changed and animated on the website):
    a. Arithmetic Logic Unit
    b. Control Unit
    c. Clock
    d. General-Purpose Registers
    e. Buses
    f. Dedicated Registers:
        i. Program Counter
        ii. Current Instruction Register
        iii. Memory Address Register

        iv. Memory Buffer Register
        v. Status Register
4. There should be automatic syntax highlighting for any code that is written in the Assembly Language text field – this will be updated every time the code is changed.
5. There should be verbose output which displays any and all debugging for erroneous Assembly code that is written.
    a. This means creating a translator (interpreter) for converting Assembly code written in the Assembly Language text field into JavaScript, which can then be outputted on the website in order to show what each error is, and where it has happened.
6. Users should have the option to upload text files (.txt), which contain Assembly Language instructions, using the same Instruction Set as the website.
    a. This should automatically be converted into the appropriate format for the Assembly Language text field, and should be able to run immediately, if the user decides to start the animations of the visualisation.
7. Users should be able to view the Instruction Set (refer to the Document Analysis section) on the same main page as the visualisations.
    a. There should be an additional link on the main page that goes to another page, which contains full descriptions and explanations of each component of the CPU that has been added (refer to requirement 3), as well as the A-Level Instruction Set (refer to requirement 2).
8. Users should be able to see step-by-step outputs of the details of each process being run, in relation to what is happening within the visualisation.
    a. This should include an indicator of which part of the FDE cycle is being run – including the processes with the 5 aforementioned Dedicated Registers (refer to requirement 3).
9. There should be a list of coding challenges that can be solved by converting the given Python code (along with instructions of what it does) into Assembly code.
    a. There should also be some predefined small challenges for which the solutions are available e.g., adding 2 numbers in the Assembly Language.
10. There should be an introductory tutorial for new users of the resource who have never visited the website.
    a. This means showing the basic features of each component with a short description and how to use each and what each of them does.
11. Accounts
    a. Student account
        i. Should be able to view and complete challenges (without a teacher having to set these challenges).
        ii. Should be able to then self-mark challenges and submit them to their teacher.
        iii. Should be able to view a log of their previously completed challenges.

iv. Should be able to view a log of their previously saved code (including non-challenges), which means creating an option to save this code.

b. Teacher account
   i. Teachers have classes which are groups of students and have access to view a list of their classes and the students within them.
   ii. Should be able to set challenges to be completed by all students in a class, or specific students.
   iii. Can view students' submitted challenges – there will be an indicator that the student has clicked on the submit button.
   iv. Can view students' challenges even if not submitted – there will be an indicator to notify the teacher that the student has not clicked the submit button.
   v. Should not be able to view students' personal saved code (non-challenges).

c. No account
   i. These users can still view visualisations and write code in the Assembly Language.
   ii. Should be able to view, complete, and self-mark challenges, but will not be able to view a log of previously completed challenges.
   iii. Cannot submit challenges to a teacher as they will have no assigned class and therefore no assigned teacher.
   iv. Will not be able to see a log of their previously saved code (non-challenges).

As for the conversation itself, this conversation was a lot longer than the other ones, so I have decided to omit the actual words spoken and simply summarise what we discussed.

Summary
Firstly, Mr. Wood informed me that the AQA Instruction Set may no longer be used, so, after a lengthy discussion, we decided that using my own instruction set would be acceptable if I make it easy for people to understand (this slightly alters requirement 7 from above and means that Figure 3.18 can be ignored).

Hence, I have also decided to remove the requirement for Logical Bitwise Operators and Left/Right Shift in the Assembly code as that will rarely ever be used, and it would also make the use of the visualisation much more complicated (this removes requirements 2f and 2g from above). At the same time, although I will still show the General-Purpose Registers, I will not explicitly allow users to use the registers as the process may become too complicated, so all data will simply be stored in RAM, and this will be explained on an explanation page (this removes requirement 3d from above).

We also discussed the syntax highlighting part of the Assembly code and decided that it isn't really a core feature that is super important, so we can consider it out-of-scope for this project, because it is an extremely complicated process (this removes requirement 4 from above).

The verbose output remains helpful for students to understand the processes; however, a translator does not explicitly need to be built for this purpose. Using JavaScript and other jQuery functions, such as window and event listeners, the visualisations can still be accomplished (this alters requirement 5a from above).

As mentioned before, we discussed the Instruction Set at length. Since I will no longer be using the AQA Instruction Set, the users will probably not have any familiarity with the Instruction Set, so, even though I have talked about the benefits of putting the instruction set on the same page as the visualisation and Assembly code text editor area, we realised that it may not make sense to have it on the same page because it can obstruct the view. Instead, we will be putting the Instruction Set on a different page with explanations to make it easier to understand (this alters requirement 7 from above further).

The longest part of our conversation, however, was about the challenges feature. We decided that most users without an account would probably only come to the website to see the visualisations themselves, and they would probably have their own challenges they want to try to solve, which means that there is no need for general challenges without an account. Hence, the main functionality for users without an account is simply to be able to write assembly code and view the visualisations.

When it comes to users with accounts, a similar thing applies for general challenges – they are not necessary, because students can find those on their own. Furthermore, there will anyways be a select few Assembly codes that users can load into the text area to understand how to use the resource and how Assembly code works. Therefore, we decided that the teachers need only be able to create a challenge – which means giving it a challenge title and a description, nothing else – and be able to assign these challenges to separate classes or students. Of course, the teachers also need to be able to view the students' solutions, but, as Mr. Wood said, there isn't a need to provide explicit feedback to each student because any common errors can be discussed in class, so the website doesn't require this functionality. On the student side, the only thing they need to be able to do are see their assigned challenges, the due dates, and be able to submit their solutions. How this is accomplished doesn't really matter, as long as it is relatively straightforward. Taking all of these into account, requirements 9 and 11 from above, respectively need to be altered.

Having discussed these requirements with Mr. Wood, explored the technologies to use, and gained a sharper understanding of what everything will be coded in and how it will link with each other, I can now fully detail the complete and final requirements.

Final Requirements

1. The resource should be a Web Application which is coded using NodeJS as the framework and MySQL as the database.
    a. The website and all of its components should be easy to use, navigate, and understand.
2. Users should be allowed to write code in the Assembly Language and be able to run the machine-code operations which do the following:
    a. Load
    b. Add
    c. Subtract
    d. Branching (conditional & unconditional)
    e. Compare
    f. Halt
3. Users should be able to view visualisations of the assembly code that they have written, with the following components included in the visualisation (being smoothly changed and animated on the website):
    a. Arithmetic Logic Unit
    b. Control Unit
    c. Clock
    d. Buses
    e. Dedicated Registers:
        i. Program Counter
        ii. Current Instruction Register
        iii. Memory Address Register
        iv. Memory Buffer Register
        v. Status Register
4. There should be verbose output which displays any and all debugging for erroneous Assembly code that is written.
    a. This means using jQuery event listeners for converting Assembly code written in the Assembly Language text field into JavaScript, which can then be outputted on the website in order to show what each error is, and where it has happened.
5. Users should have the option to upload text files (.txt), which contain Assembly Language instructions, using the same Instruction Set as the website.
    a. This should automatically be converted into the appropriate format for the Assembly Language text field, and should be able to run immediately, if the user decides to start the animations of the visualisation.
6. Users should be able to view the Instruction Set on a separate page from the visualisations.
    a. There should be an additional link on the main page that goes to another page, which contains full descriptions and explanations of each component of the CPU

that has been added (refer to requirement 3), as well as the Instruction Set (refer to requirement 2).

7. Users should be able to see step-by-step outputs of the details of each process being run, in relation to what is happening within the visualisation.
    a. This should include an indicator of which part of the FDE cycle is being run – including the processes with the 5 aforementioned Dedicated Registers (refer to requirement 3).
8. On the same page where users write their assembly code and view visualisations, there should be a dropdown list of pre-created Assembly codes that can be loaded into the text area, ready for visualisation.
9. There should be an introductory tutorial for new users of the resource who have never visited the website.
    a. This means showing the basic features of each component with a short description and how to use each and what each of them does.
10. Accounts – users should be able to login and sign up for a student or teacher account, depending on whether they are a student or teacher at Dubai College, using their Dubai College email address.
    a. Student account
        i. Should be able to view their assigned challenges and due dates (set by a teacher).
        ii. Should be able to submit these assigned challenges by pasting in their Assembly code into a text area.
        iii. Should be able to view a log of their previously completed challenges.
    b. Teacher account
        i. Teachers have classes which are groups of students.
        ii. Should be able to create and set challenges to be completed by all students in a class, or specific students.
        iii. Can view students' submitted challenge responses – there will be an indicator that the student has clicked on the submit button and the student can no longer edit it.
        iv. Should be able to view all logs of challenges and differ between completed and non-completed challenges.
    c. No account
        i. These users can still view visualisations and write code in the Assembly Language.
        ii. Cannot submit challenges to a teacher as they will have no assigned class and therefore no assigned teacher.

Now that I have finalised all of the requirements, I feel as though I have a much better understanding of how everything links together. I have therefore drawn a few diagrams and

models of how I think everything links together. This has been shown in the Modelling section below.

Modelling

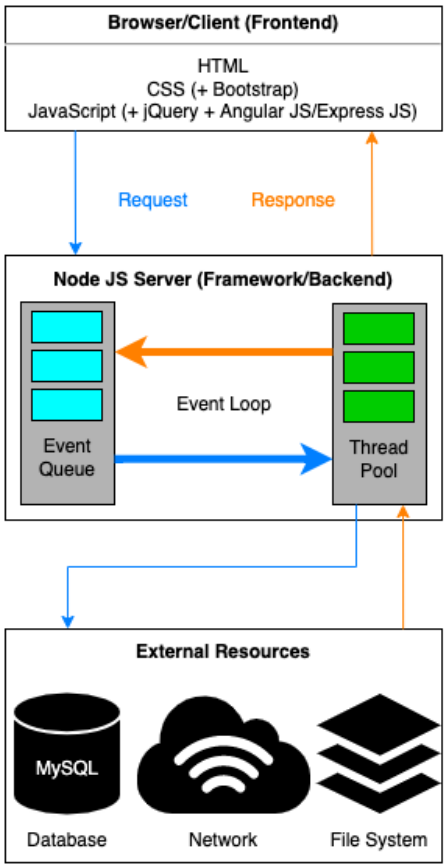

Figure 3.25 – Relationship between Technologies

Figure 3.25 to the left shows how the different technologies (that I've chosen to create this web application with) link together.
Every time the user makes a request on the browser – which could be entering in some input, clicking the mouse, or clicking a button – the request is sent to the NodeJS backend framework, so that the request can be processed. If the processing requires access to external resources, such as getting some records or data from the tables in the database, NodeJS handles the retrieval of this data from the necessary external resource. Once the data from the external resource is received again by the NodeJS framework, it processes the data in the necessary way, and then the output is sent as a response back to the client's browser, updating the page, without having to reload it.
Requests with websites use the HTTP GET and HTTP POST methods in order to request data and send a response. In order to update the pages, rather than the HTML files themselves being updated (as they are templates), the JavaScript instead handles the dynamic changing of components on the actual webpage that is displayed to the user.

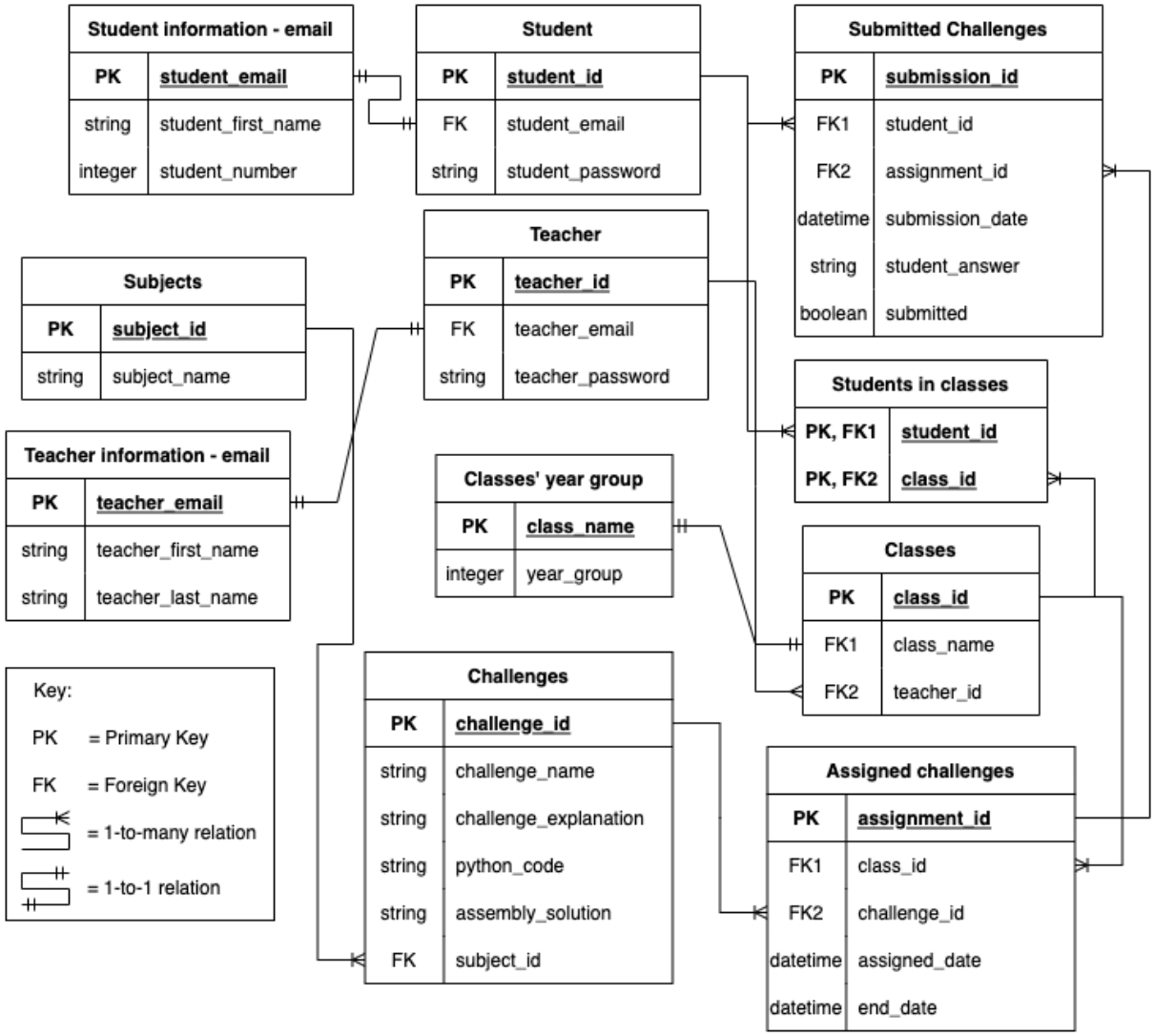

Figure 3.26 – Entity-Relationship Diagram for Overall Database

Figure 3.26 shows my basic preliminary understanding of what the database should look like for the entire project. I have created the above diagram based on all of the information that I have collated and evaluated during the entire Analysis section. However, as this is still just the initial understanding of how the database should look, I am sure that altercations may be made throughout the rest of this document, such as in the Design section.

Chapter 4 Documented Design

Overview

There are many parts to this web application that need to be created, designed, and prototyped, so I will first begin by going through an overview of the technology I have decided to use for this project, along with the default files created for the project, an explanation of each, and then slight alterations that have to be made to work with the main features of this project. After this, I will group the client requirements into a few different sections – as the main features – and will explore each of these in-depth after the exploration of the technology being used.

Technologies

Recalling what was spoken about in the Exploring Technologies section in the Analysis, I determined that the most suitable technology for the backend would be NodeJS, linked with MySQL, while the frontend would need to be JavaScript and HTML. In this section, I will explain the basic files when creating a new project with the Express JS template engine, explain some of the syntax that is used, and explore any additional libraries or other dependencies that I may need throughout the process of coding the project. However, it is important to remember that some libraries for very specific, small purposes may not be covered here, even if they are used in the final project, because these small things can sometimes only appear during the actual coding and cannot really be thought about beforehand. With regards to the libraries, I will be analysing the drawbacks to using these, as well as how I could potentially work around using them.

As previously mentioned, one of the biggest advantages of NodeJS is the Node Package Manner, better known as simply NPM, which provides extensive customisation opportunities. All one would really need to do is to look online at https://www.npmjs.com/ for packages that could seem useful, then run `npm install <package name>` in the terminal. Then, all that needs to be done is to read the documentation and understand how the package works. This could, however, be risky if the documentation isn't up-to-date or quite as readable as one would like – things may not be explained in the best way.

Regardless, using dependencies is far more feasible and practical than recreating and rewriting snippets of code that have already been implemented by someone else in a library. These are extremely useful for large features that would otherwise require extensive code, for features that require fast and efficient algorithms, and also for features that would basically be impossible to implement from scratch without using any dependencies, such as a strict web framework. On the other hand, the drawbacks of using libraries and modules must be considered. Most NPM packages are usually made for heavy customisation, and while this is good from a visual and quickly-writing-code perspective, it also means that the libraries may result in slower processing and thus, slower web pages. Also, though libraries do provide vast

customisation opportunities, the only way to completely customise every single detail of a website would be to do it all from scratch, rather than relying on a library. In addition, libraries also take a large amount of memory and disk storage space, and though this is not a big concern for now because I will be running the website locally and my device has enough storage, the website will eventually be hosted on servers, which may have additional costs for a certain amount of memory and disk space, so I should try to avoid extremely large libraries.

Link to Requirements
Recalling requirement 1:

1. The resource should be a Web Application which is coded using NodeJS as the framework and MySQL as the database.
    a. The website and all of its components should be easy to use, navigate, and understand.

Hence, I will now explore the different parts of NodeJS and Express.

Express JS

Express JS (or just Express) is a flexible web framework that provides robust features for web applications such as basic routing, template engines, middleware, and serving static files. It is a thin framework built on top of NodeJS, so it does not compromise the performance of the latter, which is extremely useful. However, one of the main potential drawbacks of using Express is the fact that there is a lack of standardisation for how to use it, and this may create more work for managing my code, unlike other frameworks such as Django. However, since there is a lack of standardisation, this also means that I would be quite free to arrange things in a way that makes sense to me, and since I will be using JavaScript for most of the backend processing anyways, the lack of standardisation is unlikely to cause major issues for me.

Middleware

Middleware are all NPM packages that have full access to the request and response methods and objects between the client and server and are therefore an essential part of NodeJS. They are responsible for processing the data between the client and server, before and after the request/response objects, and they often pass data through other middleware before passing a request object to a route handler or returning a response object to a client. NPM usually handles the installation of middleware and Express allows for seamless integration, so I will not go over the installation aspect of this.

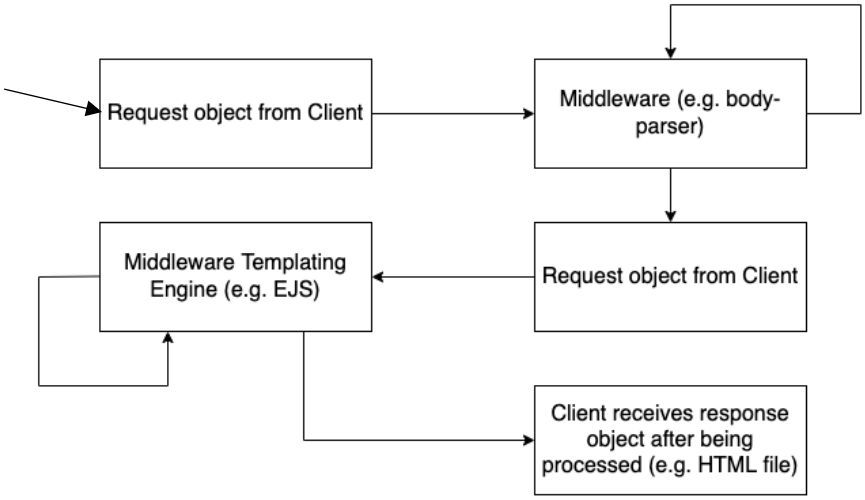

Figure 4.1 – Middleware flowchart

Figure 4.1 shows an overview of how middleware is used with Express. When the request object is sent from a web page, the middleware first pre-processes the data. The given example of body-parser essentially pre-processes HTML form data into a JSON format that can be read by the backend. After the backend routers and controllers handle the route and process the necessary data, the response object is returned to the client web page, and this object contains files that are incompatible with most browsers, e.g., EJS files. Hence, the middleware is used to post-process the data and convert it into simply HTML and JavaScript files, which are compatible with almost all browsers, and certainly with all well-known and widely used browsers.

Template Engine

Overview

Template engines allow backend variables to be embedded into HTML files. With appropriate middleware, template files are rendered to plain HTML format with variables fetched from the backend with their stored values. By default, Express uses Pug, but there are other templates like EJS and Moustache that integrate with Express.

Pug vs EJS

Though both Pug and EJS are equally useful, Pug's syntax is overall easier to read, due to the style of indentation that it uses. On the other hand, EJS has syntax that is extremely similar to HTML and is also a lot easier for a beginner to learn. EJS also allows the direct embedding of JavaScript code into the templates, while Pug does not. Pug, however, processes instructions slightly faster than EJS.

Taking the above into account, EJS would probably be the better choice to go with, simply because of the JavaScript embedding, since the main backend will be done in JavaScript, so this

makes the most sense. Also, the slight speed decrease in processing instructions should not make much of a difference here, since the processing is actually done by the backend, and I will mostly just need to be able to display the stored variables using the template engine.

Express Explanation

Routing

In general, Express works using 3 main things: routes, controllers, and views. Routes forwards requests (from the URL) to controller files; controllers are responsible for processing data, whether that includes the reading/writing of data from the database or getting a form object after being filled out in the frontend; and views are simply HTML or EJS files that act as the templates for basic routes.

This is slightly easier to see in a diagram:

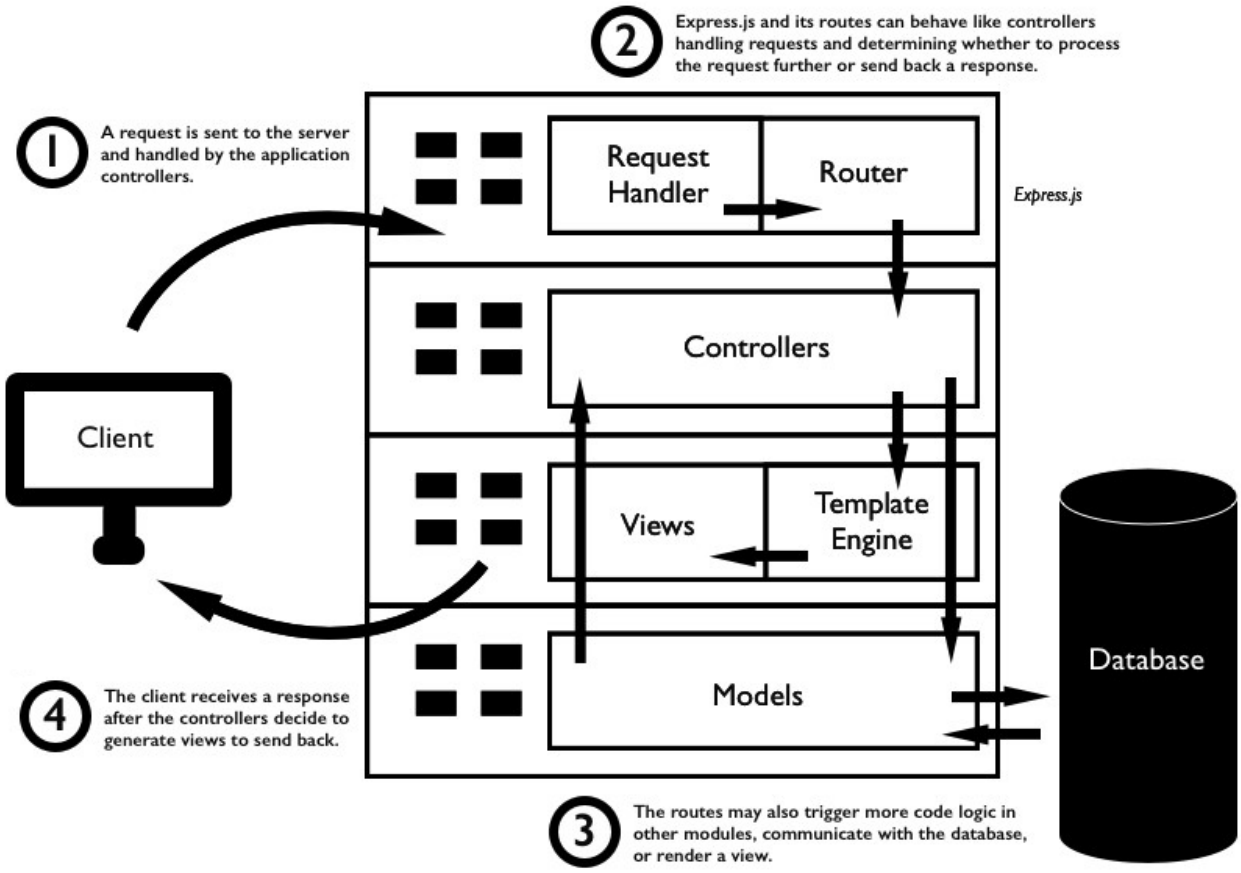

Figure 4.2 - Express Routing Flowchart (Manning Publications, 2018)

Though the logic of using Express and NodeJS works as above, the actual file structure is slightly different to what would be thought of as conventional or intuitive for the flowchart above.

File Structure

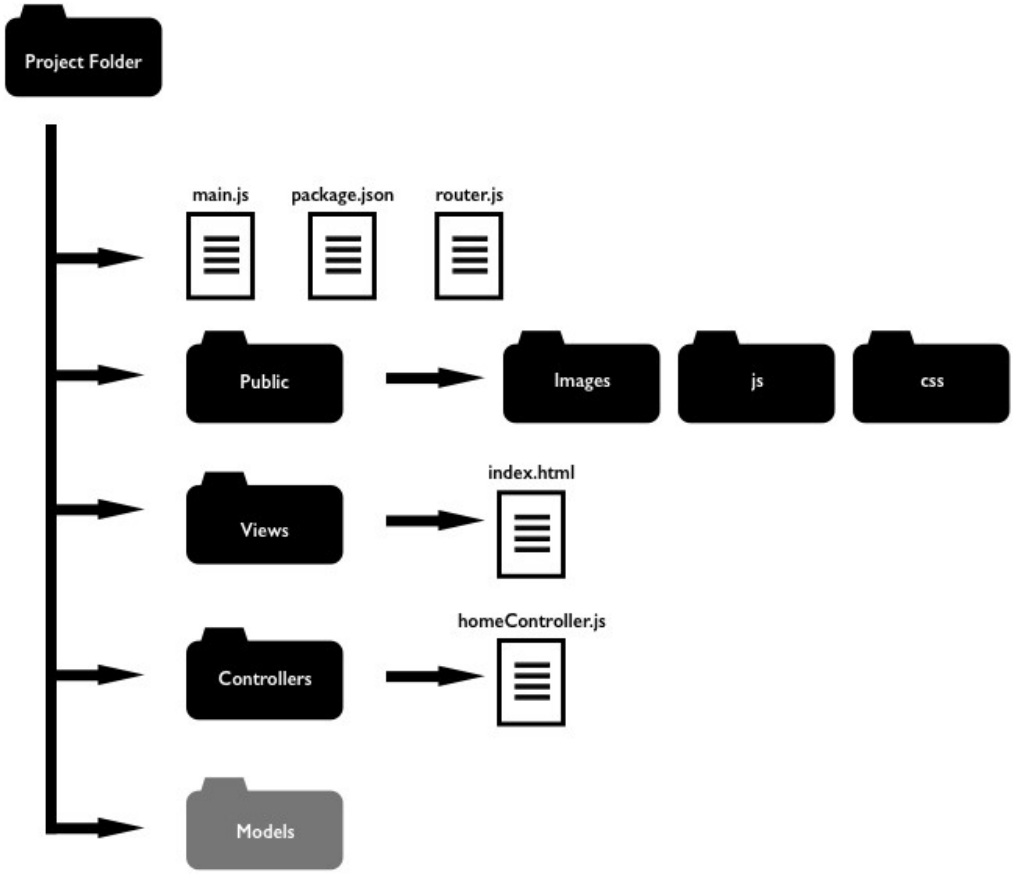

Figure 4.3 - File Structure of basic Express application (Manning Publications, 2018)

The diagram above shows us that firstly, we have a **main.js** file, which handles all of the libraries, packages, and modules that will be used by the application. It is used to set up and initialise the entire project, as well as connect things like databases and handle the routes for basic pages such as errors.

The **router.js** file simply links a URL or a URL directory to a specific controller within the **controllers/** folder and is essentially just a modularised file that is imported into the **main.js** file, since, when we run the application with `nodemon start`, the main.js file is really the only thing that runs, and then everything is linked from there. Note that I have called my **controllers/** directory **routes/**.

Note that although there aren't further sub-folders shown in the image above, my application will have many sub-folders for ease of access and to modularise the project.

The **public/** directory contains all static files, including images, JavaScript scripts for things such as a ToolTip (text that only appears when the mouse hovers over a certain area in the web

page), as well as stylesheets, because these aren't necessary for the website to function, and so are fine to be accessible by any website visitor, if needed.

The **views/** directory contains either HTML files or EJS files for the templates of the main pages. EJS files are essentially just HTML files, but they can contain additional tags, such as **<% %>**, **<%= %>**, and **<%- %>** which are used to run JavaScript directly in the pages or insert variables into certain places where they could be useful within the HTML template, for example in id attributes for tags.

I will now go through all of the important packages that I have already identified I will need to use, in order to make sure that everything connects between the backend and frontend, as well as to make sure that all the features of the website will work and be quite simple to do. For libraries that are obviously required, I will not go into too much detail about considerations of whether or not to use them, but for the others, I will consider alternatives to using the modules, as I mentioned previously.

Http Errors

This package allows the handling and generation of errors with NodeJS. It is quite easy to use and has a list of all status codes with error messages in the documentation and is almost always a required module when creating a NodeJS application, so I will be using it.

I will store the package in a **createError()** variable and will use have it running and listening to the localhost with the port chosen to run this locally (3000).
Size: 5.8 kB

Path

The Path module provides utilities to work with files and directory paths extensively. Once again, this is a required module if I want to use static files, which I will for images, JavaScript files and CSS files, so I will be using this.

The following line of code will be used to set up the **views/** and **public/** directories:

```
1. app.set('views', path.join(__dirname, 'views'));
2. app.use(express.static(path.join(__dirname, 'public')));
```

Size: 17.1kB

Cookie Parser

This populates **req.cookies** with an object keyed by the cookies' names. Signed cookies are also supported by passing a "secret" string through it and encrypting it, which assigns **req.secret** so it can be used by other middleware. These properties are name-value pairs for data. Once again, this is almost always required, and will be in order to check the sessions and ensure that cookies do not need to be refreshed, so I will be using it.

I will not be using the secret string with cookie-parser because I have instead used it with express-session, which is explained below. There is also no need to encrypt it twice since the website will only be used by Dubai College students.
Size: 3.6kB

Morgan

This package is a http request logger middleware for NodeJS. It is a great tool that logs requests along with some other information depending upon its configuration and the preset used.

The only thing that I will be using this for is so that I can have some output for developers in order to debug my code and view what processes are running and when, and to save these log files.
Size: 13.3kB

Express Flash

Express-Flash allows the use of flash messages with the application, which is especially helpful when considering displaying errors to the user – for example, with a failed login or sign-up attempt, we can use this to display an error message beneath the submit button, or with a successful attempt, we can display a success message. This is also extremely useful because it has the ability to define a flash message and render it without redirecting the request, hence I will be using this as well.

From now on, the following two kinds of lines will be used in the controller files for these flash messages:

```
1. req.flash('error', ' An error occured');
2. req.flash('success', ' A success occured');
```

Additionally, the following line is required to save the flash message and display it:

```
1. res.locals.message = req.flash();
```

Size: 1.1kB

Express-Session

This is middleware that allows sessions to be used with Express and store data, which can be accessed from any of the web pages. The way this stores data is similar to cookies on website (which is explained further on in this Documented Design section). I will be using it to store the different types of accounts.

Express-Session will be used with the following information and configurations (some of these are required fields):

- ⇒ **Secret:** Secret password used to secure the session.
- ⇒ **Cookies:**

- **Secure:** Boolean value that either allows or disallows using http (https is always allowed).
- **httpOnly:** Boolean value that, if enabled, prevents client-side JavaScript from reading cookies.
- **maxAge**: The amount of time, in milliseconds, that a user does not need to create a new cookie for – their cookie is saved. I will be using this in tandem with a "remember me" checkbox during the login process to handle how long a user stays signed in.

⇒ **Store:** Database that saves all cookies for each client – kind of like a client-cookie lookup table, but this does not require SQL. I will be using Redis and RedisStore, which I will explain after this section.

⇒ **SaveUninitialized:** Boolean value that determines whether a session should be stored or not for a user who is not logged in.

⇒ **Resave:** Boolean value that determines whether the session is refreshed automatically or note (i.e., user does not need to login again ever).

Size: 20.8kB

Redis (Redis Store, Connect Redis, and Redis Client)

Redis itself is an open-source, in-memory data store that can be used as a database, cache, and message broker. Its persistence (keeping the dataset in memory for fast access, but also persisting all writes to permanent storage to survive reboots and system failures) and programmability (server-side scripting and server-side stored procedures with Redis Functions) means that it is ideal to use for storing the cookies for the application.

I will install Redis using Docker, and this is required because RedisStore requires an existing Redis client to be running locally already, and from there, any clients compatible with the redis API will work (such as Connect Redis for Express).

To setup Redis, we simply need to create the client and then connect to it, which is done as follows:

```
var redisClient = createClient({
    legacyMode: true,
    host: REDIS_HOST,
    port: REDIS_PORT,
    database: REDIS_DATABASE
});

redisClient.connect().catch(console.error);
```

I have put temporary variables for the actual values, as these will be stored in the **.env** file (explained below), and depends on how Redis is setup through Docker itself.

Size: 242.9kB

Dotenv

Dotenv (or better known as .env) is a zero-dependency module that loads environment variables from a **.env** file into **process.env**. Storing these configurations in an environment separated from the code allows better security and also means that the application is further modularised. Once again, this is useful as I do not need to go through every file to check for constants, and instead can just store them in the **.env** file – this will be used mostly for the MySQL and Redis configurations, as well as any secret keys used.
Size: 1.6kB

Passport (and Passport Local)

This is an extremely useful authentication middleware for NodeJS. Being extremely flexible and modular, Passport can basically be used with any Express application.

Though I could use strategy support authenticators such as Google, Facebook, Twitter, or Outlook, I have decided to instead create a local strategy (hence, Passport Local), because this allows for exact customisability with the usernames and passwords. In other words, using a local strategy would allow me to ensure that a student user only has a student Dubai College email (as discussed in the Analysis section), and that a teacher user only has a teacher Dubai College email. For this, two different strategies have to be created, and thus imported into the initialisation in **app.js**.

Additionally, instead of using their username itself to identify the user, I will use their email, because that is already unique, as discussed in the Analysis section. Hence, the email and password will be stored in each cookie.

This package requires a lot of setup and configuration, and so, I will do this within **lib/** in a separate file. Essentially, queries need to be created to serialise and deserialise the user and data, and this needs to be done separately for each type of account. Since there are different tables within the database for Students and Teachers, these will therefore require different queries. The different queries are shown below with a comment explaining what they are each for:

```
1. 'SELECT * FROM Student WHERE student_email = ?;' // to deserialise and get the student user from the database
2. 'SELECT * FROM Teacher WHERE teacher_email = ?;' // to deserialise and get the teacher user from the database
```

For the serialisation of a user, we simply need to call the JavaScript **done()** function and pass in the necessary attributes. This is as follows:

```
1. passport.serializeUser(function(user, done) {
2.     process.nextTick(function() {
3.         if (user.role === 'student') {
4.             done(null, { id: user.student_id, email: user.student_email, role: user.role });
5.         } else {
6.             done(null, { id: user.teacher_id, email: user.teacher_email, role: user.role });
7.         }
```

```
        });
    });
```

Size: 1.6kB

MySQL

This is essentially a NodeJS driver for MySQL. As it is written in JavaScript, it does not require compiling, and therefore will run faster than other packages. The reason I have decided to use this is because I am familiar with using MySQL, so it will save me time, rather than learning how to use a new package like SQLite3, for example. I have to use some sort of SQL package in order to query the database, hence, I will be using this package.

The configuration for this package is quite simple, but I will still put it in as a separate file in the **lib/** directory. Essentially, all that needs to be done is to create a new connection and then connect to it as follows:

```
var connection = mysql.createConnection({
    host: DB_HOST,
    user: DB_USER,
    password: DB_PASSWORD,
    database: DB_DATABASE,
    port: DB_PORT
})

connection.connect();
```

Here, I have just put in placeholders for the actual values, but they will be stored using .env, and then put in here, whichever way MAMP is set up to run the database on my device (I believe it is run on port 8888).

Size: 271.9kB

Bcrypt

A library to help hash passwords, Bcrypt itself is a password-hashing function based on the Blowfish cipher. It incorporates a salt to protect against rainbow table attacks, and is an adaptive function, which means that over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks, even with increasing computation power. It is the default password hash algorithm for OpenBSD and some other Linux distributions, so it seems quite reasonable to use this to hash the passwords for the database.

Instead of using Bcrypt, I could potentially work on my own hashing algorithm to hash the passwords for the database. This would not be too complicated to do, however, I do not think that this is the best idea because it definitely will not be as secure as using the Bcrypt library. Hence, it makes sense to do this using a library.

We will be using 2 different Bcrypt functions through the application, the **hash()** function in order to actually hash the password and insert the hashed value into the database for the

password entry, as well as the **compare()** password, which will be used during sign-up to ensure that the entered password is the same as the password stored in the database (when not hashed). We do not have to worry too much about how this actually works, but just how to implement it. The two lines of code will be similar to the following:

```
var hashed_password = await bcrypt.hash(password, 10);
var are_same = await bcrypt.compare(password, rows[0].teacher_password);
```

Again, the password variable is just a placeholder in this case.

Notice the **await** keyword before calling the functions. This is done because hashing the password can take a while, so by using Async/Await (asynchronous processing), all processes that do not require these variables and work completely independently of these variables will be carried out, and the functions that do require these variables will only be run after the values are returned from the **await** variable/function.
Size: 143kB

Lodash

Lodash is a JavaScript library that provides utility function for common programming tasks. It uses the functional programming paradigm, and helps programmers write more concise and easier to maintain JavaScript code. Lodash contains tools to simplify programming with strings, numbers, arrays, functions, and objects. The reason that I will be using this library is that it will make processing arrays and objects a lot easier for me, and is therefore worth using, even though it is a relatively large library.

Though I could simply write more code to do the necessary processes with the arrays and objects (which would save more memory and be slightly more readable), using Lodash is far more efficient and reduces the time that I have to waste by repeating complex processes.
Size: 71kB

Moment

This JavaScript library allows parsing, validating, manipulating, and formatting dates. Since I have a feature that requires datetimes in the database (assigning challenges and setting deadlines), this is a required library. Though this is a legacy project, and there are more recent packages that may be more usable, I have previously used moment, and it is also a lot more extensively documented, which makes it easier to use for me.

Since there is a time difference from the database (GMT time) and my local time on which the application will be run for now (GMT+4:00), I need to consider this offset and then select from the database (because, weirdly, MySQL is fine while inserting into the database, but not while selecting from it).

The setup of moment and the offset is as follows:

```
1. moment.locale('en-gb');
2. moment.defaultFormat = 'L LT';
3. var utc_datetime = moment.utc().format();
4. moment(element['due_date']).utc(utc_datetime).local().format()
```

The 1st line above sets the locale to British English, which makes the dates in the format of DD/MM/YYYY, instead of the default American locale which formats the date as MM/DD/YYYY. The 2nd line is just my preferred way of displaying datetimes (e.g. 04/02/2023 2:59 PM), and the 4th line above shows how data will be offset when retrieving from the database (using the 3rd line's UTC datetime to find the offset).
Size: 60.6kB

Chosen

This library makes long select boxes far more user friendly, specifically select boxes where users are able to select multiple option.

Though I could do without using chosen, and simply customise the multi-select boxes using simple CSS, chosen makes the boxes much easier for users to use, and will be much more helpful for teachers when choosing which students to assign a challenge to, because chosen will allow the teachers to actually search the names of students and then select them. At the same time, it will also show a message if that student does not exist in the directory. This will make it much easier to navigate the students in classes, instead of having to scroll through a list of around 20 students and hope that they are able to find the correct student. Therefore, it makes sense to use this library.

This library does not really require configuration, apart from being **required** in (imported into) the application. All that needs to be done is that **chosen-toggle** should be added as a class to the multi-select box that needs to use the library.
Size: 182kB

Leader Line

This is a way of having dynamic arrows on a web page (https://anseki.github.io/leader-line/).

For the 3 main buses that will be shown in the main visualisation for the assembly code (the address bus, control bus, and data bus), I could simply draw shapes using JavaScript to show these buses. However, to show how data travels across these buses would be a significant task, as the positions (x-y coordinate values) on the screen would be different for each of the 100 RAM memory locations. Instead, using Leader Line, I could set the start points and endpoints of the Leader Lines to **div** tags in the HTML, which would automate the process of moving the arrows. So, instead of showing data/bits traveling across the buses, I will instead make the buses point to where the data is going. At the same time, I will also highlight the bus that is actively/currently being used to ensure that students are able to visualise absolutely everything.

Using this idea of moving the arrows instead of having moving data/bits means that I cannot have moving data/bits (drawn rectangles using JavaScript) for all the other components as well. Hence, a good alternative to this is to simply highlight the container and border of each component in the CPU when it is being used. This should still be relatively easy to follow, and if I include a text box within this container, the text box could be used to show the current value/data/bits being transferred across the main parts of the CPU.

The following lines of code will be used to create the 3 Leader Lines:

```
function create_buses() {

  address_bus = new LeaderLine(
    document.getElementById('MAR_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
      x: 0,
      y: 0
    }),
    {
      color: '#AAAAAA',
      size: 8,
      middleLabel: LeaderLine.pathLabel({
        text: 'Address Bus',
        color: 'black'
      }),
      path: 'straight',
      startSocket: 'right',
      endSocket: 'left',
      endPlug: 'arrow3'
    }
  );

  control_bus = new LeaderLine(
    document.getElementById('control_unit_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_50_wrapper'), {
      x: 0,
      y: 0
    }),
    {
      color: '#AAAAAA',
      size: 8,
      middleLabel: LeaderLine.pathLabel({
        text: 'Control Bus',
        color: 'black'
      }),
      path: 'straight',
      startSocket: 'right',
      endSocket: 'left',
      startPlug: 'arrow3',
      endPlug: 'arrow3'
    }
  );

```

```
  data_bus = new LeaderLine(
    document.getElementById('MBR_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
      x: 0,
      y: 0
    }),
    {
      color: '#AAAAAA',
      size: 8,
      middleLabel: LeaderLine.pathLabel({
        text: 'Data Bus',
        color: 'black'
      }),
      path: 'straight',
      startSocket: 'right',
      endSocket: 'left',
      startPlug: 'arrow3',
      endPlug: 'arrow3',
    }
  );

};
```

In order to update the Leader Lines, we can do things similar to the following:

```
address_bus.setOptions({
      end: LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
        x: 0,
        y: '23%'
      }),
      path: 'straight',
      color: '#AAAAAA'
    });
    control_bus.setOptions({
      color: '#AAAAAA'
    });
    data_bus.setOptions({
      end: LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
        x: 0,
        y: '86%'
      }),
      path: 'straight',
      color: '#AAAAAA'
    });
```

The above block of code just contains some examples of ways to update the options of the Leader Lines and make them appear dynamically on the screen. Since this code is located in a JavaScript partial file, the Leader Lines will automatically be updated if we put this inside a **document.ready()** function.

Size: 148kB

We can now partially tick off requirement 1, and I will now consider the actual user experience and landing page that should first show up.

User Experience

The landing page of the website should include the actual Little Man Computer simulation itself, where any user can write their assembly code. This is because the primary purpose of this application is to actually use it to see what happens within the CPU of a computer. On every page on the website, there should be easily accessible links to challenges and to a login/logout button (or signup button). This will perhaps be most easily accomplished using a navigation bar that appears at the top of the screen, with the appropriate names that send the user to the aforementioned links. Hopefully, this should make the web application very easy to use, even for new users. I will also include animations or transitions where possible in order to make the user experience seem seamless and dynamic.

Landing Page

On the landing page itself, the user will be able to click on a login or signup button from this page. They also need to be able to access an explanation of how the resource works. However, this will also be the main page for a user who doesn't have an account, as described in the Final Requirements in the Analysis section (above).

Navigation Menu

No Account

The navigation bar for a user who is not signed in could look something like the following:

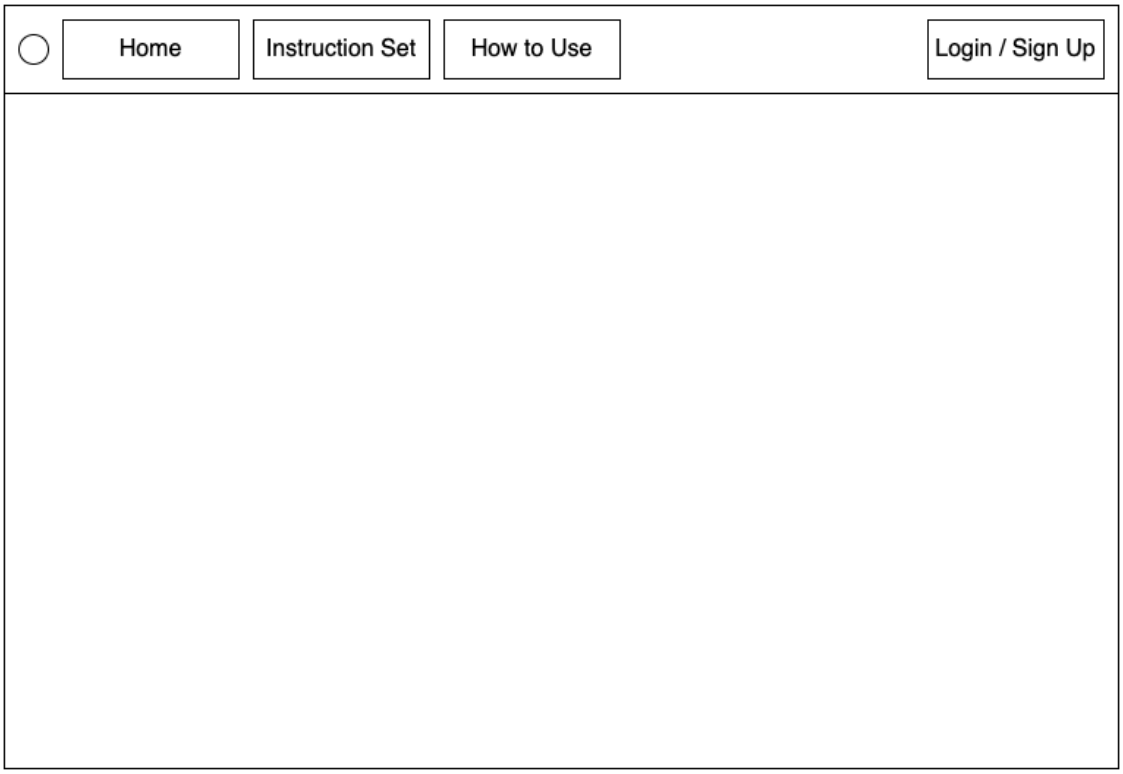

Figure 4.4 - No account Navigation Menu

The dot in the top left could be an indicator of whether the user is 1) not logged in, 2) a teacher, or 3) a student.

The colours could be as follows:

Type of Account	Colour
No account	Red
Teacher account	Yellow
Student account	Green

Table 4.1 – Table of colours for different accounts

Student Account

The navigation bar for a student account would probably be something like this:

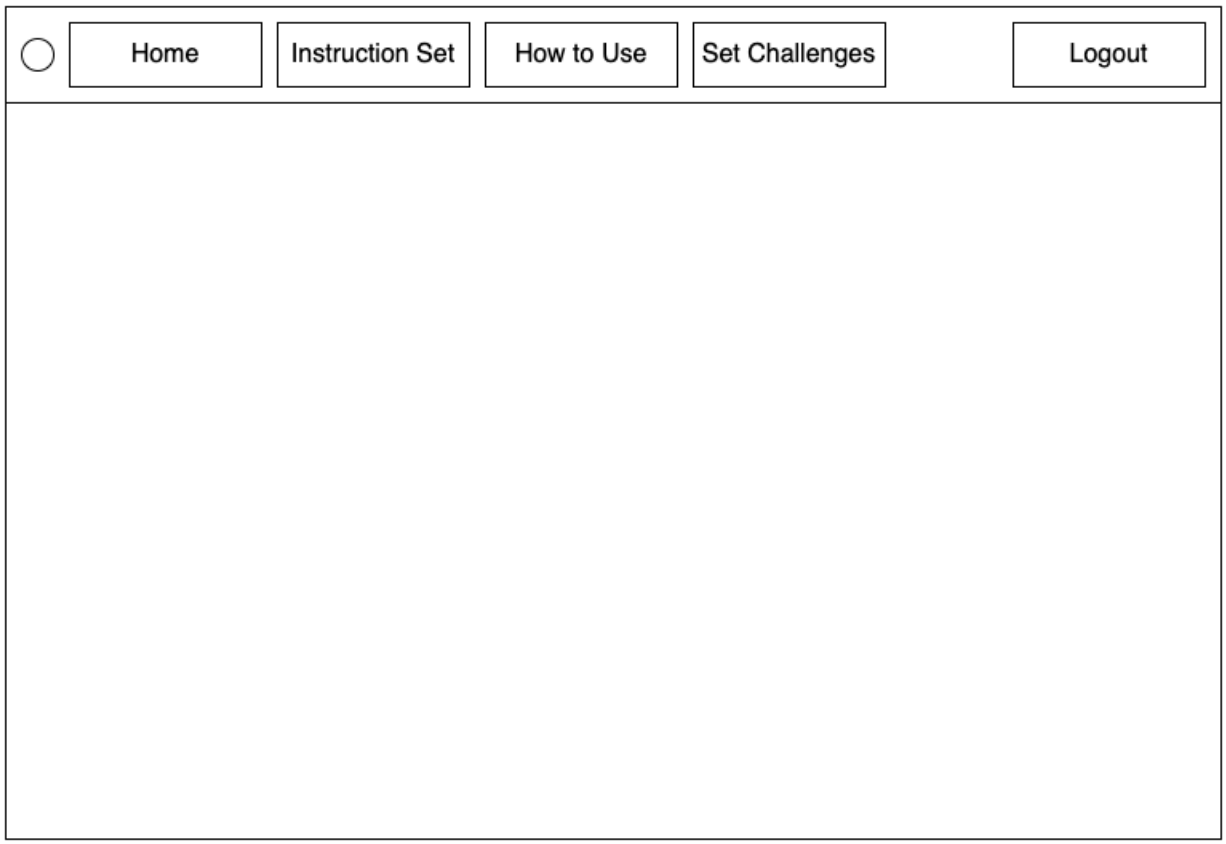

Figure 4.5 - Student account Navigation Menu

Obviously, a student needs to be signed in, so the button in the top right is now a logout button instead. The first three buttons from the left stay the same, while there is a new button to view the challenges and complete them as well.

Teacher Account

The difference between a teacher account and student account, apart from the colour of the indicator dot in the top left, is the fact that teachers need to create new challenges and assign new challenges (which can be done from the same page), and also need to view the students' challenges. Hence the navigation menu for teachers can look something like the following:

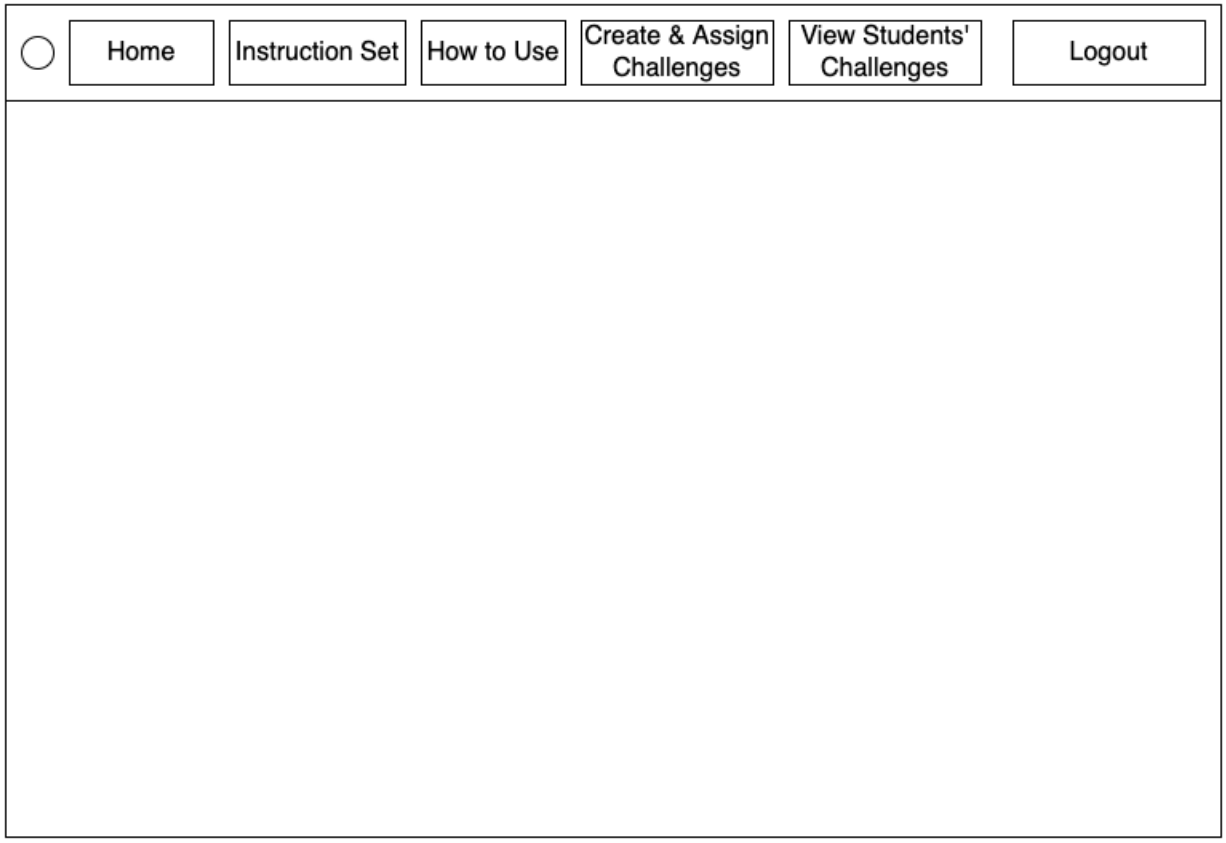

Figure 4.6 – Teacher account Navigation Menu

Once again, this may be subject to changes if, during the actual coding process, some new considerations appear or a different kind of idea would be a little simpler to code, but I still think that things will look relatively similar to this design.

Having considered what these will look like, we can now try to prototype a navigation menu (better known as a navbar in Bootstrap for HTML, which is likely what I will be using).

Prototype

```
// set active tab in the navbar automatically
$(document).ready(function () {
    var element = $('meta[name="active-menu"]').attr('content');
    $('#' + element).addClass('active');
});

```

```
// activate tooltips (the thing that shows up underneath the navbar user indicator)
$(document).ready(function () {
    $('[data-toggle="tooltip"]').tooltip();
});
```

The code above is a prototype for the tooltip feature that will be used to explain the coloured indicator in the top left of the navigation bar. Depending on the type of account, certain information will be shown. If the user is not logged in, only “red: no account” will be shown. If a student is logged in, the red information, as well as “green: student account” will be shown. If a teacher is logged in, all of the above will be shown, as well as “yellow: teacher account”. This will actually be implemented within the navbar code itself.

Having looked at the file structures for NodeJS already, I will be putting the navbar code as a partial file because it will be required in every single other page.

The navbar code could look similar to this:

```
<div class="row" id="navbar_row">
    <div class="col-md-12">
        <nav class="navbar navbar-default navbar-expand-sm navbar-dark bg-dark fixed-top">
            <div class="container-fluid navbar_wrapper">

                <!-- hidden form for POST method for logout tab in navbar -->
                <form id="logout_form" name="logout_form" action="/logout" method="post">
                    <button type="submit">LOG OUT</button>
                </form>

                <div class="navbar_left">
                    <ul class="navbar-nav nav_items_left">
                        <li>
                            <!-- coloured dot icon (3 diff colours for different possible accounts / no account) - needs to be coded -->
                            <!-- tooltip should change depending on account type - no account: red; teacher: red, yellow, green; student: red, green; -->
                            <span>
                                <% if (typeof role === "undefined" ) { %>
                                    <img class="tooltip-inner" id="navbar_user_indicator" src="/images/red_dot_indicator.png" alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account" />
                                <% } else { %>
                                    <% if (role === 'student') { %>
                                        <img class="tooltip-inner" id="navbar_user_indicator" src="/images/green_dot_indicator.png" alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account
green: student account" />
                                    <% } else { %>
                                        <img class="tooltip-inner" id="navbar_user_indicator" src="/images/yellow_dot_indicator.png" alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account
yellow: teacher account
green: student account" />
                                    <% } %>
                                <% } %>
                            </span>
                        </li>
                        <li>
```

```
                <span id="navbar_title">Assembly Code Visualiser</span>
            </li>
        </ul>
    </div>

    <div class="collapse navbar-collapse" id="main_navbar">

        <div class="navbar_left">
            <!-- main options on the left -->
            <ul class="navbar-nav nav_items_left" id="navbar_options">
                <li class="nav-item">
                    <a class="nav-link" id="home" href="/">HOME</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="instruction_set" href="/instruction_set">INSTRUCTION SET</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="how_to_use" href="/how_to_use">HOW TO USE THIS RESOURCE</a>
                </li>
                <% if (typeof role==="undefined" ) { %>
                <% } else { %>
                    <% if (role==='student' ) { %>
                        <li class="nav-item">
                            <a class="nav-link" id="view_set_challenges" href="/view_set_challenges">VIEW SET CHALLENGES</a>
                        </li>
                    <% } else { %>
                        <li class="nav-item">
                            <a class="nav-link" id="create_challenges" href="/create_challenges">CREATE CHALLENGES</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" id="view_student_challenges" href="/view_student_challenges">VIEW STUDENT CHALLENGES</a>
                        </li>
                    <% } %>
                <% } %>
            </ul>
        </div>

        <div class="navbar_right">
            <!-- login / sign up button on the right -->
            <ul class="navbar-nav nav_items_right">
                <% if (typeof role === "undefined" ) { %>
                    <li class="navbar-item">
                        <a class="nav-link navbar_right" id="login" href="/login">LOGIN</a>
                    </li>
                    <li class="navbar-item">
                        <a class="nav-link navbar_right" id="sign_up" href="/sign_up">SIGN UP</a>
                    </li>
                <% } else { %>
                    <li class="navbar-item">
                        <a class="nav-link navbar_right" href="#" onclick="post_logout()">LOG OUT</a>
                    </li>
                <% } %>
            </ul>
```

```
            </div>
        </div>

        <!-- collapsible menu w/ 3 horizontal lines for when screen too small -->
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#main_navbar"
          aria-controls="main_navbar" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>

      </div>
    </nav>
  </div>
</div>
```

There is some EJS included in this code, because the navbar will change depending on the type of account, and this is all the EJS code does – if-else statements to check whether there is an account type logged in, and if so, which kind. Rather than doing 3 separate files for the navbars, it is easier to simply do it in 1 file like this, and this way, I do not have to think about which kind of account each page will be accessible by and then link the specific partial file. Also, with 3 separate files, for the pages that are the same, it would be quite difficult to ensure that only 1 of the partial files is called at a time, so doing this in this way just makes things easier for me later on.

Having looked at the navigation menu and basic landing page idea, we can say that designing requirement 1a has been completed, and I will now look at the actual contents of the landing page – the main visualisation.

Little Man Computer Visualisation

Link to Requirements

Recalling requirements 3, 4, 5, 6, 7, and 9:

3. Users should be able to view visualisations of the assembly code that they have written, with the following components included in the visualisation (being smoothly changed and animated on the website):
    a. Arithmetic Logic Unit
    b. Control Unit
    c. Clock
    d. Buses
    e. Dedicated Registers:
        i. Program Counter
        ii. Current Instruction Register
        iii. Memory Address Register
        iv. Memory Buffer Register
        v. Status Register

4. There should be verbose output which displays any and all debugging for erroneous Assembly code that is written.
    a. This means using jQuery event listeners for converting Assembly code written in the Assembly Language text field into JavaScript, which can then be outputted on the website in order to show what each error is, and where it has happened.

5. Users should have the option to upload text files (.txt), which contain Assembly Language instructions, using the same Instruction Set as the website.
    a. This should automatically be converted into the appropriate format for the Assembly Language text field, and should be able to run immediately, if the user decides to start the animations of the visualisation.

6. Users should be able to view the Instruction Set on a separate page from the visualisations.
    a. There should be an additional link on the main page that goes to another page, which contains full descriptions and explanations of each component of the CPU that has been added (refer to requirement 3), as well as the Instruction Set (refer to requirement 2).

7. Users should be able to see step-by-step outputs of the details of each process being run, in relation to what is happening within the visualisation.
    a. This should include an indicator of which part of the FDE cycle is being run – including the processes with the 5 aforementioned Dedicated Registers (refer to requirement 3).

9. There should be an introductory tutorial for new users of the resource who have never visited the website.
    a. This means showing the basic features of each component with a short description and how to use each and what each of them does.

The requirements shown above clearly outline which parts of the CPU must be included within the visualisation, as well as any additional components that need to be shown with the necessary information. Taking the requirements into account, I have created a basic view of the layout for the visualisation.

Basic Template for Layout

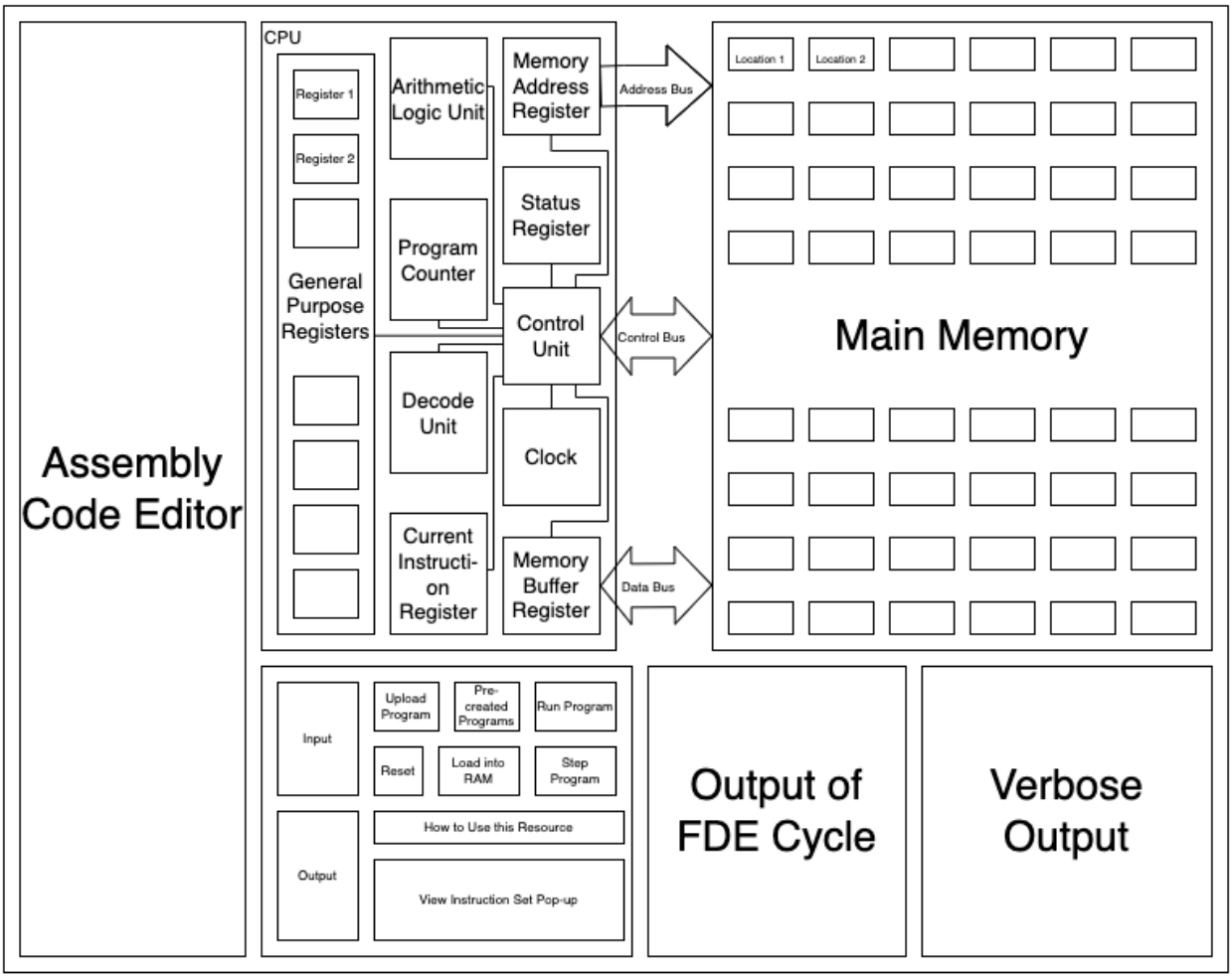

Figure 4.7 – Layout of Main Visualisation

The actual amount of memory locations and number of general-purpose registers cannot be determined right now – I will have to do this during the implementation phase so that I can see how much space I have left in the browser window, and thus, how many of each component I can add.

Requirement 3 is satisfied with the components located in the CPU section of the visualisation, while there is a verbose output field for requirement 4, and a button in the buttons section for requirement 5 – to be able to "upload" a text file from a user's computer (this will be discussed in further detail in the Assembly Code and Editor section below). Requirement 6 is also fulfilled with a button to "View Instruction Set Pop-up" – this will open up a new page which will display the instruction set. Requirement 7, like requirement 4, has its own field to show the output of the FDE Cycle (which processes are being run at which moments). Finally, there is a button called "How to Use this Resource" which will link to a new tab with an explanation of each

component on the visualisation, as well as a video created by myself which explains how to use the visualisation and each of its fields. This satisfies requirement 9.

All arrows and lines in the image above represent the buses, through which the data will travel. As discussed in the technologies section, I will no longer be showing objects for the data as they move, and instead simply highlighting the containers of each component in the CPU, so the smaller buses aren't required to be shown.

Something similar to the following HTML/EJS code can be used to show the main visualisation, while something like the following NodeJS code (I will label which one is which below) can be used to highlight the **div**s containing the required component.

Prototyped Code

Main Visualisation Layout

```
<div class="row">

  <div class="col-md-2">
    <span>Assembly Code Editor</span>
  </div>

  <div class="col-md-4">
    <span>  Processor</span>
  </div>

   <div class="col-md-1"></div>

   <div class="col-md-5">
     <span>RAM</span>
   </div>

</div>

<div class="row" id="main_visualisation_wrapper_row_1">

   <div class="col-md-2">
     <div id="code_wrapper">
       <div class="line-numbers col-md-2 scroll_together">
         <span></span>
       </div>
       <textarea class="scroll_together" id="code_area" rows=27 wrap="off"></textarea>
     </div>
   </div>

   <div class="col-md-4">
     <div id="CPU_wrapper">

       <div class="row">
         <div class="col-md-4">
           <span id="general_registers_title">General Purpose Registers</span>
         </div>
```

```
        </div>

        <div class="row">

          <div class="col-md-4" id="general_register_wrapper">
            <!-- create General Purpose Registers (from 0 to 9) -->
            <% for (var i = 0; i < 10; i++) { %>
              <div id="register_<%= i %>_wrapper">
                <tr>
                  <td>Register <%= i.toString() %><input readonly id="general_register_<%= i %>" type="text"
value="0000" /></td>
                </tr>
              </div>
            <% } %>
          </div>

          <div class="col-md-4" id="CPU_column_2">

            <div id="ALU_wrapper">
              Arithemtic Logic Unit
              <input readonly id="ALU" type="text" value="" />
            </div>

            <div id="accumulator_wrapper">
              Accumulator
              <input readonly id="accumulator" type="text" value="00" />
            </div>

            <div id="PC_wrapper">
              Program Counter
              <input readonly id="PC" type="text" value="00" />
            </div>

            <div id="decode_unit_wrapper">
              Decode Unit
              <input readonly id="decode_unit" type="text" value="" />
            </div>

            <div id="CIR_wrapper">
              Current Instruction Register
              <input readonly id="CIR" type="text" value="00" />
            </div>

          </div>

          <div class="col-md-4" id="CPU_column_3">

            <div id="MAR_wrapper">
              Memory Address Register
              <input readonly id="MAR" type="text" value="00" />
            </div>

            <div id="status_register_wrapper">
              Status Register
              <input readonly id="status_register" type="text" value="00" />
```

```
            </div>

            <div id="control_unit_wrapper">
              Control Unit
              <input readonly id="control_unit" type="text" value="" />
            </div>

            <div id="clock_wrapper">
              Clock (cycles)
               <input readonly id="clock" type="text" value="0" />
             </div>

             <div id="MBR_wrapper">
               Memory Buffer Register
               <input readonly id="MBR" type="text" value="0000" />
             </div>

           </div>

           </div>
      </div>
    </div>

    <div class="col-md-1"></div>

    <div class="col-md-5">
      <div id="RAM_wrapper">

        <table id="RAM">
          <!-- create RAM (memory locations from 00 to 99) -->
          <% for (var i = 0; i < 10; i++) { %>
            <tr>
              <% for (var j = 0; j < 10; j++) { %>

                <% var k = (i * 10) + j %>
                <% var k_str = k.toString() %>

                <%# for memory locations 0-9, we want to display 00, 01, etc. %>
                <% while (k_str.length < 2) { %>
                  <% k_str = '0' + k_str %>
                <% } %>

                <td id="memory_<%= k_str %>_wrapper"><%= k_str %><input id="memory_location_<%= k %>"
type="text" value="0000" /></td>

              <% } %>
            </tr>
          <% } %>
        </table>
      </div>

    </div>
</div>

<div class="row" id="second_title_text_row">
```

```

    <div class="col-md-6">
      <span>
        <div class="row">
          <div class="col-md-5">User Controlbox for Visualisation</div>
          <div class="col-md-7"><input type="range" id="clock_speed" min="5" max="5000" value="25"></div>
        </div>
      </span>
    </div>

    <!-- log outputs -->
    <div class="col-md-3">
      <span>FDE Cycle Output</span>
    </div>

    <div class="col-md-3" id="verbose_output_title">
      <span>Verbose Output</span>
    </div>

</div>

<div class="row" id="main_visualisation_wrapper_row_2">

    <div class="col-md-6" id="controlbox_wrapper">
      <div class="row">

        <div class="col-md-3">
          <div id="input_wrapper">
            <tr>
              <td>Input<input id="input" type="text" value="00" /></td>
            </tr>
          </div>
        </div>

        <div class="col-md-9" id="controlbox_top_two_rows_column_2">
          <div class="row" id="controlbox_row_1_column_2">

            <div class="col-md-3" id="upload_program_wrapper">
              <button id="upload_program"><label for="upload_program_actual">Upload Program</label></button>
            </div>

            <div class="col-md-3" id="reset_RAM_wrapper">
              <button id="reset_RAM" onclick="reset_RAM(LMC)">Reset RAM</button>
            </div>

            <div class="col-md-3" id="load_into_RAM_wrapper">
              <button id="load_into_RAM" onclick="load_into_RAM(LMC)">Load into RAM</button>
            </div>

            <div class="col-md-3" id="run_program_wrapper">
              <button id="run_program" onclick="run_program(LMC)">Run Program</button>
            </div>

          </div>

```

```
            <div class="row" id="controlbox_row_2_column_2">

              <div class="col-md-4" id="clear_editor_wrapper">
                <button id="clear_editor" onclick="clear_editor()">Clear Editor</button>
              </div>

              <div class="col-md-4" id="pre-created_programs_wrapper">
                <select id="pre-created_programs" name="pre-created_programs"></select>
              </div>

              <div class="col-md-4" id="step_program_wrapper">
                <button id="step_program" onclick="step_program(LMC)">Step Program</button>
              </div>

            </div>
          </div>

        </div>

        <div class="row">

          <div class="col-md-3">
            <div id="output_wrapper">
              <tr>
                <td>Output<input readonly id="output" type="text" value="00" /></td>
              </tr>
            </div>
          </div>

          <div class="col-md-9" id="controlbox_bottom_two_rows_column_2">

            <div class="row" id="how_to_use_this_resource_wrapper">
              <button id="how_to_use_this_resource" onclick="open_link_new_tab('how_to_use')">How to Use this Resource</button>
            </div>

            <div class="row" id="view_instruction_set_wrapper">
              <button id="view_instruction_set" onclick="open_link_new_tab('instruction_set')">View Instruction Set</button>
            </div>

          </div>

        </div>
      </div>

      <div class="col-md-3">
        <div id="FDE_output_wrapper">
          <textarea id="FDE_output" wrap="on">Output of FDE cycle</textarea>
        </div>
      </div>

      <div class="col-md-3">
        <div id="verbose_output_wrapper">
          <textarea id="verbose_output" wrap="on">Verbose Output</textarea>
```

```
        </div>
    </div>

</div>
```

The above HTML/EJS file creates something similar to the template shown above.

Div Highlighting Example

```
for (var i = 0; !resolve(); i++) {
  if (i == 0) {
    LMC.log_output('HLT');
    LMC.log_output('Executing instruction...');
  } else if (i == 1) {
    LMC.deactivate_wrapper('control_unit_wrapper');
    LMC.deactivate_wrapper('decode_unit_wrapper');
    control_bus.setOptions({
    color: '#AAAAAA'
     });
     LMC.log_output('Program stopped');
     LMC.cycles++;
   } else {
     LMC.log_output('####################');
     LMC.log_output('Program executed in ' + LMC.cycles + ' FDE cycles');
     LMC.paused = true;
     LMC.stop = true;
     document.getElementById('clock').value = LMC.cycles;
     LMC.deactivate_wrapper('clock_wrapper');
     LMC.cycles = 0;
     resolve();
     clearInterval(LMC.carry_on);
   };
}
```

This JavaScript is an example of the processes needed for the HALT instruction of the instruction set, as well as which **div**s need to be highlighted. The rest of the instruction set will be done in a similar fashion to this below.

Instruction Set Page

```
<table id="instruction_set_table">
  <caption>Above is an explanation of the Instruction Set for the LMC. This has been taken from <a
        href="https://en.wikipedia.org/wiki/Little_man_computer#Instructions" target="_blank">here</a>.</caption>
  <tbody>
     <tr>
        <th>Numeric code
        </th>
        <th>Mnemonic code
        </th>
         <th>Instruction
         </th>
         <th>Description
         </th>
      </tr>
      <tr>
```

```
      <td>1xx</td>
      <td>ADD</td>
      <td>ADD
      </td>
      <td>Add the value stored in mailbox xx to whatever value is currently on the accumulator (calculator).
        <dl>
          <dd>Note: the contents of the mailbox are not changed, and the actions of the accumulator
            (calculator) are not defined for add instructions that cause sums larger than 3 digits.
            Similarly to SUBTRACT, one could set the negative flag on overflow.</dd>
        </dl>
      </td>
    </tr>
    <tr>
      <td>2xx</td>
      <td>SUB</td>
      <td>SUBTRACT
      </td>
      <td>Subtract the value stored in mailbox xx from whatever value is currently on the accumulator
        (calculator).
        <dl>
          <dd>Note: the contents of the mailbox are not changed, and the actions of the accumulator are not
            defined for subtract instructions that cause negative results - however, a negative flag will be
            set so that <b>7xx (BRZ)</b> and <b>8xx (BRP)</b> can be used properly.</dd>
        </dl>
      </td>
    </tr>
    <tr>
      <td>3xx</td>
      <td>STA</td>
      <td>STORE
      </td>
      <td>Store the contents of the accumulator in mailbox xx (destructive).
        <dl>
          <dd>Note: the contents of the accumulator (calculator) are not changed (non-destructive), but
            contents of mailbox are replaced regardless of what was in there (destructive)</dd>
        </dl>
      </td>
    </tr>
    <tr>
      <td>5xx</td>
      <td>LDA</td>
      <td>LOAD
      </td>
      <td>Load the value from mailbox xx (non-destructive) and enter it in the accumulator (destructive).
      </td>
    </tr>
    <tr>
      <td>6xx</td>
      <td>BRA</td>
      <td>BRANCH (unconditional)</td>
      <td>Set the program counter to the given address (value xx). That is, the value in mailbox xx will be the
        next instruction executed.
      </td>
    </tr>
    <tr>
```

```
      <td>7xx</td>
      <td>BRZ</td>
      <td>BRANCH IF ZERO (conditional)
      </td>
      <td>If the accumulator (calculator) contains the value 000, set the program counter to the value xx.
        Otherwise, do nothing. Whether the negative flag is taken into account is undefined. When a SUBTRACT
        underflows the accumulator, this flag is set, after which the accumulator is undefined, potentially
        zero, causing behavior of BRZ to be undefined on underflow. Suggested behavior would be to branch if
        accumulator is zero and negative flag is not set.
        <dl>
          <dd>Note: since the program is stored in memory, data and program instructions all have the same
            address/location format.</dd>
        </dl>
      </td>
    </tr>
    <tr>
      <td>8xx</td>
      <td>BRP</td>
      <td>BRANCH IF POSITIVE (conditional)
      </td>
      <td>If the accumulator (calculator) is 0 or positive, set the program counter to the value xx. Otherwise, do
        nothing. As LMC memory cells can only hold values between 0 and 999, this instruction depends solely on
        the negative flag set by an underflow on SUBTRACT and potentially on an overflow on ADD (undefined).
        <dl>
          <dd>Note: since the program is stored in memory, data and program instructions all have the same
            address/location format.</dd>
        </dl>
      </td>
    </tr>
    <tr>
      <td>901</td>
      <td>INP</td>
      <td>INPUT
      </td>
      <td>Go to the INBOX, fetch the value from the user, and put it in the accumulator (calculator)
        <dl>
          <dd>Note: this will overwrite whatever value was in the accumulator (destructive)</dd>
        </dl>
      </td>
    </tr>
    <tr>
      <td>902</td>
      <td>OUT</td>
      <td>OUTPUT
      </td>
      <td>Copy the value from the accumulator (calculator) to the OUTBOX.
        <dl>
          <dd>Note: the contents of the accumulator are not changed (non-destructive).</dd>
        </dl>
      </td>
    </tr>
    <tr>
      <td>000</td>
      <td>HLT</td>
      <td>HALT</td>
```

```
        <td>Stop working/end the program.</td>
      </tr>
      <tr>
        <td>N/A</td>
        <td>DAT</td>
        <td>DATA</td>
        <td>This is an assembler instruction which
          simply loads the value into the next available mailbox. DAT can also be used in conjunction with labels
          to declare variables. For example, DAT 984 will store the value 984 into a mailbox at the address of the
          DAT instruction.
        </td>
      </tr>
    </tbody>
</table>
```

The HTML code above shows an example of a table that can be used to display the Instruction Set for the user on a separate page. The idea is to show the opcode number that is used in the RAM, followed by the 3-letter opcode name that will be used in the assembly code editor area, then the full form of the opcode name, and finally an explanation of the opcode. This is almost exactly the same as the table shown at https://en.wikipedia.org/wiki/Little_man_computer#Instructions.

How To Use This Resource Page

```
<!DOCTYPE html>
<html lang="en">

<head>
  <% include ../partials/head %>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <!-- HOW TO USE THIS WEBSITE/RESOURCE INFO -->

    <div class="row">

      <h4 class="centered_header sub_header">LMC:</h4>
      <table id="image_explanation_table">
        <tr>
          <td>
            <a href="/images/LMC_top_half_explanation.png" target="_blank"><img style="width: 100%; height: 100%; object-fit: scale-down;" src="/images/LMC_top_half_explanation.png" alt="Explanation of Top Half of Home Page" /></a>
          </td>
          <td>
            <a href="/images/LMC_top_half_explanation.png" target="_blank"><img style="width: 100%; height: 100%; object-fit: scale-down;" src="/images/LMC_bottom_half_explanation.png" alt="Explanation of Top Half of Home Page" /></a>
          </td>
        </tr>
      </table>

```

```
    </div>

    <br />

    <div class="row centered_header">

      <h4 class="sub_header">Challenges:</h4>

      <div class="col-md-6" id="sub_header_2_t">
        <h5>Teachers:</h5>
      </div>
      <div class="col-md-6" id="sub_header_2_s">
        <h5>Students:</h5>
      </div>

      <table class="centered_header" id="pages_info_table">

        <tr>
          <td>
            <p>From the <b>CREATE CHALLENGES</b> page, Teachers have the ability to create, edit, and delete challenges. In addition to this, they can also assign any of these challenges to specific Students, or to a whole class by clicking the "Select All" button next to the respective class.</p>
          </td>
          <td rowspan="2">
            <p>From the <b>VIEW SET CHALLENGES</b> page, Students are able to view their pending challenges to complete, their completed challenges, and the challenges that they have missed, along with all the available information - due data, and completion date if possible.<br /><br />For the challenges left to complete, the Student is able to submit their solution by clicking on the button next to the challenge, and this will take them to a new page which displays the Challenge Title and Challenge Description, and has an input field for the Student to paste in their Assembly Code solution to the problem.<br /><br />For completed challenges, Students can also see what their answer was by clicking the "View Answer" button next to the challenge.</p>
          </td>
        </tr>

        <tr>
          <td>
            <p>From the <b>VIEW STUDENT CHALLENGES</b> page, Teachers can first view their challenges that have been assigned to at least 1 Student, and those that simply haven't been assigned yet.<br /><br />After clicking on the "More Information" button for a challenge that has been assigned already, the Teacher will be able to see the Challenge Title and Challenge Description, as well as all the Students who have submitted, have yet to submit, or have failed to submit the challenge - with information about the due date, and completion date, if available.<br /><br />The Teacher can then view the Student's solution if the Student has already submitted this assignment.</p>
          </td>
        </tr>

      </table>

    </div>

  </div>
</body>

<% include ../partials/script %>

</html>
```

The above code is an example for a page to explain how the resource will work. First, there will be 2 images that are screenshots of the main visualisation page, along with annotations of what each component does. These could be similar to the following 2 images:

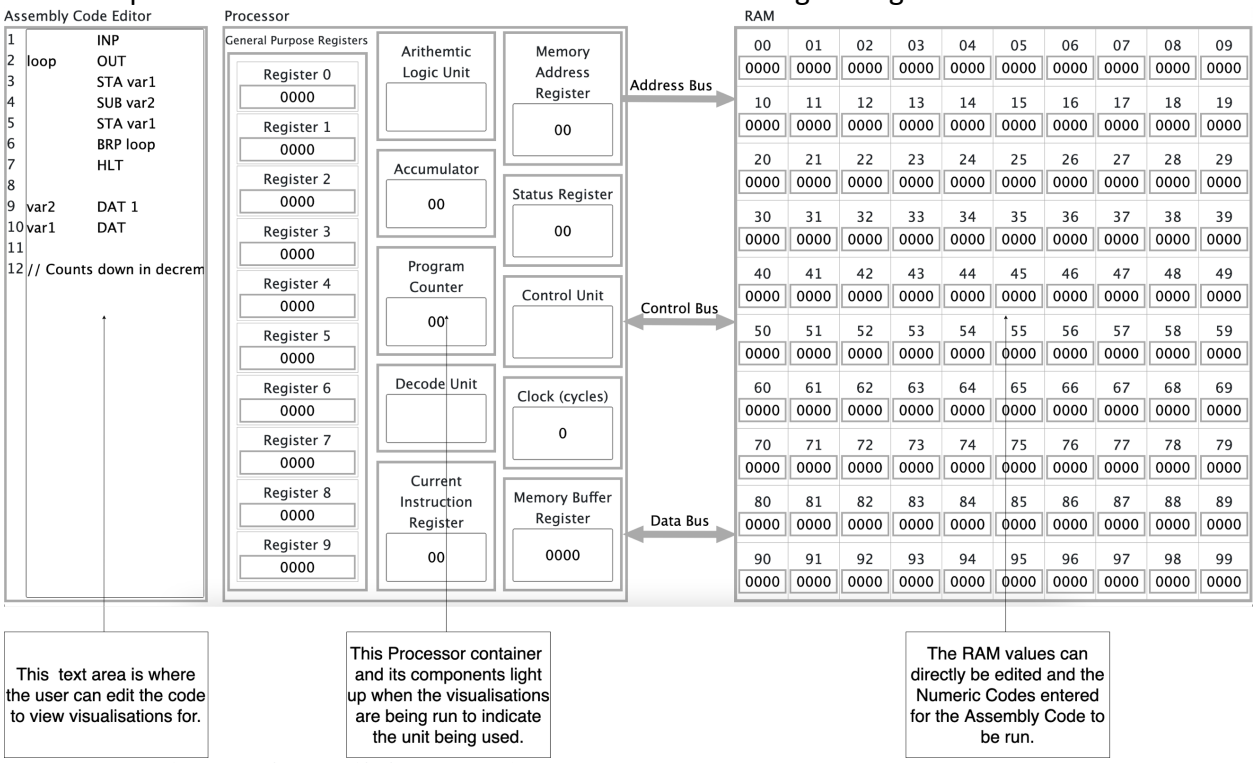

Figure 4.8 – Explanation of Top Half of Main Visualisation

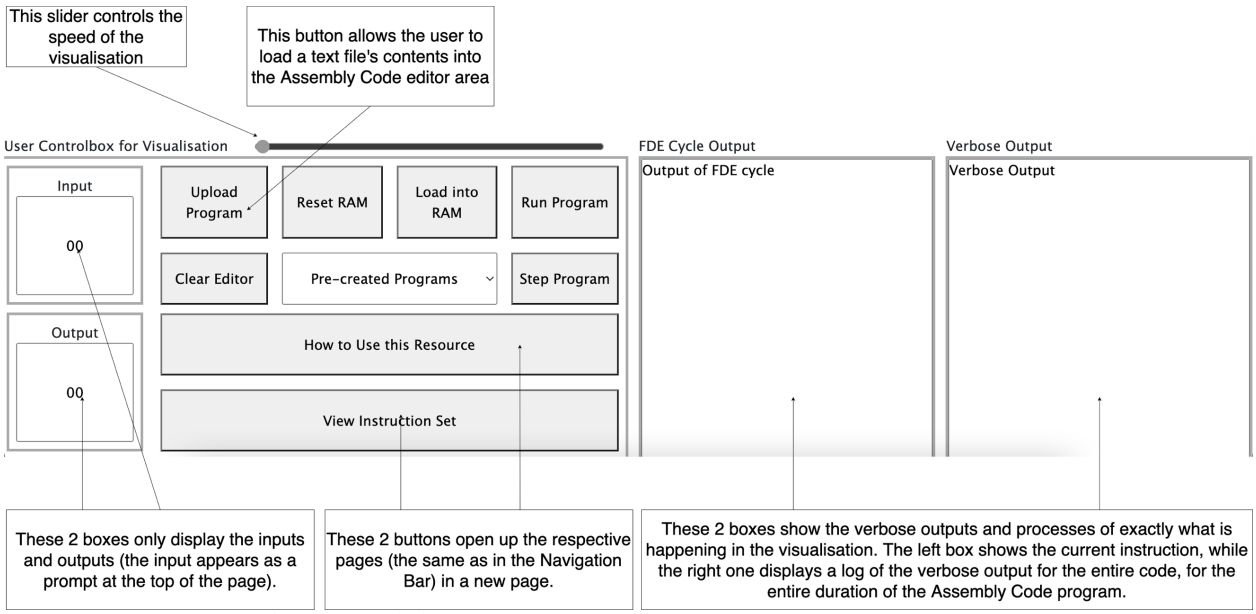

Figure 4.9 – Explanation of Bottom Half of Main Visualisation

Having completed looking at requirements 3, 6, and 9, I will now go on to look at the actual assembly code editor.

Assembly Code and Editor

Link to Requirements

Recalling requirements 2, 4, 5, 7, and 8:

2. Users should be allowed to write code in the Assembly Language and be able to run the machine-code operations which do the following:
    a. Load
    b. Add
    c. Subtract
    d. Branching (conditional & unconditional)
    e. Compare
    f. Halt

4. There should be verbose output which displays any and all debugging for erroneous Assembly code that is written.
    a. This means using jQuery event listeners for converting Assembly code written in the Assembly Language text field into JavaScript, which can then be outputted on the website in order to show what each error is, and where it has happened.

5. Users should have the option to upload text files (.txt), which contain Assembly Language instructions, using the same Instruction Set as the website.
    a. This should automatically be converted into the appropriate format for the Assembly Language text field, and should be able to run immediately, if the user decides to start the animations of the visualisation.

7. Users should be able to see step-by-step outputs of the details of each process being run, in relation to what is happening within the visualisation.
    a. This should include an indicator of which part of the FDE cycle is being run – including the processes with the 5 aforementioned Dedicated Registers (refer to requirement 3).

8. On the same page where users write their assembly code and view visualisations, there should be a dropdown list of pre-created Assembly codes that can be loaded into the text area, ready for visualisation.

The requirements shown above specify which instructions the users should be able to use in the editor, as well as giving a general idea of how the editor should work. It also gives an insight

into some specific requirements of the “translator” e.g., keeping track of each process that is occurring so that it can debugged (for requirement 4).

The first thing that I need to design here is the actual layout of the editor.

Editor Layout and Translator

```
1   loop:
2           STR R1, #1
3           STR R2, 32
4           CMP R1, R2
5           BLT loop
6           HLT
7
8
9
10
```

Figure 4.10 – Basic Idea for Editor

The image to the left shows a very basic illustration of how the Assembly Code editor will look. The idea is that indents will be done using tabs (with a tab being the equivalent of 4 spaces), and this will be done automatically by the frontend system in place. Of course, the user will still be able to indent the code themselves. The greyed-out lines show each tab for the indentation, which will also help the user keep track of their code. The editor also shows line numbers, which is extremely important for the editor, as discussed in the Analysis section.

The “translator” will include the Instruction Set instructions as discussed in the Analysis section, which satisfies requirement 2. The translator will automatically convert all correctly formatted assembly code in the editor to NodeJS so it can be used for the visualisation. This means that requirement 5 will be satisfied by requirement 4a.

We can now look at the processes for each instruction in the instruction set, namely what arts need to be highlighted in the visualisation. This will also include the logic that will be done in the backend.

Instruction Set Prototypes

HLT

I already showed this in the previous prototyping section, so I will not do it again.

INP

```
for (var i = 0; !resolve(); i++) {
  // fetch instruction
  LMC.check_error();

  if (LMC.assembly_code_error == true) {
    resolve();
    clearInterval(LMC.carry_on);
  } else {
    if (i == 0) {
      LMC.log_output('INP');
```

```
        LMC.log_output('Executing instruction...');
      } else if (i == 1) {
        LMC.deactivate_wrapper('control_unit_wrapper');
        LMC.deactivate_wrapper('decode_unit_wrapper');
        control_bus.setOptions({
          color: '#AAAAAA'
        });
        LMC.log_output('Waiting for user input...');
        LMC.cycles++;
      } else if (i == 2) {
        LMC.inp = prompt('User input:');
        document.getElementById('input').value = LMC.inp;
        document.getElementById('input').scrollTop = document.getElementById('input').scrollHeight;
      } else {
        LMC.log_output('Store user input in Accumulator: ' + LMC.inp);
        document.getElementById('accumulator').value = LMC.inp;
        document.getElementById('clock').value = LMC.cycles;
        LMC.deactivate_wrapper('clock_wrapper');
        resolve();
        clearInterval(LMC.carry_on);
      };
    }
}
```

OUT

```
for (var i = 0; !resolve(); i++) {
  // fetch instruction
  LMC.check_error();

  if (LMC.assembly_code_error == true) {
    resolve();
    clearInterval(LMC.carry_on);
  } else {
    if (i == 0) {
        LMC.log_output('OUT');
        LMC.log_output('Executing instruction...');
      } else if (i == 1) {
        LMC.deactivate_wrapper('control_unit_wrapper');
        LMC.deactivate_wrapper('decode_unit_wrapper');
        control_bus.setOptions({
          color: '#AAAAAA'
        });
        LMC.activate_wrapper('accumulator_wrapper');
      } else if (i == 2) {
        LMC.deactivate_wrapper('accumulator_wrapper');
        LMC.log_output('Output value held in Accumulator: ' + document.getElementById('accumulator').value);
        LMC.cycles++;
        document.getElementById('output').value = document.getElementById('accumulator').value;
        document.getElementById('output').scrollTop = document.getElementById('output').scrollHeight;
      } else {
        document.getElementById('clock').value = LMC.cycles;
        LMC.deactivate_wrapper('clock_wrapper');
        resolve();
        clearInterval(LMC.carry_on);
```

```
        };
        i++;
    };
}
```

ADD

```
for (var i = 0; !resolve(); i++) {
    // fetch instruction
    LMC.check_error();

    if (LMC.assembly_code_error == true) {
        resolve(MBR, accumulator);
        clearInterval(LMC.carry_on);
    } else {
        if (i == 0) {
            LMC.log_output('ADD');
            LMC.log_output('Executing instruction...');
        } else if (i == 1) {
            LMC.activate_wrapper('ALU_wrapper');
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
                color: '#AAAAAA'
            });
            LMC.cycles++;
        } else if (i == 2) {
            if (direct) {
                MBR = parseInt(operand);
                LMC.activate_wrapper('MBR_wrapper');
            } else {
                MBR = parseInt(document.getElementById('memory_location_' + operand).value);
                LMC.activate_wrapper('MAR_wrapper');
                LMC.deactivate_wrapper('MBR_wrapper');
            };
        } else if (i == 3) {
            if (direct) {
                LMC.activate_wrapper('MBR_wrapper');
                LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
                document.getElementById('MBR').value = MBR;
            } else {
                LMC.deactivate_wrapper('MAR_wrapper');
                LMC.deactivate_wrapper('MBR_wrapper');
                LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
                LMC.log_output('Fetch data at location held by MAR and store it in MBR: ' + MBR);
                document.getElementById('MAR').value = operand;
                document.getElementById('MBR').value = MBR;
            }
        } else if (i == 4) {
            LMC.activate_wrapper('MBR_wrapper');
            LMC.activate_wrapper('accumulator_wrapper');
        } else if (i == 5) {
            LMC.deactivate_wrapper('MBR_wrapper');
            LMC.deactivate_wrapper('accumulator_wrapper');
```

```
48.         LMC.log_output('Add MBR value to Accumulator and store the result in Accumulator: ' + accumulator + '+' + MBR +
'=' + (accumulator + MBR));
49.         accumulator += MBR;
50.         document.getElementById('accumulator').value = accumulator;
51.       } else {
52.         document.getElementById('clock').value = LMC.cycles;
53.         LMC.deactivate_wrapper('clock_wrapper');
54.         LMC.deactivate_wrapper('ALU_wrapper');
55.         resolve(MBR, accumulator);
56.         clearInterval(LMC.carry_on);
57.       };
58.     };
59. }
```

SUB

```
1. for (var i = 0; !resolve(); i++) {
2.   // fetch instruction
3.   LMC.check_error();
4.
5.   if (LMC.assembly_code_error == true) {
6.     resolve(MBR, accumulator);
7.     clearInterval(LMC.carry_on);
8.   } else {
9.     if (i == 0) {
10.         LMC.log_output('SUB');
11.         LMC.log_output('Executing instruction...');
12.       } else if (i == 1) {
13.         LMC.activate_wrapper('ALU_wrapper');
14.         LMC.deactivate_wrapper('control_unit_wrapper');
15.         LMC.deactivate_wrapper('decode_unit_wrapper');
16.         control_bus.setOptions({
17.           color: '#AAAAAA'
18.         });
19.         LMC.cycles++;
20.       } else if (i == 2) {
21.         if (direct) {
22.           MBR = parseInt(operand);
23.           LMC.activate_wrapper('MBR_wrapper');
24.         } else {
25.           MBR = parseInt(document.getElementById('memory_location_' + operand).value);
26.           LMC.activate_wrapper('MAR_wrapper');
27.           LMC.deactivate_wrapper('MBR_wrapper');
28.         };
29.       } else if (i == 3) {
30.         if (direct) {
31.           LMC.activate_wrapper('MBR_wrapper');
32.           LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
33.           document.getElementById('MBR').value = MBR;
34.         } else {
35.           LMC.deactivate_wrapper('MAR_wrapper');
36.           LMC.deactivate_wrapper('MBR_wrapper');
37.           LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
38.           LMC.log_output('Fetch data at location held by MAR and store it in MBR: ' + MBR);
39.           document.getElementById('MAR').value = operand;
```

```
            document.getElementById('MBR').value = MBR;
        }
    } else if (i == 4) {
        LMC.activate_wrapper('MBR_wrapper');
        LMC.activate_wrapper('accumulator_wrapper');
    } else if (i == 5) {
        LMC.deactivate_wrapper('MBR_wrapper');
        LMC.deactivate_wrapper('accumulator_wrapper');
        LMC.log_output('Subtract MBR value from Accumulator and store the result in Accumulator: ' + accumulator + '-' +
MBR + '=' + (accumulator - MBR));
        accumulator -= MBR;
        document.getElementById('accumulator').value = accumulator;
    } else {
        document.getElementById('clock').value = LMC.cycles;
        LMC.deactivate_wrapper('clock_wrapper');
        LMC.deactivate_wrapper('ALU_wrapper');
        resolve(MBR, accumulator);
        clearInterval(LMC.carry_on);
    };
  };
}
```

LDA

```
for (var i = 0; !resolve(); i++) {
  // fetch instruction
  LMC.check_error();

  if (LMC.assembly_code_error == true) {
    resolve(MBR);
    clearInterval(LMC.carry_on);
  } else {
    if (i == 0) {
        LMC.log_output('LDA');
        LMC.log_output('Executing instruction...');
    } else if (i == 1) {
        LMC.deactivate_wrapper('control_unit_wrapper');
        LMC.deactivate_wrapper('decode_unit_wrapper');
        control_bus.setOptions({
            color: '#AAAAAA'
        });
        LMC.cycles++;
    } else if (i == 2) {
        if (direct) {
            MBR = parseInt(operand);
            LMC.activate_wrapper('MBR_wrapper');
        } else {
            MBR = parseInt(document.getElementById('memory_location_' + operand).value);
            LMC.activate_wrapper('MAR_wrapper');
            LMC.deactivate_wrapper('MBR_wrapper');
        };
    } else if (i == 3) {
        if (direct) {
            LMC.activate_wrapper('MBR_wrapper');
            LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
```

```
            document.getElementById('MBR').value = MBR;
        } else {
            LMC.deactivate_wrapper('MAR_wrapper');
            LMC.deactivate_wrapper('MBR_wrapper');
            MBR = parseInt(document.getElementById('memory_location_' + operand).value);
            LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
            LMC.log_output('Fetch data at location held by MAR (' + operand + ') and store it in MBR: ' + MBR);
            document.getElementById('MAR').value = operand;
            document.getElementById('MBR').value = MBR;
        }
    } else if (i == 4) {
        LMC.activate_wrapper('MBR_wrapper');
        LMC.activate_wrapper('accumulator_wrapper');
    } else if (i == 5) {
        LMC.deactivate_wrapper('MBR_wrapper');
        LMC.deactivate_wrapper('accumulator_wrapper');
        document.getElementById('accumulator').value = MBR;
        LMC.log_output('Store MBR value in Accumulator: ' + MBR);
    } else {
        document.getElementById('clock').value = LMC.cycles;
        LMC.deactivate_wrapper('clock_wrapper');
        resolve(MBR);
        clearInterval(LMC.carry_on);
    };
  };
}
```

STA

```
for (var i = 0; !resolve(); i++) {
  // fetch instruction
  LMC.check_error();

  if (LMC.assembly_code_error == true) {
    resolve(MBR);
    clearInterval(LMC.carry_on);
  } else {
    if (i == 0) {
        LMC.log_output('STA');
        LMC.log_output('Executing instruction...');
    } else if (i == 1) {
        LMC.deactivate_wrapper('control_unit_wrapper');
        LMC.deactivate_wrapper('decode_unit_wrapper');
        control_bus.setOptions({
            color: '#AAAAAA'
        });
        LMC.cycles++;
    } else if (i == 2) {
        MBR = parseInt(document.getElementById('accumulator').value);
        LMC.activate_wrapper('MBR_wrapper');
        LMC.activate_wrapper('accumulator_wrapper');
    } else if (i == 3) {
        LMC.deactivate_wrapper('accumulator_wrapper');
        LMC.log_output('Set MBR to value held in Accumulator: ' + MBR);
        document.getElementById('MBR').value = MBR;
```

```
      } else if (i == 4) {
        LMC.deactivate_wrapper('MBR_wrapper');
        LMC.activate_wrapper('MAR_wrapper');
        LMC.log_output('Store MBR value ' + MBR + ' at memory location held in MAR: ' + operand);
        document.getElementById('MAR').value = operand;
        document.getElementById('memory_location_' + operand).value = MBR;
      } else {
        LMC.deactivate_wrapper('MAR_wrapper');
        document.getElementById('clock').value = LMC.cycles;
        LMC.deactivate_wrapper('clock_wrapper');
        resolve(MBR);
        clearInterval(LMC.carry_on);
      };
    };
}
```

BRA

```
for (var i = 0; !resolve(); i++) {
  // fetch instruction
  LMC.check_error();

  if (LMC.assembly_code_error == true) {
    resolve();
    clearInterval(LMC.carry_on);
  } else {
    if (i == 0) {
      LMC.log_output('BRA');
      LMC.log_output('Executing instruction...');
    } else if (i == 1) {
      LMC.activate_wrapper('status_register_wrapper');
      LMC.deactivate_wrapper('control_unit_wrapper');
      LMC.deactivate_wrapper('decode_unit_wrapper');
      control_bus.setOptions({
        color: '#AAAAAA'
      });
      LMC.cycles++;
    } else if (i == 2) {
      LMC.activate_wrapper('PC_wrapper');
    } else if (i == 3) {
      document.getElementById('PC').value = operand;
      LMC.deactivate_wrapper('PC_wrapper');
      LMC.program_counter = parseInt(operand);
      LMC.log_output('Set PC to operand of instruction: ' + operand);
    } else {
      document.getElementById('clock').value = LMC.cycles;
      LMC.deactivate_wrapper('clock_wrapper');
      LMC.deactivate_wrapper('status_register_wrapper');
      resolve();
      clearInterval(LMC.carry_on);
    };
  };
}
```

BRZ

```
for (var i = 0; !resolve(); i++) {
  // fetch instruction
  LMC.check_error();

  if (LMC.assembly_code_error == true) {
    resolve();
    clearInterval(LMC.carry_on);
  } else {
    if (i == 0) {
      LMC.log_output('BRZ');
      LMC.log_output('Executing instruction...');
    } else if (i == 1) {
      LMC.activate_wrapper('status_register_wrapper');
      LMC.deactivate_wrapper('control_unit_wrapper');
      LMC.deactivate_wrapper('decode_unit_wrapper');
      control_bus.setOptions({
        color: '#AAAAAA'
      });
      LMC.cycles++;
    } else if (i == 2) {
      LMC.log_output('Check if value held in Accumulator is 0');
    } else if (i == 3) {
      if (accumulator == 0) {
        LMC.log_output('0 == 0 - true');
        LMC.activate_wrapper('PC_wrapper');
      } else {
        LMC.log_output(accumulator + ' == 0 - false');
      };
    } else if (i == 4) {
      if (accumulator == 0) {
        LMC.deactivate_wrapper('PC_wrapper');
        LMC.log_output('Set PC to operand of instruction: ' + operand);
        document.getElementById('PC').value = operand;
        LMC.program_counter = parseInt(operand);
      } else {
        document.getElementById('clock').value = LMC.cycles;
        LMC.deactivate_wrapper('clock_wrapper');
        LMC.deactivate_wrapper('status_register_wrapper');
        resolve();
        clearInterval(LMC.carry_on);
      };
    } else {
      document.getElementById('clock').value = LMC.cycles;
      LMC.deactivate_wrapper('clock_wrapper');
      LMC.deactivate_wrapper('status_register_wrapper');
      resolve();
      clearInterval(LMC.carry_on);
    };
  };
}
```

BRP

```
for (var i = 0; !resolve(); i++) {
  // fetch instruction
  LMC.check_error();

  if (LMC.assembly_code_error == true) {
    resolve();
    clearInterval(LMC.carry_on);
  } else {
    if (i == 0) {
      LMC.log_output('BRP');
      LMC.log_output('Executing instruction...');
    } else if (i == 1) {
      LMC.activate_wrapper('status_register_wrapper');
      LMC.deactivate_wrapper('control_unit_wrapper');
      LMC.deactivate_wrapper('decode_unit_wrapper');
      control_bus.setOptions({
        color: '#AAAAAA'
      });
      LMC.cycles++;
    } else if (i == 2) {
      LMC.log_output('Check if value held in Accumulator is positive (>= 0)');
    } else if (i == 3) {
      if (accumulator >= 0) {
        LMC.log_output(accumulator + ' >= 0 - true');
        LMC.activate_wrapper('PC_wrapper');
      } else {
        LMC.log_output(accumulator + ' >= 0 - false');
      };
    } else if (i == 4) {
      if (accumulator >= 0) {
        LMC.deactivate_wrapper('PC_wrapper');
        LMC.log_output('Set PC to operand of instruction: ' + operand);
        document.getElementById('PC').value = operand;
        LMC.program_counter = parseInt(operand);
      } else {
        document.getElementById('clock').value = LMC.cycles;
        LMC.deactivate_wrapper('status_register_wrapper');
        LMC.deactivate_wrapper('clock_wrapper');
        resolve();
        clearInterval(LMC.carry_on);
      };
    } else {
      document.getElementById('clock').value = LMC.cycles;
      LMC.deactivate_wrapper('clock_wrapper');
      LMC.deactivate_wrapper('status_register_wrapper');
      resolve();
      clearInterval(LMC.carry_on);
    };
  };
}
```

Unfortunately, there is no better way to automate this process, so instead, everything simply had to be done with a for loop and different conditions. Using the AQA specification, I looked at what components should be used when, and accordingly highlighted the appropriate container.

I can now mark requirements 2 and 7 (because the above also includes the log outputs for the step-by-step outputs) as designed. The next easiest thing to look at would be upload.

Uploading Text File and Converting to Assembly Code

The basic idea of uploading a text file is to use FTP (File Transfer Protocol). However, as I am using NodeJS and Express for this website, this process doesn't explicitly refer to FTP.

NodeJS, Express, and File Upload Template

```
const express = require('express');
const fileUpload = require('express-fileupload');
const cors = require('cors');
const bodyParser = require('body-parser');
const morgan = require('morgan');
const _ = require('lodash');

const app = express();

// enable files upload
app.use(fileUpload({
    createParentPath: true
}));

//add other middleware
app.use(cors());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(morgan('dev'));

//start app
const port = process.env.PORT || 3000;

app.listen(port, () =>
    console.log(`App is listening on port ${port}.`)
);
```

Figure 4.11 – Code for Basic Template of File Upload

The code above first sets up the express-fileupload middleware to enable multipart/form-data requests. After that, other Express middleware is added to allow Cross-Origin Resource Sharing (CORS) – which is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served – as well as request body parsing, and finally, HTTP request logging. The final few lines of code start the server on port 3000.

Next, a route needs to be created to allow users to upload a text file.

Single File Upload Prototyped Code

```
app.post('/upload-textfile', async (req, res) => {
    try {
        if (!req.files) {
            res.send({
                status: false,
                message: 'No file uploaded'
            });
        } else {

            let textfile = req.files.textfile;

            // use mv() method to place the file in upload directory
            textfile.mv('./uploads/' + textfile.name);

            //send response
            res.send({
                status: true,
                message: 'File is uploaded',
                data: {
                    name: textfile.name,
                    mimetype: textfile.mimetype,
                    size: textfile.size
                }
            });
        }
    } catch (err) {
        res.status(500).send(err);
    }
});
```

Figure 4.12 – Code for Uploading a Single File

This code is a HTTP POST function. When the user sends a multipart/form-data request to the /upload-textfile route in order to upload a file, this function saves the file to the uploads folder on the server.

Since there's only 1 file that needs to be uploaded here, there is no functionality requirement to upload multiple files, hence the route above for uploading a single file is sufficient.

When the file is uploaded, the text needs to be put directly into the editor and auto formatted. However, this doesn't require any additional code or processes because this process is already taken care of by the "translator". Therefore, all that this process needs to do is copy the contents of the text file directly into the editor for the visualisations.

Requirement 5 is met by this. We can now look at the pre-created codes.

Pre-Created Assembly Codes

My idea for this is to simply use a select dropdown, where the users can click on one of the options, and in the backend JavaScript, a function will be called that takes the name of the challenge, calls an appropriate file (stored in the **public/** directory), and replaces the current text in the Assembly Code text editor area with the contents of the text file called.

```
<textarea class="scroll_together" id="code_area" rows=27 wrap="off">
      INP
loop    OUT
      STA var1
      SUB var2
      STA var1
      BRP loop
      HLT

var2    DAT 1
var1    DAT

// Counts down in decrements of var2 (1), starting with the inputted number</textarea>
<!-- pre-created dropdown logic in lmc.js -->
            <div class="col-md-4" id="pre-created_programs_wrapper">
              <select id="pre-created_programs" name="pre-created_programs">
                <option value="" selected hidden>Pre-created Programs</option>
                <option value="add_2_nums" id="add_2_nums">add 2 nums</option>
                <option value="sub_2_nums" id="sub_2_nums">sub 2 nums</option>
                <option value="mult_2_nums" id="mult_2_nums">mult 2 nums</option>
                <option value="count_down" id="count_down">count down</option>
                <option value="immediate_addressing" id="immediate_addressing">immediate addressing</option>
                <option value="square_inp_until_0" id="square_inp_until_0">square inp until 0</option>
                <option value="bubble_sort" id="bubble_sort">bubble sort</option>
              </select>
            </div>
```

The first part of the code above shows what I will immediately place into the Assembly Code text editor area (one of the pre-created programs, which is marked as selected at the beginning). The second part shows the HTML syntax for the select which will have the dropdown options.

As for the logic of getting the text files and putting them into the text editor area, something like the following could work:

```
$("#pre-created_programs").on("change", function () {

  try {

    var selected_option = $("#pre-created_programs").val().toString();
    var directory = '../text_files/pre_created_programs/' + selected_option + '.txt';

    // get file with same name as pre-created_programs selection and put that into text area
    fetch(directory).then(response => response.text()).then(text => {
        document.getElementById("code_area").value = text;
     });

  } catch (error) {
     document.getElementById("code_area").value = "does not exist";
  }

});
```

This ensures that every time the dropdown menu is clicked, the assembly code text editor area will automatically be updated with the relevant text file's contents.

I will not explicitly go through the text files, as these are basic examples that have just been found online or created by me to demonstrate a simple idea using my instruction set for the Assembly code.

We can now tick requirement 8 off, which leaves requirement 4, the main visualisation and how it actually works.

Little Man Computer

Firstly, I have created a UML diagram for how I expect the instruction set and instructions to link with the Little Man Computer.

UML Diagram

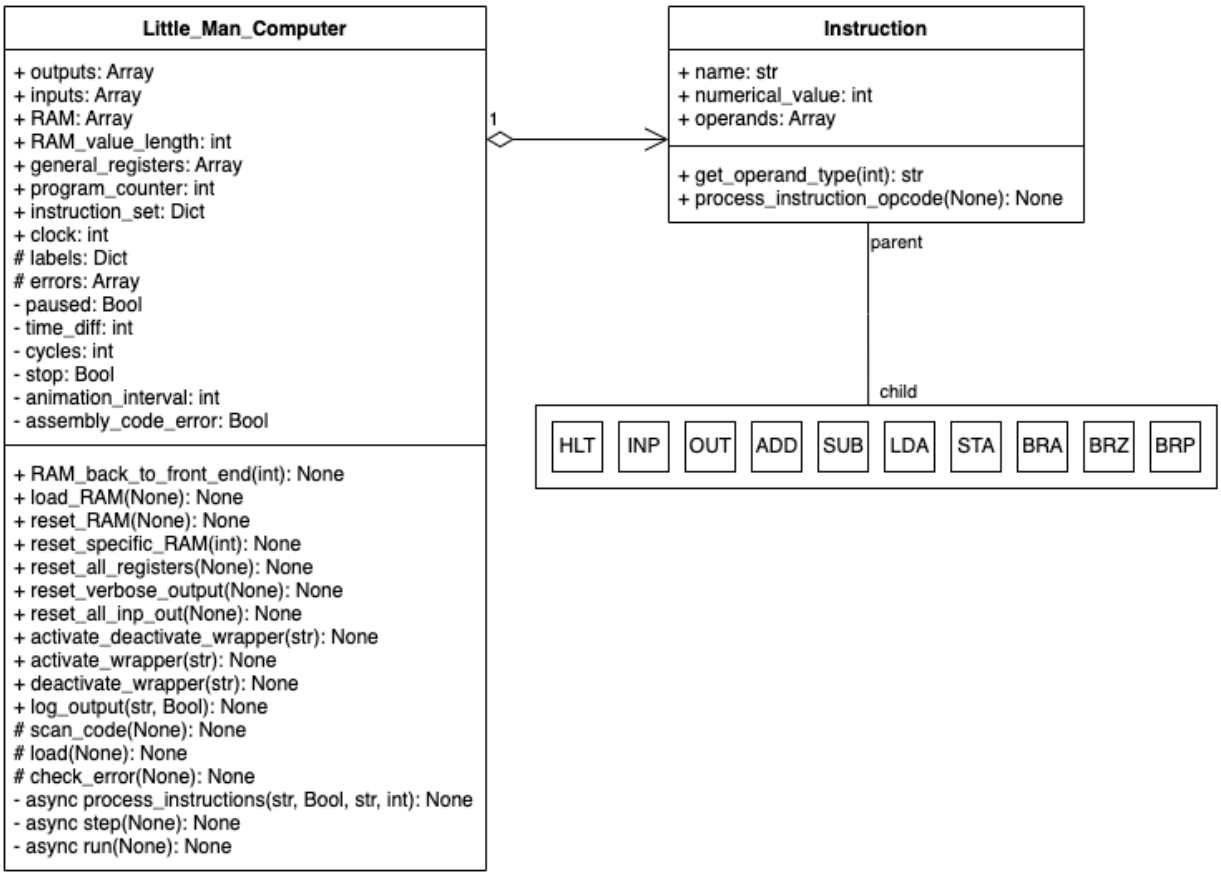

Figure 4.13 - UML diagram for Main Visualisation

Each instruction above is its own object, and each inherits the Instruction class and extends it. These instructions will then be stored in a dictionary, which is then used in the Little Man Computer class.

From here, I will now try to prototype the main methods in the Little Man Computer class, because we have already looked at the main ones in the Instruction class. I will not prototype some of the smaller methods because they will either be relatively straightforward, or only contain outputs for myself, or be a combination of other methods already implemented for the Little Man Computer class e.g. load_RAM().

Scanning the Assembly Code Text Editor and Loading into RAM

For this method, the main idea here is to trim whitespaces and think about how many words there will be on a single line. If there are two or three words with and any of them have the word “DAT”, then we need to store this as a variable, and this is what the following code will be used for:

```
1. this.labels = {};
```

```
var counter = 0;
var assembly_code = document.getElementById('code_area').value.toUpperCase();
var lines = assembly_code.split('\n');

for (var i = 0; i < lines.length; i++) {

  var line = lines[i].trim();

  if (line != '') {

    var instruction = line.split(/\s+/);

    // variable that stores input
    if (instruction.length == 2) {

      if (instruction[1] == 'DAT') {
        this.labels[instruction[0]] = counter;
        this.reset_specific_RAM(counter);
      } else if (!(instruction[0] in this.instruction_set)) {
        this.labels[instruction[0]] = counter;
      };

    };

    // variable that stores value after DAT
    if (instruction.length == 3) {

      if (instruction[1] == 'DAT') {
        this.labels[instruction[0]] = counter;
        this.RAM[counter] = instruction[2].toString();
      } else if (!(instruction[0] in this.instruction_set)) {
        this.labels[instruction[0]] = counter;
      }

    };

    counter++;

  };
};
```

We will save all the variables in the dictionary “labels” as keys, and the values will be the line in which the variable is stored/called.

The next part is to load the Assembly code into RAM, and the code is scanned at the same time as we do this. We also have to take into account direct and immediate addressing, using “@” and “#”. The following code will be similar to what can be used for this:

```
this.scan_code();

// load program in RAM
var counter = 0;
var assembly_code = document.getElementById('code_area').value.toUpperCase();
```

```
var lines = assembly_code.split('\n');

for (var i = 0; i < lines.length; i++) {

    var line = lines[i].trim();

    if (line != '') {

        var instruction = line.split(/\s+/);

        if (instruction.length == 3) {

            var label = instruction[0];
            var opcode = instruction[1];
            var operand = instruction[2];

            this.labels[label] = counter;

            if (operand in this.labels) {
                operand = this.labels[operand];
            };

            if (operand.length > 1) {

                if (operand.substring(0, 1) == '@') {

                    operand = operand.substring(1);

                    if (isNaN(operand)) {
                        if (operand in this.labels) {
                            operand = '@' + this.labels[operand];
                        };
                    } else {
                        operand = '@' + operand;
                    };

                } else if (operand.substring(0, 1) == '#') { // immediate addressing

                    operand = operand.substring(1);

                    if (isNaN(operand)) {
                        if (operand in this.labels) {
                            operand = '#' + this.labels[operand];
                        };
                    } else {
                        operand = '#' + operand;
                    };

                };

            };

            if (opcode in this.instruction_set) {

                var encode = this.instruction_set[opcode].numerical_value + operand;
```

```
            this.RAM[counter] = encode.toString();

            counter++;

        } else if (opcode == 'DAT') {

            counter++;

        } else {

            this.log_output('Error at line ' + counter + ': Invalid opcode');

        };

    } else if (instruction.length == 2) {

        if (instruction[0] in this.instruction_set) {

            var opcode = instruction[0];
            var operand = instruction[1];

            if (operand in this.labels) {
                operand = this.labels[operand];
            };

            // Indirect Addressing
            if (operand.length > 1) {

                if (operand.substring(0, 1) == '@') {

                    operand = operand.substring(1);

                    if (isNaN(operand)) {
                        if (operand in this.labels) {
                            operand = '@' + this.labels[operand];
                        };
                    } else {
                        operand = '@' + operand;
                    };

                } else if (operand.substring(0, 1) == '#') {

                    operand = operand.substring(1);

                    if (isNaN(operand)) {
                        if (operand in this.labels) {
                            operand = '#' + this.labels[operand];
                        };
                    } else {
                        operand = '#' + operand;
                    };

                };

            };
```

```

            var encode = this.instruction_set[opcode].numerical_value + operand;
            this.RAM[counter] = encode.toString();

            counter++;

        } else {

            this.labels[instruction[0]] = counter;
            var opcode = instruction[1];

            if (opcode in this.instruction_set) {

                var encode = this.instruction_set[opcode].numerical_value;
                this.RAM[counter] = encode.toString();

                counter++;

            } else if (instruction[1] == 'DAT') {

                counter++;

            } else {

                this.log_output('Error at line ' + counter + ': Invalid instruction');

            };

        };

    } else {

        if (line in this.instruction_set) {
            this.RAM[counter] = this.instruction_set[line].numerical_value;
            counter++;
        } else {
            this.log_output('Error at line ' + counter + ': Invalid instruction');
        };

    };

  };

  this.load_RAM_from_backend();

};

for (var i = 0; i < this.RAM.length; i++) {
    if (this.RAM[i] == '') {
        this.reset_specific_RAM(i);
    };
};
this.load_RAM_from_backend();
```

Once again, this uses the idea of different numbers of words on a single line. Namely, a single word means that it is the name of a loop, or a HALT statement. With 2 words, if not a DAT statement, we know that it is an instruction from the instruction set, so we can use the instruction set dictionary with the instruction objects to get the numerical values and store them in the RAM at the necessary memory locations (each location corresponds to a single line of code, except for the variables which are stored elsewhere where no other important information will be in the RAM).

I will now go on to look at how we will actually run, step, and process instructions.

Running Instructions

Starting with processing instructions, these will literally need to be hardcoded because there is no other way to ensure that the instructions are the same as the values required, so the following is an example of what could be used:

```
if (this.assembly_code_error == false) {

  // must be done in order, so await is used
  var instruction = await this.process_instruction_main_beginning();

  // halt instruction
  if (instruction == '000') {

    await this.process_instruction_HLT();

  } else if (instruction == '901') { // input instruction

    await this.process_instruction_INP();

  } else if (instruction == '902') { // output instruction

    await this.process_instruction_OUT();

  } else {

    var opcode = instruction.substr(0, 1);
    var operand_address = instruction.substr(1);
    var operand;
    var direct = false;

    if (operand_address.substr(0, 1) == '@') { // indirect addressing
      operand = parseInt(document.getElementById('memory_location_' + operand_address.substr(1)).value);
    } else if (operand_address.substr(0, 1) == '#') { // direct addressing
      direct = true;
      operand = operand_address.substr(1);
    } else {
      operand = operand_address;
    };

    var accumulator = parseInt(document.getElementById('accumulator').value);

```

```
    if (opcode == '1') { // add instruction
      var MBR, accumulator = await this.process_instruction_ADD(direct, operand, accumulator);
    } else if (opcode == '2') { // subtract instruction
      var MBR, accumulator = await this.process_instruction_SUB(direct, operand, accumulator);
    } else if (opcode == '5') { // load to accumulator
      var MBR = await this.process_instruction_LDA(direct, operand);
    } else if (opcode == '3') { // store accumulator value in memory
      var MBR = await this.process_instruction_STA(operand);
    } else if (opcode == '6') { // always branch
      await this.process_instruction_BRA(operand);
    } else if (opcode == '7') { // branch if zero
      await this.process_instruction_BRZ(operand, accumulator);
    } else if (opcode == '8') { // branch if positive or 0
      await this.process_instruction_BRP(operand, accumulator);
    };
  };
} else { // invalid instruction - not recongnised
  alert('Error in assembly code! Executing HLT instruction!');
  this.activate_wrapper('status_register_wrapper');
  document.getElementById('status_register').value = '-1';
  await this.process_instruction_HLT();
  this.paused = true;
  this.stop = true;
};
```

Note that this code calls separate **process_instruction()** methods, which will basically just be defined by calling the appropriate instruction object, and then calling its method: **process_instruction()**, with whatever necessary parameters there are.

This is the main bit of code, and the run and step functions essentially just call this function.

The step function would be something like the following:

```
this.paused = true;
this.animation_interval = parseInt(this.time_numerator / document.getElementById('clock_speed').value);
this.process_instruction();
```

The run function, similarly, would be something like this:

```
this.cycles = 0;
this.stop = false;

this.reset_all_registers();
this.reset_all_inp_out();

this.scan_code();

this.paused = false;
this.animation_interval = parseInt(this.time_numerator / document.getElementById('clock_speed').value);

while (!this.stop) {
  await this.process_instruction();
};
```

Having looked at the actual important parts of running the Little Man Computer and loading into RAM, we now need to look at how errors will be dealt with.

Dealing with Errors

This will quite simply be done by the **check_error()** function. Since we have already saved included the actual outputting of errors in the lines above, this will only be done for checking the errors, and, as above, run whenever the next line of the Assembly code is being looked at.

This is relatively straightforward and doesn't really need to be explained, so it can look akin to the following:

```
1. let FDE = document.getElementById('FDE_output').value;
2. let verbose = document.getElementById('verbose_output').value;
3.
4. let FDE_list = FDE.split('\n');
5. let verbose_list = verbose.split('\n');
6.
7. // dealing with errors of log_outputs
8. if (FDE_list.length >= 3 && verbose_list.length >= 3) {
9.     if (FDE_list[(FDE_list.length - 1) - 1] == FDE_list[(FDE_list.length - 1) - 2] || verbose_list[(verbose_list.length - 1) - 1] ==
verbose_list[(verbose_list.length - 1) - 2]) {
10.         this.assembly_code_error = true;
11.     } else {
12.         this.assembly_code_error = false;
13.     };
14. } else {
15.     this.assembly_code_error = false;
16. };
```

Having now completed designing the Little Man Computer and main visualisation, the next important parts to look at are the actual challenges, and therefore, the process of signing up and logging in – the accounts.

Login and Sign-up

Link to Requirements

Recalling requirement 10:

10. Accounts – users should be able to login and sign up for a student or teacher account, depending on whether they are a student or teacher at Dubai College, using their Dubai College email address.
    a. Student account
        i. Should be able to view their assigned challenges and due dates (set by a teacher).
        ii. Should be able to submit these assigned challenges by pasting in their Assembly code into a text area.
        iii. Should be able to view a log of their previously completed challenges.
    b. Teacher account
        i. Teachers have classes which are groups of students.
        ii. Should be able to create and set challenges to be completed by all students in a class, or specific students.
        iii. Can view students' submitted challenge responses – there will be an indicator that the student has clicked on the submit button and the student can no longer edit it.
        iv. Should be able to view all logs of challenges and differ between completed and non-completed challenges.
    c. No account
        i. These users can still view visualisations and write code in the Assembly Language.
        ii. Cannot submit challenges to a teacher as they will have no assigned class and therefore no assigned teacher.

Obviously, it is very important that the actual sign-up and login processes are designed, so these will be done first, followed by the challenges part of this requirement.

Flowcharts

Sign-up

When a user enters the website, although they may just want to use the visualisation really quickly for an extremely quick task, most users will likely want to login to their existing account, or sign-up for an account. Below, I have delineated the processes of signing up via a Flow diagram.

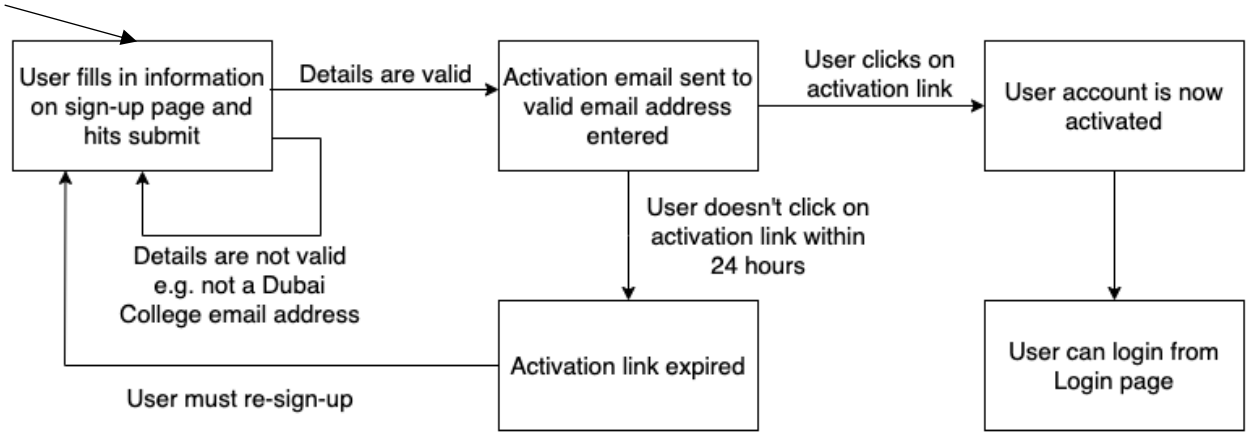

Figure 4.14 – Flowchart for user sign-up

From my own personal experience of sign-ups that require an activation link, I know that they usually expire within 24 hours or less, and so, as I believe that I will use some of the same techniques in order to create an activation link, I have added the fact that if the user doesn't activate their account within 24 hours, then they will need to sign-up again. As this is important information, once a user has submitted the necessary valid information for signing-up, there will be a notification on the screen, like an alert, that tells the user that they must activate their account within 24 hours, or they will need to sign-up again.

Login

Once a user has signed up and has activated their account, the next thing will be for them to login to their account. I have outlined the procedures that will take place while a user is logging in in the Flow diagram below.

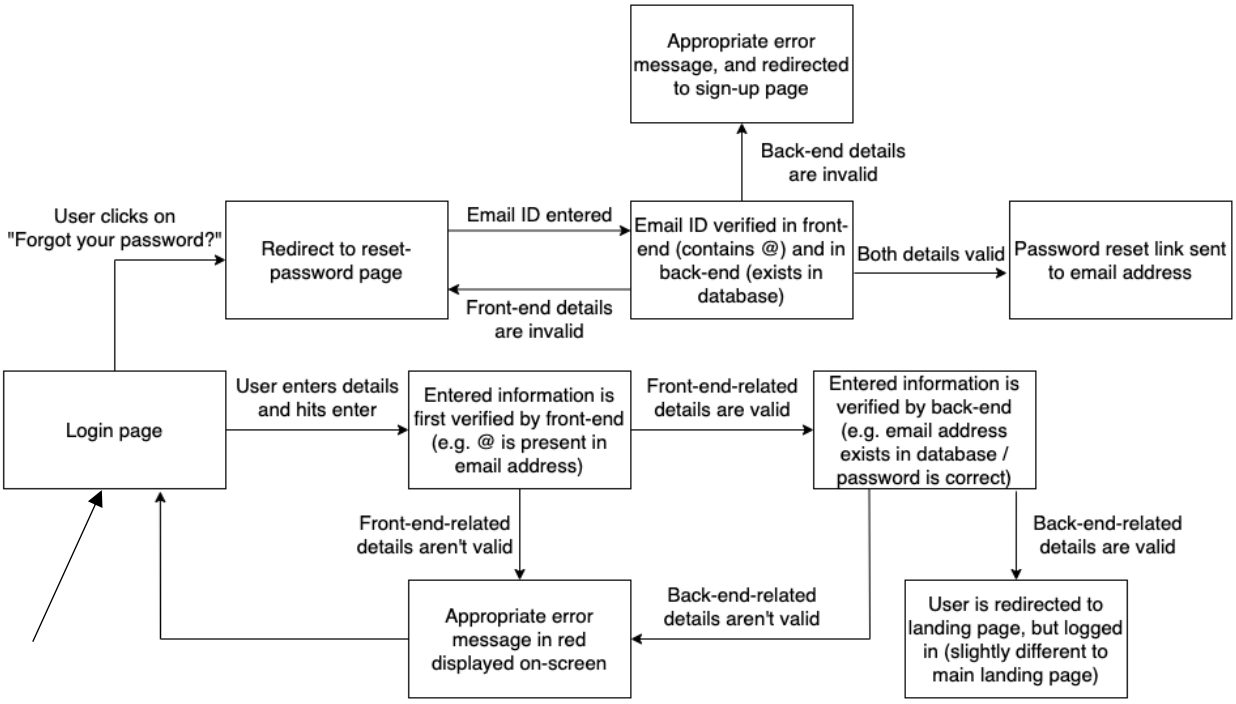

Figure 4.15 – Flowchart for user login and password reset

The entered information for the login page will have to go through 2 sets of verifications, 1 by the front-end on the surface-level, without needing to access the database, where basic things like an "@" present in the email address are checked. After that, a more thorough verification procedure takes place, where things like a valid Dubai College email address, correct password, etc. are checked.

I have also included the steps that will be taken for a user who clicks on the "Forgot your password?" option on the login screen. This follows a similar sort of procedures to the sign-up page with regards to the reset password link, i.e., if a user takes over 24 hours to respond to the email and reset their password, they will have to redo the procedure. Of course, they will be notified of this.

Login and Sign-up Combined

As all 3 processes (logging in, signing up, and resetting a password) are closely related, I have combined them all into a single Flow diagram shown below, adding in the new links between each of them.

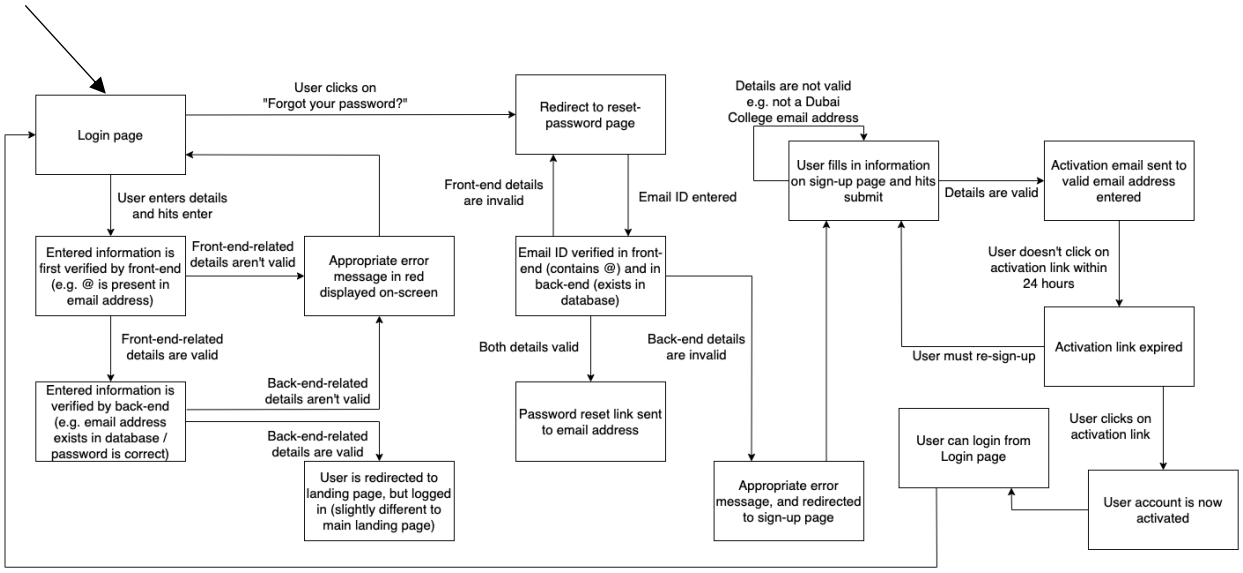

Figure 4.16 – Combined Login, Sign-up, and Reset password Flowchart

The diagrams have stayed pretty much the same, with a bit of redesigning so that the entire flowchart is easier to follow. The new links added are the link from the redirection to the sign-up page, to the information being entered into and submitted from the sign-up page, as well as the long arrow going from the user being able to login from the Login page, to actually being redirected to the Login page.

Although a prototype was made and discussed in the Analysis section above, there are a few changes that need to be made.

As things are currently designed, when a user signs up, they must choose the name of their teacher from a dropdown list, for example “Mr. Wood”. However, looking at the database that we have, this would be quite hard to accomplish, having to store a teacher’s prefix (e.g., “Mr.” or “Mrs”). An alternative is to store the teacher’s Dubai College-assigned 3-letter code e.g., “MWD”. This is more efficient in terms of memory, as only 3 additional characters need to be stored for each teacher as part of the newly created “class_code” attribute.

The updated version of the prototype and entity-relationship diagram for signing up as a student are shown below.

Entity-Relationship Diagram

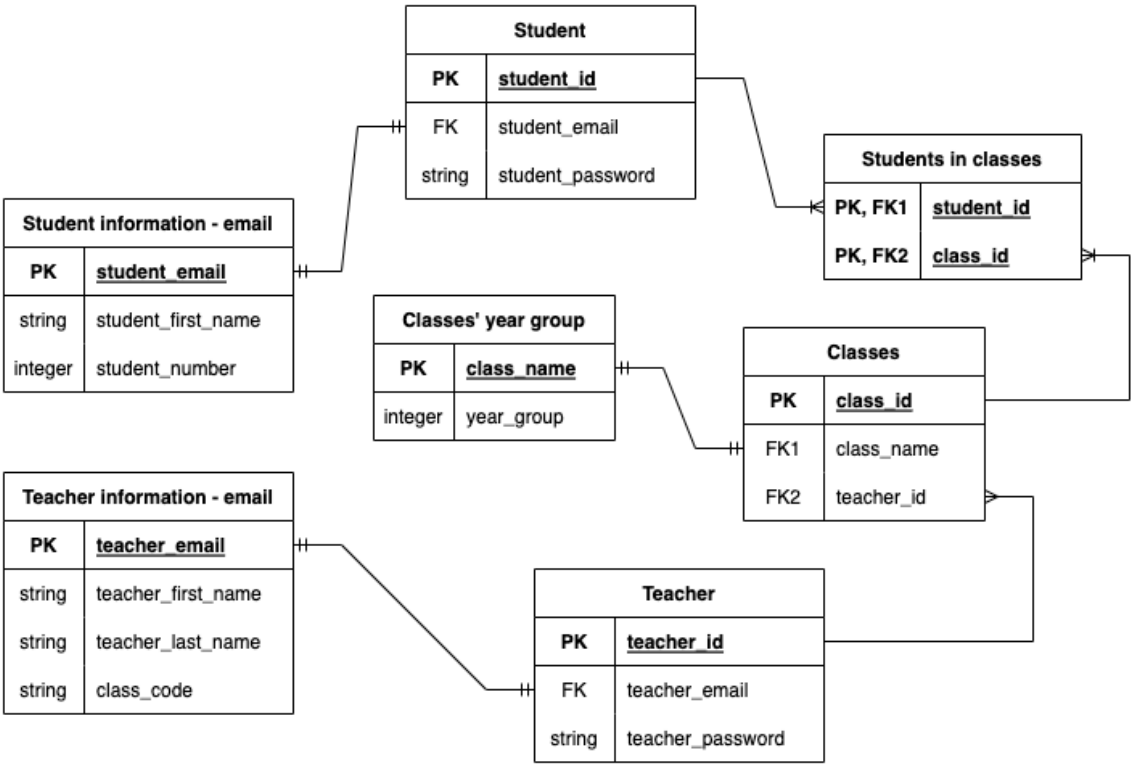

Figure 4.17 – Updated Entity-relationship diagram for Login and Sign-up

Prototypes

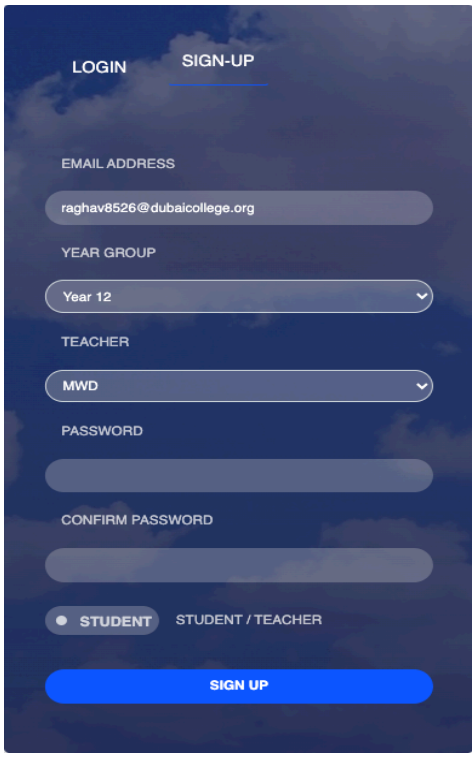

Figure 4.18 – Updated student sign-up page

Using a class_code also means that when a teacher is signing up, they must have the option to enter their class_code, and this needs to be verified with the back-end database to make sure that no other account already exists with the same class_code (as class_code must be unique – this is how the Dubai College system works). Although this step of verification would require another modification to the flowchart for signing up (Figure 4.14), it can be abstracted into the "details are validated" arrows, which will check whether the class_code already exists or not when a teacher is signing up for an account.

The prototype for a teacher's point of view when signing up and logging in were not show previously in the Analysis section, and so, I have decided to show the 2 prototypes below.

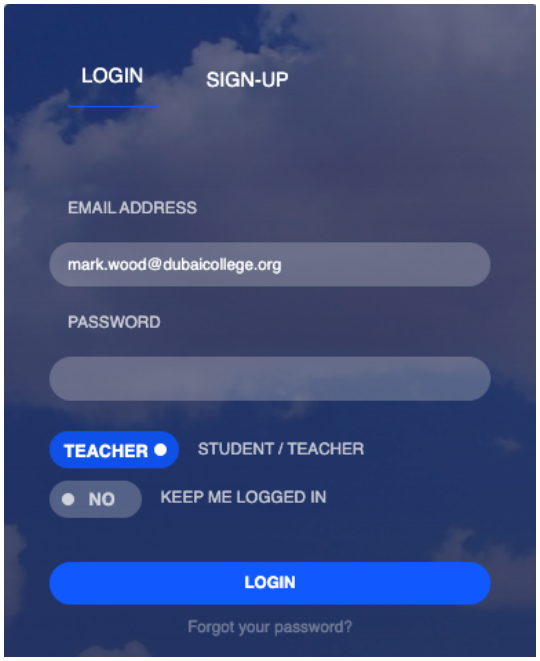

Figure 4.19 – Prototype login page for teachers

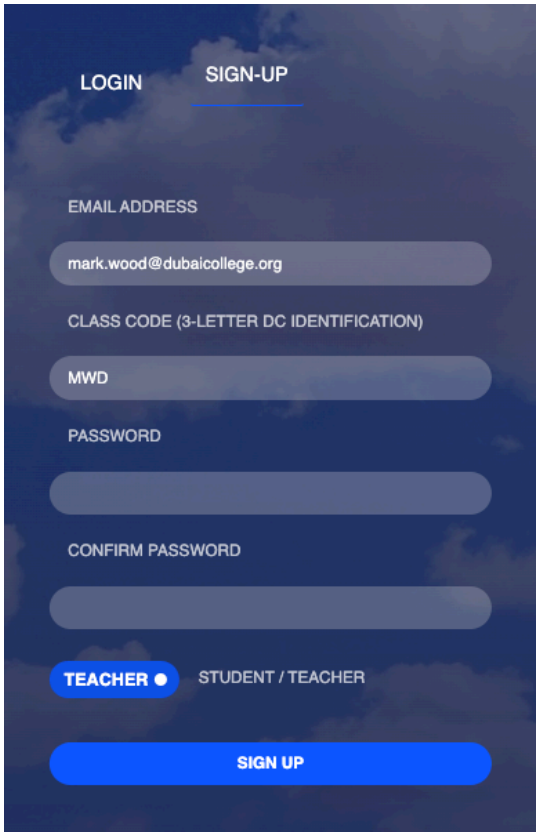

Figure 4.20 – Prototype sign-up page for teachers

The logins and sign-ups have now been covered for both students and teachers, however, even though I have outlined the process for resetting a user's password in a Flow diagram above (Figure 4.15), I have not shown the prototype for resetting the password. This is done below.

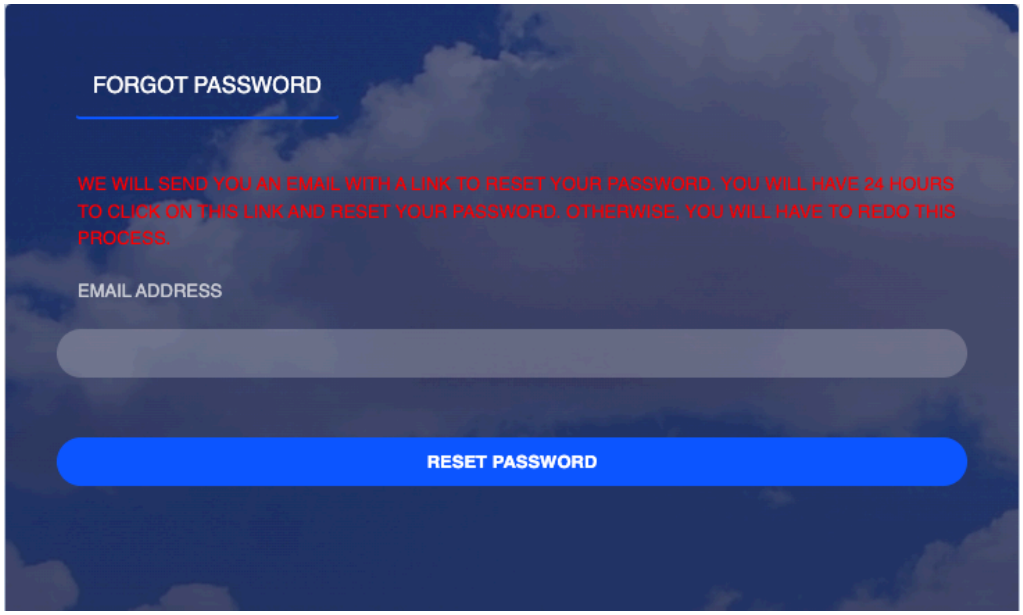

Figure 4.21 – Reset Password / "Forgot your password?" page

One of the major design decisions that I made when considering the forgot password page was whether to include the forgot password as simply another tab, next to the sign-up button, as in the figure below (Figure 4.22).

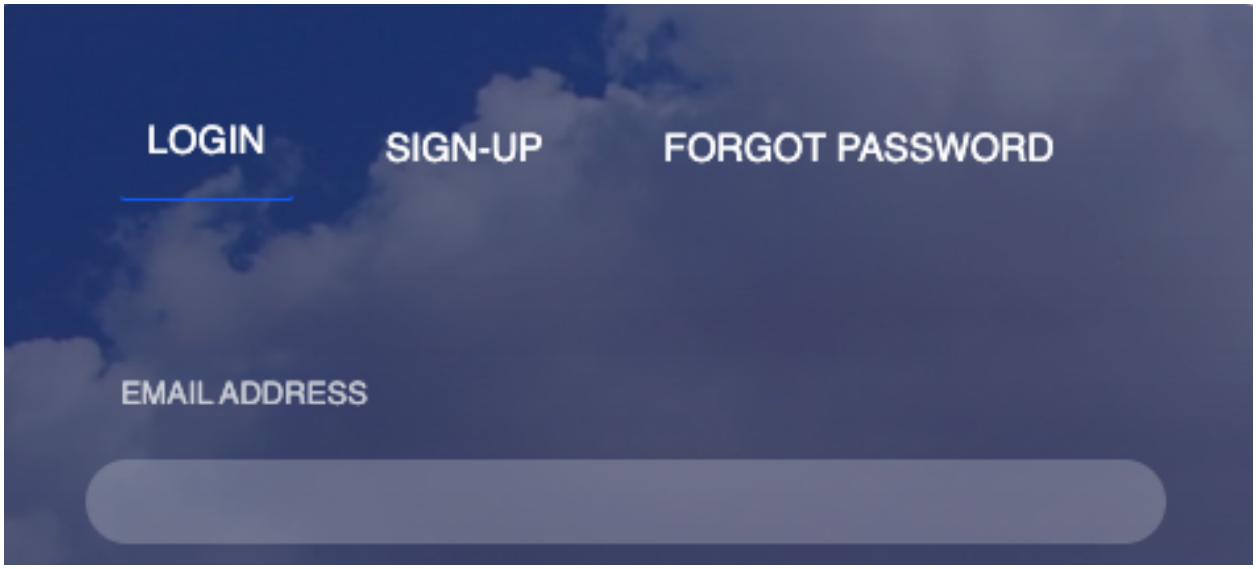

Figure 4.22 – Possible layout for Forgot Password page/tab

I decided to avoid this for multiple reasons. For one, it would require increasing the width of the rest of the forms included within the login and sign-up pages, as the forgot password tab wouldn't fit within the original space. Furthermore, including a forgot password tab would render the "Forgot your password?" button useless. Also, there is no need for a person to be able to go back to the login or sign-up page from the forgot password page, as they would need

to go to their email and reset their password before doing anything else, in which case they would just open a new tab to access the website and land on the main landing page (essentially, I thought that the forgot password page needs to have its own page).

Cookies and Sessions

Persistent Cookies

For users who already have an account within the database and have logged into their account on this device before, they may want to be remembered, and not have to re-enter their credentials every time they visit the website (hence the remember me button in the prototype above). In order to include this feature within the application, it will require the use of "persistent cookies" to store the login of a user. A persistent cookie stores data for an extended duration and comes up with an expiration date.

First-Party Cookies

In addition to persistent cookies, the resource will also need to use "first-party" or "session" cookies, which are temporary cookies that get deleted when a user closes their browser, and their main purpose is to keep items even if a user navigates to other tabs within the browser. This type of cookie will mostly be used to save the progress of a user in challenges, as well as make sure that the code entered into the text field for the assembly code for the visualisation is kept while the user doesn't close the browser.

Persistent cookies and session cookies are implemented in a similar way in-tandem, and I have included an outline of how this can be accomplished below.

Issuing Session Tokens

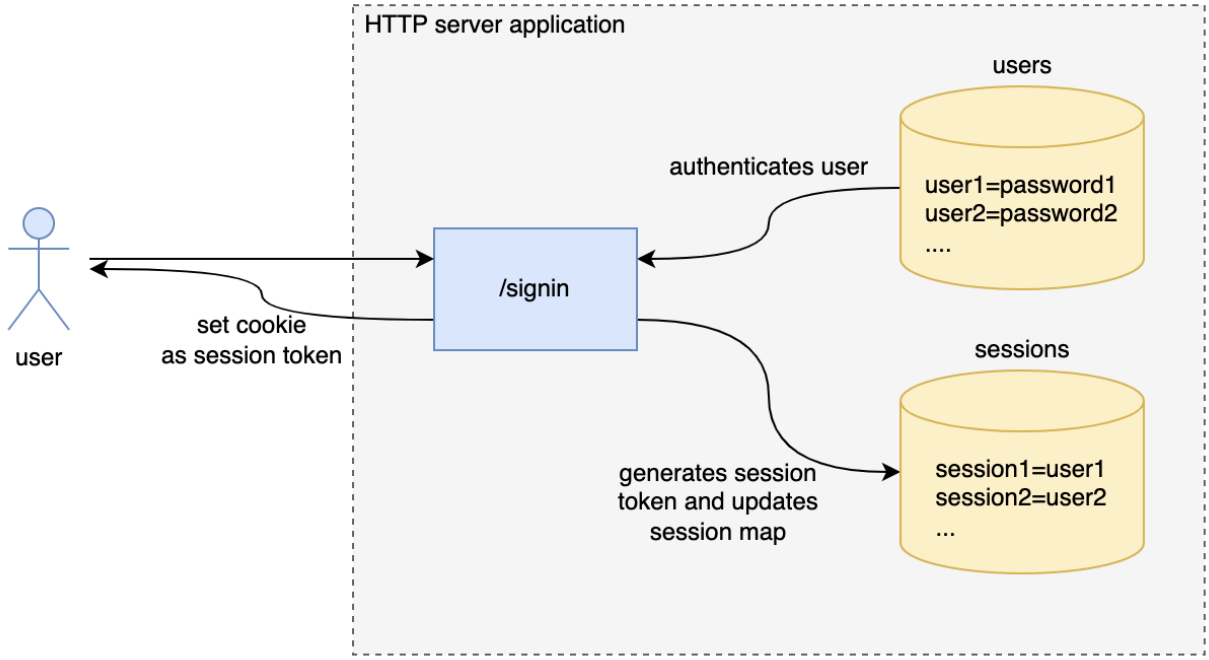

Figure 4.23 – Session Token Process (Session Cookie Authentication in Node.js (With Complete Examples) [Screenshot], 2022)

The first thing that needs to be done is create a session token, and the image above shows how this process is used by the backend when the user is authenticated. The basic idea is that when the user is authenticated and is redirected to the login page, the backend generates a session token and adds that session token to a database, where the session token is matched to the user. Then, one cookie is set as the user's session token.

The code below shows a very simple way of how a session token can actually be created and set.

```
// sessions contain username and expiration time
class Session {
    constructor(username, expiresAt) {
        this.username = username
        this.expiresAt = expiresAt
    }

    // used later to see if session has expired
    isExpired() {
        this.expiresAt < (new Date())
    }
}

const sessions = {} // this will be implemented using a database to store
sessions

const signinHandler = (req, res) => {
    // get user's credentials from the JSON body
    const {username, password} = req.body
    if (!username) {
        // username not present
        res.status(401).end()
        return
    }

    // validate password
    const expectedPassword = users[username]
    if (!expectedPassword || expectedPassword !== password) {
        res.status(401).end()
        return
    }

    const sessionToken = uuid.v4() // random UUID for session token

    // expiry time as 1 hour after current time
    const now = new Date()
    const expiresAt = new Date(+now + 1 * 3600 * 1000)

    // create a session - will be added to database
    const session = new Session(username, expiresAt)
    sessions[sessionToken] = session // add session info to session map

    // in response, set a cookie on the client with the name "session_cookie"
    and the value as the UUID as well as expiration time
    res.cookie("session_token", sessionToken, { expires: expiresAt })
    res.end()
}
```

Figure 4.24 – Code for Creating and Setting Cookie Session Token

The code above clearly delineates how an object of the Session class is created and what attributes it has (a user's username and the expiration time of the cookie). Although these are stored in an array in the code above, in the actual implementation of the cookies, the sessions will be stored within the database in a separate table for the web resource. There are more validations that will need to be done than are actually shown, but the image includes the most basic validation – where a user's password isn't correct (this process has actually been outlined in Figure 4.15), or where a user's username isn't available. I have used a random UUID v4 (Universally unique identifier) for the session token because, although there is a chance of collisions with v4 (albeit miniscule), since the cookies expire after a relatively short time, the chance of 2 of the same UUIDs existing at a given moment are even lower – practically none.

Once logged in clients have cookies which store their session information, the website can should be able to authenticate any subsequent requests by verifying the user using the already stored cookies.

Authenticating Users via Cookies

The main idea here is that the user sends the website server their cookie which contains their session token. The user's information is then retrieved from the database which contains that session token, and finally, the process can be carried out. For a welcome message, this process it shown below.

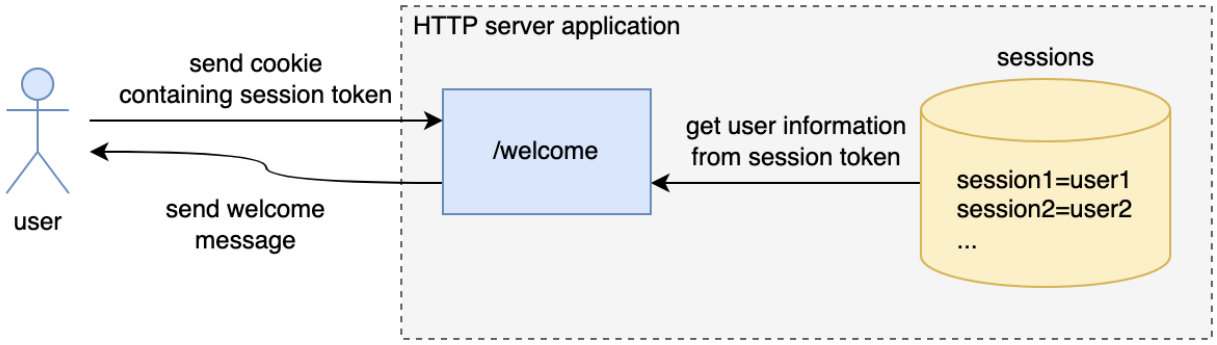

Figure 4.25 – Authenticating with Session Token (Session Cookie Authentication in Node.js (With Complete Examples) [Screenshot], 2022)

We can call the process shown above the "welcome handler" as it shows how the cookies and session tokens process the user logging in and displaying the welcome page (for this resource, the welcome page will be like the landing page shown in **Error! Reference source not found.**, b ut without the login/sign-up button, as the user will already be logged in).

A simple prototype for this system implemented in JavaScript (NodeJS) is shown below.

```
const welcomeHandler = (req, res) => {
    // no cookies means not authenticated
    if (!req.cookies) {
        res.status(401).end()
        return
    }

    // session token from requested cookies
    const sessionToken = req.cookies['session_token']
    if (!sessionToken) {
        res.status(401).end() // cookie not set
        return
    }

    // get the user's session
    userSession = sessions[sessionToken]
    if (!userSession) {
        // session token not in session map
        res.status(401).end()
        return
    }

    // session expired
    if (userSession.isExpired()) {
        delete sessions[sessionToken] // delete session token from map
        res.status(401).end()
        return
    }

    // all checks passed
    res.send(`Welcome  ${userSession.username}!`).end()
}
```

Figure 4.26 – Code for Authenticating Users via Session Cookies

As before, the necessary validation is in place – checking that the cookie is set, the user is authenticated, the session token exists, and, of course, the session isn't expired. If the cookie passes all the verifications, then we know that the cookie is valid and so, we can send the welcome message.

The reason that I've chosen to use session-based authentication is because it keeps the user's session secure in multiple ways: the session tokens are randomly generated, as mentioned previously, so it is almost impossible for a malicious attacker to use a brute-force method to get into the system, and if a user's session token were to be compromised in some way, then it cannot be used once it is expired (this is the reason that we set an expiry time limit). The actual amount of time that I will need to set the expiry timeout to cannot be decided until the website is fully functional, and I can test how long users spend (on average) on the website in one go.

Since session tokens expire quickly, and the expiry time is kept small (so that we can prevent against hackers), validated users would also need to re-login every time their token expires,

which could be extremely annoying for a user if they haven't left the website. Hence, there is a need to refresh session tokens in a secure way.

Refreshing Session Tokens

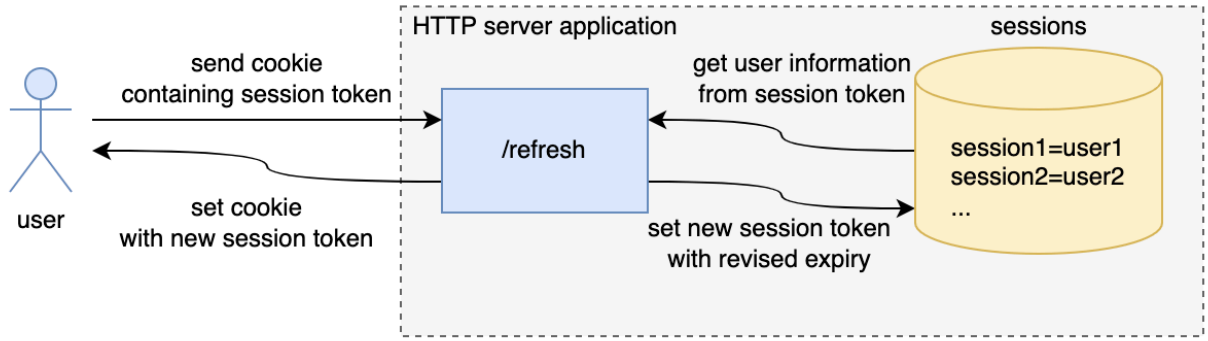

Figure 4.27 – Process to Refresh Session Tokens (Session Cookie Authentication in Node.js (With Complete Examples) [Screenshot], 2022)

The process above shows how the process of refreshing session tokens works from the user's browser and the website's backend.

Akin to the process of authenticating a user via cookies, this process requires the cookie to go from the user's browser to the website's database to retrieve the user's information from the session token. However, there is a new process which is to set the new session token with the new expiry time and add this to the database.

The way that this can be implemented with JavaScript (NodeJS) is shown below, but in the actual implementation of the website, NodeJS will be used in tandem with a SQL statement to access and write to the database.

```
const refreshHandler = (req, res) => {
    // SAME CODE AS WELCOME HANDLER GOES BELOW THIS COMMENT
    // ...
    // ...
    // CODE FROM WELCOME HANDLER STOPS HERE

    // create a new session token
    const newSessionToken = uuid.v4()

    // renew the expiry time
    const now = new Date()
    const expiresAt = new Date(+now + 1 * 3600 * 1000)
    const session = new Session(userSession.username, expiresAt)

    // add new session to map, delete the old session
    // this is done with an array right now, but will be implemented with SQL
    sessions[newSessionToken] = session
    delete sessions[sessionToken]

    // set session with renewed expiration time
    res.cookie("session_token", newSessionToken, { expires: expiresAt })
    res.end()
}
```

Figure 4.28 – Code for Refreshing Session Token

Majority of the code is the same as the welcome handler, but the old session token needs to be deleted and replaced by the new session.

The only thing remaining is the process of logging a user out from their session.

Implementing User Logout

This process is quite similar to logging in, except that we are now deleted cookie data from the user's browser rather than adding it. Once again, this process is shown in a diagram.

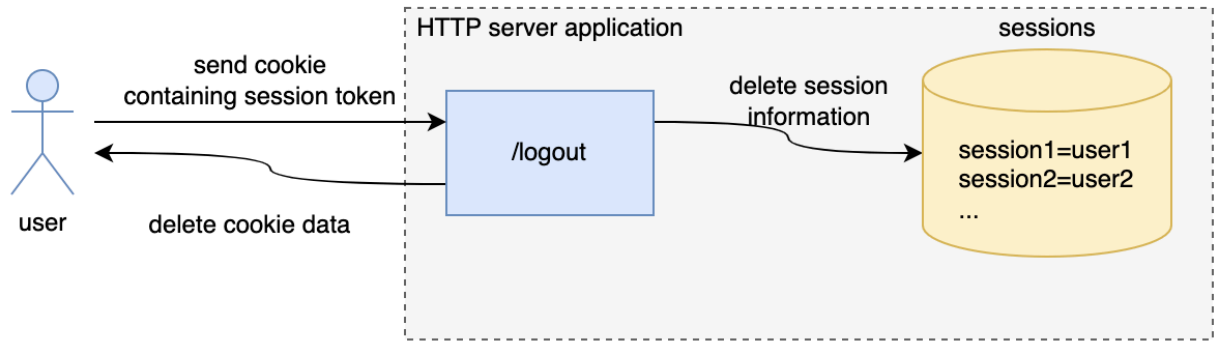

Figure 4.29 – Process of Logging a User Out of their Session

The cookie data is still required in order to find out which session needs to be removed from the database, but once this is done, all that needs to happen is that the cookie data needs to be deleted.

This implementation is shown in the code below for the logout handler.

```
const logoutHandler = (req, res) => {
    if (!req.cookies) { // no cookies means not authenticated
        res.status(401).end()
        return
    }

    const sessionToken = req.cookies['session_token']
    if (!sessionToken) { // session token not in session map
        res.status(401).end()
        return
    }

    delete sessions[sessionToken]
    res.cookie("session_token", "", { expires: new Date() })
    res.end()
}
```

Figure 4.30 – Code for Deleting User Session and Cookies

Now that the overall login system and cookie management have been designed, the next thing that I need to look at are the challenges.

Challenges

Link to Requirements

Requirement 10 essentially says that there will be 3 different pages to do with challenges. The first of these is for teachers to create new challenges, edit challenges, and delete challenges, as well as assign challenges to students or classes. The second one is for students to be able to view their assigned challenges, view their completed challenges, and complete their assigned challenges. The third and final one is a page for teachers to be able to see the challenges that have been assigned and which students have completed them, as well as the students' answers.

I will go over these 3 pages below, with a template of approximately how each will look, as well as prototype some of the code and look at the SQL statements that will be required for each page. However, before that, we need to finalise the database's tables and the links between them.

Final Database

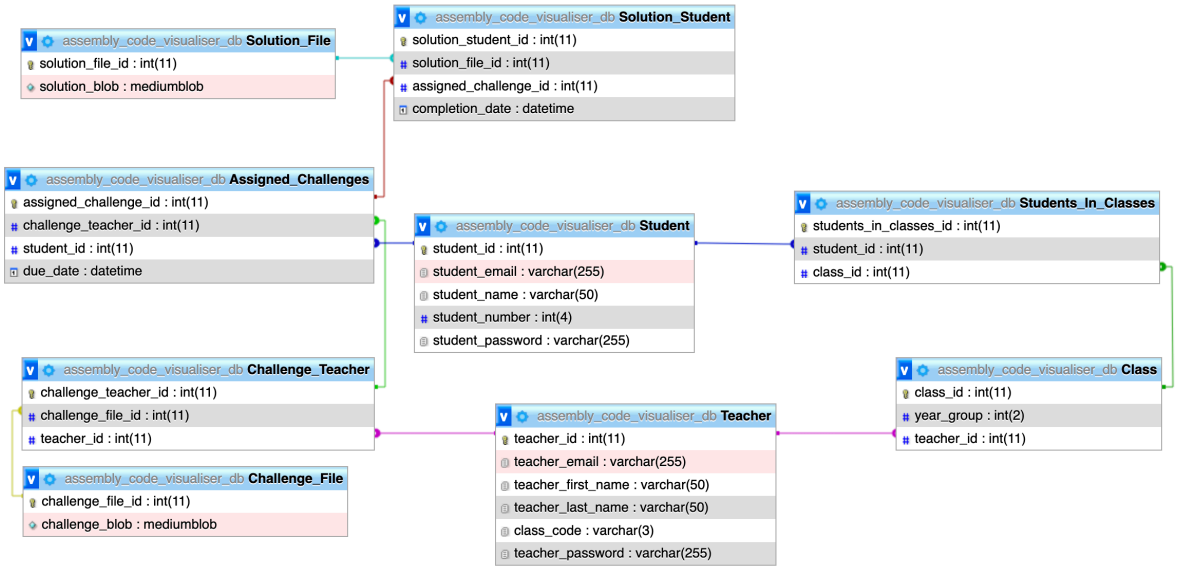

Figure 4.31 - Connected tables for the entire final database

We can see here that the login and sign-up tables have remained the same, while some changes have been made for the creation of answers to challenges, and completion of them.

Notice that I have decided to store the challenge data and solution data as MEDIUMBLOBs, instead of simply text files, because these can be compressed better. Also, BLOBs are file-like immutable, raw data, and we want the solutions to be immutable because students should not be able to change it afterwards. In addition, they can be read as text or binary data, or

converted into a ReadableStream so its methods can be used for processing the data. They can also represent data that isn't necessarily in a JavaScript-native format, which is quite useful.

Additionally, I have decided to put the MEDIUMBLOB in a separate table from the actual Solution_Student table for data reading since everything is loaded into RAM, so this being in a separate table means that only the necessary row will be loaded into the local device's memory, rather than all rows in that specific table (this is based on how data is actually retrieved from databases on local devices).

I will now look at the teacher page for creating, editing, and deleting challenges, as well as assigning challenges.

Teacher Creating and Assigning Challenges

Creating a New Challenge

The required fields for a new challenge will simply be the Challenge Title and Challenge Description. The maximum character length for the title should be 100 characters, while there is no maximum for the description, however, as both are required fields, there needs to be at least 1 character for each field.

I have created an illustration of what I think the page to create a new challenge should look like below.

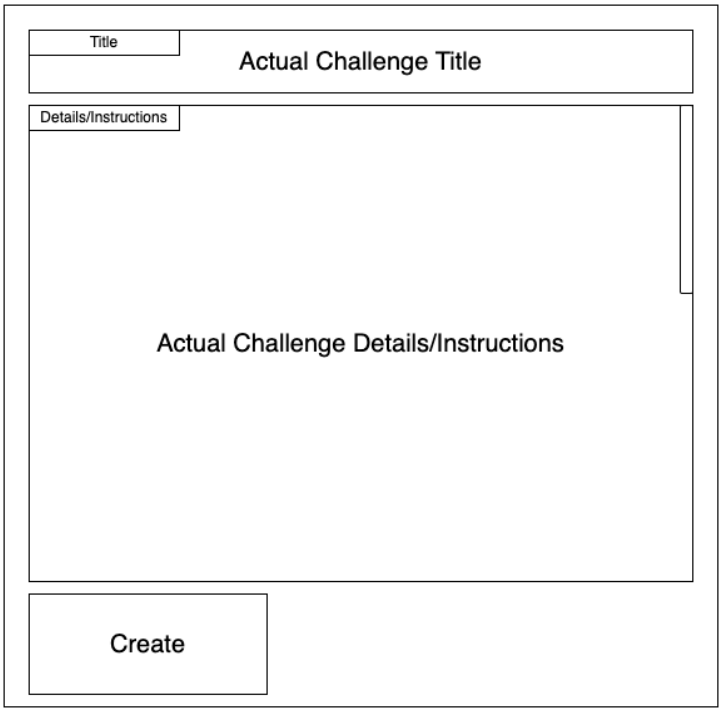

Figure 4.32 – Page to create a new challenge

Obviously, this page will require an INSERT SQL statement in order to actual put the data into the database, but first, the data needs to be converted into a BLOB, as previously discussed.

The way that I will do this is a relatively roundabout way of doing it, but it ensures that the user of the website will only ever be able to see the data for the challenge they just created. Otherwise, all previously created data would be viewable by any user in the **public/text_files/** directory because the challenge data needs to be stored in a .txt file before being converted into a BLOB.

```
var create_challenge_file_path = './public/text_files/teacher_create_challenge.txt';
try {
  if (fs.existsSync(create_challenge_file_path)) {
    // file exists
    console.log('Teacher Create Challenge file already exists');

    try {
      fs.unlinkSync(create_challenge_file_path);
      // file removed
      console.log('Teacher Create Challenge file deleted');
    } catch (error) {
      console.error(error);
    };
  }
} catch (err) {
  console.error(err);
};
var file_data = '###CHALLENGE TITLE###\n' + req.body.challenge_title.toString() + '\n###CHALLENGE DESCRIPTION###\n' +
req.body.challenge_description.toString();
    fs.writeFileSync(create_challenge_file_path, file_data,
    // callback function that is called after writing file is done
    function (err) {
      if (err) {
        console.log(err);
      };
      console.log('Data written to Teacher Create Challenge file successfully')
    });
var data_to_db = fs.readFileSync(create_challenge_file_path);
```

The above code shows the process of editing the file, writing the new data to the file, and converting it into a BLOB.

Then, it can be put into the database with the following SQL statements:

```
// put file as MEDIUMBLOB in db
'INSERT INTO Challenge_File (challenge_blob) VALUES (?);'
// record relation b/w teacher and challenge
'INSERT INTO Challenge_Teacher (challenge_file_id, teacher_id) VALUES ((SELECT challenge_file_id FROM Challenge_File
WHERE challenge_blob = ?), ?);';
```

The first SQL statement above puts the challenge file into the database, and the second one connects the challenge file to the teacher and challenge id.

Editing a Challenge

Similar to creating a challenge, the idea of a text file will be used, and all that needs to be done is to UPDATE the row with the new challenge file.

The following SQL could be used to do this:

```
1. // put file as MEDIUMBLOB in db
2. 'UPDATE Challenge_File SET challenge_blob = ? WHERE challenge_file_id = (SELECT challenge_file_id FROM
Challenge_Teacher WHERE challenge_teacher_id = ?);';
```

This also takes care of re-linking the challenge file to the teacher.

However, there is one other thing that needs to be done for this page. Though the page will be the same as the creating challenges page, the challenge title and challenge description boxes need to be pre-populated with the data that is already stored in the database, to make the process of editing easier.

This data can be retrieved from the database with the following SQL query:

```
1. // get challenge text from teacher_id
2. 'SELECT * FROM Challenge_File, Challenge_Teacher WHERE Challenge_File.challenge_file_id =
Challenge_Teacher.challenge_file_id AND Challenge_Teacher.teacher_id = ?;';
```

Now that that is done, we can look at the process of deleting a challenge.

Deleting a Challenge

All that we really need to do here is to write the SQL query.

```
1. // delete challenge
2. 'DELETE FROM Challenge_File WHERE challenge_file_id = (SELECT challenge_file_id FROM Challenge_Teacher WHERE
challenge_teacher_id = ?);';
```

The query above could be used for this, and since I will be using a router for each individual challenge, I can get exactly which challenge_teacher_id needs to be deleted.

Next up is the process of assigning challenges.

Assigning a Challenge

Though this will be on the same page as the place where we can create, edit, and delete challenges, there will be buttons that link us to separate views.

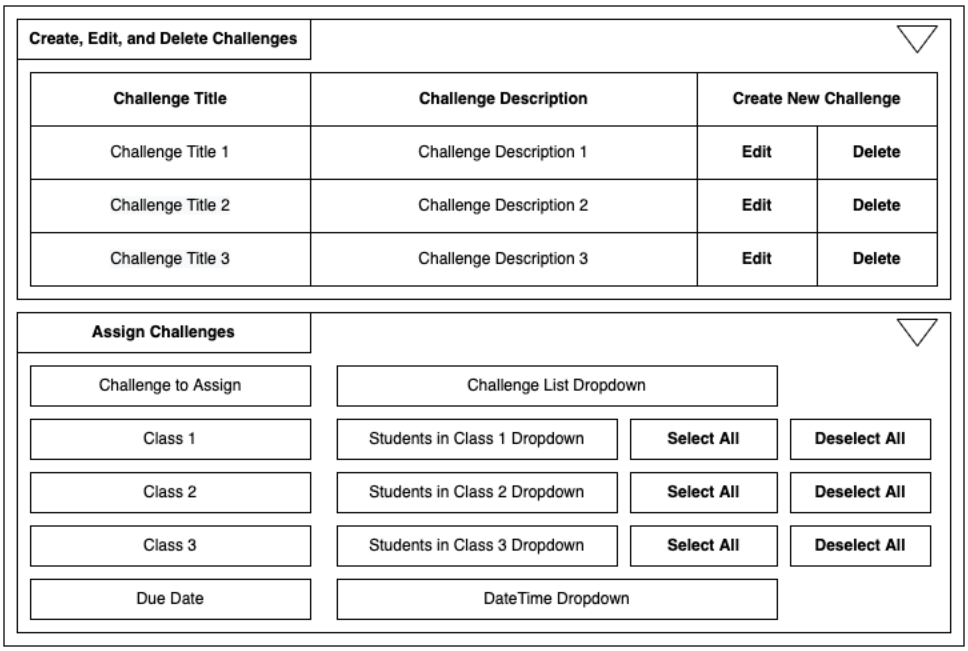

Figure 4.33 - Creating, Editing, Deleting, and Assigning Challenges Page Example

An example of the page layout is shown above.

We will be focusing on the Assign Challenges section right now because we have already covered the process of creating, editing, and deleting challenges in the previous section. Firstly, we need to look at the query that can populate the student in class dropdown menu, as we have already seen the query that can populate the challenge list dropdown menu above.

```
// get classes of currently signed in teacher
'SELECT CONCAT(year_group, class_code), class_id FROM Class, Teacher WHERE Class.teacher_id = Teacher.teacher_id AND Class.teacher_id = ?;';

// get students in each class of currently signed in teacher
'SELECT CONCAT(student_name, student_number) FROM Student, Students_In_Classes WHERE Student.student_id = Students_In_Classes.student_id AND class_id = ?;';
```

The first query above gets a list of all the classes and the class ids that belong to the currently signed in teacher. Following this, the next query can be used to get the students in each class from the class ids retrieved above. Using this, we can populate the class student dropdowns.

After this, we need to actually assign the challenge. However, before that, we need to check if any of the students have already been assigned the challenge we are trying to assign, and if so, display an error message.

```
// get the students who have already been assigned the specific challenge
```

```
2. 'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id FROM Assigned_Challenges, Challenge_Teacher, Student WHERE Assigned_Challenges.challenge_teacher_id = Challenge_Teacher.challenge_teacher_id AND Assigned_Challenges.student_id = Student.student_id AND Assigned_Challenges.challenge_teacher_id = (SELECT challenge_teacher_id FROM Challenge_Teacher WHERE challenge_teacher_id = ?);';
```

If we already have the challenge teacher id, then we can use the SQL query above to retrieve all the students who have been assigned this challenge already, and then compare this to the students who have been selected.

Once this is done, we can actually assign the challenge now, which can be done using the INSERT SQL statement below.

```
1. 'INSERT INTO Assigned_Challenges (challenge_teacher_id, student_id, due_date) VALUES (?, (SELECT student_id FROM Student WHERE student_name = ? AND student_number = ?), ?);';
```

Having now looked at all of this, we can move on to the second page to do with challenges, which is a student's view of their assigned challenges, as well as the ability to complete the challenges.

Student Viewing and Completing Assigned Challenges
Firstly, I have drawn a diagram of how I expect the page to look.

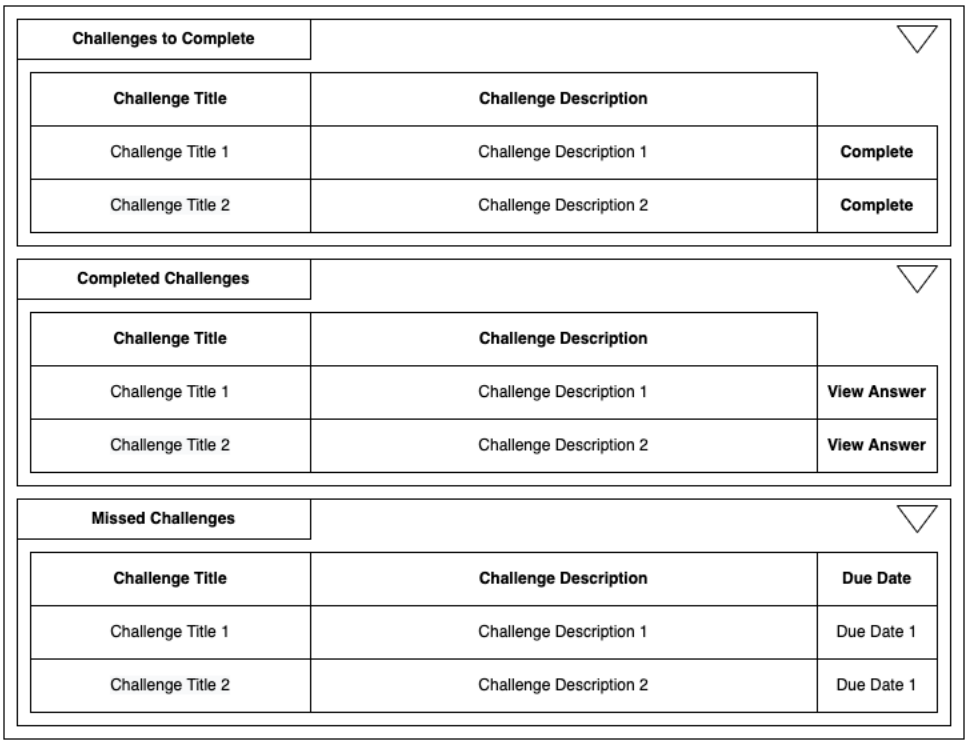

Figure 4.34 – Example of Students' View Student Challenges Page

I will now go through the three SQL queries to retrieve the data for this page, followed by the queries for completing a challenge and viewing a completed challenge.

```
1. // all challenges that can still be solved by the student, the due date is after the current time
2. 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE Challenge_File.challenge_file_id =
Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id =
Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND
Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student,
Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) AND
Assigned_Challenges.due_date >= NOW() ORDER BY Assigned_Challenges.due_date ASC;';
```

The query above is for getting all the challenges that have yet to be completed by the student. The query for getting all the missed challenges is almost the same, but the only difference is that the due date has to be before the current time. This query is shown below:

```
1. // get all challenges that were due before the current date - i.e. missed
2. 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE Challenge_File.challenge_file_id =
Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id =
Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND
Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student,
Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) AND
Assigned_Challenges.due_date < NOW() ORDER BY Assigned_Challenges.due_date ASC;';
```

Next, the query for getting the challenges that have already been completed is slightly similar to the previous two, but we do not need to consider the time or due date now.

```
1. 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges, Solution_Student, Solution_File WHERE
Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id =
Assigned_Challenges.challenge_teacher_id AND Solution_Student.assigned_challenge_id =
Assigned_Challenges.assigned_challenge_id AND Solution_File.solution_file_id = Solution_Student.solution_file_id AND
Assigned_Challenges.student_id = ? AND Assigned_Challenges.assigned_challenge_id IN (SELECT
Solution_Student.assigned_challenge_id FROM Solution_Student, Assigned_Challenges WHERE
Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) ORDER BY
Assigned_Challenges.due_date ASC;';
```

Having looked at the 3 main queries on the main page, I will look at the queries to complete challenges and to view completed challenges, although these are extremely similar to the query to create challenges and the functionality in the edit challenges page where the text is loaded into the text fields from the database's BLOB files.

```
1. 'INSERT INTO Solution_File (solution_blob) VALUES (?);';
2. 'INSERT INTO Solution_Student (solution_file_id, assigned_challenge_id, completion_date) VALUES ((SELECT solution_file_id
FROM Solution_File WHERE solution_blob = ?), ?, NOW());';
```

The 2 queries above are used for completing the challenges. Similar to the logic of creating a new challenge, a solution is created and inserted into the database as a BLOB file. The 2nd line

above takes care of linking the completed solution to the student and the specific challenge that was assigned.

In order to view the completed challenges, it is the same as the previous section, but the only differences this time are the fact that the text fields should be non-editable and the button, instead of allowing the user to update the challenge, should simply be a “go back” button.

Now that we have looked at the student side of the challenges, there is one more page that we need to look at for teachers.

Teacher Viewing Students’ Assigned Challenges

This will work by first, on a page similar to the students’ challenges page, displaying two fields. One being all the assigned challenges, and the other being all the challenges that have been created by the teacher, but not assigned. The queries for this are as follows:

```
1. // all challenge_teacher_id's that have been assigned to some student
2. 'SELECT * FROM Challenge_Teacher, Challenge_File WHERE Challenge_Teacher.challenge_file_id =
Challenge_File.challenge_file_id AND Challenge_Teacher.challenge_teacher_id IN(SELECT challenge_teacher_id FROM
Assigned_Challenges) AND Challenge_Teacher.teacher_id = ?;';
3. // all challenge_teacher_id's that have not been assigned to some student
4. 'SELECT * FROM Challenge_Teacher, Challenge_File WHERE Challenge_Teacher.challenge_file_id =
Challenge_File.challenge_file_id AND Challenge_Teacher.challenge_teacher_id NOT IN (SELECT challenge_teacher_id FROM
Assigned_Challenges) AND Challenge_Teacher.teacher_id = ?;';
```

The first one shows us the challenges that have been assigned, and the other shows the opposite. The only difference between these 2 queries is the IN vs NOT IN, because the NOT obviously negates the entire statement (as we have a sub-query within the query).

For all the challenges that have been assigned, there will be a button to open onto a new page. I will call this the “more information” button. From here, the new page will contain 4 fields: 1st, the challenge title and challenge description; 2nd, the students that have submitted the challenge; 3rd, the students that are yet to submit the challenge; and finally, the students that have missed their deadline for the challenge.

The first field does not need to be gone over, as we have gone through the queries to display the challenge data multiple times before.

The other 3 queries are shown below:

```
1. // get all students who have been assigned challenge_teacher_id and solved
2. 'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id, Assigned_Challenges.due_date,
Solution_Student.completion_date FROM Solution_Student, Solution_File, Student, Assigned_Challenges WHERE
Solution_Student.solution_file_id = Solution_File.solution_file_id AND Assigned_Challenges.assigned_challenge_id =
Solution_Student.assigned_challenge_id AND Assigned_Challenges.student_id = Student.student_id AND
```

```
Assigned_Challenges.challenge_teacher_id = ? AND Assigned_Challenges.assigned_challenge_id IN (SELECT
assigned_challenge_id FROM Solution_Student);';
3. // get all students who have been assigned challenge_teacher_id and not solved
4. 'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id, Assigned_Challenges.due_date
FROM Student, Assigned_Challenges WHERE Assigned_Challenges.student_id = Student.student_id AND
Assigned_Challenges.challenge_teacher_id = ? AND Assigned_Challenges.assigned_challenge_id NOT IN (SELECT
assigned_challenge_id FROM Solution_Student);';
5. // get answer blob
6. 'SELECT * FROM Solution_File, Solution_Student WHERE Solution_File.solution_file_id = Solution_Student.solution_file_id
AND Solution_Student.assigned_challenge_id = (SELECT assigned_challenge_id FROM Assigned_Challenges WHERE
challenge_teacher_id = ? AND student_id = ?);';
```

The first query above gets all the challenges that have already been solved. The second query gets all the challenges that have not been solved. Again, the only difference is the NOT. The way that I will differentiate between missed challenges and challenges yet to solve is by using the JavaScript “new Date()” variable, and comparing it, similarly to above, with < and >.

The third query is just a query to show the challenge data for the completed challenges and will be the same page as the student page to view the completed challenges.

We can now mark requirement 10 as completed which is the final one.

Now that all of the requirements have been accounted for, I will proceed to the Technical Solution section.

Chapter 5 Technical Solution

Technical Skills

The following 3 tables refer to the complex models, algorithms, and coding techniques I have used in my project. I have referenced where in the code each appears to make it easier to find evidence for each skill.

Technical A Skills

Group	Model (including data model/structure)	Page number(s) evidenced in my NEA	Algorithms	Page number(s) evidenced in my NEA
A	Complex data model in database (eg several interlinked tables)	252	Cross-table parameterised SQL	223
			User/CASE-generated DDL script	259
	Files(s) organised for direct access	152	Complex List operations	260
			Hashing	222
	Complex user-defined use of object orientated programming (OOP) model, eg classes, inheritance, composition, polymorphism, interfaces	167	Async/Await functions	233, 234
			Regular expressions	216
			Cross-table queries	263
	Complex Client-Server Model	152, 162		

Table 5.1 – Group A complexities

Technical B Skills

Group	Model (including data model/structure)	Page number(s) evidenced in my NEA	Algorithms	Page number(s) evidenced in my NEA
B	Multi-dimensional arrays	249	Single table or non-parameterised SQL	224
	Dictionaries	194		
	Text files	203	Writing and reading from files	203

Table 5.2 – Group B complexities

Technical C Skills

Group	Model (including data model/structure)	Page number(s) evidenced in my NEA
C	Single-dimensional arrays	178, 179
	Appropriate choice of simple data types	167

Table 5.3 – Group C complexities

Coding Practices/Styles

Coding Styles

This table below is similar to the 3 Technical Skills tables in that they are recommended and generally good to use. I have similarly referenced where each row lies in my code, but note that almost all everywhere, throughout my code.

Style	Characteristic	Page number(s) evidenced in my NEA
Excellent	Modules (subroutines) with appropriate interfaces	170
	Loosely coupled modules (subroutines) – module code interacts with other parts of the program through its interface only	180
	Cohesive modules (subroutines) – module code does just one thing	179
	Modules (collections of subroutines) – subroutines with common purpose grouped	220
	Defensive programming	203
	Good exception handling	233
Good	Good use of local variables	176
	Managed casting of types	179
	Use of constants	201
	Appropriate indentation	154
	Self-documenting code	156
	Consistent style throughout	154
	File paths parameterised	159
Basic	Meaningful identifier names for variables and subroutines	156, 202
	Annotation used effectively where required	202

Table 5.4 – General coding styles

My Coding Practices

Below is a table of my personal coding practices that I have adhered to throughout the entire project.

Coding Practice	Page number(s) evidenced in my NEA
Variables and functions will be **aptly named** to describe what their purpose is, **apart from counter values** in relatively small for loops (the letter 'i' is okay for this).	168

Variables and functions will all be written using **snake_case**.	169
There should be consistent use of **1 whitespace character** between operator and operands, e.g. “x == 5”.	171
All **modules, packages, and libraries** should be imported at the top of files.	155
Where possible, **partial files** will be used where lots of sections of code are used and repeated in multiple different files.	152
Comments should only be used for **overviews or general explanations**, meaning there can be large gaps between comments if I feel the code is quite self-explanatory. However, there **can also be commented out chunks of codes** for approaching a particular part of a solution in a different way, e.g. asynchronously / synchronously writing to and reading from files.	172
Comments should be located **above the line** they are describing, or to the **right of the line** we are talking about, e.g. during the else section of an if-else clause.	158
Efficiency should always be prioritized over readability because comments can explain what is happening.	274
All files should end with an empty line.	154

Table 5.5 – My Coding Practices

I will now go on to explain my technical solution below. Referencing the client requirements, I will show and explain snippets of code that satisfy the complexities from above, the coding practices, and the final requirements, as well as referencing the relevant design section. The full source code can be found in Appendix I.

Working File Directory

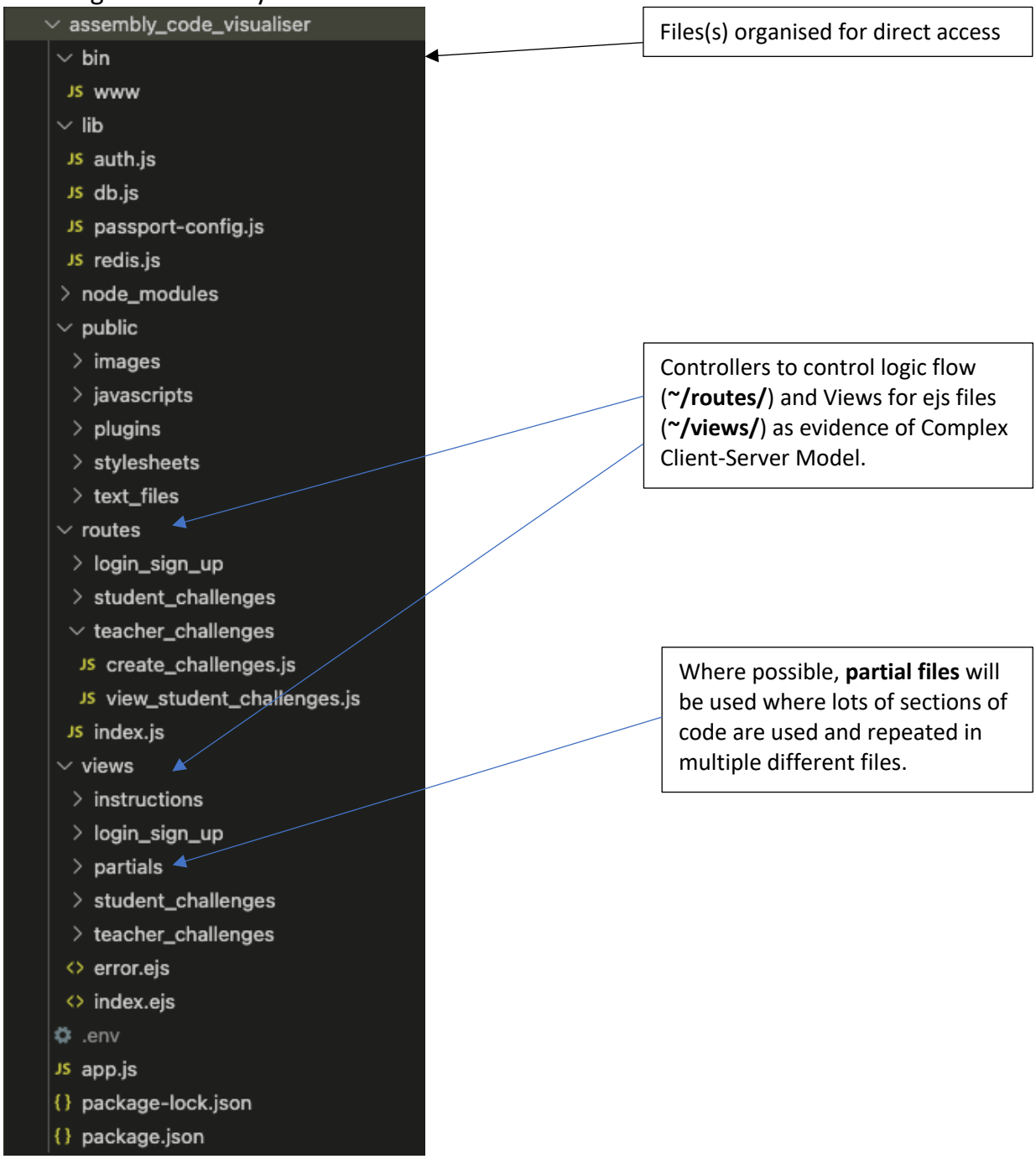

Figure 5.1 – Working directory overview

Figure 5.1 shows the working file directory, in which all the files for this project are located. To run NodeJS and refer to other files, this is necessary, and from here on out, I will be referring to the **assembly_code_visualiser/** directory as the home directory, **~/**. Having organised and

grouped important modules that work together into separate folders, it becomes easier to access and understand where I am currently working.

Database Table Organisation

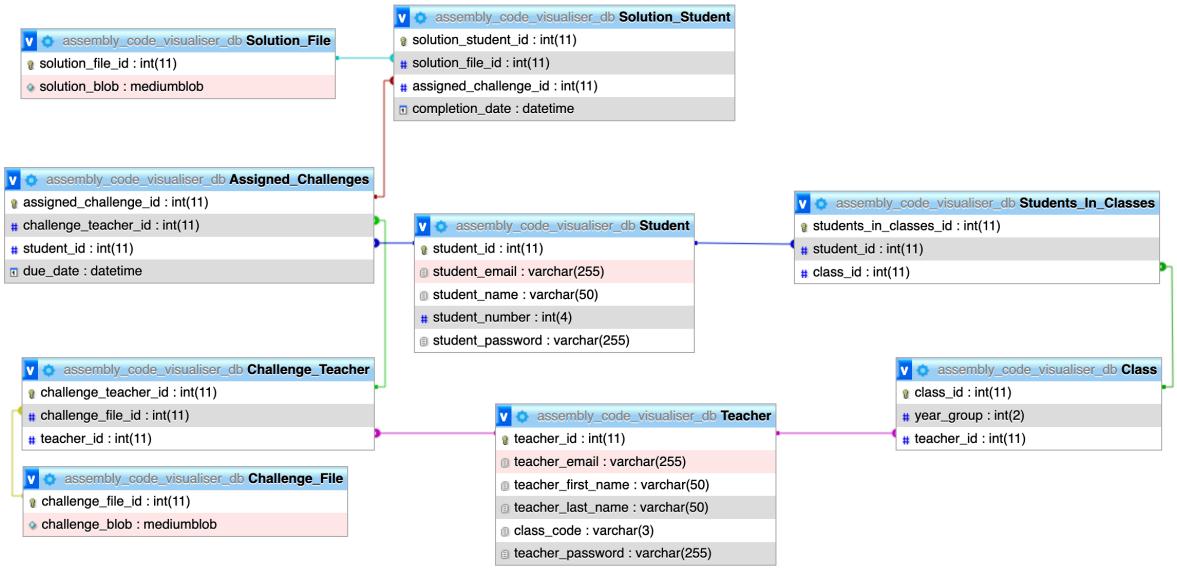

Figure 5.2 – Database Designer Image from PHPMyAdmin

The image above shows the actual implementation of the database used for this entire project. This was already discussed at length in the Documented Design section, including information about why I chose to do it this way, as well as how I actually created the database tables using SQL commands, so I do not feel the need to further explain this image.

NodeJS and EJS Default Files

Recall that in my Documented Design section, I decided to build my project using NodeJS, with the EJS (Embedded JavaScript Templating) engine. Some of the files from above were automatically created by NodeJS and EJS when creating the project, and I will explain each file's use, as well as what each now contains below.

~/node_modules/

This folder contains all of the default plugins and libraries, as well as the ones that I have added, for NodeJS. **~/node_modules/.package-lock.json** is a JSON file that contains the names of each of these modules used, as well as their versions, dependencies, integrity checks, and download location links.

~/views/

As mentioned in the Documented Design section, this directory contains all EJS/HTML files that will be rendered by the routes and their GET methods. It is further split up into folders (which will be seen later), but the actual files all have the extension **.ejs** because they, though technically are HTML code, contain EJS constructs, for passing variables from the backend into them.

~/views/error.ejs

```
<!-- default error message template -->
<!DOCTYPE html>
<html lang="en">

<head>
    <% include partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include partials/menu %>

        <div class="row">
            <div class="col-md-12">

                <h1><%= message %></h1>
                <h2><%= error.status %></h2>
                <pre><%= error.stack %></pre>

            </div>
        </div>

    </div>
</body>

<% include partials/script %>

</html>

```

Appropriate indentation

All files should end with an empty line – Consistent style throughout

This file is the default for when any error messages are to be shown. It simply shows a page that contains a red box with text inside it, containing an error message.

~/views/index.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
```

```
5.    <% include partials/head %>
6.    <% include partials/main_visualisation_head %>
7. </head>
8.
9. <body>
10.    <div class="container-fluid">
11.
12.        <% include partials/menu %>
13.        <!-- route to ejs template with main assembly visualisation -->
14.        <% include partials/main_visualisation %>
15.
16.    </div>
17. </body>
18.
19. <% include partials/script %>
20.
21. </html>
22.
```

This file contains the default page that should load up when the NodeJS file is run, i.e. the home directory of the website, **/**. The only thing I have altered from the original is to include an EJS reference to use **~/views/partials/main_visualisation.ejs**, which will be looked at later on, as well as **~/views/partials/main_visualisation_head.ejs**, which simply links the file to a separate CSS file (**~/public/stylesheets/main_visualisation_style.css**) and will not be specifically talked about as it does not add much information to do with Technical skills and good coding practices.

~/views/partials/

This directory contains snippets of lines of code that are used across multiple files. They are there to ensure that the entire project has been modularised.

~/routes/

As mentioned in the Documented Design section, this directory contains all JS files that will handle the backend of the application, namely GET and POST methods, as well as all processing. Once again, it is further split up into folders (the same as those in **~/views/**).

~/routes/index.js

```
1. // all libs/modules
2. var express = require('express');
3. var router = express.Router();
4.
5. var auth = require('../lib/auth');
6.
7. /* GET home page. */
8. router.get('/', function(req, res, next) {
9.
```

All **modules, packages, and libraries** should be imported at the top of files.

```
10.   console.log('req.user:', req.user);
11.
12.   res.locals.message = req.flash();
13.
14.   // no user
15.   if (typeof req.user == 'undefined') {
16.     res.render('index', {
17.       title: 'Home',
18.       menu_id: 'home',
19.       session_id: req.sessionID,
20.       session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
21.     });
22.   } else { // user exists
23.     res.render('index', {
24.       title: 'Home',
25.       menu_id: 'home',
26.       role: req.user.role,
27.       email: req.user.email,
28.       session_id: req.sessionID,
29.       session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
30.     });
31.   }
32.
33. });
34.
35. /* GET instruction set page */
36. router.get('/instruction_set', function (req, res, next) {
37.
38.   res.locals.message = req.flash();
39.
40.   if (typeof req.user == 'undefined') {
41.     res.render('instructions/instruction_set', {
42.       title: 'Instruction Set',
43.       menu_id: 'instruction_set',
44.     });
45.   } else {
46.     res.render('instructions/instruction_set', {
47.       title: 'Instruction Set',
48.       menu_id: 'instruction_set',
49.       role: req.user.role,
50.     });
51.   }
52. });
53.
54. /* GET how to use page */
55. router.get('/how_to_use', function (req, res, next) {
56.
57.   res.locals.message = req.flash();
58.
59.   if (typeof req.user == 'undefined') {
60.     res.render('instructions/how_to_use', {
61.       title: 'How To Use This Resource',
62.       menu_id: 'how_to_use',
63.     });
64.   } else {
```

Self-documenting code

Meaningful identifier names for variables and subroutines

```
65.         res.render('instructions/how_to_use', {
66.           title: 'How To Use This Resource',
67.           menu_id: 'how_to_use',
68.           role: req.user.role,
69.         });
70.     }
71. });
72.
73. /* POST logout button request */
74. router.post('/logout', function(req, res) {
75.     res.locals.message = req.flash();
76.     req.logout(function(err) {
77.
78.         if (err) {
79.           console.log(err);
80.           return next(err);
81.         }
82.
83.         // remove cookies and destroy session
84.         req.session.destroy((error) => {
85.
86.           if (error) {
87.             console.log(error);
88.             return next(error);
89.           }
90.
91.           res.redirect('/login');
92.         });
93.     });
94. });
95.
96. module.exports = router;
97.
```

This file is usually the default routing file for loading and rendering the default **/** page. At the beginning of the file, I have imported all libraries and modules that will be used in this file. From there, what to do when the GET method for the default **/** page is called is defined, which essentially just renders **~/views/index.ejs**. If a user is logged in, this needs to be passed as an argument into the EJS template, but if not, nothing needs to be passed in, and the variable will be considered "undefined", which is used a lot throughout the code in the EJS template files for validation. This route also takes care of the GET method for the 2 information-filled pages, **~/views/instructions/how_to_use.ejs** and **~/views/instructions/instruction_set.ejs**, which, again, will be looked at later on. In addition, the POST method for when the logout button is clicked on the navbar (which only appears after being signed in, as we'll soon see), is contained here.

The original file generated by NodeJS does not include the two above GET methods or the logout POST method. For all original files after the initialisation of the NodeJS and EJS template

project, https://github.com/raghav2005/Assembly-Code-Visualiser contains all my commit message and history of the entire project.

~/app.js

```
// all packages
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var flash = require('express-flash');
var session = require('express-session');

var passport = require('passport');
var method_override = require('method-override');

// redis for cookie and session management
var connectRedis = require('connect-redis');
var redisClient = require('./lib/redis');

// all routes
var index_router = require('./routes/index');
var login_router = require('./routes/login_sign_up/login');
var sign_up_router = require('./routes/login_sign_up/sign_up');
var teacher_challenges_router = require('./routes/teacher_challenges/create_challenges');
var student_challenges_router = require('./routes/student_challenges/view_set_challenges');
var teacher_challenges_view_student_challenges_router =
require('./routes/teacher_challenges/view_student_challenges');

// use dotenv (local configurations)
require('dotenv').config()

// initialise express
var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

// initialise all packages
app.use(logger('dev'));
app.use(express.json());

// allow getting information from forms (sign up & login)
app.use(express.urlencoded({ extended: true }));

// set static directory
app.use(express.static(path.join(__dirname, 'public')));

app.use(cookieParser());
```

Comments should be located **above the line** they are describing, or to the **right of the line** we are talking about, e.g. during the else section of an if-else clause.

```
48. // redis setup
49. var RedisStore = connectRedis(session);
50. // session
51. app.use(session({
52.     secret: process.env.SESSION_SECRET_KEY,
53.     cookie: {
54.         secure: false, // if true, would only transmit cookies over https
55.         httpOnly: false, // if true, would prevent client-side JS from reading cookies
56.         maxAge: 7200000, // session max age in milliseconds (2 hours)
57.     },
58.     store: new RedisStore({ client: redisClient }),
59.     saveUninitialized: false,
60.     resave: false,
61. }));
62.
63. app.use(flash());
64.
65. // set up passport
66. app.use(passport.initialize());
67. app.use(passport.session());
68. var initializePassport = require('./lib/passport-config');
69. initializePassport(passport);
70.
71. app.use(method_override('_method'))
72.
73. // initialise all routes
74. app.use('/', index_router);
75. app.use('/login', login_router);
76. app.use('/sign_up', sign_up_router);
77. app.use('/logout', index_router);
78. app.use('/instruction_set', index_router);
79. app.use('/how_to_use', index_router);
80. app.use('/create_challenges', teacher_challenges_router);
81. app.use('/create_challenges/delete', teacher_challenges_router);
82. app.use('/create_challenges/assign', teacher_challenges_router);
83. app.use('/create_challenges/edit', teacher_challenges_router);
84. app.use('/view_set_challenges', student_challenges_router);
85. app.use('/view_student_challenges', teacher_challenges_view_student_challenges_router);
86.
87. // catch 404 and forward to error handler
88. app.use(function (req, res, next) {
89.     next(createError(404));
90. });
91.
92. // error handler
93. app.use(function (err, req, res, next) {
94.     // set locals, only providing error in development
95.     res.locals.message = err.message;
96.     res.locals.error = process.env.NODE_ENV === 'development' ? err : {};
97.
98.     // render the error page
99.     res.status(err.status || 500);
100.     res.render('error');
101. });
102.
```

File paths parameterised

```
103. module.exports = app;
104.
```

Again, as mentioned in the Documented Design section, this file is the main controller file for the entire application. Firstly, as usual, all libraries and modules to be used in the file are defined at the top.

From there, a multitude of router variables are defined, that contain references to all the JS files in **~/routes/**, so that once again, the project has been modularised and not all GET/POST method request handlers are located in the same file, since not all directories or resources require the same packages or libraries either.

The **~/public/** directory is also set as the static folder, which will be covered later.

I then had to use and define RedisStore cache (Remote Dictionary Server), a fast, open source, in-memory, key-value data store, to store the session and cookies for logged in users, defining a maximum age for how long a user should be allowed to stay logged in (2 hours). From there I included the passport library and my configuration for it to allow for multiple login account types – more information about this later, when talking about **~/lib/passport-config.js**.

From here, I assigned the previously defined route files to the possible routes that could be entered into the search bar, so that all URLs use the specific routes based on the different modules. After this, I had to define an error handler, and the final line is a NodeJS convention that allows the use of multiple routers by exporting the entire app and adding it to **module.exports**, which acts like a global variable, but is not one.

~/lib/
This directory contains all configuration files, to do with databases, as well as any configurations needed to do for external libraries.

~/bin/
This directory contains all start-up scripts, which are run when starting the express app as a webserver.

~/bin/www

```
1. #!/usr/bin/env node
2.
3. /**
4.  * Module dependencies.
5.  */
6.
7. var app = require('../app');
```

```
var debug = require('debug')('assembly-code-visualiser:server');
var http = require('http');

require('dotenv').config()

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT);
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

```

```
63.  var bind = typeof port === 'string'
64.    ? 'Pipe ' + port
65.    : 'Port ' + port;
66.
67.  // handle specific listen errors with friendly messages
68.  switch (error.code) {
69.    case 'EACCES':
70.      console.error(bind + ' requires elevated privileges');
71.      process.exit(1);
72.      break;
73.    case 'EADDRINUSE':
74.      console.error(bind + ' is already in use');
75.      process.exit(1);
76.      break;
77.    default:
78.      throw error;
79.  }
80. }
81.
82. /**
83.  * Event listener for HTTP server "listening" event.
84.  */
85.
86. function onListening() {
87.  var addr = server.address();
88.  var bind = typeof addr === 'string'
89.    ? 'pipe ' + addr
90.    : 'port ' + addr.port;
91.  debug('Listening on ' + bind);
92. }
93.
```

This file essentially sets up the port for running the webserver with Express, as well as a HTTP event listener, which allows all interactivity once web pages load up. It defines using **~/app.js** as the main handler for the application, and also configures the assembly-code-visualiser server, as that is the name of the directory and the application, which in this case is used as a package.

Line 11 above uses the dotenv package so that hard-coded values for variables such as the port are not required, and instead, only contained inside a single hidden environment file, **~/.env**.

~/.env

```
1. DB_HOST = '127.0.0.1'
2. DB_USER = 'root'
3. DB_PASSWORD = '(this is hidden for security reasons)'
4. DB_DATABASE = 'assembly_code_visualiser_db'
5. DB_PORT = 8889
6.
7. REDIS_HOST = '127.0.0.1'
8. REDIS_PORT = 6379
9. REDIS_DATABASE = 0
```

Use of localhost, database, and redis as evidence for Complex Client-Server Model

```
10.
11. SESSION_SECRET_KEY = '(this is hidden for security reasons)'
12. PORT = 3000
13.
14. NODE_ENV = 'development'
15.
```

All constants used in other configuration files are contained here, for setting up the database, Redis, and the web server port. Since I use MAMP to run the web server, this port has to be the same as the default that MAMP defines (3000), and these are used to connect to the localhost server.

~/public/

This directory is where, as mentioned above, all static files, such as images, CSS files, JavaScript files (for common things like the ToolTip on hover), and text files (for the Assembly Code pre-created programs) are stored.

~/public/stylesheets/

This directory contains all CSS files for the entire project. Since CSS and styling is not directly linked to the Implementation of the solution (and only somewhat to requirement 1, which will be shown in the Testing section and cannot explicitly be shown in this section), I will not be going over this directory or its contained files.

~/public/plugins/

This directory contains all plugins that I have externally had to download and use in my files because they did not meet the dependencies of the other NodeJS libraries, packages, and modules. I had to alter a few of these files slightly to work with these dependencies, but I will not be explaining those changes as they are not directly relevant to the project.

Configurations

Recall requirement 1:

1. The resource should be a Web Application which is coded using NodeJS as the framework and MySQL as the database.
    a. The website and all of its components should be easy to use, navigate, and understand.

As mentioned above, 1.a cannot be shown in this section, and we have already shown the default files for NodeJS, but the configuration for the MySQL database, as well as Redis for storing the cookies and session information can be shown and is done so below.

~/lib/db.js

```
var mysql = require('mysql');

require('dotenv').config()

// from .env
var DB_HOST = process.env.DB_HOST
var DB_USER = process.env.DB_USER
var DB_PASSWORD = process.env.DB_PASSWORD
var DB_DATABASE = process.env.DB_DATABASE
var DB_PORT = process.env.DB_PORT

// connect to db, creation of tables done in phpMyAdmin UI
var connection = mysql.createConnection({
    host: DB_HOST,
    user: DB_USER,
    password: DB_PASSWORD,
    database: DB_DATABASE,
    port: DB_PORT
})

connection.connect(function (error) {
    if (!!error) {
        console.log('Could not establish a connection with the database:' + error);
    } else {
        console.log('Connected to Database..!');
    }
});

module.exports = connection;

```

Using **~/.env**, the database host, username, password, database name, and database port are stored in the DB_ variables, which are then used to create the connection to the database using

the MySQL package of NodeJS. Once the connection is established, the connection will need to be used in other files, so it is added to **module.exports**.

~/lib/redis.js

```
1. // redis for cookie and session management
2. var { createClient } = require('redis');
3.
4. require('dotenv').config()
5.
6. // from .env
7. var REDIS_HOST = process.env.REDIS_HOST
8. var REDIS_PORT = process.env.REDIS_PORT
9. var REDIS_DATABASE = process.env.REDIS_DATABASE
10.
11. // redis configuration
12. var redisClient = createClient({
13.     legacyMode: true,
14.     host: REDIS_HOST,
15.     port: REDIS_PORT,
16.     database: REDIS_DATABASE
17. });
18.
19. redisClient.connect().catch(console.error);
20.
21. redisClient.on('error', function (err) {
22.     console.log('Could not establish a connection with redis:' + err);
23. });
24. redisClient.on('connect', function (err) {
25.     console.log('Connected to Redis..!');
26. });
27.
28. module.exports = redisClient;
29.
```

Similar to connecting to the MySQL database, the Redis database uses the .env variables stored in the REDIS_ variables in the local file in order to create the client. The Redis database itself never needs to be added to or edited unless I want to delete all previous cookies, which is unnecessary until too many sessions are stored, but Redis anyways has a fallback for that case, so I have not needed to access Redis through other means, like I did with the MySQL database and phpMyAdmin.

Main Visualisation / Home Page

Recall requirements 2, 3, 4, 5, 7, and 8:

2. Users should be allowed to write code in the Assembly Language and be able to run the machine-code operations which do the following:
    a. Load
    b. Add
    c. Subtract
    d. Branching (conditional & unconditional)
    e. Compare
    f. Halt
3. Users should be able to view visualisations of the assembly code that they have written, with the following components included in the visualisation (being smoothly changed and animated on the website):
    a. Arithmetic Logic Unit
    b. Control Unit
    c. Clock
    d. Buses
    e. Dedicated Registers:
        i. Program Counter
        ii. Current Instruction Register
        iii. Memory Address Register
        iv. Memory Buffer Register
        v. Status Register
4. There should be verbose output which displays any and all debugging for erroneous Assembly code that is written.
    a. This means using jQuery event listeners for converting Assembly code written in the Assembly Language text field into JavaScript, which can then be outputted on the website in order to show what each error is, and where it has happened.
5. Users should have the option to upload text files (.txt), which contain Assembly Language instructions, using the same Instruction Set as the website.
    a. This should automatically be converted into the appropriate format for the Assembly Language text field, and should be able to run immediately, if the user decides to start the animations of the visualisation.
3. Users should be able to see step-by-step outputs of the details of each process being run, in relation to what is happening within the visualisation.
    a. This should include an indicator of which part of the FDE cycle is being run – including the processes with the 5 aforementioned Dedicated Registers (refer to requirement 3).

4. On the same page where users write their assembly code and view visualisations, there should be a dropdown list of pre-created Assembly codes that can be loaded into the text area, ready for visualisation.

This subsection will cover all of these requirements, showing the backend for the processing of the Assembly Code, as well as the code for displaying the frontend visualisations, but obviously, this will be quite hard to show with just code, so will be ticked off in the Testing section.

~/public/javascripts/lmc.js

```
// define classes

// AQA A level instruction template
class Instruction {

  constructor(args) {

    this.name = args.name;
    this.numerical_value = args.numerical_value;

    this.operands = args.operands; // list of 'R's, 'M's, 'B's and/or 'L', where R = Register, M = Memory, B = Both/Either, L = Label (for Branch)

  };

  no_of_operands() {
    return this.operands.length;
  };

  get_operand_type(operand_num) {
    return this.operands[operand_num - 1];
  }; // e.g. for LDR instruction, operand_num = 1 would return R because the first operand is Rd

  process_instruction_opcode() {
    return;
  };

};

class HLT_Instruction extends Instruction {

  process_instruction_opcode(LMC) {

    return new Promise((resolve, reject) => {
      var i = 0;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        if (i == 0) {
          LMC.log_output('HLT');
          LMC.log_output('Executing instruction...');
        } else if (i == 1) {
          LMC.deactivate_wrapper('control_unit_wrapper');
```

Inheritance and Polymorphism as evidence of Complex user-defined use of object orientated programming (OOP) model, eg classes, inheritance, composition, polymorphism, interfaces

Appropriate choice of simple data types

```
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.log_output('Program stopped');
            LMC.cycles++;
          } else {
            LMC.log_output('####################');
            LMC.log_output('Program executed in ' + LMC.cycles + ' FDE cycles');
            LMC.paused = true;
            LMC.stop = true;
            document.getElementById('clock').value = LMC.cycles;
            LMC.deactivate_wrapper('clock_wrapper');
            LMC.cycles = 0;
            resolve();
            clearInterval(LMC.carry_on);
          };
          i++;
        }, LMC.animation_interval);
      });

    };

};

class INP_Instruction extends Instruction {

  process_instruction_opcode(LMC) {

    return new Promise((resolve, reject) => {
      var i = 0;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve();
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('INP');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.log_output('Waiting for user input...');
            LMC.cycles++;
          } else if (i == 2) {
            LMC.inp = prompt('User input:');
            document.getElementById('input').value = LMC.inp;
            document.getElementById('input').scrollTop = document.getElementById('input').scrollHeight;
          } else {
```

Variables and functions will be **aptly named** to describe what their purpose is, **apart from counter values** in relatively small for loops (the letter 'i' is okay for this).

```
                LMC.log_output('Store user input in Accumulator: ' + LMC.inp);
                document.getElementById('accumulator').value = LMC.inp;
                document.getElementById('clock').value = LMC.cycles;
                 LMC.deactivate_wrapper('clock_wrapper');
                 resolve();
                 clearInterval(LMC.carry_on);
              };
              i++;
            };
          }, LMC.animation_interval);
        });

      };

    };

    class OUT_Instruction extends Instruction {

      process_instruction_opcode(LMC) {

        return new Promise((resolve, reject) => {
          var i = 0;
          LMC.carry_on = setInterval(() => {
            // fetch instruction
            LMC.check_error();

            if (LMC.assembly_code_error == true) {
              resolve();
              clearInterval(LMC.carry_on);
            } else {
              if (i == 0) {
                LMC.log_output('OUT');
                LMC.log_output('Executing instruction...');
              } else if (i == 1) {
                LMC.deactivate_wrapper('control_unit_wrapper');
                LMC.deactivate_wrapper('decode_unit_wrapper');
                control_bus.setOptions({
                  color: '#AAAAAA'
                });
                LMC.activate_wrapper('accumulator_wrapper');
              } else if (i == 2) {
                LMC.deactivate_wrapper('accumulator_wrapper');
                LMC.log_output('Output value held in Accumulator: ' + document.getElementById('accumulator').value);
                LMC.cycles++;
                document.getElementById('output').value = document.getElementById('accumulator').value;
                document.getElementById('output').scrollTop = document.getElementById('output').scrollHeight;
              } else {
                document.getElementById('clock').value = LMC.cycles;
                LMC.deactivate_wrapper('clock_wrapper');
                resolve();
                clearInterval(LMC.carry_on);
              };
              i++;
            };
          }, LMC.animation_interval);
```

Variables and functions will all be written using **snake_case**.

```
    });

  };

};

class ADD_Instruction extends Instruction {

  process_instruction_opcode(LMC, direct, operand, accumulator) {

    return new Promise((resolve, reject) => {
      var i = 0;
      var MBR;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve(MBR, accumulator);
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('ADD');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.activate_wrapper('ALU_wrapper');
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.cycles++;
          } else if (i == 2) {
            if (direct) {
              MBR = parseInt(operand);
              LMC.activate_wrapper('MBR_wrapper');
            } else {
              MBR = parseInt(document.getElementById('memory_location_' + operand).value);
              LMC.activate_wrapper('MAR_wrapper');
              LMC.deactivate_wrapper('MBR_wrapper');
            };
          } else if (i == 3) {
            if (direct) {
              LMC.activate_wrapper('MBR_wrapper');
              LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
              document.getElementById('MBR').value = MBR;
            } else {
              LMC.deactivate_wrapper('MAR_wrapper');
              LMC.deactivate_wrapper('MBR_wrapper');
              LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
              LMC.log_output('Fetch data at location held by MAR and store it in MBR: ' + MBR);
              document.getElementById('MAR').value = operand;
              document.getElementById('MBR').value = MBR;
            }
          } else if (i == 4) {
```

Modules (subroutines) with appropriate interfaces.

```
                LMC.activate_wrapper('MBR_wrapper');
                LMC.activate_wrapper('accumulator_wrapper');
            } else if (i == 5) {
                LMC.deactivate_wrapper('MBR_wrapper');
                LMC.deactivate_wrapper('accumulator_wrapper');
                LMC.log_output('Add MBR value to Accumulator and store the result in Accumulator: ' + accumulator + '+' + MBR + '=' + (accumulator + MBR));
                accumulator += MBR;
                document.getElementById('accumulator').value = accumulator;
            } else {
                document.getElementById('clock').value = LMC.cycles;
                LMC.deactivate_wrapper('clock_wrapper');
                LMC.deactivate_wrapper('ALU_wrapper');
                resolve(MBR, accumulator);
                clearInterval(LMC.carry_on);
            };
            i++;
        };
    }, LMC.animation_interval);
  });

  };

};

class SUB_Instruction extends Instruction {

  process_instruction_opcode(LMC, direct, operand, accumulator) {

    return new Promise((resolve, reject) => {
      var i = 0;
      var MBR;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve(MBR, accumulator);
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('SUB');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.activate_wrapper('ALU_wrapper');
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.cycles++;
          } else if (i == 2) {
            if (direct) {
              MBR = parseInt(operand);
              LMC.activate_wrapper('MBR_wrapper');
```

There should be consistent use of **1 whitespace character** between operator and operands, e.g. “x == 5”.

```
                } else {
                  MBR = parseInt(document.getElementById('memory_location_' + operand).value);
                  LMC.activate_wrapper('MAR_wrapper');
                  LMC.deactivate_wrapper('MBR_wrapper');
                };
              } else if (i == 3) {
                if (direct) {
                  LMC.activate_wrapper('MBR_wrapper');
                  LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
                  document.getElementById('MBR').value = MBR;
                } else {
                  LMC.deactivate_wrapper('MAR_wrapper');
                  LMC.deactivate_wrapper('MBR_wrapper');
                  LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
                  LMC.log_output('Fetch data at location held by MAR and store it in MBR: ' + MBR);
                  document.getElementById('MAR').value = operand;
                  document.getElementById('MBR').value = MBR;
                }
              } else if (i == 4) {
                LMC.activate_wrapper('MBR_wrapper');
                LMC.activate_wrapper('accumulator_wrapper');
              } else if (i == 5) {
                LMC.deactivate_wrapper('MBR_wrapper');
                LMC.deactivate_wrapper('accumulator_wrapper');
                LMC.log_output('Subtract MBR value from Accumulator and store the result in Accumulator: ' + accumulator + '-' + MBR + '=' + (accumulator - MBR));
                accumulator -= MBR;
                document.getElementById('accumulator').value = accumulator;
              } else {
                document.getElementById('clock').value = LMC.cycles;
                LMC.deactivate_wrapper('clock_wrapper');
                LMC.deactivate_wrapper('ALU_wrapper');
                resolve(MBR, accumulator);
                clearInterval(LMC.carry_on);
              };
              i++;
            };
          }, LMC.animation_interval);
        });

      };

    };

    class LDA_Instruction extends Instruction {

      process_instruction_opcode(LMC, direct, operand) {

        return new Promise((resolve, reject) => {
          var i = 0;
          var MBR;
          LMC.carry_on = setInterval(() => {
            // fetch instruction
            LMC.check_error();

```

Comments should only be used for **overviews or general explanations**, meaning there can be large gaps between comments if I feel the code is quite self-explanatory. However, there **can also be commented out chunks of codes** for approaching a particular part of a solution in a different way, e.g. asynchronously / synchronously writing to and reading from files.

```
            if (LMC.assembly_code_error == true) {
              resolve(MBR);
              clearInterval(LMC.carry_on);
            } else {
              if (i == 0) {
                LMC.log_output('LDA');
                LMC.log_output('Executing instruction...');
              } else if (i == 1) {
                LMC.deactivate_wrapper('control_unit_wrapper');
                LMC.deactivate_wrapper('decode_unit_wrapper');
                control_bus.setOptions({
                  color: '#AAAAAA'
                });
                LMC.cycles++;
              } else if (i == 2) {
                if (direct) {
                  MBR = parseInt(operand);
                  LMC.activate_wrapper('MBR_wrapper');
                } else {
                  MBR = parseInt(document.getElementById('memory_location_' + operand).value);
                  LMC.activate_wrapper('MAR_wrapper');
                  LMC.deactivate_wrapper('MBR_wrapper');
                };
              } else if (i == 3) {
                if (direct) {
                  LMC.activate_wrapper('MBR_wrapper');
                  LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
                  document.getElementById('MBR').value = MBR;
                } else {
                  LMC.deactivate_wrapper('MAR_wrapper');
                  LMC.deactivate_wrapper('MBR_wrapper');
                  MBR = parseInt(document.getElementById('memory_location_' + operand).value);
                  LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
                  LMC.log_output('Fetch data at location held by MAR (' + operand + ') and store it in MBR: ' + MBR);
                  document.getElementById('MAR').value = operand;
                  document.getElementById('MBR').value = MBR;
                }
              } else if (i == 4) {
                LMC.activate_wrapper('MBR_wrapper');
                LMC.activate_wrapper('accumulator_wrapper');
              } else if (i == 5) {
                LMC.deactivate_wrapper('MBR_wrapper');
                LMC.deactivate_wrapper('accumulator_wrapper');
                document.getElementById('accumulator').value = MBR;
                LMC.log_output('Store MBR value in Accumulator: ' + MBR);
              } else {
                document.getElementById('clock').value = LMC.cycles;
                LMC.deactivate_wrapper('clock_wrapper');
                resolve(MBR);
                clearInterval(LMC.carry_on);
              };
              i++;
            };
          }, LMC.animation_interval);
        });
```

```
    };

};

class STA_Instruction extends Instruction {

    process_instruction_opcode(LMC, operand) {

        return new Promise((resolve, reject) => {
            var i = 0;
            var MBR;
            LMC.carry_on = setInterval(() => {
                // fetch instruction
                LMC.check_error();

                if (LMC.assembly_code_error == true) {
                    resolve(MBR);
                    clearInterval(LMC.carry_on);
                } else {
                    if (i == 0) {
                        LMC.log_output('STA');
                        LMC.log_output('Executing instruction...');
                    } else if (i == 1) {
                        LMC.deactivate_wrapper('control_unit_wrapper');
                        LMC.deactivate_wrapper('decode_unit_wrapper');
                        control_bus.setOptions({
                            color: '#AAAAAA'
                        });
                        LMC.cycles++;
                    } else if (i == 2) {
                        MBR = parseInt(document.getElementById('accumulator').value);
                        LMC.activate_wrapper('MBR_wrapper');
                        LMC.activate_wrapper('accumulator_wrapper');
                    } else if (i == 3) {
                        LMC.deactivate_wrapper('accumulator_wrapper');
                        LMC.log_output('Set MBR to value held in Accumulator: ' + MBR);
                        document.getElementById('MBR').value = MBR;
                    } else if (i == 4) {
                        LMC.deactivate_wrapper('MBR_wrapper');
                        LMC.activate_wrapper('MAR_wrapper');
                        LMC.log_output('Store MBR value ' + MBR + ' at memory location held in MAR: ' + operand);
                        document.getElementById('MAR').value = operand;
                        document.getElementById('memory_location_' + operand).value = MBR;
                    } else {
                        LMC.deactivate_wrapper('MAR_wrapper');
                        document.getElementById('clock').value = LMC.cycles;
                        LMC.deactivate_wrapper('clock_wrapper');
                        resolve(MBR);
                        clearInterval(LMC.carry_on);
                    };
                    i++;
                };
            }, LMC.animation_interval);
        });
```

```

    };

};

class BRA_Instruction extends Instruction {

    process_instruction_opcode(LMC, operand) {

        return new Promise((resolve, reject) => {
            var i = 0;
            LMC.carry_on = setInterval(() => {
                // fetch instruction
                LMC.check_error();

                if (LMC.assembly_code_error == true) {
                    resolve();
                    clearInterval(LMC.carry_on);
                } else {
                    if (i == 0) {
                        LMC.log_output('BRA');
                        LMC.log_output('Executing instruction...');
                    } else if (i == 1) {
                        LMC.activate_wrapper('status_register_wrapper');
                        LMC.deactivate_wrapper('control_unit_wrapper');
                        LMC.deactivate_wrapper('decode_unit_wrapper');
                        control_bus.setOptions({
                            color: '#AAAAAA'
                        });
                        LMC.cycles++;
                    } else if (i == 2) {
                        LMC.activate_wrapper('PC_wrapper');
                    } else if (i == 3) {
                        document.getElementById('PC').value = operand;
                        LMC.deactivate_wrapper('PC_wrapper');
                        LMC.program_counter = parseInt(operand);
                        LMC.log_output('Set PC to operand of instruction: ' + operand);
                    } else {
                        document.getElementById('clock').value = LMC.cycles;
                        LMC.deactivate_wrapper('clock_wrapper');
                        LMC.deactivate_wrapper('status_register_wrapper');
                        resolve();
                        clearInterval(LMC.carry_on);
                    };
                    i++;
                };
            }, LMC.animation_interval);
        });

    };

};

class BRZ_Instruction extends Instruction {

```

```
process_instruction_opcode(LMC, operand, accumulator) {

  return new Promise((resolve, reject) => {
    var i = 0;
    LMC.carry_on = setInterval(() => {
      // fetch instruction
      LMC.check_error();

      if (LMC.assembly_code_error == true) {
        resolve();
        clearInterval(LMC.carry_on);
      } else {
        if (i == 0) {
          LMC.log_output('BRZ');
          LMC.log_output('Executing instruction...');
        } else if (i == 1) {
          LMC.activate_wrapper('status_register_wrapper');
          LMC.deactivate_wrapper('control_unit_wrapper');
          LMC.deactivate_wrapper('decode_unit_wrapper');
          control_bus.setOptions({
            color: '#AAAAAA'
          });
          LMC.cycles++;
        } else if (i == 2) {
          LMC.log_output('Check if value held in Accumulator is 0');
        } else if (i == 3) {
          if (accumulator == 0) {
            LMC.log_output('0 == 0 - true');
            LMC.activate_wrapper('PC_wrapper');
          } else {
            LMC.log_output(accumulator + ' == 0 - false');
          };
        } else if (i == 4) {
          if (accumulator == 0) {
            LMC.deactivate_wrapper('PC_wrapper');
            LMC.log_output('Set PC to operand of instruction: ' + operand);
            document.getElementById('PC').value = operand;
            LMC.program_counter = parseInt(operand);
          } else {
            document.getElementById('clock').value = LMC.cycles;
            LMC.deactivate_wrapper('clock_wrapper');
            LMC.deactivate_wrapper('status_register_wrapper');
            resolve();
            clearInterval(LMC.carry_on);
          };
        } else {
          document.getElementById('clock').value = LMC.cycles;
          LMC.deactivate_wrapper('clock_wrapper');
          LMC.deactivate_wrapper('status_register_wrapper');
          resolve();
          clearInterval(LMC.carry_on);
        };
        i++;
      };
    }, LMC.animation_interval);
```

Good use of local variables

```
535.        });
536.
537.    };
538.
539. };
540.
541. class BRP_Instruction extends Instruction {
542.
543.    process_instruction_opcode(LMC, operand, accumulator) {
544.
545.        return new Promise((resolve, reject) => {
546.            var i = 0;
547.            LMC.carry_on = setInterval(() => {
548.                // fetch instruction
549.                LMC.check_error();
550.
551.                if (LMC.assembly_code_error == true) {
552.                    resolve();
553.                    clearInterval(LMC.carry_on);
554.                } else {
555.                    if (i == 0) {
556.                        LMC.log_output('BRP');
557.                        LMC.log_output('Executing instruction...');
558.                    } else if (i == 1) {
559.                        LMC.activate_wrapper('status_register_wrapper');
560.                        LMC.deactivate_wrapper('control_unit_wrapper');
561.                        LMC.deactivate_wrapper('decode_unit_wrapper');
562.                        control_bus.setOptions({
563.                            color: '#AAAAAA'
564.                        });
565.                        LMC.cycles++;
566.                    } else if (i == 2) {
567.                        LMC.log_output('Check if value held in Accumulator is positive (>= 0)');
568.                    } else if (i == 3) {
569.                        if (accumulator >= 0) {
570.                            LMC.log_output(accumulator + ' >= 0 - true');
571.                            LMC.activate_wrapper('PC_wrapper');
572.                        } else {
573.                            LMC.log_output(accumulator + ' >= 0 - false');
574.                        };
575.                    } else if (i == 4) {
576.                        if (accumulator >= 0) {
577.                            LMC.deactivate_wrapper('PC_wrapper');
578.                            LMC.log_output('Set PC to operand of instruction: ' + operand);
579.                            document.getElementById('PC').value = operand;
580.                            LMC.program_counter = parseInt(operand);
581.                        } else {
582.                            document.getElementById('clock').value = LMC.cycles;
583.                            LMC.deactivate_wrapper('status_register_wrapper');
584.                            LMC.deactivate_wrapper('clock_wrapper');
585.                            resolve();
586.                            clearInterval(LMC.carry_on);
587.                        };
588.                    } else {
589.                        document.getElementById('clock').value = LMC.cycles;
```

```
                LMC.deactivate_wrapper('clock_wrapper');
                LMC.deactivate_wrapper('status_register_wrapper');
                resolve();
                clearInterval(LMC.carry_on);
            };
            i++;
        };
    }, LMC.animation_interval);
  });

};

};

// LMC template w/ methods used later
class Little_Man_Computer {

  constructor(args) {

    this.outputs = [];
    this.inputs = args.inputs || [];

    this.RAM = args.RAM || [];
    this.RAM_value_length = args.RAM_value_length || 3; // same as general_registers_value_length

    this.general_registers = args.general_registers || [];
    this.program_counter = 0;

    this.instruction_set = args.instruction_set || {};
    this.instruction_set_objs = args.instruction_set_objs || {};

    // this.clock = 25;
    // this.time_lapse = 10000;
    this.time_numerator = 10000;
    this.paused = true;

    this.labels = {};
    this.errors = [];
    this.inp;
    this.carry_on;
    this.cycles = 0;

    this.stop = false;

    this.animation_interval = 300; // milliseconds

    this.assembly_code_error = false;

  };

  // ensure backend RAM values are 2 digits long
  RAM_backend_to_RAM_value_length_digits(location) {

    // ensure 4 digits long
    if (this.RAM[location].length < this.RAM_value_length) {
```

Single-dimensional arrays

```
645.
646.        while (this.RAM[location].length < this.RAM_value_length) {
647.           this.RAM[location] = '0' + this.RAM[location];
648.        }
649.
650.     } else if (this.RAM[location].length > this.RAM_value_length) {
651.
652.        var location_as_string = location.toString();
653.
654.        while (location_as_string.length < 2) {
655.           location_as_string = '0' + location_as_string;
656.        }
657.
658.        alert('RAM value at memory location ' + location_as_string + ' must be ' + this.RAM_value_length.toString() + ' or
less digits long');
659.
660.        this.reset_specific_RAM(location);
661.
662.     } else {
663.        // do nothing - correct length
664.     };
665.
666.  };
667.
668.  // populate frontend visualisation RAM locations with the values stored in this.RAM
669.  load_RAM_from_backend() {
670.     for (var i = 0; i < this.RAM.length; i++) {
671.        // will always be 2 digits long because backend is always 2 digits long
672.        document.getElementById("memory_location_" + i.toString()).value = this.RAM[i];
673.     };
674.  };
675.
676.  // populate this.RAM with the values in the frontent visualisation RAM locations
677.  load_RAM_from_frontend() {
678.
679.     for (var i = 0; i < this.RAM.length; i++) {
680.        this.RAM[i] = document.getElementById("memory_location_" + i.toString()).value;
681.        // this.RAM_backend_to_RAM_value_length_digits(i);
682.     };
683.
684.     this.load_RAM_from_backend();
685.
686.  };
687.
688.  // reset all values in RAM location in frontend + backend to 00
689.  reset_RAM() {
690.     for (var i = 0; i < this.RAM.length; i++) {
691.        this.RAM[i] = '0'.repeat(this.RAM_value_length);
692.     };
693.     this.load_RAM_from_backend();
694.  };
695.
696.  // reset the value at a specific RAM location in frontend + backend to 00
697.  reset_specific_RAM(location) {
698.     this.RAM[location] = '0'.repeat(this.RAM_value_length);
```

Single-dimensional arrays

Managed casting of types

Cohesive modules (subroutines) – module code does just one thing.

```
    document.getElementById("memory_location_" + location.toString()).value = this.RAM[location];
};

// ensure backend general registers' values are 2 digits long
general_registers_backend_to_RAM_value_length_digits(location) {

    if (this.general_registers[location].length < this.RAM_value_length) {

        while (this.general_registers[location].length < this.RAM_value_length) {
            this.general_registers[location] = '0' + this.general_registers[location];
        }

    } else if (this.general_registers[location].length > this.RAM_value_length) {

        var location_as_string = location.toString();

        while (location_as_string.length < this.RAM_value_length) {
            location_as_string = '0' + location_as_string;
        }

        alert('General Purpose Register ' + location_as_string + '\'s value must be ' + this.RAM_value_length.toString() + ' or less digits long');

        this.reset_specific_general_register(location);

    } else {
        // do nothing - correct length
    };

};

// populate frontend visualisation general purpose registers with the values stored in this.general_registers
load_general_registers_from_backend() {
    for (var i = 0; i < this.general_registers.length; i++) {
        // will always be 2 digits long because backend is always 2 digits long
        document.getElementById("general_register_" + i.toString()).value = this.general_registers[i];
    };
};

// reset all values in the general registers in backend (automatically update to frontend because read-only)
reset_general_registers() {
    for (var i = 0; i < this.general_registers.length; i++) {
        this.general_registers[i] = '0'.repeat(this.RAM_value_length);
    };
    this.load_general_registers_from_backend();
};

// reset the value of a specific general register in frontend + backend to 00
reset_specific_general_register(location) {
    this.general_registers[location] = '0'.repeat(this.RAM_value_length);
    document.getElementById("general_register_" + location.toString()).value = this.general_registers[location];
};

reset_all_registers() {
    this.reset_general_registers();
```

Loosely coupled modules (subroutines) – module code interacts with other parts of the program through its interface only.

```

    document.getElementById('MAR').value = '00';
    document.getElementById('accumulator').value = '00';
    document.getElementById('status_register').value = '00';
    document.getElementById('CIR').value = '00';
    document.getElementById('PC').value = '00';
    document.getElementById('clock').value = '0';
    document.getElementById('MBR').value = '0'.repeat(this.RAM_value_length);

    this.program_counter = 0;

    this.deactivate_wrapper('ALU_wrapper');
    this.deactivate_wrapper('accumulator_wrapper');
    this.deactivate_wrapper('PC_wrapper');
    this.deactivate_wrapper('CIR_wrapper');
    this.deactivate_wrapper('MAR_wrapper');
    this.deactivate_wrapper('control_unit_wrapper');
    this.deactivate_wrapper('clock_wrapper');
    this.deactivate_wrapper('MBR_wrapper');
    this.deactivate_wrapper('status_register_wrapper');

    // leaderlines + wrappers back to normal
    address_bus.setOptions({
      end: LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
        x: 0,
        y: '23%'
      }),
      path: 'straight',
      color: '#AAAAAA'
    });
    control_bus.setOptions({
      color: '#AAAAAA'
    });
    data_bus.setOptions({
      end: LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
        x: 0,
        y: '86%'
      }),
      path: 'straight',
      color: '#AAAAAA'
    });
  };

  reset_verbose_output() {
    document.getElementById('verbose_output').value = 'Verbose Output';
  };

  reset_all_inp_out() {
    document.getElementById('input').value = '00';
    document.getElementById('output').value = '00';
    document.getElementById('FDE_output').value = 'Output of FDE cycle';
    this.reset_verbose_output();
  };

  // change border color of element to blue if gray, and vice-versa
```

```
activate_deactivate_wrapper(element) {

    var element_info = document.getElementById(element);

    if ($(element_info).css('border-color') !== 'rgb(64, 64, 255)') {
        $(element_info).css('border-color', '#4040FF');
    } else {
        $(element_info).css('border-color', '#AAAAAA');
    };
};

activate_wrapper(element) {
    var element_info = document.getElementById(element);
    $(element_info).css('border-color', '#4040FF');
};

deactivate_wrapper(element) {
    var element_info = document.getElementById(element);
    $(element_info).css('border-color', '#AAAAAA');
};

// log to FDE Cycle Output (based on reset), and to Verbose Output
log_output(text, reset = false) {

    if (reset) {
        document.getElementById('FDE_output').value = text.replace('   ', '   ') + '\r\n';
    } else {
        document.getElementById('FDE_output').value = document.getElementById('FDE_output').value + text.replace('   ',
'   ') + '\r\n';
    }

    document.getElementById('verbose_output').value = document.getElementById('verbose_output').value +
text.replace('   ', '   ') + '\r\n';
    document.getElementById('verbose_output').scrollTop = document.getElementById('verbose_output').scrollHeight;

};

// get code from Assembly Code text area
scan_code() {

    this.labels = {};
    var counter = 0;
    var assembly_code = document.getElementById('code_area').value.toUpperCase();
    var lines = assembly_code.split('\n');

    for (var i = 0; i < lines.length; i++) {

        var line = lines[i].trim();

        if (line != '') {

            var instruction = line.split(/\s+/);

            // variable that stores input
            if (instruction.length == 2) {
```

```
861.
862.                if (instruction[1] == 'DAT') {
863.                    this.labels[instruction[0]] = counter;
864.                    this.reset_specific_RAM(counter);
865.                } else if (!(instruction[0] in this.instruction_set)) {
866.                    this.labels[instruction[0]] = counter;
867.                };
868.
869.            };
870.
871.            // variable that stores value after DAT
872.            if (instruction.length == 3) {
873.
874.                if (instruction[1] == 'DAT') {
875.                    this.labels[instruction[0]] = counter;
876.                    this.RAM[counter] = instruction[2].toString();
877.                } else if (!(instruction[0] in this.instruction_set)) {
878.                    this.labels[instruction[0]] = counter;
879.                }
880.
881.            };
882.
883.            counter++;
884.
885.        };
886.    };
887. };
888.
889. // into RAM from Assembly Code after converting to opcode + operand form
890. load() {
891.
892.    this.load_RAM_from_frontend();
893.
894.    this.paused = true;
895.    this.cycles = 0;
896.    this.assembly_code_error = false;
897.
898.    // reset all registers (general purpose and special purpose)
899.    this.reset_all_registers();
900.    // reset all inputs / outputs / displays
901.    this.reset_all_inp_out();
902.    // reset RAM
903.    this.reset_RAM();
904.    // reset general registers
905.    this.reset_general_registers();
906.
907.    // scan code for labels
908.    this.scan_code();
909.
910.    // load program in RAM
911.    var counter = 0;
912.    var assembly_code = document.getElementById('code_area').value.toUpperCase();
913.    var lines = assembly_code.split('\n');
914.
915.    for (var i = 0; i < lines.length; i++) {
```

```
916.
917.        var line = lines[i].trim();
918.
919.        if (line != '') {
920.
921.          var instruction = line.split(/\s+/);
922.
923.          if (instruction.length == 3) {
924.
925.            var label = instruction[0];
926.            var opcode = instruction[1];
927.            var operand = instruction[2];
928.
929.            this.labels[label] = counter;
930.
931.            if (operand in this.labels) {
932.              operand = this.labels[operand];
933.            };
934.
935.            if (operand.length > 1) {
936.
937.              if (operand.substring(0, 1) == '@') {
938.
939.                operand = operand.substring(1);
940.
941.                if (isNaN(operand)) {
942.                  if (operand in this.labels) {
943.                    operand = '@' + this.labels[operand];
944.                  };
945.                } else {
946.                  operand = '@' + operand;
947.                };
948.
949.              } else if (operand.substring(0, 1) == '#') { // immediate addressing
950.
951.                operand = operand.substring(1);
952.
953.                if (isNaN(operand)) {
954.                  if (operand in this.labels) {
955.                    operand = '#' + this.labels[operand];
956.                  };
957.                } else {
958.                  operand = '#' + operand;
959.                };
960.
961.              };
962.
963.            };
964.
965.            if (opcode in this.instruction_set) {
966.
967.              var encode = this.instruction_set[opcode].numerical_value + operand;
968.              this.RAM[counter] = encode.toString();
969.
970.              counter++;
```

```

            } else if (opcode == 'DAT') {

                counter++;

            } else {

                this.log_output('Error at line ' + counter + ': Invalid opcode');

            };

        } else if (instruction.length == 2) {

            if (instruction[0] in this.instruction_set) {

                var opcode = instruction[0];
                var operand = instruction[1];

                if (operand in this.labels) {
                    operand = this.labels[operand];
                };

                // Indirect Addressing
                if (operand.length > 1) {

                    if (operand.substring(0, 1) == '@') {

                        operand = operand.substring(1);

                        if (isNaN(operand)) {
                            if (operand in this.labels) {
                                operand = '@' + this.labels[operand];
                            };
                        } else {
                            operand = '@' + operand;
                        };

                    } else if (operand.substring(0, 1) == '#') {

                        operand = operand.substring(1);

                        if (isNaN(operand)) {
                            if (operand in this.labels) {
                                operand = '#' + this.labels[operand];
                            };
                        } else {
                            operand = '#' + operand;
                        };

                    };

                };

                var encode = this.instruction_set[opcode].numerical_value + operand;
                this.RAM[counter] = encode.toString();
```

```

                counter++;

            } else {

                this.labels[instruction[0]] = counter;
                var opcode = instruction[1];

                if (opcode in this.instruction_set) {

                    var encode = this.instruction_set[opcode].numerical_value;
                    this.RAM[counter] = encode.toString();

                    counter++;

                } else if (instruction[1] == 'DAT') {

                    counter++;

                } else {

                    this.log_output('Error at line ' + counter + ': Invalid instruction');

                };

            };

        } else {

            if (line in this.instruction_set) {
                this.RAM[counter] = this.instruction_set[line].numerical_value;
                counter++;
            } else {
                this.log_output('Error at line ' + counter + ': Invalid instruction');
            };

        };

    };

    this.load_RAM_from_backend();

  };

  for (var i = 0; i < this.RAM.length; i++) {
    if (this.RAM[i] == '') {
      this.reset_specific_RAM(i);
    };
  };
  this.load_RAM_from_backend();
};

check_error() {
  let FDE = document.getElementById('FDE_output').value;
  // alert(FDE);
```

```
    // alert(FDE.split('\n')[(FDE.split('\n').length - 1) - 1]);
    // alert(FDE.split('\n')[(FDE.split('\n').length - 1) - 2]);
    let verbose = document.getElementById('verbose_output').value;
    // alert(verbose);
    // alert(verbose.split('\n')[(verbose.split('\n').length - 1) - 1]);
    // alert(verbose.split('\n')[(verbose.split('\n').length - 1) - 2]);

    let FDE_list = FDE.split('\n');
    let verbose_list = verbose.split('\n');

    // dealing with errors of log_outputs
    if (FDE_list.length >= 3 && verbose_list.length >= 3) {
      if (FDE_list[(FDE_list.length - 1) - 1] == FDE_list[(FDE_list.length - 1) - 2] || verbose_list[(verbose_list.length - 1) - 1] == verbose_list[(verbose_list.length - 1) - 2]) {
        this.assembly_code_error = true;
      } else {
        this.assembly_code_error = false;
      };
    } else {
      this.assembly_code_error = false;
    };
  }

  process_instruction_main_beginning() {
    return new Promise((resolve, reject) => {
      var i = 0;
      var instruction, k_str;
      this.carry_on = setInterval(() => {
        // fetch instruction
        this.check_error();

        if (this.assembly_code_error == true) {
          resolve(instruction);
          clearInterval(this.carry_on);
        } else {
          if (i == 0) {
            this.program_counter = parseInt(document.getElementById('PC').value);
            this.log_output('####################', true);
            this.log_output('Fetching instruction...');
            this.activate_wrapper('control_unit_wrapper');
            this.activate_wrapper('clock_wrapper');
            control_bus.setOptions({
              color: '#4040FF'
            });
          } else if (i == 1) {
            this.activate_wrapper('PC_wrapper');
          } else if (i == 2) {
            instruction = document.getElementById('memory_location_' + this.program_counter).value;
            this.activate_wrapper('MAR_wrapper');
            document.getElementById('MAR').value = this.program_counter;
          } else if (i == 3) {
            this.deactivate_wrapper('PC_wrapper');
            this.deactivate_wrapper('MAR_wrapper');
            this.log_output('Set MAR to value of PC: ' + this.program_counter);
          } else if (i == 4) {
```

```
            this.activate_wrapper('PC_wrapper');
            this.program_counter++;
            document.getElementById('PC').value = this.program_counter;
        } else if (i == 5) {
            this.deactivate_wrapper('PC_wrapper');
            this.log_output('Increment PC by 1');
        } else if (i == 6) {
            this.log_output('Fetch instruction from address stored in MAR');
        } else if (i == 7) {
            this.activate_wrapper('MAR_wrapper');
        } else if (i == 8) {

            // get memory wrapper location as 2 digits e.g. 00, 01, 92, etc.
            k_str = document.getElementById('MAR').value.toString();
            while (k_str.length < 2) {
                k_str = '0' + k_str;
            };

            // move address bus + change color
            address_bus.setOptions({
                end: document.getElementById('memory_' + k_str + '_wrapper'),
                path: 'fluid',
                color: '#4040FF'
            });

        } else if (i == 9) {
            this.activate_wrapper('memory_location_' + document.getElementById('MAR').value.toString());
        } else if (i == 10) {
            // move data bus + change color
            data_bus.setOptions({
                end: document.getElementById('memory_' + k_str + '_wrapper'),
                path: 'fluid',
                color: '#4040FF'
            });
        } else if (i == 11) {
            this.activate_wrapper('MBR_wrapper');
            document.getElementById('MBR').value = instruction;
        } else if (i == 12) {

            // leaderlines + wrappers back to normal
            address_bus.setOptions({
                end: LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
                    x: 0,
                    y: '23%'
                }),
                path: 'straight',
                color: '#AAAAAA'
            });
            data_bus.setOptions({
                end: LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
                    x: 0,
                    y: '86%'
                }),
                path: 'straight',
                color: '#AAAAAA'
```

```
                });

                this.deactivate_wrapper('MAR_wrapper');
                this.deactivate_wrapper('memory_location_' + document.getElementById('MAR').value.toString());
                this.deactivate_wrapper('MBR_wrapper');

                this.log_output('Fetched instruction ' + instruction + ' stored in MBR');

            } else if (i == 13) {
                this.deactivate_wrapper('control_unit_wrapper');
                control_bus.setOptions({
                    color: '#AAAAAA'
                });
                this.activate_wrapper('MBR_wrapper');
            } else if (i == 14) {
                this.activate_wrapper('CIR_wrapper');
                document.getElementById('CIR').value = instruction;
            } else if (i == 15) {
                this.deactivate_wrapper('MBR_wrapper');
                this.deactivate_wrapper('CIR_wrapper');
                this.log_output('Copied instruction from MBR to CIR');
            } else {
                this.log_output('Decoding instruction stored in CIR...');
                this.activate_wrapper('decode_unit_wrapper');
                this.activate_wrapper('control_unit_wrapper');
                control_bus.setOptions({
                    color: '#4040FF'
                });
                resolve(instruction);
                clearInterval(this.carry_on);
            }
            i++;
        };
    }, this.animation_interval);
  });
};

process_instruction_HLT() {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['HLT'].process_instruction_opcode(LMC_Obj);
    resolve();
  });
};

process_instruction_INP() {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['INP'].process_instruction_opcode(LMC_Obj);
    resolve();
  });
};

process_instruction_OUT() {
  return new Promise(async (resolve, reject) => {
```

```
        var LMC_Obj = this;
        await this.instruction_set_objs['OUT'].process_instruction_opcode(LMC_Obj);
        resolve();
    });
};

process_instruction_ADD(direct, operand, accumulator) {
    return new Promise(async (resolve, reject) => {
        var LMC_Obj = this;
        await this.instruction_set_objs['ADD'].process_instruction_opcode(LMC_Obj, direct, operand, accumulator);
        resolve();
    });
};

process_instruction_SUB(direct, operand, accumulator) {
    return new Promise(async (resolve, reject) => {
        var LMC_Obj = this;
        await this.instruction_set_objs['SUB'].process_instruction_opcode(LMC_Obj, direct, operand, accumulator);
        resolve();
    });
};

process_instruction_LDA(direct, operand) {
    return new Promise(async (resolve, reject) => {
        var LMC_Obj = this;
        await this.instruction_set_objs['LDA'].process_instruction_opcode(LMC_Obj, direct, operand);
        resolve();
    });
};

process_instruction_STA(operand) {
    return new Promise(async (resolve, reject) => {
        var LMC_Obj = this;
        await this.instruction_set_objs['STA'].process_instruction_opcode(LMC_Obj, operand);
        resolve();
    });
};

process_instruction_BRA(operand) {
    return new Promise(async (resolve, reject) => {
        var LMC_Obj = this;
        await this.instruction_set_objs['BRA'].process_instruction_opcode(LMC_Obj, operand);
        resolve();
    });
};

process_instruction_BRZ(operand, accumulator) {
    return new Promise(async (resolve, reject) => {
        var LMC_Obj = this;
        await this.instruction_set_objs['BRZ'].process_instruction_opcode(LMC_Obj, operand, accumulator);
        resolve();
    });
};

process_instruction_BRP(operand, accumulator) {
```

```
    return new Promise(async (resolve, reject) => {
      var LMC_Obj = this;
      await this.instruction_set_objs['BRP'].process_instruction_opcode(LMC_Obj, operand, accumulator);
      resolve();
    });
  };

  // asynchronous so order of events is followed
  async process_instruction() {

    if (this.assembly_code_error == false) {

      // must be done in order, so await is used
      var instruction = await this.process_instruction_main_beginning();

      // halt instruction
      if (instruction == '000') {

        await this.process_instruction_HLT();

      } else if (instruction == '901') { // input instruction

        await this.process_instruction_INP();

      } else if (instruction == '902') { // output instruction

        await this.process_instruction_OUT();

      } else {

        var opcode = instruction.substr(0, 1);
        var operand_address = instruction.substr(1);
        var operand;
        var direct = false;

        if (operand_address.substr(0, 1) == '@') { // indirect addressing
          operand = parseInt(document.getElementById('memory_location_' + operand_address.substr(1)).value);
        } else if (operand_address.substr(0, 1) == '#') { // direct addressing
          direct = true;
          operand = operand_address.substr(1);
        } else {
          operand = operand_address;
        };

        var accumulator = parseInt(document.getElementById('accumulator').value);

        if (opcode == '1') { // add instruction
          var MBR, accumulator = await this.process_instruction_ADD(direct, operand, accumulator);
        } else if (opcode == '2') { // subtract instruction
          var MBR, accumulator = await this.process_instruction_SUB(direct, operand, accumulator);
        } else if (opcode == '5') { // load to accumulator
          var MBR = await this.process_instruction_LDA(direct, operand);
        } else if (opcode == '3') { // store accumulator value in memory
          var MBR = await this.process_instruction_STA(operand);
        } else if (opcode == '6') { // always branch
```

```
                    await this.process_instruction_BRA(operand);
                } else if (opcode == '7') { // branch if zero
                    await this.process_instruction_BRZ(operand, accumulator);
                } else if (opcode == '8') { // branch if positive or 0
                    await this.process_instruction_BRP(operand, accumulator);
                };
            };
        } else { // invalid instruction - not recongnised
            alert('Error in assembly code! Executing HLT instruction!');
            this.activate_wrapper('status_register_wrapper');
            document.getElementById('status_register').value = '-1';
            await this.process_instruction_HLT();
            this.paused = true;
            this.stop = true;
        };
    };

    step() {

        this.paused = true;
        this.animation_interval = parseInt(this.time_numerator / document.getElementById('clock_speed').value);
        this.process_instruction();

    };

    async run() {

        this.cycles = 0;
        this.stop = false;

        this.reset_all_registers();
        this.reset_all_inp_out();

        this.scan_code();

        this.paused = false;
        this.animation_interval = parseInt(this.time_numerator / document.getElementById('clock_speed').value);

        while (!this.stop) {
            await this.process_instruction();
        };

    };

};

// functions to initialize values

// create instruction and LMC objects
function initialise_LMC() {

    var INP = new Instruction({ // get user input + store in accumulator
        name: 'INP',
        numerical_value: '901',
        operands: []
```

```
});
var OUT = new Instruction({ // output value stored in accumulator
    name: 'OUT',
    numerical_value: '902',
    operands: []
});
var LDA = new Instruction({ // load accumulator w/ contents of given RAM address
    name: 'LDA',
    numerical_value: '5',
    operands: ['M']
});
var STA = new Instruction({ // store value in accumulator to RAM address OR register
    name: 'STA',
    numerical_value: '3',
    operands: ['B']
});
var ADD = new Instruction({ // add accumulator value to RAM address value OR register value
    name: 'ADD',
    numerical_value: '1',
    operands: ['B']
});
var SUB = new Instruction({ // subtract contents of RAM address OR register contents from accumulator value
    name: 'SUB',
    numerical_value: '2',
    operands: ['B']
});
var BRP = new Instruction({ // branch to RAM address if accumulator value is zero or positive
    name: 'BRP',
    numerical_value: '8',
    operands: ['M']
});
var BRZ = new Instruction({ // branch to RAM address if accumulator value is zero
    name: 'BRZ',
    numerical_value: '7',
    operands: ['M']
});
var BRA = new Instruction({ // branch to RAM address
    name: 'BRA',
    numerical_value: '6',
    operands: ['M']
});
var HLT = new Instruction({ // stop assembly code
    name: 'HLT',
    numerical_value: '000',
    operands: []
});
var DAT = new Instruction({ // associates a label to a free RAM address. optional value used to store at RAM address
    name: 'DAT',
    numerical_value: '', // will be 4 or something like that
    operands: [] // will be number, RAM address
});

// dictionary of Instruction objects
var instruction_set = {
    'INP': INP,
```

```
    'OUT': OUT,
    'LDA': LDA,
    'STA': STA,
    'ADD': ADD,
    'SUB': SUB,
    'BRP': BRP,
    'BRZ': BRZ,
    'BRA': BRA,
    'HLT': HLT,
    'DAT': DAT
};

var INP_Obj = new INP_Instruction({
    name: 'INP',
    numerical_value: '901',
    operands: []
});
var OUT_Obj = new OUT_Instruction({
    name: 'OUT',
    numerical_value: '902',
    operands: []
});
var LDA_Obj = new LDA_Instruction({
    name: 'LDA',
    numerical_value: '5',
    operands: ['M']
});
var STA_Obj = new STA_Instruction({
    name: 'STA',
    numerical_value: '3',
    operands: ['B']
});
var ADD_Obj = new ADD_Instruction({
    name: 'ADD',
    numerical_value: '1',
    operands: ['B']
});
var SUB_Obj = new SUB_Instruction({
    name: 'SUB',
    numerical_value: '2',
    operands: ['B']
});
var BRP_Obj = new BRP_Instruction({
    name: 'BRP',
    numerical_value: '8',
    operands: ['M']
});
var BRZ_Obj = new BRZ_Instruction({
    name: 'BRZ',
    numerical_value: '7',
    operands: ['M']
});
var BRA_Obj = new BRA_Instruction({
    name: 'BRA',
    numerical_value: '6',
```

Dictionary

```
        operands: ['M']
    });
    var HLT_Obj = new HLT_Instruction({
        name: 'HLT',
        numerical_value: '000',
        operands: []
    });

    // similar dictionary but different use (only for process_instruction)
    var instruction_set_objs = {
        'INP': INP_Obj,
        'OUT': OUT_Obj,
        'LDA': LDA_Obj,
        'STA': STA_Obj,
        'ADD': ADD_Obj,
        'SUB': SUB_Obj,
        'BRP': BRP_Obj,
        'BRZ': BRZ_Obj,
        'BRA': BRA_Obj,
        'HLT': HLT_Obj
    }

    RAM = [];
    var RAM_value_length = 4;

    for (var i = 0; i < 100; i++) {
        RAM.push('0'.repeat(RAM_value_length));
    };

    general_registers = [];

    for (var i = 0; i < 10; i++) {
        general_registers.push('0'.repeat(RAM_value_length));
    }

    // LMC object with all processing
    var LMC = new Little_Man_Computer({
        instruction_set: instruction_set,
        instruction_set_objs: instruction_set_objs,
        RAM: RAM,
        RAM_value_length: RAM_value_length,
        general_registers: general_registers
    });

    return LMC
};

// create leader-lines for address, control, & data buses
function create_buses() {

    address_bus = new LeaderLine(
        document.getElementById('MAR_wrapper'),
        LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
            x: 0,
            y: '23%'
```

```
    }),
    {
      color: '#AAAAAA',
      size: 8,
      // outline: true,
      // endPlugOutline: true,
      // outlineColor: '#AAAAAA',
      // endPlugSize: 0.75,
      middleLabel: LeaderLine.pathLabel({
        text: 'Address Bus',
        color: 'black'
      }),
      path: 'straight',
      startSocket: 'right',
      endSocket: 'left',
      endPlug: 'arrow3'
    }
  );

  control_bus = new LeaderLine(
    document.getElementById('control_unit_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_50_wrapper'), {
      x: 0,
      y: '14%'
    }),
    {
      color: '#AAAAAA',
      size: 8,
      // outline: true,
      // endPlugOutline: true,
      // outlineColor: '#AAAAAA',
      // endPlugSize: 0.75,
      middleLabel: LeaderLine.pathLabel({
        text: 'Control Bus',
        color: 'black'
      }),
      path: 'straight',
      startSocket: 'right',
      endSocket: 'left',
      startPlug: 'arrow3',
      endPlug: 'arrow3'
    }
  );

  data_bus = new LeaderLine(
    document.getElementById('MBR_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
      x: 0,
      y: '86%'
    }),
    {
      color: '#AAAAAA',
      size: 8,
      // outline: true,
      // endPlugOutline: true,
```

```
        // outlineColor: '#AAAAAA',
        // endPlugSize: 0.75,
        middleLabel: LeaderLine.pathLabel({
          text: 'Data Bus',
          color: 'black'
        }),
        path: 'straight',
        startSocket: 'right',
        endSocket: 'left',
        startPlug: 'arrow3',
        endPlug: 'arrow3',
      }
  );

};

LMC = initialise_LMC();

// all window event listeners from HTML

// on loading the page
window.addEventListener('load', function () {
  // draw leaderlines for 3 buses
  create_buses();
  // disable spellcheck for all input/output fields
  $('body').attr("spellcheck", false);
});

// run everytime anything on the page is clicked
window.addEventListener('click', function () {

  LMC.load_RAM_from_frontend();
  LMC.load_general_registers_from_backend();

});

// all other initialisations / scripts

// create line numbers for assembly code area
var assembly_code_area = document.getElementById('code_area');
var line_numbers = document.querySelector('.line-numbers');
assembly_code_area.addEventListener('keyup', event => {
  var number_of_lines = event.target.value.split('\n').length;
  line_numbers.innerHTML = Array(number_of_lines).fill('<span></span>').join('');
});
$(document).ready(function () {
  var number_of_lines = assembly_code_area.value.split('\n').length;
  line_numbers.innerHTML = Array(number_of_lines).fill('<span></span>').join('');
});

// make line numbers scroll with assembly code area
(function ($) {

  var methods = {

```

```
    /**
     * @var options default options
     */
    options: {
      /**
       * @var eventTimeOut time in milliseconds
       */
      eventTimeOut: 200,

      /**
       * @var scrollSpeed time in milliseconds the other light boxes should take to scroll to their relevant point.
       */
      scrollSpeed: 100
    },

    /**
     * init
     *
     * method that applies brings everything together
     *
     * @param options object an object that overwrites the default options
     * @return null
     */
    init: function (options) {
      // overwrite defaults
      $.extend(methods.options, options);

      /**
       * @var $this object cached global jQuery variable
       */
      var $this = $(this);

      // Assign IDs. Used to prevent jittering of element that is calling.
      var id_num = 0;
      $(this).each(function () {
        $(this).data("scrollid", id_num);
        id_num++;
      });

      // bind on scroll
      var callback = function (el) {

        var
          /**
           * @var scroll_percent number percentage of top of element scrolled
           */
          scroll_percent = methods.convertScrollToPercent.call(el),
          /**
           * @var me number scroll id of element calling the scroll method
           */
          me = el.data("scrollid");

        // Loop through each
        $.each($this, function () {
          // Set scroll to false to prevent this element from calling same event
```

```
            $(this).data("scroll", false);
            // if it's the same element, skip
            if (me !== $(this).data("scrollid")) {
              // Scroll the element to percent
              methods.scrollToPercent.call(this, scroll_percent);
            }
            // Turn scroll back on
            $(this).data("scroll", true);
          });

        }

        // Assign scroll handler
        $(this).scroll(function (event) {
          /**
           * @var scroll_on boolean get scroll data value to see if it's safe to scroll
           */
          var scroll_on = $(this).data("scroll");

          // cancel call if it's not safe
          if (scroll_on == false) return;

          // Call method callback
          callback($(this));
        });

      },

      /**
       * convertScrollToPercent
       *
       * converts the scroll position to percent and returns it.
       *
       * @return number the percentage (duh)
       */
      convertScrollToPercent: function () {
        // Defaults

        var
          horrizontal = $(this).data("horizontal") || false,
          scrollStyle = (horrizontal != false ? "scrollLeft" : "scrollTop"),
          scrollWidth = (horrizontal != false ? "scrollWidth" : "scrollHeight");
        // get top and scroll position

        var
          top = $(this)[scrollStyle](),
          height = (horrizontal ? $(this)[0].scrollWidth - $(this).width() : $(this)[0].scrollHeight - $(this).height());

        // console.log("top :" + top + ", height :" + height);
        // height == 100 then what does top equal? ()
        var
          answer = 100 * top,
          answer = answer / height

        //   return answer;
```

```
        return Math.round(answer);

    },

    /**
     * scrollToPercent
     *
     * This method takes a percent and scrolls an element to that given point.
     *
     * @param percent numher the percent to which the element will be scrolled to.
     */
    scrollToPercent: function (percent) {

        var
            /**
             * @var horrizontal boolean what type of scroll to use. Horizontal or vertical
             */
            horrizontal = $(this).data("horizontal") || false,
            /**
             * @var height number the height or width of the element
             */
            height = (horrizontal ? $(this)[0].scrollWidth - $(this).width() : $(this)[0].scrollHeight - $(this).height()),
            // Perform equation to determin relative scroll positon
            answer = percent * height,
            /**
             * @var answer number the top or left position
             */
            answer = answer / 100;

        // scroll element to answer position
        if (horrizontal == true) {
            $(this).scrollLeft(answer);
        }
        else {
            $(this).scrollTop(answer);
        }
    }
}

/**
 * bindScroll
 *
 * a jQuery plugin that will bind the scroll to the given elements.
 * Each element should have a data attribute called scrollid. This
 * will let the plugin know not to apply the scrolling bind to that active
 * element. If element is to be a horizontal scroll, specify by setting
 * data-horizontal attribute to true.
 *
 * @param options object object with options (currently no options exist);
 * @return object returns the jQuery DOM object to maintain chainability
 */
$.fn.bindScroll = function (options) {
    methods.init.call(this, options);
    return this;
}
```

```

})($);
$(document).ready(function () {
    // apply bindScroll to all elements with .scroll.
    $(".scroll_together").bindScroll();
});

// allow tabs in assembly code area
document.getElementById('code_area').addEventListener('keydown', function (key) {
    if (key.keyCode == 9) {

        // get caret position / selection
        var start = this.selectionStart;
        var end = this.selectionEnd;

        var target = key.target;
        var value = target.value;

        // set textarea value to `text before caret` + tab + `text after caret`
        target.value = value.substring(0, start) + '\t' + value.substring(end);

        // put caret at right position again (add 1 for tab)
        this.selectionStart = this.selectionEnd = start + 1;

        // prevent losing focus
        key.preventDefault();

    };
}, false);

// triggered when a file is chosen for upload program
document.getElementById('upload_program_actual').addEventListener('change', () => {
    let files = document.getElementById('upload_program_actual').files;

    if (files.length == 0) return;

    const file = files[0];

    let reader = new FileReader();

    reader.onload = (element) => {
        const file = element.target.result;

        // regular expression to identify carriage
        // returns and line breaks
        const lines = file.split(/\r\n|\n/);
        document.getElementById('code_area').value = lines.join('\n');
    };

    reader.onerror = (element) => alert(element.target.error.name);
    reader.readAsText(file);

});

// functions called from HTML buttons
```

Use of constants

```
// remove all text froma assembly code editor
function clear_editor() {
    document.getElementById('code_area').value = '';
};

// set all values in RAM back to 00
function reset_RAM(LMC) {
    LMC.reset_RAM();
    LMC.reset_all_registers();
    LMC.reset_all_inp_out();
    LMC.stop = false;
    LMC.assembly_code_error = false;
    LMC = initialise_LMC();
};

// open new tab with /instruction_set
function open_link_new_tab(location) {

    // ternary operator used in case window.location doesn't include '/', works as follows:
    // bool_expr ? result_if_bool_expr_true : result_if_bool_expr_false;
    var redirect_to = window.location.toString() + (window.location.toString().includes('/') ? location : ('/' + location));

    window.open(redirect_to, '_blank');
};

function load_into_RAM(LMC) {
    LMC.load();
};
function run_program(LMC) {

    if (LMC.RAM[0] == '0000') {
        load_into_RAM(LMC);
    };

    if (!LMC.stop) {
        LMC.run();
    } else {
        alert('Program finished. To re-run, please reset RAM and load the program into RAM again.');
    };

};
function step_program(LMC) {

    if (LMC.RAM[0] == '0000') {
        load_into_RAM(LMC);
    };

    if (!LMC.stop) {
        LMC.step();
    } else {
        alert('Program finished. To re-run, please reset RAM and load the program into RAM again.');
    };

};
```

Annotation used effectively where required

Meaningful identifier names for variables and subroutines

```
// for pre-created programs loading into assembly code area
$("#pre-created_programs").on("change", function () {

  try {

    var selected_option = $("#pre-created_programs").val().toString();
    var directory = '../text_files/pre_created_programs/' + selected_option + '.txt';

    // get file with same name as pre-created_programs selection and put that into text area
    fetch(directory).then(response => response.text()).then(text => {
      document.getElementById("code_area").value = text;
    });

  } catch (error) {
    document.getElementById("code_area").value = "does not exist";
  }

});
```

Defensive programming

Text files

Writing and reading from files

This file contains the entire backend and processing done for converting Assembly Code to JavaScript and running the processes, changing the frontend values to show the animations and visualisations, as well as verbose output of what is actually happening and producing the actual output from the inputs. It also has the functions for each button contained within the control box on the frontend, such as “Load into RAM”, and “upload file”.

Objects of class **Instruction** are the valid instructions that are allowed with this Little Man Computer’s instruction set. They are all stored in a dictionary **instruction_set**, and then passed into the **Little_Man_Computer** class. There is one instance of this second class, called **LMC**, which essentially does all the processing. The reason that OOP has been used here is because it allowed for abstraction in that I was able to code for interface rather than implementation.

Within the **Little_Man_Computer** class, there are many methods that all take care of processing the information passed in from the frontend. When it comes to the actual conversions of the instructions from the instruction set, if-else statements had to be used, in order to explicitly say what to do with each instruction, but I had to learn to use Promises, a JavaScript construct that allows asynchronous functions so that certain procedures only happen after other ones.

Along with processing the information, there are also other miscellaneous methods such as resetting the RAM for the user to see on the frontend.

The **create_buses()** function uses the Leader_Line plugin from **~/public/plugins/leader_line/**, and is used for the address, control, and data bus. It allows seamless transitions and animations and was much easier to use than creating separate rectangles and triangles from JavaScript and

moving them to specific positions on the screen, which would have to be hard coded for each memory and dedicated register location. Furthermore, these locations would change based on the resolution of the screen of the device using the resource, so using Leader_Lines instead allows the buses to be "directly connected" to the div tag surrounding the locations.

Afterwards, there is an event listener for the Assembly Code text area which I have used to create the line numbers in the text area. The function below that allows the user to just scroll through their code, and the line numbers will scroll automatically as well – bounded scrolling – and without this, the line numbers would stay static while the code is being scrolled through vertically. Since an event listener with the method on "change" is used for the Assembly Code text area, the line numbers only update whenever there is a change in the text. I decided to do this instead of a constant event listener that always listens because that had the potential to really slow down the visualisations and cause a lot of delay and lag, which would no longer make the website, visualisations, and animations smooth.

Followed by an event listener for the "upload program" button, there are also a lot of functions that take the **LMC** object as an argument. This is because I did not want the frontend buttons to have direct access to the **Little_Man_Computer** class for security reasons, so each onclick function for the button was a separate function that called the **LMC** object's methods.

Finally, the pre-created programs dropdown code uses the dropdown value selected and chooses the appropriate file from **~/public/text_files/pre_created_programs/** to upload into the Assembly Code text area.

We have now completed explanations for the implementations of requirements 2, 4.a, 5.a, 7.a, and 8 in this section. That leaves the frontend code to show, but again, some of the requirements can only be met after seeing the Testing section.

~/views/partials/main_visualisation.ejs

```
<div class="row">

  <div class="col-md-2">
    <span>Assembly Code Editor</span>
  </div>

  <div class="col-md-4">
    <span>  Processor</span>
  </div>

  <div class="col-md-1"></div>

  <div class="col-md-5">
    <span>RAM</span>
  </div>
```

```
16.
17. </div>
18.
19. <div class="row" id="main_visualisation_wrapper_row_1">
20.
21.     <div class="col-md-2">
22.         <div id="code_wrapper">
23.             <div class="line-numbers col-md-2 scroll_together">
24.                 <span></span>
25.             </div>
26.             <textarea class="scroll_together" id="code_area" rows=27 wrap="off">
27.         INP
28. loop    OUT
29.         STA var1
30.         SUB var2
31.         STA var1
32.         BRP loop
33.         HLT
34.
35. var2    DAT 1
36. var1    DAT
37.
38. // Counts down in decrements of var2 (1), starting with the inputted number</textarea>
39.         </div>
40.     </div>
41.
42.     <div class="col-md-4">
43.         <div id="CPU_wrapper">
44.
45.             <div class="row">
46.                 <div class="col-md-4">
47.                     <span id="general_registers_title">General Purpose Registers</span>
48.                 </div>
49.             </div>
50.
51.             <div class="row">
52.
53.                 <div class="col-md-4" id="general_register_wrapper">
54.                     <!-- create General Purpose Registers (from 0 to 9) -->
55.                     <% for (var i = 0; i < 10; i++) { %>
56.                         <div id="register_<%= i %>_wrapper">
57.                             <tr>
58.                                 <td>Register <%= i.toString() %><input readonly id="general_register_<%= i %>" type="text"
value="0000" /></td>
59.                             </tr>
60.                         </div>
61.                     <% } %>
62.                 </div>
63.
64.                 <div class="col-md-4" id="CPU_column_2">
65.
66.                     <div id="ALU_wrapper">
67.                         Arithemtic Logic Unit
68.                         <input readonly id="ALU" type="text" value="" />
69.                     </div>
```

```

            <div id="accumulator_wrapper">
              Accumulator
              <input readonly id="accumulator" type="text" value="00" />
            </div>

            <div id="PC_wrapper">
              Program Counter
              <input readonly id="PC" type="text" value="00" />
            </div>

            <div id="decode_unit_wrapper">
              Decode Unit
              <input readonly id="decode_unit" type="text" value="" />
            </div>

            <div id="CIR_wrapper">
              Current Instruction Register
              <input readonly id="CIR" type="text" value="00" />
            </div>

          </div>

          <div class="col-md-4" id="CPU_column_3">

            <div id="MAR_wrapper">
              Memory Address Register
              <input readonly id="MAR" type="text" value="00" />
            </div>

            <div id="status_register_wrapper">
              Status Register
              <input readonly id="status_register" type="text" value="00" />
            </div>

            <div id="control_unit_wrapper">
              Control Unit
              <input readonly id="control_unit" type="text" value="" />
            </div>

            <div id="clock_wrapper">
              Clock (cycles)
              <input readonly id="clock" type="text" value="0" />
            </div>

            <div id="MBR_wrapper">
              Memory Buffer Register
              <input readonly id="MBR" type="text" value="0000" />
            </div>

          </div>

          </div>
      </div>
  </div>
```

```

    <div class="col-md-1"></div>

    <div class="col-md-5">
        <div id="RAM_wrapper">

            <table id="RAM">
                <!-- create RAM (memory locations from 00 to 99) -->
                <% for (var i = 0; i < 10; i++) { %>
                    <tr>
                        <% for (var j = 0; j < 10; j++) { %>

                            <% var k = (i * 10) + j %>
                            <% var k_str = k.toString() %>

                            <%# for memory locations 0-9, we want to display 00, 01, etc. %>
                            <% while (k_str.length < 2) { %>
                                <% k_str = '0' + k_str %>
                            <% } %>

                            <td id="memory_<%= k_str %>_wrapper"><%= k_str %><input id="memory_location_<%= k %>"
type="text" value="0000" /></td>

                        <% } %>
                    </tr>
                <% } %>
            </table>
        </div>

    </div>
</div>

<div class="row" id="second_title_text_row">

    <div class="col-md-6">
        <span>
            <div class="row">
                <div class="col-md-5">User Controlbox for Visualisation</div>
                <div class="col-md-7"><input type="range" id="clock_speed" min="5" max="5000" value="25"></div>
            </div>
        </span>
    </div>

    <!-- log outputs -->
    <div class="col-md-3">
        <span>FDE Cycle Output</span>
    </div>

    <div class="col-md-3" id="verbose_output_title">
        <span>Verbose Output</span>
    </div>

</div>

<div class="row" id="main_visualisation_wrapper_row_2">
```

```

<div class="col-md-6" id="controlbox_wrapper">
  <div class="row">

    <div class="col-md-3">
      <div id="input_wrapper">
        <tr>
          <td>Input<input id="input" type="text" value="00" /></td>
        </tr>
      </div>
    </div>

    <div class="col-md-9" id="controlbox_top_two_rows_column_2">
      <div class="row" id="controlbox_row_1_column_2">

        <div class="col-md-3" id="upload_program_wrapper">
          <!-- button acts as label for hidden input file field -->
          <input hidden type="file" id="upload_program_actual" name="Upload Program" accept="text/plain"/>
          <button id="upload_program"><label for="upload_program_actual">Upload Program</label></button>
        </div>

        <div class="col-md-3" id="reset_RAM_wrapper">
          <button id="reset_RAM" onclick="reset_RAM(LMC)">Reset RAM</button>
        </div>

        <div class="col-md-3" id="load_into_RAM_wrapper">
          <button id="load_into_RAM" onclick="load_into_RAM(LMC)">Load into RAM</button>
        </div>

        <div class="col-md-3" id="run_program_wrapper">
          <button id="run_program" onclick="run_program(LMC)">Run Program</button>
        </div>

      </div>

      <div class="row" id="controlbox_row_2_column_2">

        <div class="col-md-4" id="clear_editor_wrapper">
          <button id="clear_editor" onclick="clear_editor()">Clear Editor</button>
        </div>

        <!-- pre-created dropdown logic in lmc.js -->
        <div class="col-md-4" id="pre-created_programs_wrapper">
          <select id="pre-created_programs" name="pre-created_programs">
            <option value="" selected hidden>Pre-created Programs</option>
            <option value="add_2_nums" id="add_2_nums">add 2 nums</option>
            <option value="sub_2_nums" id="sub_2_nums">sub 2 nums</option>
            <option value="mult_2_nums" id="mult_2_nums">mult 2 nums</option>
            <option value="count_down" id="count_down">count down</option>
            <option value="immediate_addressing" id="immediate_addressing">immediate addressing</option>
            <option value="square_inp_until_0" id="square_inp_until_0">square inp until 0</option>
            <option value="bubble_sort" id="bubble_sort">bubble sort</option>
          </select>
        </div>

```

```
                <div class="col-md-4" id="step_program_wrapper">
                  <button id="step_program" onclick="step_program(LMC)">Step Program</button>
                </div>

              </div>
            </div>

        </div>

        <div class="row">

            <div class="col-md-3">
              <div id="output_wrapper">
                <tr>
                  <td>Output<input readonly id="output" type="text" value="00" /></td>
                </tr>
              </div>
            </div>

            <div class="col-md-9" id="controlbox_bottom_two_rows_column_2">

              <div class="row" id="how_to_use_this_resource_wrapper">
                <button id="how_to_use_this_resource" onclick="open_link_new_tab('how_to_use')">How to Use this Resource</button>
              </div>

              <div class="row" id="view_instruction_set_wrapper">
                <button id="view_instruction_set" onclick="open_link_new_tab('instruction_set')">View Instruction Set</button>
              </div>

            </div>

        </div>
    </div>

    <div class="col-md-3">
        <div id="FDE_output_wrapper">
            <textarea id="FDE_output" wrap="on">Output of FDE cycle</textarea>
        </div>
    </div>

    <div class="col-md-3">
        <div id="verbose_output_wrapper">
            <textarea id="verbose_output" wrap="on">Verbose Output</textarea>
        </div>
    </div>

</div>

<% include ./lmc_script %>

```

This file contains all the HTML elements for the frontend to be loaded and look nice for the viewer. As I mentioned while explaining the previous file, the buttons have onclick functions that pass in the **LMC** object as a parameter instead of having direct access, and these functions are located in **~/public/javascripts/lmc.js**.

The **<% %>** tags are EJS constructions to run JavaScript while creating the template, and the reason this is done in this file rather than the previous one is because it is much easier to directly create HTML objects in an EJS file rather than insert them as innerHTML code from a JavaScript file.

Most of the code here is self-explanatory – just divs and sections for each component of the CPU and the controlbox, while the actual handling of this is done in the previous file.

This file completes requirements 3, 4, 5, 7, and 8.

~/views/partials/lmc_script.ejs

```
<!-- Local files -->
<!-- general js file for main visualisation -->
<script type="text/javascript" src="/javascripts/lmc.js"></script>
<!-- leader_line plugin for creating arrows b/w divs -->
<script src="/plugins/leader_line/leader-line.min.js"></script>

```

This partial file connects the previous file to **~/public/javascripts/lmc.js** so that it is used as the main JavaScript file for it. It also sources the external plugin Leader_Line.

Having explained the main visualisation, I will now briefly touch on the required pages to understand and use this resource, followed by an explanation of the login and sign-up system.

Instruction Set and How to Use the Resource

Recall requirements 6 and 9:

6. Users should be able to view the Instruction Set on a separate page from the visualisations.
    a. There should be an additional link on the main page that goes to another page, which contains full descriptions and explanations of each component of the CPU that has been added (refer to requirement 3), as well as the Instruction Set (refer to requirement 2).
5. There should be an introductory tutorial for new users of the resource who have never visited the website.
    a. This means showing the basic features of each component with a short description and how to use each and what each of them does.

From the first NodeJS and EJS Default Files section, we saw that **~/routes/index.js** contains the GET methods for the two pages that satisfy the above requirements.

For requirement 6, I simply needed a separate page which contained a table of the instruction set and their explanations. For requirement 9, similar to what I discussed in the Documented Design section, I decided to create a page in a text and image-based format because I thought that that would be the best way to show how to use the resource, with labels on the images of the home / main visualisation page.

It is important to note that these 2 requirements can only really be shown in the Testing section, because they are mostly text with a basic HTML document outline, so there is no explanation needed for the Implementation of these, apart from the fact that the 2 files which are rendered in **~/routes/index.js** are **~/views/instructions/instruction_set.ejs** and **~/views/instructions/how_to_use.ejs**. Once again, to see these pages explicitly, they can be viewed in Appendix I.

Login and Sign-Up

Recall requirement 10:

6. Accounts – users should be able to login and sign up for a student or teacher account, depending on whether they are a student or teacher at Dubai College, using their Dubai College email address.
    a. Student account
        i. Should be able to view their assigned challenges and due dates (set by a teacher).
        ii. Should be able to submit these assigned challenges by pasting in their Assembly code into a text area.
        iii. Should be able to view a log of their previously completed challenges.
    b. Teacher account
        i. Teachers have classes which are groups of students.
        ii. Should be able to create and set challenges to be completed by all students in a class, or specific students.
        iii. Can view students' submitted challenge responses – there will be an indicator that the student has clicked on the submit button and the student can no longer edit it.
        iv. Should be able to view all logs of challenges and differ between completed and non-completed challenges.
    c. No account
        i. These users can still view visualisations and write code in the Assembly Language.
        ii. Cannot submit challenges to a teacher as they will have no assigned class and therefore no assigned teacher.

We can clearly see that a user should be able to sign up for a teacher or a student account, and then be able to login to that kind of account. Since there are these different types of accounts as well as a no-account option, there needs to be some sort of indicator of the type of account, and this is a red/yellow/green dot shown on the navbar. This is shown below.

~/views/partials/menu.ejs

```
<div class="row" id="navbar_row">
  <div class="col-md-12">
    <nav class="navbar navbar-default navbar-expand-sm navbar-dark bg-dark fixed-top">
      <div class="container-fluid navbar_wrapper">

        <!-- hidden form for POST method for logout tab in navbar -->
        <form id="logout_form" name="logout_form" action="/logout" method="post">
          <button type="submit">LOG OUT</button>
        </form>
```

```
10.
11.         <div class="navbar_left">
12.           <ul class="navbar-nav nav_items_left">
13.             <li>
14.               <!-- coloured dot icon (3 diff colours for different possible accounts / no account) - needs to be coded -->
15.               <!-- tooltip should change depending on account type - no account: red; teacher: red, yellow, green;
student: red, green; -->
16.               <span>
17.                 <% if (typeof role === "undefined" ) { %>
18.                   <img class="tooltip-inner" id="navbar_user_indicator" src="/images/red_dot_indicator.png" alt="User
Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account" />
19.                 <% } else { %>
20.                   <% if (role === 'student') { %>
21.                     <img class="tooltip-inner" id="navbar_user_indicator" src="/images/green_dot_indicator.png"
alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account
22. green: student account" />
23.                   <% } else { %>
24.                     <img class="tooltip-inner" id="navbar_user_indicator" src="/images/yellow_dot_indicator.png"
alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account
25. yellow: teacher account
26. green: student account" />
27.                   <% } %>
28.                 <% } %>
29.               </span>
30.             </li>
31.             <li>
32.               <span id="navbar_title">Assembly Code Visualiser</span>
33.             </li>
34.           </ul>
35.         </div>
36.
37.         <div class="collapse navbar-collapse" id="main_navbar">
38.
39.           <div class="navbar_left">
40.             <!-- main options on the left -->
41.             <ul class="navbar-nav nav_items_left" id="navbar_options">
42.               <li class="nav-item">
43.                 <a class="nav-link" id="home" href="/">HOME</a>
44.               </li>
45.               <li class="nav-item">
46.                 <a class="nav-link" id="instruction_set" href="/instruction_set">INSTRUCTION SET</a>
47.               </li>
48.               <li class="nav-item">
49.                 <a class="nav-link" id="how_to_use" href="/how_to_use">HOW TO USE THIS RESOURCE</a>
50.               </li>
51.               <% if (typeof role==="undefined" ) { %>
52.               <% } else { %>
53.                 <% if (role==='student' ) { %>
54.                   <li class="nav-item">
55.                     <a class="nav-link" id="view_set_challenges" href="/view_set_challenges">VIEW SET
CHALLENGES</a>
56.                   </li>
57.                 <% } else { %>
58.                   <li class="nav-item">
59.                     <a class="nav-link" id="create_challenges" href="/create_challenges">CREATE CHALLENGES</a>
```

```
                    </li>
                    <li class="nav-item">
                      <a class="nav-link" id="view_student_challenges" href="/view_student_challenges">VIEW STUDENT CHALLENGES</a>
                    </li>
                  <% } %>
                <% } %>
              </ul>
            </div>

            <div class="navbar_right">
              <!-- login / sign up button on the right -->
              <ul class="navbar-nav nav_items_right">
                <% if (typeof role === "undefined" ) { %>
                  <li class="navbar-item">
                    <a class="nav-link navbar_right" id="login" href="/login">LOGIN</a>
                  </li>
                  <li class="navbar-item">
                    <a class="nav-link navbar_right" id="sign_up" href="/sign_up">SIGN UP</a>
                  </li>
                <% } else { %>
                  <li class="navbar-item">
                    <a class="nav-link navbar_right" href="#" onclick="post_logout()">LOG OUT</a>
                  </li>
                <% } %>
              </ul>
            </div>
          </div>

          <!-- collapsible menu w/ 3 horizontal lines for when screen too small -->
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#main_navbar"
            aria-controls="main_navbar" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>

        </div>
      </nav>
    </div>
  </div>

```

The above partial file contains the links to all of the main pages that need to be linked. Furthermore, we can also see the EJS template constructs for the different types of accounts. The expression **(typeof role === "undefined")** is true when there is no account, and **(role === 'student')** and **(role === 'teacher')** are quite self-explanatory. We can see that there is the **data-toggle** and **title** attributes in the image tag, which shows the explanation of the types of account when the mouse pointer is hovered over the image. For this to work, the data-tooltip must be enabled in a different file, and since the navbar is in every page, this is located in **~/public/javascripts/menu.js**.

~/public/javascripts/menu.js

```
// set active tab in the navbar automatically
$(document).ready(function () {
    var element = $('meta[name="active-menu"]').attr('content');
    $('#' + element).addClass('active');
});

// activate tooltips (the thing that shows up underneath the navbar user indicator)
$(document).ready(function () {
    $('[data-toggle="tooltip"]').tooltip();
});

// post logout method when logout clicked on navbar (because links can't post, only get)
function post_logout() {
    document.forms['logout_form'].submit();
};

```

This file takes care of setting up the tooltip, logging out only when possible if the logout button is clicked, and setting the active tab in the navbar, changing the colour slightly.

~/public/images/

This file contains 3 images, the red, yellow, and green dots, which, as seen in **~/views/partials/menu.ejs** are used to indicate the account type currently active.

Sign-up

~/views/login_sign_up/sign_up.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
    <!-- ensure that the page reloads on first redirect so all data from db is up-to-date -->
    <script type="text/javascript">
        $(document).ready(function () {
            //Check if the current URL contains '#'

            //if (document.URL.slice(-8, document.URL.length - 1) == '/assign' || document.URL.slice(-8, document.URL.length - 1) === '/assign') {
            //  url = document.URL.slice(0, document.URL.length - 8);
            //  location.replace(url);
            //};

            if (document.URL.indexOf("#") == -1) {
                // Set the URL to whatever it was plus "#".
                url = document.URL + "#";
                location = "#";

```

```
            //Reload the page
            location.reload(true);
        }
    });
  </script>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <div class="row justify-content-center" id="student_card">
      <div class="col-md-8">
        <div class="card">

          <div class="card-header" style="background-color: #93c572;">
            <div class="row">
              <div class="col-md-3">
                <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Student Sign-up</h5>
              </div>
              <div class="col-md-9">
                <button class="btn btn-secondary switch_to_teacher" onclick="change_form_login()" style="float: right;">Switch to Teacher Sign-up</button>
              </div>
            </div>
          </div>

          <div class="card-body">

            <!-- student form -->
            <form action="/sign_up/student" method="POST" id="student_form">

              <!-- student's email -->
              <div class="form-group row">

                <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                <div class="col-md-8">
                  <!-- automatically check for correctly formatted student dc email (regex in JavaScript: '\d{4}' is the same as '[0-9][0-9][0-9][0-9]' ) -->
                  <input type="email" id="email" name="email" pattern="^[A-Za-z]+\d{4}\@dubaicollege.org$" class="form-control" placeholder="student_name_and_number@dubaicollege.org" required autofocus />
                </div>
              </div>

              <!-- student's year group -->
              <div class="form-group row">
                <label for="year_group" class="col-md-3 col-form-label">YEAR GROUP</label>
                <div class="col-md-8">
                  <!-- drop-down menu options -->
                  <!-- NOTE: ASSUMPTION THAT EVERY TEACHER TEACHES EVERY YEAR (THIS MAKES IT EASIER TO SIGN UP, EVEN IF THOSE CLASSES DON'T ACTUALLY HAVE STUDENTS IN THEM) -->
                  <select id="year_group" name="year_group" class="form-control" required>
                    <option value="" selected hidden>select your year group</option>
                    <option value="10">Year 10</option>
```

Regular expression

```
                        <option value="11">Year 11</option>
                        <option value="12">Year 12</option>
                        <option value="13">Year 13</option>
                      </select>
                    </div>
                  </div>

                  <!-- student's teacher (3-letter code) -->
                  <div class="form-group row">
                    <label for="teacher_initials" class="col-md-3 col-form-label">CLASS (TEACHER CODE)</label>
                    <div class="col-md-8">
                      <!-- drop-down menu options -->
                      <select id="teacher_initials" name="teacher_initials" class="form-control" required>
                        <option value="" selected hidden>select your class</option>
                        <% if (typeof class_codes===="undefined" ) { %>
                          <option value="no_teacher">no teacher has signed up yet</option>
                        <% } else { %>
                          <% for (var class_code_counter=0; class_code_counter < class_codes.length;
class_code_counter++) { %>
                            <option value="<%= class_codes[class_code_counter].class_code %>">
                              <%= class_codes[class_code_counter].class_code %>
                            </option>
                          <% } %>
                        <% } %>
                      </select>
                    </div>
                  </div>

                  <!-- student's password -->
                  <div class="form-group row">
                    <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
                    <div class="col-md-8">
                      <input type="password" id="password" name="password" class="form-control" placeholder="enter a
password" required />
                    </div>
                  </div>

                  <!-- student's password confirmation -->
                  <div class="form-group row">
                    <label for="password_confirmation" class="col-md-3 col-form-label">CONFIRM PASSWORD</label>
                    <div class="col-md-8">
                      <input type="password" id="password_confirmation" name="password_confirmation" class="form-
control" placeholder="enter the same password again" required />
                    </div>
                  </div>

                  <!-- submit button -->
                  <div class="row">
                    <div class="col-md-8 offset-md-3">
                      <button type="submit" class="btn btn-primary">SIGN UP</button>
                    </div>
                  </div>

                  <% if (message.error) { %>
                    <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
```

```
                        <strong>
                          <%= message.error %>
                        </strong>
                      </div>
                    <% } %>

                    <% if (message.success) { %>
                      <div class="alert alert-success" role="alert" style="margin-top: 1em;">
                        <strong>
                          <%= message.success %>
                        </strong>
                      </div>
                    <% } %>

                  </form>

                  <h5 style="margin-top: 1em;">Already have an account? <a href="/login">Login...</a></h5>

                </div>

              </div>
            </div>
          </div>

          <div class="row justify-content-center" id="teacher_card">
            <div class="col-md-8">
              <div class="card">

                <div class="card-header" style="background-color: #cccc00;">
                  <div class="row">
                    <div class="col-md-3">
                      <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Teacher Sign-up</h5>
                    </div>
                    <div class="col-md-9">
                      <button class="btn btn-secondary switch_to_student" onclick="change_form_login()" style="float: right;">Switch to Student Sign-up</button>
                    </div>
                  </div>
                </div>

                <div class="card-body">

                  <!-- teacher form -->
                  <form action="/sign_up/teacher" method="POST" id="teacher_form">

                    <!-- teacher's email -->
                    <div class="form-group row">
                      <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                      <div class="col-md-8">
                        <!-- automatically check for correctly formatted teacher dc email -->
                        <input type="email" id="email" name="email" pattern="^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$" class="form-control" placeholder="teacher_first_name.teacher_last_name@dubaicollege.org" required />
                      </div>
                    </div>

```

```
177.                <!-- teacher's class code (3-letter code) -->
178.                <div class="form-group row">
179.                  <label for="teacher_initials" class="col-md-3 col-form-label">CLASS CODE</label>
180.                  <div class="col-md-8">
181.                    <input type="text" id="teacher_initials" name="teacher_initials" class="form-control" pattern="^[A-
Z]{3}$" placeholder="3-letter DC teacher identification" required />
182.                  </div>
183.                </div>
184.
185.                <!-- teacher's password -->
186.                <div class="form-group row">
187.                  <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
188.                  <div class="col-md-8">
189.                    <input type="password" id="password" name="password" class="form-control" placeholder="enter a
password" required />
190.                  </div>
191.                </div>
192.
193.                <!-- teacher's password confirmation -->
194.                <div class="form-group row">
195.                  <label for="password_confirmation" class="col-md-3 col-form-label">CONFIRM PASSWORD</label>
196.                  <div class="col-md-8">
197.                    <input type="password" id="password_confirmation" name="password_confirmation" class="form-
control" placeholder="enter the same password again" required />
198.                  </div>
199.                </div>
200.
201.                <!-- submit button -->
202.                <div class="row">
203.                  <div class="col-md-8 offset-md-3">
204.                    <button type="submit" class="btn btn-primary">SIGN UP</button>
205.                  </div>
206.                </div>
207.
208.                <% if (message.error) { %>
209.                  <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
210.                    <strong>
211.                      <%= message.error %>
212.                    </strong>
213.                  </div>
214.                <% } %>
215.
216.                <% if (message.success) { %>
217.                  <div class="alert alert-success" role="alert" style="margin-top: 1em;">
218.                    <strong>
219.                      <%= message.success %>
220.                    </strong>
221.                  </div>
222.                <% } %>
223.
224.              </form>
225.
226.              <h5 style="margin-top: 1em;">Already have an account? <a href="/login">Login...</a></h5>
227.
228.            </div>
```

```
229.
230.            </div>
231.          </div>
232.       </div>
233.
234.    </div>
235. </body>
236.
237. <% include ../partials/script %>
238. <% include partials/script %>
239.
240. </html>
241.
```

We can see here the frontend for the teacher and student forms for signing up, with the fields as **required** and the email using regex for validation. Some of the validation is done in this file, but most of it is in the backend.

~/routes/login_sign_up/sign_up.js

```
1. var express = require('express');
2. var router = express.Router();
3. var passport = require('passport');
4.
5. var auth = require('../../lib/auth');
6. // for connecting to database
7. var db_connection = require('../../lib/db');
8.
9. // for decrypting password
10. var bcrypt = require('bcrypt');
11. const { format } = require('../../lib/db');
12.
13. // coming from index page -> sign_up
14. router.get('/', auth.check_not_authenticated, function (req, res, next) {
15.
16.    try {
17.
18.       // want all class_codes from the Teacher table to fill out the options for the student sign up
19.       db_connection.query(
20.          'SELECT class_code FROM Teacher;',
21.          function (err, rows) {
22.
23.             if (err) {
24.                console.log(err);
25.                req.flash('error', ' An error occured');
26.                res.locals.message = req.flash();
27.                res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
28.             }
29.
30.             if (rows.length === 0) {
31.                res.locals.message = req.flash();
32.                res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
```

Modules (collections of subroutines) – subroutines with common purpose grouped.

All routes' functions to do with signing up for teachers and students – separated from login file.

```
            } else {
                // everything works
                res.locals.message = req.flash();
                res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up', class_codes: rows });
            }
        }
    );

  } catch (error) {

    res.locals.message = req.flash();
    res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });

  }
});

// sign up form - student
router.post('/student', auth.check_not_authenticated, async function(req, res) {
  try {

    // all user-side scripting for validation
    var error_message = false;

    // ensure everything filled out
    if (req.body.email.length === 0) {
      error_message = true;
      req.flash('error', ' Email is required');
    }
    if (req.body.year_group.length === 0) {
      error_message = true;
      req.flash('error', ' Year Group is required');
    }
    if (req.body.teacher_initials.length === 0) {
      error_message = true;
      req.flash('error', ' Class is required');
    }
    if (req.body.password.length === 0) {
      error_message = true;
      req.flash('error', ' Password is required');
    }
    if (req.body.password_confirmation.length === 0) {
      error_message = true;
      req.flash('error', ' Confirm password is required');
    }

    // check email matches regex for student email
    var regex_student = /^[A-Za-z]+\d{4}\@dubaicollege.org$/;
    if (req.body.email.match(regex_student) === null) {
      error_message = true;
      req.flash('error', ' Invalid email format (DC student email required)');
    }

    // ensure a teacher has signed up
    if (req.body.teacher_initials === 'no_teacher') {
      error_message = true;
```

```
88.          req.flash('error', ' A teacher needs to sign up first');
89.      }
90.
91.      // ensure password length is >= 8 characters
92.      if (req.body.password.length < 8) {
93.          error_message = true;
94.          req.flash('error', ' Password length must be at least 8 characters');
95.      }
96.
97.      // ensure password contains at least 1 alphabet, 1 number, and 1 special character
98.      var special_chars = /[ `!@#$%^&*()_+\-=\[\]{};':"\\|,.<>\/?~]/; // regex for special characters
99.      var alphabet_chars = /[a-zA-Z]/; // regex for alphabets
100.     var number_chars = /\d/; // regex for digit
101.     if (!(special_chars.test(req.body.password) && alphabet_chars.test(req.body.password) &&
number_chars.test(req.body.password))) {
102.         error_message = true;
103.         req.flash('error', ' Password must contain at least 1 alphabet, 1 integer, and 1 special character');
104.     }
105.
106.     // ensure confirm password is the same as password
107.     if (req.body.password !== req.body.password_confirmation) {
108.         error_message = true;
109.         req.flash('error', ' Confirm password does not match password');
110.     }
111.
112.     // return with the error message
113.     if (error_message) {
114.         res.locals.message = req.flash();
115.         return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
116.     }
117.
118.     // all server-side scripting for validation
119.
120.     // ensure email is not already being used
121.     db_connection.query(
122.         'SELECT * FROM Student WHERE student_email = ?;', [req.body.email],
123.         function (err, rows) {
124.             if (!rows || rows.length <= 0) {
125.                 // good, do not want a result
126.                 // pass - nothing needs to happen
127.             } else {
128.                 // bad, 1 email cannot be used multiple times
129.                 req.flash('error', ' Email is already being used');
130.                 res.locals.message = req.flash();
131.                 return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
132.             }
133.         }
134.     );
135.
136.     // validation complete, so add all data to the database
137.     var hashed_password = await bcrypt.hash(req.body.password, 10);
138.     db_connection.query(
139.         'INSERT INTO Student (student_email, student_name, student_number, student_password) VALUES (?, ?, ?, ?);',
140.         [req.body.email, req.body.email.split('@')[0].slice(0, -4), req.body.email.split('@')[0].substr(-4), hashed_password],
141.         function (err, rows) {
```

Hashing

```
            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
            }
            console.log(rows);
          }
        );
        // cross-parameterised SQL (tells MySQL exactly where the parameters are supposed to go (where the ? marks are))
        db_connection.query(
          'INSERT INTO Students_In_Classes (student_id, class_id) VALUES ((SELECT student_id FROM Student WHERE student_email = ?), (SELECT class_id FROM Class WHERE year_group = ? AND teacher_id = (SELECT teacher_id FROM Teacher WHERE class_code = ?)));',
          [req.body.email, req.body.year_group, req.body.teacher_initials],
          function (err, rows) {
            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
            }
            console.log(rows);
          }
        );

        return res.redirect('/login');

      } catch {
        return res.redirect('/sign_up');
      }
    });

    // sign up form - teacher
    router.post('/teacher', auth.check_not_authenticated, async function(req, res) {
      try {

        // all user-side scripting for validation
        var error_message = false;

        // ensure everything filled out
        if (req.body.email.length === 0) {
          error_message = true;
          req.flash('error', ' Email is required');
        }
        if (req.body.teacher_initials.length === 0) {
          error_message = true;
          req.flash('error', ' Class code is required');
        }
        if (req.body.password.length === 0) {
          error_message = true;
          req.flash('error', ' Password is required');
        }
        if (req.body.password_confirmation.length === 0) {
          error_message = true;
```

? for parameterisation, and multiple SELECT statements within SELECT statements as evidence of Cross-table parameterised SQL

```
195.        req.flash('error', ' Confirm password is required');
196.     }
197.
198.     // check email matches regex for teacher email
199.     var regex_teacher = /^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$/;
200.     if (req.body.email.match(regex_teacher) === null) {
201.        error_message = true;
202.        req.flash('error', ' Invalid email format (DC teacher email required)');
203.     }
204.
205.     // ensure class code is 3 letters
206.     var regex_class_code = /^[A-Z]{3}$/;
207.     if (!(req.body.teacher_initials.match(regex_class_code))) {
208.        error_message = true;
209.        req.flash('error', ' Invalid class code format (3 capital letters)');
210.     }
211.
212.     // ensure password length is >= 8 characters
213.     if (req.body.password.length < 8) {
214.        error_message = true;
215.        req.flash('error', ' Password length must be at least 8 characters');
216.     }
217.
218.     // ensure password contains at least 1 alphabet, 1 number, and 1 special character
219.     var special_chars = /[ `!@#$%^&*()_+\-=\[\]{};':"\\|,.<>\/?~]/; // regex for special characters
220.     var alphabet_chars = /[a-zA-Z]/; // regex for alphabets
221.     var number_chars = /\d/; // regex for digit
222.     if (!(special_chars.test(req.body.password) && alphabet_chars.test(req.body.password) &&
number_chars.test(req.body.password))) {
223.        error_message = true;
224.        req.flash('error', ' Password must contain at least 1 alphabet, 1 integer, and 1 special character');
225.     }
226.
227.     // ensure confirm password is the same as password
228.     if (req.body.password !== req.body.password_confirmation) {
229.        error_message = true;
230.        req.flash('error', ' Confirm password does not match password');
231.     }
232.
233.     // return with the error message
234.     if (error_message) {
235.        res.locals.message = req.flash();
236.        return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
237.     }
238.
239.     // all server-side scripting for validation
240.
241.     // ensure email is not already being used
242.     db_connection.query(
243.        'SELECT * FROM Teacher WHERE teacher_email = ?;', [req.body.email],
244.        function (err, rows) {
245.           if (!rows || rows.length <= 0) {
246.              // good, do not want a result
247.              // pass - nothing needs to happen
248.           } else {
```

Single table or non-parameterised SQL

```
            // bad, 1 email cannot be used multiple times
            req.flash('error', ' Email is already being used');
            res.locals.message = req.flash();
            return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
          }
        }
      );

      // ensure class code is not already being used
      db_connection.query(
        'SELECT * FROM Teacher WHERE class_code = ?;', [req.body.teacher_initials],
        function (err, rows) {
          if (!rows || rows.length <= 0) {
            // good, do not want a result
            // pass - nothing needs to happen
          } else {
            // bad, 1 email cannot be used multiple times
            req.flash('error', ' Class code is already being used');
            res.locals.message = req.flash();
            return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
          }
        }
      );

      // validation complete, so add all data to the database
      // add teacher info to Teacher table
      var hashed_password = await bcrypt.hash(req.body.password, 10);
      db_connection.query(
        'INSERT INTO Teacher (teacher_email, teacher_first_name, teacher_last_name, class_code, teacher_password) VALUES (?, ?, ?, ?, ?);',
        [req.body.email, req.body.email.split('@')[0].split('.')[0].charAt(0).toUpperCase() + req.body.email.split('@')[0].split('.')[0].slice(1), req.body.email.split('@')[0].split('.')[1].charAt(0).toUpperCase() + req.body.email.split('@')[0].split('.')[1].slice(1), req.body.teacher_initials, hashed_password],
        function (err, rows) {
          if (err) {
            console.log(err);
            req.flash('error', ' An error occured');
            res.locals.message = req.flash();
            return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
          }
          console.log(rows);
        }
      );
      // add 4 new classes (year 10-13 inclusive), with the teacher
      for (var year = 10; year <= 13; year++) {
        db_connection.query(
          'INSERT INTO Class (year_group, teacher_id) VALUES (?, (SELECT teacher_id FROM Teacher WHERE teacher_email = ?));',
          [year, req.body.email],
          function (err, rows) {
            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
```

```
                }
                console.log(rows);
            }
        );
    }

    return res.redirect('/login');

  } catch {
    return res.redirect('/sign_up');
  }
});

module.exports = router;

```

I implemented the sign-up page using two forms that are always present on the screen as seen in the previous file, but one of the forms is always hidden and cannot be access. The only way to do this is to click on a button that toggles between the two forms. The file that contains the JavaScript for this functionality will be explained once I finish explaining the rest of this file.

Since the student file appears first on the page, we need to dynamically update the dropdown selection menu for the class_codes field, which contains the teacher class name, such as "MWD". This must be done immediately when the page is loaded, so is run when the GET method for the page is called and is done using a simple SELECT query from the Teacher table.

From there, I then take care of the student form's POST method. Firstly, I did all of the user-side validation for the form – making sure everything is filled out, correct lengths, passwords matching, etc. and then reloaded the page with the appropriate error message if there was an error, with **res.locals.message = req.flash()**, which, as seen at the beginning of this section, is used for displaying errors.

Afterwards, I went on to do some validation with regards to the database (checking if the user already exists), and without any errors, proceeded to INSERT the necessary values into multiple tables through cross-parameterised SQL.

It is also worth noting that the reason the function for this GET method is asynchronous is because we need to wait for the **bcrypt** library to hash the password, hence the **await**, and only then be able to INSERT the values into the database.

After this, I did almost the same thing for the teacher form's POST method, but obviously doing the new validation for the new fields.

~/public/javascripts/form_toggler.js

```
// immediately hide teacher form when page loads
$(document).ready(function () {
    $('#teacher_form').hide();
    $('#teacher_card').hide();
});

function change_form_login() {
    $('#teacher_form').toggle();
    $('#student_form').toggle();
    $('#teacher_card').toggle();
    $('#student_card').toggle();
}

```

This JavaScript file takes care of toggling the forms when the button is clicked. As soon as the page is loaded, the teacher form needs to be hidden, and luckily, there is a simple JavaScript function called **toggle()** to take care of toggling the forms whenever the button is clicked.

~/views/login_sign_up/partials/script.ejs

```
<!-- Local files -->
<script type="text/javascript" src="/javascripts/form_toggler.js"></script>

```

We have previously seen the overall partials directory and its contents, but each module can also have its own partials directory and EJS scripts. Here, we connect **~/views/login_sign_up/sign_up.ejs** to **~/public/javascripts/form_toggler.js**.

Login

~/views/login_sign_up/login.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <div class="row justify-content-center" id="student_card">
            <div class="col-md-8">
                <div class="card">

                    <div class="card-header" style="background-color: #93c572;">
                        <div class="row">
```

```
                <div class="col-md-3">
                  <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Student Login</h5>
                </div>
                <div class="col-md-6"></div>
                <div class="col-md-3">
                  <button class="btn btn-secondary switch_to_teacher" onclick="change_form_login()">Switch to Teacher Login</button>
                </div>
              </div>
            </div>

            <div class="card-body">

              <!-- student form -->
              <form action="/login/student" method="post" id="student_form">

                <!-- student's email -->
                <div class="form-group row">
                  <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                  <div class="col-md-8">
                    <!-- automatically check for correctly formatted student dc email (regex in JavaScript: '\d{4}' is the same as '[0-9][0-9][0-9][0-9]' ) -->
                    <input type="email" id="email" name="email" class="form-control" pattern="^[A-Za-z]+\d{4}\@dubaicollege.org$"
                      placeholder="student_name_and_number@dubaicollege.org" required autofocus />
                  </div>
                </div>

                <!-- student's password -->
                <div class="form-group row">
                  <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
                  <div class="col-md-8">
                    <input type="password" id="password" name="password" class="form-control" placeholder="enter a password" required />
                  </div>
                </div>

                <!-- remember me button -->
                <div class="form-group row">
                  <div class="col-md-8 offset-md-3">
                    <label for="remember_me" style="margin-left: 0.5em;">REMEMBER ME</label>
                    <div class="remember_me_float">
                      <input type="checkbox" id="remember_me" name="remember_me" value="1" />
                    </div>
                  </div>
                </div>

                <!-- submit button -->
                <div class="row">
                  <div class="col-md-8 offset-md-3">
                    <button type="submit" class="btn btn-primary" value="Login">LOGIN</button>
                    <a href="#" class="btn btn-link">Forgot Your Password?</a>
                  </div>
                </div>

```

```
                    <% if (message.error) { %>
                      <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
                        <strong>
                          <%= message.error %>
                        </strong>
                      </div>
                    <% } %>

                    <% if (message.success) { %>
                      <div class="alert alert-success" role="alert" style="margin-top: 1em;">
                        <strong>
                          <%= message.success %>
                        </strong>
                      </div>
                    <% } %>

                  </form>

                  <h5 style="margin-top: 1em;">Don't have an account? <a href="/sign_up">Sign up...</a></h5>

                </div>

              </div>
            </div>
          </div>

          <div class="row justify-content-center" id="teacher_card">
            <div class="col-md-8">
              <div class="card">

                <div class="card-header" style="background-color: #cccc00;">
                  <div class="row">
                    <div class="col-md-3">
                      <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Teacher Login</h5>
                    </div>
                    <div class="col-md-6"></div>
                    <div class="col-md-3">
                      <button class="btn btn-secondary switch_to_student" onclick="change_form_login()">Switch to Student Login</button>
                    </div>
                  </div>
                </div>

                <div class="card-body">

                  <!-- teacher form -->
                  <form action="/login/teacher" method="post" id="teacher_form">

                    <!-- teacher's email -->
                    <div class="form-group row">
                      <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                      <div class="col-md-8">
                        <!-- automatically check for correctly formatted teacher dc email -->
                        <input type="email" id="email" name="email" class="form-control" pattern="^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$" placeholder="teacher_first_name.teacher_last_name@dubaicollege.org" required />
```

```
                    </div>
                  </div>

                  <!-- teacher's password -->
                  <div class="form-group row">
                    <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
                    <div class="col-md-8">
                      <input type="password" id="password" name="password" class="form-control" placeholder="enter a password" required />
                    </div>
                  </div>

                  <!-- remember me button -->
                  <div class="form-group row">
                    <div class="col-md-8 offset-md-3">
                      <label for="remember_me" style="margin-left: 0.5em;">REMEMBER ME</label>
                      <div class="remember_me_float">
                        <input type="checkbox" id="remember_me" name="remember_me" value="1" />
                      </div>
                    </div>
                  </div>

                  <!-- submit button -->
                  <div class="row">
                    <div class="col-md-8 offset-md-3">
                      <button type="submit" class="btn btn-primary" value="Login">LOGIN</button>
                      <a href="#" class="btn btn-link">Forgot Your Password?</a>
                    </div>
                  </div>

                  <% if (message.error) { %>
                    <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
                      <strong>
                        <%= message.error %>
                      </strong>
                    </div>
                  <% } %>

                  <% if (message.success) { %>
                    <div class="alert alert-success" role="alert" style="margin-top: 1em;">
                      <strong>
                        <%= message.success %>
                      </strong>
                    </div>
                  <% } %>

                </form>

                <h5 style="margin-top: 1em;">Don't have an account? <a href="/sign_up">Sign up...</a></h5>

              </div>

            </div>
          </div>
        </div>
```

```

    </div>
</body>

<% include ../partials/script %>
<% include partials/script %>

</html>

```

Using the same logic as **~/views/login_sign_up/sign_up.ejs**, this file also has 2 forms, 1 for the student, and another for the teacher.

~/routes/login_sign_up/login.js

```
var express = require('express');
var router = express.Router();

// set up passport
var passport = require('passport');
var initializePassport = require('../../lib/passport-config');
var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// coming from index page -> login
router.get('/', auth.check_not_authenticated, function (req, res, next) {
    res.locals.message = req.flash();
    res.render('login_sign_up/login', { title: 'Login', menu_id: 'login' });
});

// login form - student
router.post('/student', auth.check_not_authenticated, passport.authenticate('local_student', {
    failureRedirect: '/login',
    failureFlash: true,
    session: true
}), function(req, res) {
    // on successful authentication
    // if remember me box not ticked
    if (typeof req.body.remember_me === "undefined") {
        console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
        res.redirect('/');
    } else {
        req.session.cookie.maxAge = 2592000000; // 30 days
        console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
        res.redirect('/');
    }
});

// login form - teacher
router.post('/teacher', auth.check_not_authenticated, passport.authenticate('local_teacher', {
    failureRedirect: '/login',
```

```
39.    failureFlash: true,
40.    session: true
41. }), function (req, res) {
42.    // on successful authentication
43.    // if remember me box not ticked
44.    if (typeof req.body.remember_me === "undefined") {
45.       console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
46.       res.redirect('/');
47.    } else {
48.       req.session.cookie.maxAge = 2592000000; // 30 days
49.       console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
50.       res.redirect('/');
51.    }
52. });
53.
54. module.exports = router;
55.
```

What is different in this file is the fact that there has to be logic for the “remember me” button. Here, all I do is that, when the “remember me” button is clicked **(typeof req.body.remember_me !== “undefined”)**, I change the cookie session’s max age to be larger, so that a user only has to login once every 30 days instead of every 2 hours.

Also, notice that **passport.authenticate()**, **auth.check_athenticated**, and **auth.check_not_authenticated** are used quite frequently when dealing with login and other GET methods. This is to ensure that certain pages can only be accessed if logged into an account or vice-versa. I will now explain how I set up the authentication using the passport library.

~/lib/auth.js

```
1. function check_authenticated(req, res, next) {
2.    if (req.isAuthenticated()) {
3.       return next();
4.    } else {
5.       res.redirect('/login');
6.    }
7. };
8.
9. function check_not_authenticated(req, res, next) {
10.    if (req.isAuthenticated()) {
11.       return res.redirect('/')
12.    } else {
13.       next();
14.    }
15. }
16.
17. module.exports = {
18.    check_authenticated,
19.    check_not_authenticated
```

```
};

```

Here, we use the **req.isAuthenticated()** function to get the authentication Boolean value, and based on this, redirect to a page. The two functions **check_authenticated()** and **check_not_authenticated()**, as mentioned before, are used during router and method requests based on whether a user should be logged in or not. Again, this is exported to **module.exports** so that it can be used in other files throughout the application.

~/lib/passport-config.js

```
var LocalStrategy = require('passport-local').Strategy;
var bcrypt = require('bcrypt');

// for connecting to database
var db_connection = require('./db');

var get_student_by_email = (email) => {
  return new Promise((resolve, reject) => {
    try {
      db_connection.query(
        'SELECT * FROM Student WHERE student_email = ?;', [email],
        function (err, rows) {

          if (err) {
            console.log(err);
            reject(err);
          }

          console.log(rows);

          resolve(rows);
        }
      );
    } catch (error) {
      reject(error);
    }
  });
};

var get_teacher_by_email = (email) => {
  return new Promise((resolve, reject) => {
    try {
      db_connection.query(
        'SELECT * FROM Teacher WHERE teacher_email = ?;', [email],
        function (err, rows) {

          if (err) {
            console.log('1:', err);
            reject(err);
          }
```

Promise as evidence of Async/Await functions

Good exception handling.

```
41.
42.                 console.log('2:', rows);
43.
44.                 resolve(rows);
45.             }
46.         );
47.     } catch (error) {
48.         reject('3:', error);
49.     }
50.   });
51. };
52.
53. function initialize(passport) {
54.   console.log('Initialized Passport..!');
55.
56.   var authenticate_user_student = async (req, email, password, done) => {
57.
58.     // all user-side scripting for validation
59.
60.     // ensure everything filled out
61.     if (email.length === 0) {
62.       return done(null, false, { message: ' Email is required ' });
63.     }
64.     if (password.length === 0) {
65.       return done(null, false, { message: ' Password is required' });
66.     }
67.
68.     // check email matches regex for student email
69.     var regex_student = /^[A-Za-z]+\d{4}\@dubaicollege.org$/;
70.     if (email.match(regex_student) === null) {
71.       return done(null, false, { message: ' Invalid email format (DC email required)' });
72.     }
73.
74.     // ensure password length is >= 8 characters
75.     if (password.length < 8) {
76.       return done(null, false, { message: ' Password length must be at least 8 characters' });
77.     }
78.
79.     // all server-side scripting for validation
80.     try {
81.
82.       await get_student_by_email(email).then(async (rows) => {
83.
84.         if (!rows || rows.length <= 0) {
85.           return done(null, false, { message: ' No user with that email' });
86.         }
87.
88.         try {
89.
90.           if (await bcrypt.compare(password, rows[0].student_password)) {
91.             // everything correct from login
92.             rows[0].role = 'student';
93.             return done(null, rows[0])
94.           } else {
95.             return done(null, false, { message: ' Password incorrect' })
```

Async/Await precursor to functions as evidence of Async/Await functions

```
            }

          } catch (err) {
            console.log(err);
             return done(err)
          }

        });

    } catch (error) {
        console.log(error);
        return done(null, false, { message: error });
    }

};

passport.use('local_student', new LocalStrategy({
    usernameField: 'email',
    passwordField: 'password',
    passReqToCallback: true
},
authenticate_user_student));

var authenticate_user_teacher = async (req, email, password, done) => {

    // all user-side scripting for validation

    // ensure everything filled out
    if (email.length === 0) {
        return done(null, false, { message: ' Email is required ' });
    }
    if (password.length === 0) {
        return done(null, false, { message: ' Password is required' });
    }

    // check email matches regex for teacher email
    var regex_teacher = /^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$/;
    if (email.match(regex_teacher) === null) {
        return done(null, false, { message: ' Invalid email format (DC email required)' });
    }

    // ensure password length is >= 8 characters
    if (password.length < 8) {
        return done(null, false, { message: ' Password length must be at least 8 characters' });
    }

    // all server-side scripting for validation
    try {

        await get_teacher_by_email(email).then(async (rows) => {

            if (!rows || rows.length <= 0) {
                return done(null, false, { message: ' No user with that email' });
            }

```

```
            try {

              if (await bcrypt.compare(password, rows[0].teacher_password)) {
                // everything correct from login
                rows[0].role = 'teacher';
                return done(null, rows[0])
              } else {
                return done(null, false, { message: ' Password incorrect' })
              }

            } catch (err) {
              console.log('4:', err);
              return done(err)
            }

          });

        } catch (error) {
          console.log('5:', error);
          return done(null, false, { message: error });
        }

      };

      passport.use('local_teacher', new LocalStrategy({
        usernameField: 'email',
        passwordField: 'password',
        passReqToCallback: true
      },
        authenticate_user_teacher));

      passport.serializeUser(function(user, done) {
        process.nextTick(function() {
          if (user.role === 'student') {
            done(null, { id: user.student_id, email: user.student_email, role: user.role });
          } else {
            done(null, { id: user.teacher_id, email: user.teacher_email, role: user.role });
          }
        });
      });

      passport.deserializeUser(function(user, done) {
        process.nextTick(function() {
          return done(null, user);
        })
      });
    };

    module.exports = initialize

```

Using the passport library and local strategy, this file sets up the two types of accounts, **local_student** and **local_teacher**. I use the student/teacher's id from the database

(student_id/teacher_id FROM Student/Teacher), as well as their hashed password to store their session and ensure that they are logged in. These two components are saved and used to deserialise the user. I also do some final validation to ensure that nothing goes wrong when retrieving information from the database.

Having explained the login and sign-up processes, I will now go on to explain the other main feature (apart from the visualisations) of my project that are facilitated by the login and sign-up – the challenges section.

Challenges

Creating, Editing, and Assigning Challenges as a Teacher

~/views/teacher_challenges/create_challenges.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
  <% include ../partials/head %>
  <% include ./partials/script %>
  <% include ./partials/style %>
  <% include ./partials/head %>
  <% include ./partials/script_toggle %>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <!-- display created challenges -->
    <div class="row justify-content-center">
      <div class="col-md-12">
        <div class="card" style="margin-top: 0.5em;">

          <div class="card-header" style="background-color: #E6E6FA;">
            <div class="row">
              <div class="col-md-3">
                <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Created Challenges</h5>
              </div>
              <div class="col-md-6"></div>
              <div class="col-md-3">
                <button class="btn btn-secondary switch_to_student" onclick="toggle_display_challenges()" style="float: right;" id="display_challenges_toggle">Show Challenges</button>
              </div>
            </div>
          </div>

          <div class="card-body" id="active_challenges_card_body" style="background-color: #E6E6FA;">

            <% if (typeof challenges_to_display === "undefined") { %>
                <h4><a href="../create_challenges">Reload</a> Page</h4>
            <% } else  { %>
              <% if (challenges_to_display.length == 0 || challenges_to_display == []) { %>
                <h4>No Challenges Created Yet</h4>
              <% } else { %>

                <table id="active_challenges" style="width: 100%; table-layout: fixed;">

                  <tr>
                    <th style="width: 25%;">Challenge Title</th>
                    <th style="width: 60%;">Challenge Description</th>
```

```
                        <th colspan="2" style="width: 15%;">
                          <!-- create a new challenge -->
                          <form action="/create_challenges/new_challenge" method="post">
                            <!-- submit button -->
                            <button type="submit" value="Create New Challenge" class="btn btn-secondary"
id="create_challenges_create_btn">CREATE NEW CHALLENGE</button>
                          </form>
                        </th>
                      </tr>

                      <% challenges_to_display.forEach((element, index) => { %>
                        <tr>
                          <td style="word-wrap: break-word;">
                            <%= element[0].split('\n')[1].toString() %>
                          </td>
                          <td style="word-wrap: break-word;">
                            <%= element[0].split('\n')[3].toString() %>
                          </td>
                          <!-- edit challenge -->
                          <td>
                            <form action="/create_challenges/edit/<%= element[1].toString() %>" method="post">
                              <button type="submit" value="edit_<%= element[1].toString() %>" class="btn btn-secondary
create_challenges_edit_btn">Edit</button>
                            </form>
                          </td>
                          <!-- create challenge -->
                          <td>
                            <form action="/create_challenges/delete/<%= element[1].toString() %>" method="post">
                              <button type="submit" value="delete_<%= element[1].toString() %>" class="btn btn-
secondary create_challenges_delete_btn">Delete</button>
                            </form>
                          </td>
                        </tr>
                      <% }); %>
                    </table>

                  <% } %>
                <% } %>

              </div>

            </div>
          </div>
        </div>

        <!-- assign challenges -->
        <div class="row justify-content-center">
          <div class="col-md-12">
            <div class="card" style="margin-top: 0.5em;">

              <div class="card-header" style="background-color: #ADD8E6;">
                <div class="row">
                  <div class="col-md-3">
                    <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Assign Challenges</h5>
                  </div>
```

```
                <div class="col-md-6"></div>
                <div class="col-md-3">
                  <button class="btn btn-secondary switch_to_student" onclick="toggle_assign_challenges()"
                    style="float: right;" id="assign_challenges_toggle">Hide Assign Challenges</button>
                </div>
              </div>
            </div>

            <div class="card-body" id="assign_challenges_card_body" style="background-color: #ADD8E6;">

              <form action="/create_challenges/assign" method="POST" id="assign_challenge_form">

                <% if (typeof challenges_to_display===="undefined" ) { %>

                <% } else { %>

                  <div class="form-group row">
                    <label class="col-md-3 col-form-label">CHALLENGE TITLE</label>
                    <div class="col-md-8">

                      <select id="assign_challenge_dd" name="assign_challenge_dd" class="form-control" required>
                        <option value="" selected hidden>select your challenge</option>
                        <% var challenge_titles=[]; %>
                        <% challenges_to_display.forEach((element, index)=> { %>
                          <% challenge_titles.push(element[0].split('\n')[1].toString()); %>
                        <% }); %>

                        <% if (typeof challenge_titles===="undefined" ) { %>

                          <option value="no_challenge">you haven't created any challenges</option>

                        <% } else { %>

                          <% for (var challenge_titles_counter=0; challenge_titles_counter < challenge_titles.length;
challenge_titles_counter++) { %>
                            <option value="<%= challenge_titles[challenge_titles_counter] %>">
                              <%= challenge_titles[challenge_titles_counter] %>
                            </option>
                          <% } %>

                        <% } %>
                      </select>

                    </div>
                  </div>

                <% } %>

                <% if (typeof class_students===="undefined" ) { %>

                <% } else { %>
                  <!-- DROPDOWN FOR CLASSES WITH STUDENTS -->
                  <% class_students.forEach((element, index)=> { %>

                    <div class="form-group row">
```

```
154.
155.                        <label class="col-md-3 col-form-label"><%= element[0].toString() %></label>
156.
157.                        <div class="col-md-5">
158.                          <select id="class_<%= element[0].toString() %>" data-placeholder="select <%=
element[0].toString() %> students" multiple class="form-control chosen-select">
159.                            <% if (element[2].length===0 || !element[2].length) { %>
160.                              <option value="" disabled>no students in this class</option>
161.                            <% } else { %>
162.                              <% element[2].toString().split(',').forEach((element_2, index_2)=> { %>
163.                                <option value="<%= element_2 %>" class="students">
164.                                  <%= element_2 %>
165.                                </option>
166.                              <%}); %>
167.                            <% } %>
168.                          </select>
169.                        </div>
170.
171.
172.                        <!-- only show select all / deselect all buttons if there are students in that class -->
173.                        <% if (element[2].length===0 || !element[2].length) { %>
174.                          <div class="col-md-2"></div>
175.                          <div class="col-md-2"></div>
176.                        <% } else { %>
177.                            <div class="col-md-2">
178.                              <button type="button" id="select_all_<%= element[0].toString() %>" class="chosen-toggle
select btn btn-secondary switch_to_student" onclick="select_all('<%- index.toString() %>')" style="width: 100%
!important;">Select all</button>
179.                            </div>
180.                            <div class="col-md-2">
181.                              <button type="button" class="chosen-toggle deselect btn btn-secondary
create_challenges_delete_btn" onclick="deselect_all('<%- index.toString() %>')" style="width: 100% !important;">Deselect
all</button>
182.                            </div>
183.                        <% } %>
184.
185.                      </div>
186.
187.                    <% }); %>
188.                  <% } %>
189.
190.                  <div class="form-group row">
191.                    <label class="col-md-3 col-form-label">DUE DATE</label>
192.                    <div class="col-md-8">
193.                      <input id="due_date" type="datetime-local" name="due_date" class="form-control" required />
194.                    </div>
195.
196.                    <div class="col-md-6"></div>
197.
198.                  </div>
199.
200.                  <!-- submit button -->
201.                  <div class="row">
202.                    <div class="col-md-8 offset-md-3" style="margin-top: 0.75em;">
203.                      <button type="submit" class="btn btn-primary">ASSIGN</button>
```

```
                    </div>
                </div>

                <!-- error message -->
                <% if (message.error) { %>
                    <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
                        <strong>
                            <%= message.error %>
                        </strong>
                    </div>
                <% } %>

                <% if (message.success) { %>
                    <div class="alert alert-success" role="alert" style="margin-top: 1em;">
                        <strong>
                            <%= message.success %>
                        </strong>
                    </div>
                <% } %>

                <div class="row" hidden aria-hidden="true">
                    <div class="col-md-12">
                        <label for="class_students">class_students</label>
                        <textarea id="class_students" name="class_students"><% class_students.forEach((element, index) => {
%><%= element + '\n'%><%}); %></textarea>
                    </div>
                </div>

                <div class="row" hidden aria-hidden="true">
                    <div class="col-md-12">
                        <label for="challenge_title_match_id">challenge_title_match_id</label>
                        <textarea id="challenge_title_match_id" name="challenge_title_match_id"><%-
challenge_title_match_id %></textarea>
                    </div>
                </div>

                <div class="row" hidden aria-hidden="true">
                    <div class="col-md-12">
                        <label for="challenges_to_display">challenges_to_display</label>
                        <textarea id="challenges_to_display" name="challenges_to_display"><%- challenges_to_display
%></textarea>
                    </div>
                </div>

                <div class="row" hidden aria-hidden="true">
                    <div class="col-md-12">
                        <label for="students_selected_2">SELECTED STUDENTS TO ASSIGN CHALLENGE TO</label>
                        <textarea id="students_selected_2" name="students_selected_2"></textarea>
                    </div>
                </div>

                <!-- hidden to be retrievable in js file -->
                <div class="row" hidden aria-hidden="true">
                    <div class="col-md-12">
                        <label for="challenge_title_match_id">SELECTED CHALLENGE TO ASSIGN ID</label>
```

```
                        <% if (typeof challenges_title_match_id=="undefined" ) { %>
                            <textarea id="challenge_title_match_id" name="challenge_title_match_id"></textarea>
                        <% } else { %>
                            <textarea id="challenge_title_match_id" name="challenge_title_match_id"><%
challenge_title_match_id.forEach(element => { %><%= element[0] + 'thisisaveryspeficbreak' + element[1] +
'thisisaveryspeficbreak' %><% }); %></textarea>
                        <% } %>
                    </div>
                </div>

            </form>

        </div>

      </div>
    </div>
  </div>

</body>

<% include ../partials/script %>

</html>

```

This page contains a table with all of the teacher's created challenges, each with a button that allows them to edit or delete the challenge. There is also a create challenge button which leads to a new page to create a new challenge. In addition, there is a form to select a created challenge, students, and a datetime in order to assign a challenge to students.

Again, this page is just the frontend, and I will go through the backend once I explain all of the frontend pages in **~/views/teacher_challenges/** because the router file for this section handles all requests that come from all of the EJS pages in this directory.

Note that there are a few **hidden** div tags which will be explained later, but simply put, they are so that I can access specific data from different GET/POST methods without using a global variable or putting the data as part of the URL.

~/public/javascripts/active_challenges_display_toggler.js

```
$(document).ready(function () {
    $('#active_challenges_card_body').hide();
    $('#active_challenges').hide();
    $('#challenges_havent_been_assigned').hide();
    $('#students_have_submitted').hide();
    $('#students_have_submitted_table').hide();
    $('#students_yet_to').hide();
```

```
 8.    $('#students_yet_to_table').hide();
 9.    $('#students_missed').hide();
10.    $('#students_missed_table').hide();
11. });
12.
13. function toggle_display_challenges() {
14.    $('#active_challenges_card_body').toggle();
15.    $('#active_challenges').toggle();
16.    if ($('#display_challenges_toggle').text() == "Show Challenges" || $('#display_challenges_toggle').text() === "Show
Challenges") {
17.        $('#display_challenges_toggle').text("Hide Challenges");
18.    } else {
19.        $('#display_challenges_toggle').text("Show Challenges");
20.    };
21. };
22.
23. function toggle_assign_challenges() {
24.    $('#assign_challenges_card_body').toggle();
25.    $('#assign_challenge_form').toggle();
26.    if ($('#assign_challenges_toggle').text() == "Show Assign Challenges" || $('#assign_challenges_toggle').text() === "Show
Assign Challenges") {
27.        $('#assign_challenges_toggle').text("Hide Assign Challenges");
28.    } else {
29.        $('#assign_challenges_toggle').text("Show Assign Challenges");
30.    };
31. };
32.
33. function toggle_challenges_have_been_assigned() {
34.    $('#challenges_have_been_assigned').toggle();
35.    if ($('#display_challenges_have_been_toggle').text() == "Show Challenges" ||
$('#display_challenges_have_been_toggle').text() === "Show Challenges") {
36.        $('#display_challenges_have_been_toggle').text("Hide Challenges");
37.    } else {
38.        $('#display_challenges_have_been_toggle').text("Show Challenges");
39.    };
40. };
41.
42. function toggle_challenges_havent_been_assigned() {
43.    $('#challenges_havent_been_assigned').toggle();
44.    if ($('#display_challenges_havent_been_toggle').text() == "Show Challenges" ||
$('#display_challenges_havent_been_toggle').text() === "Show Challenges") {
45.        $('#display_challenges_havent_been_toggle').text("Hide Challenges");
46.    } else {
47.        $('#display_challenges_havent_been_toggle').text("Show Challenges");
48.    };
49. };
50.
51. function toggle_challenge_info() {
52.    $('#challenge_info').toggle();
53.    if ($('#display_challenge_info').text() == "Show Information" || $('#display_challenge_info').text() === "Show
Information") {
54.        $('#display_challenge_info').text("Hide Information");
55.    } else {
56.        $('#display_challenge_info').text("Show Information");
57.    };
```

```
}

function toggle_students_have_submitted() {
    $('#students_have_submitted').toggle();
    $('#students_have_submitted_table').toggle();
    if ($('#display_students_have_submitted').text() == "Show Students" || $('#display_students_have_submitted').text() === "Show Students") {
        $('#display_students_have_submitted').text("Hide Students");
    } else {
        $('#display_students_have_submitted').text("Show Students");
    };
}

function toggle_students_yet_to() {
    $('#students_yet_to').toggle();
    $('#students_yet_to_table').toggle();
    if ($('#display_students_yet_to').text() == "Show Students" || $('#display_students_yet_to').text() === "Show Students") {
        $('#display_students_yet_to').text("Hide Students");
    } else {
        $('#display_students_yet_to').text("Show Students");
    };
}

function toggle_students_missed() {
    $('#students_missed').toggle();
    $('#students_missed_table').toggle();
    if ($('#display_students_missed').text() == "Show Students" || $('#display_students_missed').text() === "Show Students") {
        $('#display_students_missed').text("Hide Students");
    } else {
        $('#display_students_missed').text("Show Students");
    };
}

```

This JavaScript page takes care of showing and hiding all forms to do with the teacher side of challenges. Some of this may not currently make sense, but after going through this entire Implementation section, it should. It is quite similar to **~/public/javascripts/form_toggler.js**.

~/views/teacher_challenges/new_challenge.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>
```

```
12.
13.         <div class="row">
14.             <div class="col-md-12">
15.
16.                 <form action="/create_challenges/new_challenge/create" method="post">
17.
18.                     <!-- challenge title -->
19.                     <div class="row">
20.                         <div class="col-md-12">
21.                             <label for="challenge_title">CHALLENGE TITLE</label>
22.                             <textarea id="challenge_title" name="challenge_title" wrap="off" placeholder="Generic Challenge Title"
required style="overflow-y: scroll; height: 1.5em; width: 100%;"></textarea>
23.                         </div>
24.                     </div>
25.
26.                     <!-- challenge description -->
27.                     <div class="row">
28.                         <div class="col-md-12">
29.                             <label for="challenge_description">CHALLENGE DESCRIPTION</label>
30.                             <textarea id="challenge_description" name="challenge_description" wrap="off" placeholder="Generic
Challenge Description" required style="overflow-y: scroll; height: 20em; width: 100%;"></textarea>
31.                         </div>
32.                     </div>
33.
34.                     <!-- submit button -->
35.                     <button type="submit" value="Create" class="btn btn-primary">CREATE</button>
36.                 </form>
37.
38.                 <% if (message.error) { %>
39.                     <div class="alert alert-danger" role="alert">
40.                         <strong>
41.                             <%= message.error %>
42.                         </strong>
43.                     </div>
44.                 <% } %>
45.
46.                 <% if (message.success) { %>
47.                     <div class="alert alert-success" role="alert">
48.                         <strong>
49.                             <%= message.success %>
50.                         </strong>
51.                     </div>
52.                 <% } %>
53.
54.             </div>
55.         </div>
56. </body>
57.
58. <% include ../partials/script %>
59.
60. </html>
61.
```

This page contains the form for creating a new challenge. The user needs to enter text for a Challenge Title and Challenge Description, as talked about in the Documented Design section, and by using **required** fields, some of the validation is already complete. Clicking the submit button will send a POST request, which again, is handled in the router file.

~/views/teacher_challenges/edit_challenge.ejs

```
1. <!DOCTYPE html>
2. <html lang="en">
3.
4. <head>
5.    <% include ../partials/head %>
6. </head>
7.
8. <body>
9.    <div class="container-fluid">
10.
11.       <% include ../partials/menu %>
12.
13.          <div class="row">
14.             <div class="col-md-12">
15.
16.                <form action="/create_challenges/edit/update" method="post">
17.
18.                   <!-- challenge title -->
19.                   <div class="row">
20.                      <div class="col-md-12">
21.                         <label for="challenge_title">CHALLENGE TITLE</label>
22.                         <textarea id="challenge_title" name="challenge_title" wrap="off"
23.                            placeholder="Generic Challenge Title" required
24.                            style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
25.                      </div>
26.                   </div>
27.
28.                   <!-- challenge description -->
29.                   <div class="row">
30.                      <div class="col-md-12">
31.                         <label for="challenge_description">CHALLENGE DESCRIPTION</label>
32.                         <textarea id="challenge_description" name="challenge_description" wrap="off"
33.                            placeholder="Generic Challenge Description" required
34.                            style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
35.                      </div>
36.                   </div>
37.
38.                   <!-- submit button -->
39.                   <button type="submit" value="Update" class="btn btn-primary">UPDATE</button>
40.
41.                   <!-- hidden to be retrievable in js file -->
42.                   <div class="row" hidden aria-hidden="true">
43.                      <div class="col-md-12">
44.                         <label for="challenge_teacher_id">CHALLENGE TEACHER ID</label>
```

```
                    <textarea id="challenge_teacher_id" name="challenge_teacher_id"><%= challenge_teacher_id %></textarea>
                  </div>
                </div>

                <!-- hidden to be retrievable in js file -->
                <div class="row" hidden aria-hidden="true">
                  <div class="col-md-12">
                    <label for="challenge_title_og">CHALLENGE TITLE</label>
                    <textarea id="challenge_title_og" name="challenge_title_og" wrap="off" placeholder="Generic Challenge Title" required
                      style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
                  </div>
                </div>

                <!-- hidden to be retrievable in js file -->
                <div class="row" hidden aria-hidden="true">
                  <div class="col-md-12">
                    <label for="challenge_description_og">CHALLENGE DESCRIPTION</label>
                    <textarea id="challenge_description_og" name="challenge_description_og" wrap="off"
                      placeholder="Generic Challenge Description" required
                      style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
                  </div>
                </div>

              </form>

              <% if (message.error) { %>
                <div class="alert alert-danger" role="alert">
                  <strong>
                    <%= message.error %>
                  </strong>
                </div>
              <% } %>

              <% if (message.success) { %>
                <div class="alert alert-success" role="alert">
                  <strong>
                    <%= message.success %>
                  </strong>
                </div>
              <% } %>

            </div>
          </div>
</body>

<% include ../partials/script %>

</html>

```

This page allows a teacher to edit the Challenge Description and Challenge Title of a challenge. Here, the difference to the creating challenges page is that the **<%= %>** EJS tags are used to add the contents of the variables located in the tags into the HTML fields, so the Challenge Title and Challenge Description that already exists for the specific challenge are already filled out and only need to be altered, rather than rewritten. Once again, there are hidden fields for retrieving data.

~/routes/teacher_challenges/create_challenges.ejs

```
var express = require('express');
var router = express.Router();

var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// for writing to files
var fs = require('fs');

// for array/list operations
var _ = require('lodash');

// coming to create_challenge page
router.get('/', auth.check_authenticated, async function (req, res, next) {

  try {

    res.locals.message = req.flash();

    await get_challenge_txt_and_id(req, res, next).then((row_challenge_txt_id) => {

      var challenges_to_display = [];
      // var challenges_only = [];

      row_challenge_txt_id.forEach(element => {
        challenges_to_display.push([element['challenge_blob'].toString(), element['challenge_teacher_id']]);
        // challenges_only.push(element['challenge_blob'].toString());
      });

      var challenge_title_match_id = [];

      challenges_to_display.forEach(element => {
        // populate list of [challenge_title, challenge_teacher_id]
        challenge_title_match_id.push([element[0].split('\n')[1], element[1]]);
      });

      return {
        'challenges_to_display': challenges_to_display,
        // 'challenges_only': challenges_only,
        'challenge_title_match_id': challenge_title_match_id
      };
```

Multi-dimensional arrays

```
44.
45.     }).then(async (useful_vars) => {
46.
47.       // store 2d list of [class (year + teacher id, e.g. 10RAW), class_id]
48.       var classes = [];
49.
50.       await get_classes_of_teacher(req, res, next).then((row_classes_of_teacher) => {
51.         row_classes_of_teacher.forEach((element, index) => {
52.           classes.push([element['CONCAT(year_group, class_code)'], element['class_id']]);
53.         });
54.       });
55.
56.       useful_vars['classes'] = classes;
57.
58.       return useful_vars;
59.
60.     }).then(async (useful_vars) => {
61.
62.       await get_students_in_teacher_classes(req, res, next, useful_vars).then((row_class_students) => {
63.
64.         var list_of_class_students = [];
65.
66.         useful_vars['classes'].forEach((element, index) => {
67.           list_of_class_students.push([useful_vars['classes'][index][0], useful_vars['classes'][index][1],
row_class_students[index][1]]);
68.         });
69.
70.         useful_vars['list_of_class_students'] = list_of_class_students;
71.
72.       });
73.
74.       console.log('challenges_to_display');
75.       console.log(useful_vars['challenges_to_display']);
76.       console.log('challenge_title_match_id');
77.       console.log(useful_vars['challenge_title_match_id']);
78.       console.log('classes');
79.       console.log(useful_vars['classes']);
80.       console.log('list_of_class_students');
81.       console.log(useful_vars['list_of_class_students']);
82.
83.       return useful_vars;
84.
85.     }).then(async (useful_vars) => {
86.
87.       useful_vars['challenges_to_display'].forEach(async (element, index) => {
88.
89.         // delete route for each challenge using the challenge_teacher_id
90.         var route_delete = '/delete/' + element[1];
91.
92.         router.post(route_delete, async function (req, res, next) {
93.
94.           await delete_challenge(req, res, next, element).then(() => {
95.             return res.redirect('/create_challenges');
96.           });
97.
```

```
            });

            // edit route for each challenge using the challenge_teacher_id
            var route_edit = '/edit/' + element[1];

            router.post(route_edit, async function (req, res, next) {

                res.locals.message = req.flash();

                res.render('teacher_challenges/edit_challenge', {
                    title: 'Edit Challenge',
                    role: req.user.role,
                    email: req.user.email,
                    session_id: req.sessionID,
                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                    challenge_title: element[0].split('\n')[1].toString(),
                    challenge_description: element[0].split('\n')[3].toString(),
                    challenge_teacher_id: element[1].toString(),
                });

            });

        });

        res.render('teacher_challenges/create_challenges', {
            title: 'Create Challenges',
            menu_id: 'create_challenges',
            role: req.user.role,
            email: req.user.email,
            // challenge_titles: challenge_titles,
            // challenge_descriptions: challenge_descriptions,
            challenges_to_display: useful_vars['challenges_to_display'],
            challenge_title_match_id: useful_vars['challenge_title_match_id'],
            class_students: useful_vars['list_of_class_students'],
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });

    });

  } catch (any_error) {
    console.log(any_error);
    res.render('teacher_challenges/create_challenges', {
        title: 'Create Challenges',
        menu_id: 'create_challenges',
        role: req.user.role,
        email: req.user.email,
        session_id: req.sessionID,
        session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
    });
  }

});

var get_challenge_txt_and_id = (req, res, next) => {
```

```
153.    return new Promise((resolve, reject) => {
154.      try {
155.
156.        // get challenge text from teacher_id
157.        db_connection.query(
158.          'SELECT * FROM Challenge_File, Challenge_Teacher WHERE Challenge_File.challenge_file_id =
Challenge_Teacher.challenge_file_id AND Challenge_Teacher.teacher_id = ?;',
159.          [req.user.id],
160.          function (err, rows) {
161.
162.            if (err) {
163.              console.log(err);
164.              req.flash('error', ' An error occured');
165.              res.locals.message = req.flash();
166.              reject(err);
167.            };
168.
169.            resolve(rows);
170.
171.          }
172.        );
173.
174.      } catch (error) {
175.        reject(error);
176.      }
177.    });
178. };
179.
180. var get_classes_of_teacher = (req, res, next) => {
181.    return new Promise((resolve, reject) => {
182.      try {
183.
184.        // get classes of currently signed in teacher
185.        db_connection.query(
186.          'SELECT CONCAT(year_group, class_code), class_id FROM Class, Teacher WHERE Class.teacher_id =
Teacher.teacher_id AND Class.teacher_id = ?;',
187.          [req.user.id],
188.          function (err, rows) {
189.
190.            if (err) {
191.              console.log(err);
192.              req.flash('error', ' An error occured');
193.              res.locals.message = req.flash();
194.              return res.redirect('/create_challenges');
195.              reject(err)
196.            };
197.
198.            resolve(rows);
199.
200.          }
201.        );
202.
203.      } catch (error) {
204.        reject(error);
205.      }
```

SQL Statements with join as evidence of Complex data model in database (eg several interlinked tables)

```
    });
};

var get_students_in_teacher_classes = async (req, res, next, useful_vars) => {
    return new Promise(async (resolve, reject) => {
        try {

            // 2/3d list of [[class_id, [student1, student2, etc.]]]
            var class_students = [];

            await useful_vars['classes'].forEach((list_element, list_index) => {

                get_student_in_specific_class(req, res, next, list_element[1]).then((row_students) => {

                    var students = [];

                    row_students.forEach((element, index) => {
                        students.push(element['CONCAT(student_name, student_number)']);
                    });

                    class_students.push([list_element[1], students]);

                    return class_students;

                }).then((class_students) => {
                    if (list_index == useful_vars['classes'].length - 1) {
                        resolve(class_students);
                    };
                });

            });

        } catch (error) {
            reject(error);
        }
    });
};
var get_student_in_specific_class = async (req, res, next, teacher_class) => {
    return new Promise((resolve, reject) => {
        try {

            // get classes of currently signed in teacher
            db_connection.query(
                'SELECT CONCAT(student_name, student_number) FROM Student, Students_In_Classes WHERE Student.student_id = Students_In_Classes.student_id AND class_id = ?;',
                [teacher_class],
                function (err, rows) {

                    if (err) {
                        console.log(err);
                        req.flash('error', ' An error occured');
                        res.locals.message = req.flash();
                        return res.redirect('/create_challenges');
                        reject(err)
                    };
```

```
260.
261.                resolve(rows);
262.
263.            }
264.        );
265.
266.    } catch (error) {
267.        reject(error);
268.    }
269.  });
270. };
271.
272. var delete_challenge = (req, res, next, challenges_to_display_element) => {
273.  return new Promise((resolve, reject) => {
274.    try {
275.
276.      // delete challenge
277.      db_connection.query(
278.        'DELETE FROM Challenge_File WHERE challenge_file_id = (SELECT challenge_file_id FROM Challenge_Teacher
WHERE challenge_teacher_id = ?);',
279.        [challenges_to_display_element[1]],
280.        function (err, rows) {
281.
282.          if (err) {
283.            console.log(err);
284.            req.flash('error', ' An error occured');
285.            res.locals.message = req.flash();
286.            return res.redirect('/create_challenges');
287.            reject(err)
288.          };
289.
290.          resolve(rows);
291.
292.        }
293.      );
294.
295.    } catch (error) {
296.      reject(error);
297.    }
298.  });
299. };
300.
301. // redirect to new_challenge page creation - testing stuffs
302. router.post('/new_challenge', function (req, res, next) {
303.
304.  // code stuffs here
305.  console.log('new_challenge button clicked');
306.
307.  // redirect to page to create a new challenge
308.  res.redirect('/create_challenges/new_challenge');
309.
310. });
311.
312. // coming to create_challenge page
313. router.get('/new_challenge', auth.check_authenticated, function (req, res, next) {
```

```
314.
315.    res.locals.message = req.flash();
316.
317.    res.render('teacher_challenges/new_challenge', {
318.       title: 'New Challenge',
319.       role: req.user.role,
320.       email: req.user.email,
321.       session_id: req.sessionID,
322.       session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
323.    });
324.
325. });
326.
327. // create new challenge and put in database
328. router.post('/new_challenge/create', function (req, res, next) {
329.
330.    try {
331.
332.       var create_challenge_file_path = './public/text_files/teacher_create_challenge.txt';
333.
334.       // check if file exists
335.
336.       // ASYNCHRONOUS
337.       // fs.access(create_challenge_file_path, fs.F_OK, (err) => {
338.       //  if (err) {
339.       //     console.error(err);
340.       //     return;
341.       //  };
342.       //  // file exists
343.       //  console.log('Teacher Create Challenge file already exists');
344.       // });
345.
346.       // SYNCHRONOUS
347.       try {
348.          if (fs.existsSync(create_challenge_file_path)) {
349.             // file exists
350.             console.log('Teacher Create Challenge file already exists');
351.
352.             try {
353.                fs.unlinkSync(create_challenge_file_path);
354.                // file removed
355.                console.log('Teacher Create Challenge file deleted');
356.             } catch (error) {
357.                console.error(error);
358.             };
359.          }
360.       } catch (err) {
361.          console.error(err);
362.       };
363.
364.       // all user-side scripting for validation
365.       var error_message = false;
366.
367.       // ensure everything filled out
368.       if (req.body.challenge_title.length === 0) {
```

```
        error_message = true;
        req.flash('error', ' Challenge Title is required');
    }
    if (req.body.challenge_description.length === 0) {
        error_message = true;
        req.flash('error', ' Challenge Description is required');
    }

    // ensure challenge title length is <= 100 characters
    if (req.body.challenge_title.length > 100) {
        error_message = true;
        req.flash('error', ' Challenge Title length must be max 100 characters');
    }

    // return with the error message
    if (error_message) {
        res.locals.message = req.flash();
        return res.render('teacher_challenges/new_challenge', {
            title: 'New Challenge',
            role: req.user.role,
            email: req.user.email,
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });
    }

    var file_data = '###CHALLENGE TITLE###\n' + req.body.challenge_title.toString() + '\n###CHALLENGE
DESCRIPTION###\n' + req.body.challenge_description.toString();
    fs.writeFileSync(create_challenge_file_path, file_data,
    // callback function that is called after writing file is done
    function (err) {
        if (err) {
            console.log(err);
        };
        console.log('Data written to Teacher Create Challenge file successfully')
    });

    var data_to_db = fs.readFileSync(create_challenge_file_path);

    // put file as MEDIUMBLOB in db
    db_connection.query(
        'INSERT INTO Challenge_File (challenge_blob) VALUES (?);',
        [data_to_db],
        function (err, rows) {
            if (err) {
                console.log(err);
                req.flash('error', ' An error occured');
                res.locals.message = req.flash();
                return res.render('teacher_challenges/new_challenge', {
                    title: 'New Challenge',
                    role: req.user.role,
                    email: req.user.email,
                    session_id: req.sessionID,
                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                });
```

```
            }
            console.log(rows);
          }
        );

        // record relation b/w teacher and challenge
        db_connection.query(
          'INSERT INTO Challenge_Teacher (challenge_file_id, teacher_id) VALUES ((SELECT challenge_file_id FROM Challenge_File WHERE challenge_blob = ?), ?);',
          [data_to_db, req.user.id],
          function (err, rows) {
            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              return res.render('teacher_challenges/new_challenge', {
                title: 'New Challenge',
                role: req.user.role,
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
              });
            }
            console.log(rows);
          }
        );

        // redirect to page to create a new challenge
        return res.redirect('/create_challenges');

      } catch (any_error) {
        res.locals.message = req.flash();
        return res.render('teacher_challenges/new_challenge', {
          title: 'New Challenge',
          role: req.user.role,
          email: req.user.email,
          session_id: req.sessionID,
          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });
      }

});

// update the database for edited challenge
router.post('/edit/update', function (req, res, next) {

    try {

        var edit_challenge_file_path = './public/text_files/teacher_edit_challenge.txt';

        // check if file exists

        // ASYNCHRONOUS
        // fs.access(create_challenge_file_path, fs.F_OK, (err) => {
        //  if (err) {
```

```
//      console.error(err);
//      return;
//  };
//  // file exists
//  console.log('Teacher Create Challenge file already exists');
// });

// SYNCHRONOUS
try {
  if (fs.existsSync(edit_challenge_file_path)) {
    // file exists
    console.log('Teacher Edit Challenge file already exists');

    try {
      fs.unlinkSync(edit_challenge_file_path);
      // file removed
      console.log('Teacher Edit Challenge file deleted');
    } catch (error) {
      console.error(error);
    };
  }
} catch (err) {
  console.error(err);
};

// all user-side scripting for validation
var error_message = false;

// ensure everything filled out
if (req.body.challenge_title.length === 0) {
  error_message = true;
  req.flash('error', ' Challenge Title is required');
}
if (req.body.challenge_description.length === 0) {
  error_message = true;
  req.flash('error', ' Challenge Description is required');
}

// ensure challenge title length is <= 100 characters
if (req.body.challenge_title.length > 100) {
  error_message = true;
  req.flash('error', ' Challenge Title length must be max 100 characters');
}

// return with the error message
if (error_message) {
  res.locals.message = req.flash();
  res.render('teacher_challenges/edit_challenge', {
    title: 'Edit Challenge',
    role: req.user.role,
    email: req.user.email,
    session_id: req.sessionID,
    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
    challenge_title: req.body.challenge_title_og.toString(),
    challenge_description: req.body.challenge_description_og.toString(),
```

```
            challenge_teacher_id: req.body.challenge_teacher_id.toString(),
        });
    }

    var file_data = '###CHALLENGE TITLE###\n' + req.body.challenge_title.toString() + '\n###CHALLENGE
DESCRIPTION###\n' + req.body.challenge_description.toString();
    fs.writeFileSync(edit_challenge_file_path, file_data,
        // callback function that is called after writing file is done
        function (err) {
            if (err) {
                console.log(err);
            };
            console.log('Data written to Teacher Create Challenge file successfully')
        });

    var data_to_db = fs.readFileSync(edit_challenge_file_path);

    // put file as MEDIUMBLOB in db
    db_connection.query(
        'UPDATE Challenge_File SET challenge_blob = ? WHERE challenge_file_id = (SELECT challenge_file_id FROM
Challenge_Teacher WHERE challenge_teacher_id = ?);',
        [data_to_db, req.body.challenge_teacher_id],
        function (err, rows) {
            if (err) {
                console.log(err);
                req.flash('error', ' An error occured');
                res.locals.message = req.flash();
                return res.render('teacher_challenges/edit_challenge', {
                    title: 'Edit Challenge',
                    role: req.user.role,
                    email: req.user.email,
                    session_id: req.sessionID,
                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                    challenge_title: req.body.challenge_title_og.toString(),
                    challenge_description: req.body.challenge_description_og.toString(),
                    challenge_teacher_id: req.body.challenge_teacher_id.toString(),
                });
            }
            console.log(rows);
        }
    );

    // redirect to page to create a new challenge
    return res.redirect('/create_challenges');

  } catch (any_error) {
    res.locals.message = req.flash();
    return res.render('teacher_challenges/edit_challenge', {
        title: 'Edit Challenge',
        role: req.user.role,
        email: req.user.email,
        session_id: req.sessionID,
        session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        challenge_title: req.body.challenge_title_og.toString(),
        challenge_description: req.body.challenge_description_og.toString(),
```

UPDATE query with user-generated text files converted to MEDIUMBLOBs as evidence of User/CASE-generated DDL script

```
        challenge_teacher_id: req.body.challenge_teacher_id.toString(),
    });
  }

});

// assign challenges to students/classes
router.post('/assign', async function (req, res, next) {

  console.log('assign button clicked');

  var challenges_to_display_temp = req.body.challenges_to_display.toString().split('###CHALLENGE TITLE###').map(element => element.replace(/[\r]/gm, '')).filter(element => element !== '');
  // all vars to keep challenges + students updated
  var challenges_to_display = [];
  challenges_to_display_temp.forEach((element, index) => {
    if (index !== challenges_to_display_temp.length - 1) {
      challenges_to_display.push(['###CHALLENGE TITLE###' + element.slice(0, - 2 - element.split(',').reverse()[1].length), parseInt(element.split(',').reverse()[1])]);
    } else {
      challenges_to_display.push(['###CHALLENGE TITLE###' + element.slice(0, - 1 - element.split(',').reverse()[0].length), parseInt(element.split(',').reverse()[0])]);
    };
  });

  var challenge_title_match_id = req.body.challenge_title_match_id.filter(element => element !== '')[0].split(',');
  challenge_title_match_id.forEach((element, index) => {
    if (index % 2 === 0) {
      // console.log(element, index)
      challenge_title_match_id[index] = [element.toString(), parseInt(challenge_title_match_id[index + 1])];
    } else {
      challenge_title_match_id[index] = '';
    };
  });
  challenge_title_match_id = challenge_title_match_id.filter(element => element !== '');

  var class_students = req.body.class_students.split('\n').filter(element => element !== '');
  class_students.filter(element => element !== '').forEach((element, index) => {
    class_students[index] = element.replace('\r', '').split(',');
    class_students[index].forEach((element_2, index_2) => {

      if (index_2 == 1) {
        class_students[index][index_2] = parseInt(element_2);
      }

      if (index_2 == 2) {
        if (element_2 == '') {
          class_students[index][index_2] = [];
        } else {
          class_students[index][index_2] = class_students[index].slice(2, class_students[index].length + 1);
        }
      }
    });

    class_students[index] = class_students[index].slice(0, 3);
```

Complex List operations

```
637.    });
638.
639.    // var students_to_assign = req.body.students_selected.split(' | ').filter(element => element !== '');
640.    var students_to_assign = req.body.students_selected_2.split(',').filter(element => element !== '');
641.    var due_date = new Date(req.body.due_date);
642.    var today = new Date();
643.
644.    try {
645.
646.      // all user-side scripting for validation
647.      var error_message = false;
648.
649.      // ensure at least 1 student selected
650.      if (students_to_assign.length === 0 || students_to_assign === [] || typeof students_to_assign === "undefined") {
651.        error_message = true;
652.        req.flash('error', ' Select at least 1 student');
653.      }
654.
655.      // due date must be > 24 hours away from current time
656.      if (due_date.getTime() <= new Date(today.getTime() + 60 * 60 * 24 * 1000).getTime()) {
657.        error_message = true;
658.        req.flash('error', ' Due date must be at least 24 hours after current time');
659.      }
660.
661.      // return with the error message
662.      if (error_message) {
663.        res.locals.message = req.flash();
664.        return res.render('teacher_challenges/create_challenges', {
665.          title: 'Create Challenges',
666.          menu_id: 'create_challenges',
667.          role: req.user.role,
668.          email: req.user.email,
669.          challenges_to_display: challenges_to_display,
670.          challenge_title_match_id: challenge_title_match_id,
671.          class_students: class_students,
672.          session_id: req.sessionID,
673.          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
674.        });
675.      }
676.
677.      var student_ids_assigned_this_challenge_already = [];
678.      var student_names_assigned_this_challenge_already = [];
679.
680.      await get_students_already_assigned_challenge(req, res, next, challenges_to_display, challenge_title_match_id,
class_students).then((rows) => {
681.        rows.forEach((element, index) => {
682.          student_ids_assigned_this_challenge_already.push([element['CONCAT(Student.student_name,
Student.student_number)'], element['student_id']]);
683.        });
684.      }).then(() => {
685.
686.        student_ids_assigned_this_challenge_already.forEach((element, index) => {
687.          student_names_assigned_this_challenge_already.push(element[0]);
688.        });
689.
```

```
690.        var contains_1 = student_names_assigned_this_challenge_already.some(element => {
691.          return students_to_assign.includes(element);
692.        });
693.        var contains_2 = students_to_assign.some(element => {
694.          return student_names_assigned_this_challenge_already.includes(element);
695.        });
696.
697.        if (contains_1 == true || contains_2 == true) {
698.          error_message = true;
699.          req.flash('error', ' At least 1 student has already been assigned this challenge!');
700.        };
701.
702.        // return with the error message
703.        if (error_message) {
704.          res.locals.message = req.flash();
705.          return res.render('teacher_challenges/create_challenges', {
706.            title: 'Create Challenges',
707.            menu_id: 'create_challenges',
708.            role: req.user.role,
709.            email: req.user.email,
710.            challenges_to_display: challenges_to_display,
711.            challenge_title_match_id: challenge_title_match_id,
712.            class_students: class_students,
713.            session_id: req.sessionID,
714.            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
715.          });
716.        } else {
717.
718.          students_to_assign.forEach((element, index) => {
719.            // record relation b/w teacher and challenge
720.            db_connection.query(
721.              'INSERT INTO Assigned_Challenges (challenge_teacher_id, student_id, due_date) VALUES (?, (SELECT
student_id FROM Student WHERE student_name = ? AND student_number = ?), ?);',
722.              [challenge_title_match_id[_.findIndex(challenge_title_match_id, function (el) { return el[0] ==
req.body.assign_challenge_dd })][1], element.slice(0, -4), element.slice(-4), due_date.toISOString().slice(0, 19).replace('T', ' ')],
723.              function (err, rows) {
724.                if (err) {
725.                  console.log(err);
726.                  req.flash('error', ' An error occured');
727.                  res.locals.message = req.flash();
728.                  return res.render('teacher_challenges/create_challenges', {
729.                    title: 'Create Challenges',
730.                    menu_id: 'create_challenges',
731.                    role: req.user.role,
732.                    email: req.user.email,
733.                    challenges_to_display: challenges_to_display,
734.                    challenge_title_match_id: challenge_title_match_id,
735.                    class_students: class_students,
736.                    session_id: req.sessionID,
737.                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
738.                  });
739.                }
740.                // console.log(rows);
741.              }
742.            );
```

```
            });

            req.flash('success', ' Assigned challenge to student(s)');
            res.locals.message = req.flash();
            return res.render('teacher_challenges/create_challenges', {
              title: 'Create Challenges',
              menu_id: 'create_challenges',
              role: req.user.role,
              email: req.user.email,
              challenges_to_display: challenges_to_display,
              challenge_title_match_id: challenge_title_match_id,
              class_students: class_students,
              session_id: req.sessionID,
              session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            });

          }

        });

    } catch (error) {
      console.log(error);
      // redirect to page to create a new challenge
      return res.render('teacher_challenges/create_challenges', {
        title: 'Create Challenges',
        menu_id: 'create_challenges',
        role: req.user.role,
        email: req.user.email,
        challenges_to_display: challenges_to_display,
        challenge_title_match_id: challenge_title_match_id,
        class_students: class_students,
        session_id: req.sessionID,
        session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
      });
    };
});

var get_students_already_assigned_challenge = (req, res, next, challenges_to_display, challenge_title_match_id,
class_students) => {
    return new Promise((resolve, reject) => {
      try {

        // get the students who have already been assigned the specific challenge
        db_connection.query(
          'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id FROM
Assigned_Challenges, Challenge_Teacher, Student WHERE Assigned_Challenges.challenge_teacher_id =
Challenge_Teacher.challenge_teacher_id AND Assigned_Challenges.student_id = Student.student_id AND
Assigned_Challenges.challenge_teacher_id = (SELECT challenge_teacher_id FROM Challenge_Teacher WHERE
challenge_teacher_id = ?);',
          [challenge_title_match_id[_.findIndex(challenge_title_match_id, function (el) { return el[0] ==
req.body.assign_challenge_dd })][1]],
          function (err, rows) {

            if (err) {
              console.log(err);
```

Cross-table query

```
792.                req.flash('error', ' An error occured');
793.                res.locals.message = req.flash();
794.                return res.render('teacher_challenges/create_challenges', {
795.                  title: 'Create Challenges',
796.                  menu_id: 'create_challenges',
797.                  role: req.user.role,
798.                  email: req.user.email,
799.                  challenges_to_display: challenges_to_display,
800.                  challenge_title_match_id: challenge_title_match_id,
801.                  class_students: class_students,
802.                  session_id: req.sessionID,
803.                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
804.                });
805.                reject(err)
806.              }
807.
808.              resolve(rows);
809.
810.            }
811.          );
812.
813.        } catch (error) {
814.          reject(error);
815.        }
816.    });
817. };
818.
819. module.exports = router;
820.
```

This file handles all the GET/POST methods for creating, editing, deleting, and assigning challenges.

Firstly, multiple asynchronous functions must be used in order to retrieve data from the database, using **.then()** to ensure the next processes are only run after the results of the previous function are returned. After getting the data to be displayed in the table (if it exists), I then had to define the edit and delete POST routes handlers within the GET **/** handler because I had to loop through the table of challenges. For each challenge, the route is **edit/** followed by a number, which is the **challenge_teacher_id** from the database. Likewise, the same is used for the delete route. The reason these routes are within the GET **/** handler is because I did not want to use global variables, which would have otherwise been required.

After those routes are handled, we render the **~/views/teacher_challenges/create_challenges.ejs** page, passing in the variables that are used in the view files to display data (this is how the backend and frontend communicate).

As mentioned before, these variables are stored in hidden fields so that they can be retrieved in other GET/POST methods, as seen in line 596, where the data stored in those fields has to be

processed, because while they were passed from the backend into the frontend, they were converted to strings, and the backend requires them to be arrays again.

Going back a bit, the post methods on the edit and create challenges pages are handled by UPDATE / DELETE queries to the database.

In the database, I used a blob file, instead of a text file directly, which was justified during the Documented Design section. However, in order to do this for the create and edit challenges queries, I need to create a text file with the text and then convert them into blobs in order to put them into the database. I did this synchronously, and using **~/public/text_files/teacher_create_challenge.txt** and **~/public/text_files/teacher_edit_challenge.txt**, where the data from the frontend challenge text areas is inserted into these text files. Using the **readFileSync()** function, I converted the txt files into medium blobs.

Going to the assign POST handler, firstly, the required data from the **hidden** field is processed. Here, I have prioritised efficiency for converting these into the necessary formats rather than readability. After that, I did some validation, such as ensuring that no students are being assigned the same challenge twice, and then the values are finally used in an INSERT query into the Assigned_Challenges table in order to "assign" the challenges.

I will now explain how the student side of this works, viewing their assigned challenges and submitted their solutions. After that, I will then return to the teacher side in order to view students' submitted solutions.

Viewing and Submitting Assigning Challenges as a Student

~/views/student_challenges/view_set_challenges.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
    <% include ./partials/script_toggle %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <% if (typeof all_completed_challenges === "undefined" && typeof all_missed_challenges === "undefined" && typeof all_challenges_to_do === "undefined") { %>
            <h4>Go back to the <a href="../">Home</a> Page</h4>
        <% } else { %>
```

```

        <!-- CHALLENGES TO DO -->
        <div class="row">
          <div class="col-md-12">

            <div class="row justify-content-center">
              <div class="col-md-12">
                <div class="card" style="margin-top: 0.5em;">

                  <div class="card-header" style="background-color: #ADD8E6;">
                    <div class="row">
                      <div class="col-md-4">
                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenges to Complete</h5>
                      </div>
                      <div class="col-md-5"></div>
                      <div class="col-md-3">
                        <button class="btn btn-secondary switch_to_student" onclick="toggle_display_challenges()" style="float: right;" id="display_challenges_toggle">Hide Challenges</button>
                      </div>
                    </div>
                  </div>

                  <div class="card-body" id="active_challenges_card_body" style="background-color: #ADD8E6;">

                    <% if (typeof all_challenges_to_do === "undefined" || all_challenges_to_do.length == 0 || all_challenges_to_do == []) { %>
                      <h5>No Challenges set</h5>
                    <% } else { %>

                      <table style="width: 100%;" id="active_challenges">

                        <tr>
                          <th>Challenge Title</th>
                          <th>Challenge Description</th>
                          <th>Due Date</th>
                          <th></th>
                        </tr>

                        <% all_challenges_to_do.forEach((element, index)=> { %>
                          <tr>
                            <td style="word-wrap: break-word;">
                              <%= element[0].split('\n')[1].toString() %>
                            </td>
                            <td style="word-wrap: break-word;">
                              <%= element[0].split('\n')[3].toString() %>
                            </td>
                            <td>
                              <%= element[2].toString() %>
                            </td>
                            <td>
                              <form action="/view_set_challenges/complete_challenge/<%= element[1].toString() %>" method="post">
                                <button type="submit" value="complete_challenge_<%= element[1].toString() %>" class="btn btn-secondary switch_to_teacher">Complete Challenge</button>
```

```
                                    </form>
                                  </td>
                                </tr>
                              <% }); %>

                            </table>

                          <% } %>

                        </div>

                      </div>
                    </div>
                  </div>

                </div>
              </div>

              <!-- CHALLENGES FINISHED IN TIME -->
              <div class="row">
                <div class="col-md-12">

                  <div class="row justify-content-center">
                    <div class="col-md-12">
                      <div class="card" style="margin-top: 0.5em;">

                        <div class="card-header" style="background-color: #E6E6FA;">
                          <div class="row">
                            <div class="col-md-4">
                              <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Completed Challenges</h5>
                            </div>
                            <div class="col-md-5"></div>
                            <div class="col-md-3">
                                <button class="btn btn-secondary switch_to_student" onclick="toggle_display_completed()"
                                  style="float: right;" id="display_completed_toggle">Show Challenges</button>
                             </div>
                          </div>
                        </div>

                        <div class="card-body" id="completed_card_body" style="background-color: #E6E6FA;">

                          <% if (typeof all_completed_challenges=="undefined" || all_completed_challenges.length==0 || all_completed_challenges==[]) { %>
                              <h5>No Completed Challenges</h5>
                          <% } else { %>

                            <table style="width: 100%;" id="completed">

                              <tr>
                                <th style="width: 25%;">Challenge Title</th>
                                <th style="width: 45%;">Challenge Description</th>
                                <th style="width: 10%;">Due Date</th>
                                <th style="width: 10%;">Completion Date</th>
                                <th style="width: 10%;"></th>
```

```
                        </tr>

                        <% all_completed_challenges.forEach((element, index)=> { %>
                          <tr>
                            <td style="word-wrap: break-word;">
                              <%= element[0].split('\n')[1].toString() %>
                            </td>
                            <td style="word-wrap: break-word;">
                              <%= element[0].split('\n')[3].toString() %>
                            </td>
                            <td>
                              <%= element[2].toString() %>
                            </td>
                            <td>
                              <%= element[3].toString() %>
                            </td>
                            <td>
                              <form action="/view_set_challenges/view_completed/<%= element[1].toString() %>"
method="post">
                                <button type="submit" value="view_completed_<%= element[1].toString() %>"
class="btn btn-secondary switch_to_student">View Answer</button>
                              </form>
                            </td>
                          </tr>
                        <% }); %>

                      </table>

                    <% } %>

                  </div>

                </div>
              </div>
            </div>

          </div>
        </div>

        <!-- CHALLENGES NOT FINISHED IN TIME -->
        <div class="row">
          <div class="col-md-12">

            <div class="row justify-content-center">
              <div class="col-md-12">
                <div class="card" style="margin-top: 0.5em;">

                  <div class="card-header" style="background-color: #FAA0A0;">
                    <div class="row">
                      <div class="col-md-5">
                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Missed Challenges</h5>
                      </div>
                      <div class="col-md-4"></div>
                      <div class="col-md-3">
```

```
                                <button class="btn btn-secondary switch_to_student" onclick="toggle_display_missed()" style="float: right;" id="display_missed_toggle">Show Challenges</button>
                            </div>
                        </div>
                    </div>

                    <div class="card-body" id="missed_card_body" style="background-color: #FAA0A0;">

                        <% if (typeof all_missed_challenges===="undefined" || all_missed_challenges.length==0 || all_missed_challenges==[]) { %>
                            <h5>No Completed Challenges</h5>
                        <% } else { %>

                            <table style="width: 100%;" id="missed">

                                <tr>
                                    <th>Challenge Title</th>
                                    <th>Challenge Description</th>
                                    <th>Due Date</th>
                                </tr>

                                <% all_missed_challenges.forEach((element, index)=> { %>
                                    <tr>
                                        <td style="word-wrap: break-word;">
                                            <%= element[0].split('\n')[1].toString() %>
                                        </td>
                                        <td style="word-wrap: break-word;">
                                            <%= element[0].split('\n')[3].toString() %>
                                        </td>
                                        <td>
                                            <%= element[2].toString() %>
                                        </td>
                                    </tr>
                                <% }); %>

                            </table>

                        <% } %>

                    </div>

                </div>
            </div>
        </div>

    </div>
  </div>

  <% } %>

 </div>
</body>

<% include ../partials/script %>

```

```
</html>

```

This file contains 3 sections. The completed challenges that a student has submitted in time; the challenges that a student has yet to complete; and the challenges that a student failed to complete in time. Within the submitted challenges table, the user has an option to view their solutions, and within the to-do challenges table, the student has the option to submit their solution. Once again, this works similar to **~/views/teacher_challenges/create_challenges.ejs**.

~/public/javascripts/student_toggles.js

```
$(document).ready(function () {
    $('#completed_card_body').hide();
    $('#completed').hide();
    $('#missed_card_body').hide();
    $('#missed').hide();
});

function toggle_display_challenges() {
    $('#active_challenges_card_body').toggle();
    $('#active_challenges').toggle();
    if ($('#display_challenges_toggle').text() == "Show Challenges" || $('#display_challenges_toggle').text() === "Show Challenges") {
        $('#display_challenges_toggle').text("Hide Challenges");
    } else {
        $('#display_challenges_toggle').text("Show Challenges");
    };
};

function toggle_display_completed() {
    $('#completed_card_body').toggle();
    $('#completed').toggle();
    if ($('#display_completed_toggle').text() == "Show Challenges" || $('#display_completed_toggle').text() === "Show Challenges") {
        $('#display_completed_toggle').text("Hide Challenges");
    } else {
        $('#display_completed_toggle').text("Show Challenges");
    };
};

function toggle_display_missed() {
    $('#missed_card_body').toggle();
    $('#missed').toggle();
    if ($('#display_missed_toggle').text() == "Show Challenges" || $('#display_missed_toggle').text() === "Show Challenges") {
        $('#display_missed_toggle').text("Hide Challenges");
    } else {
        $('#display_missed_toggle').text("Show Challenges");
    };
};

```

This is basically the student version of **~/public/javascripts/active_challenges_display_toggler.js**, but it takes care of all toggling on a single page, as there isn't a need to do this on other pages.

~/views/student_challenges/complete_challenge.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
  <% include ../partials/head %>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <div class="row">
      <div class="col-md-12">

        <form action="/view_set_challenges/complete_challenge/complete" method="post">

          <!-- challenge title -->
          <div class="row">
            <div class="col-md-12">
              <label for="challenge_title">CHALLENGE TITLE</label>
              <textarea id="challenge_title" name="challenge_title" wrap="off"
                placeholder="Generic Challenge Title" readonly
                style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
            </div>
          </div>

          <!-- challenge description -->
          <div class="row">
            <div class="col-md-12">
              <label for="challenge_description">CHALLENGE DESCRIPTION</label>
              <textarea id="challenge_description" name="challenge_description" wrap="off"
                placeholder="Generic Challenge Description" readonly
                style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
            </div>
          </div>

          <!-- challenge answer -->
          <div class="row">
            <div class="col-md-12">
              <label for="challenge_answer">CHALLENGE ANSWER</label>
              <textarea id="challenge_answer" name="challenge_answer" wrap="off"
                placeholder="Generic Challenge Answer" required
                style="overflow-y: scroll; height: 20em; width: 100%;"></textarea>
            </div>
          </div>

```

```
            <!-- submit button -->
            <button type="submit" value="Complete" class="btn btn-primary">Complete</button>

            <!-- hidden to be retrievable in js file -->
            <div class="row" hidden aria-hidden="true">
              <div class="col-md-12">
                <label for="assigned_challenge_id">ASSIGNED CHALLENGE ID</label>
                <textarea id="assigned_challenge_id"
                  name="assigned_challenge_id"><%= assigned_challenge_id %></textarea>
              </div>
            </div>

          </form>

          <% if (message.error) { %>
            <div class="alert alert-danger" role="alert">
              <strong>
                <%= message.error %>
              </strong>
            </div>
          <% } %>

          <% if (message.success) { %>
            <div class="alert alert-success" role="alert">
              <strong>
                <%= message.success %>
              </strong>
            </div>
          <% } %>

        </div>
      </div>

    </div>
</body>

<% include ../partials/script %>

</html>

```

This page allows the user to submit their challenge solution. It is like a mix of the create challenge and edit challenge page on the teacher side in that the user can see the Challenge Title and Challenge Description already, and they can write their Challenge Solution in another text field and click submit. However, they cannot edit the Challenge Title and Challenge Description.

~/views/student_challenges/view_completed_challenge.ejs

```
<!DOCTYPE html>
<html lang="en">

```

```
<head>
  <% include ../partials/head %>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <div class="row">
      <div class="col-md-12">

          <!-- challenge title -->
          <div class="row">
            <div class="col-md-12">
              <label for="challenge_title">CHALLENGE TITLE</label>
              <textarea id="challenge_title" name="challenge_title" wrap="off"
                placeholder="Generic Challenge Title" readonly
                style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
            </div>
          </div>

          <!-- challenge description -->
          <div class="row">
            <div class="col-md-12">
              <label for="challenge_description">CHALLENGE DESCRIPTION</label>
              <textarea id="challenge_description" name="challenge_description" wrap="off"
                placeholder="Generic Challenge Description" readonly
                style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
            </div>
          </div>

          <!-- challenge answer -->
          <div class="row">
            <div class="col-md-12">
              <label for="challenge_answer">CHALLENGE ANSWER</label>
              <textarea id="challenge_answer" name="challenge_answer" wrap="off" placeholder="Generic Challenge Answer"
                readonly style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_answer %></textarea>
            </div>
          </div>

          <button onclick="location.href='../';" class="btn btn-primary">Go Back</button>

      </div>
    </div>

  </div>
</body>

<% include ../partials/script %>

</html>

```

This page works in basically the same way as the teacher side's edit challenge page, except for the fact that the student cannot edit the fields, they can only view them, and there is an additional text area – the student's solution.

~/routes/student_challenges/view_set_challenges.js

```
var express = require('express');
var router = express.Router();

var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// for writing to files
var fs = require('fs');

// for array/list operations
var _ = require('lodash');

// for DATETIME conversions
var moment = require('moment');
moment.locale('en-gb');
moment.defaultFormat = 'L LT';
var utc_datetime = moment.utc().format();

// coming to create_challenge page
router.get('/', auth.check_authenticated, async function (req, res, next) {

  console.log(req.user.id);

  try {

    res.locals.message = req.flash();

    await get_all_solved_unsolved_challenges(req, res, next, true, req.user.id).then((rows) => {

      var all_completed_challenges = [];

      // from db to list
      rows.forEach(element => {
        all_completed_challenges.push([element['challenge_blob'].toString(), element['assigned_challenge_id'],
moment(element['due_date']).utc(utc_datetime).local().format(),
moment(element['completion_date']).utc(utc_datetime).local().format(), element['solution_blob'].toString()]);
      });

      console.log('all_completed_challenges');
      console.log(all_completed_challenges);

      return {
        'all_completed_challenges': all_completed_challenges
      };

```

Efficiency should always be prioritized over readability because comments can explain what is happening.

```
    }).then(async (useful_vars) => {

      await get_all_unsolved_due_before_after_curr_date_challenges(req, res, next, true, req.user.id).then((rows) => {

        var all_missed_challenges = [];

        // from db to list
        rows.forEach(element => {
          all_missed_challenges.push([element['challenge_blob'].toString(), element['assigned_challenge_id'],
moment(element['due_date']).utc(utc_datetime).local().format()]);
        });

        console.log('all_missed_challenges');
        console.log(all_missed_challenges);

        useful_vars['all_missed_challenges'] = all_missed_challenges;

      });

      return useful_vars;

    }).then(async (useful_vars) => {

      await get_all_unsolved_due_before_after_curr_date_challenges(req, res, next, false, req.user.id).then((rows) => {

        var all_challenges_to_do = [];

        // from db to list
        rows.forEach(element => {
          all_challenges_to_do.push([element['challenge_blob'].toString(), element['assigned_challenge_id'],
moment(element['due_date']).utc(utc_datetime).local().format()]);
        });

        console.log('all_challenges_to_do');
        console.log(all_challenges_to_do);

        useful_vars['all_challenges_to_do'] = all_challenges_to_do;

      });

      return useful_vars;

    }).then((useful_vars) => {

      useful_vars['all_challenges_to_do'].forEach(async (element, index) => {

        var route_complete = '/complete_challenge/' + element[1];

        router.post(route_complete, function (req, res, next) {

          res.locals.message = req.flash();

          res.render('student_challenges/complete_challenge', {
            title: 'Complete Challenge',
            role: req.user.role,
```

```
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                challenge_title: element[0].split('\n')[1].toString(),
                challenge_description: element[0].split('\n')[3].toString(),
                assigned_challenge_id: element[1].toString()
              });

            });

          });

          useful_vars['all_completed_challenges'].forEach(async (element, index) => {

            var route_view_completed = '/view_completed/' + element[1];

            router.post(route_view_completed, async function (req, res, next) {

              res.locals.message = req.flash();

              res.render('student_challenges/view_completed_challenge', {
                title: 'View Completed Challenge',
                role: req.user.role,
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                challenge_title: element[0].split('\n')[1].toString(),
                challenge_description: element[0].split('\n')[3].toString(),
                challenge_answer: element[4].toString()
              });

            });

          });

          return res.render('student_challenges/view_set_challenges', {
            title: 'View Set Challenges',
            menu_id: 'view_set_challenges',
            role: req.user.role,
            email: req.user.email,
            all_completed_challenges: useful_vars['all_completed_challenges'],
            all_missed_challenges: useful_vars['all_missed_challenges'],
            all_challenges_to_do: useful_vars['all_challenges_to_do'],
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
          });

        });

      } catch (any_error) {
        console.log(any_error);
        res.render('student_challenges/view_set_challenges', {
          title: 'View Set Challenges',
          menu_id: 'view_set_challenges',
          role: req.user.role,
```

```
154.          email: req.user.email,
155.          session_id: req.sessionID,
156.          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
157.        });
158.    }
159.
160. });
161.
162. var get_all_unsolved_due_before_after_curr_date_challenges = (req, res, next, before, user_id) => {
163.    return new Promise((resolve, reject) => {
164.        try {
165.
166.          // get all challenges that were due before the current date - i.e. missed
167.          if (before) {
168.            var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE
Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id =
Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND
Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student,
Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) AND
Assigned_Challenges.due_date < NOW() ORDER BY Assigned_Challenges.due_date ASC;';
169.          } else { // all challenges that can still be solved by the student, the due date is after the current time
170.            var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE
Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id =
Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND
Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student,
Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) AND
Assigned_Challenges.due_date >= NOW() ORDER BY Assigned_Challenges.due_date ASC;';
171.          };
172.
173.          db_connection.query(
174.            query_to_run,
175.            [user_id],
176.            function (err, rows) {
177.
178.              if (err) {
179.                console.log(err);
180.                req.flash('error', ' An error occured');
181.                res.locals.message = req.flash();
182.                return res.render('student_challenges/view_set_challenges', {
183.                  title: 'View Set Challenges',
184.                  menu_id: 'view_set_challenges',
185.                  role: req.user.role,
186.                  email: req.user.email,
187.                  session_id: req.sessionID,
188.                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
189.                });
190.                reject(err);
191.              };
192.
193.              resolve(rows);
194.            }
195.          );
196.
197.        } catch (error) {
198.          reject(error);
```

```
      }
    });
};
var get_all_solved_unsolved_challenges = (req, res, next, solved, user_id) => {
    return new Promise((resolve, reject) => {
        try {

            if (solved) {
                var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges, Solution_Student, Solution_File WHERE Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id = Assigned_Challenges.challenge_teacher_id AND Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id AND Solution_File.solution_file_id = Solution_Student.solution_file_id AND Assigned_Challenges.student_id = ? AND Assigned_Challenges.assigned_challenge_id IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student, Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) ORDER BY Assigned_Challenges.due_date ASC;';
            } else {
                var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id = Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student, Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) ORDER BY Assigned_Challenges.due_date ASC;';
            };

            db_connection.query(
                query_to_run,
                [user_id],
                function (err, rows) {

                    if (err) {
                        console.log(err);
                        req.flash('error', ' An error occured');
                        res.locals.message = req.flash();
                        return res.render('student_challenges/view_set_challenges', {
                            title: 'View Set Challenges',
                            menu_id: 'view_set_challenges',
                            role: req.user.role,
                            email: req.user.email,
                            session_id: req.sessionID,
                            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                        });
                        reject(err);
                    };

                    resolve(rows);
                }
            );

        } catch (error) {
            reject(error);
        }
    });
};

```

```
// create new challenge and put in database
router.post('/complete_challenge/complete', async function (req, res, next) {

  console.log('attempted completion');

  try {

    var complete_challenge_file_path = './public/text_files/student_complete_challenge.txt';

    // check if file exists

    // ASYNCHRONOUS
    // fs.access(complete_challenge_file_path, fs.F_OK, (err) => {
    //  if (err) {
    //     console.error(err);
    //     return;
    //  };
    //  // file exists
    //  console.log('Teacher Create Challenge file already exists');
    // });

    // SYNCHRONOUS
    try {
      if (fs.existsSync(complete_challenge_file_path)) {
        // file exists
        console.log('Student Complete Challenge file already exists');

        try {
          fs.unlinkSync(complete_challenge_file_path);
          // file removed
          console.log('Student Complete Challenge file deleted');
        } catch (error) {
          console.error(error);
        };
      }
    } catch (err) {
      console.error(err);
    };

    console.log('student_complete_challenge.txt updated');

    // all user-side scripting for validation
    var error_message = false;

    // ensure everything filled out
    if (req.body.challenge_answer.length === 0) {
      error_message = true;
      req.flash('error', ' Challenge Answer is required');
    }

    // return with the error message
    if (error_message) {
      res.locals.message = req.flash();
      return res.render('student_challenges/view_set_challenges', {
        title: 'View Set Challenges',
```

```
297.            menu_id: 'view_set_challenges',
298.            role: req.user.role,
299.            email: req.user.email,
300.            session_id: req.sessionID,
301.            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
302.          });
303.        }
304.
305.        var file_data = req.body.challenge_answer.toString();
306.        fs.writeFileSync(complete_challenge_file_path, file_data,
307.          // callback function that is called after writing file is done
308.          function (err) {
309.            if (err) {
310.              console.log(err);
311.            };
312.            console.log('Data written to Student Complete Challenge file successfully')
313.          });
314.
315.        var data_to_db = fs.readFileSync(complete_challenge_file_path);
316.
317.        console.log('before query 1');
318.
319.        // put file as MEDIUMBLOB in db
320.        await insert_solution_blob(req, res, next, data_to_db).then(async (rows) => {
321.          console.log('query 1 done');
322.          await insert_solution_student(req, res, next, data_to_db).then(() => {
323.            console.log('query 2 done');
324.            // redirect to page to create a new challenge
325.            return res.redirect('/view_set_challenges');
326.          });
327.        });
328.
329.      } catch (any_error) {
330.        res.locals.message = req.flash();
331.        return res.render('student_challenges/view_set_challenges', {
332.          title: 'View Set Challenges',
333.          menu_id: 'view_set_challenges',
334.          role: req.user.role,
335.          email: req.user.email,
336.          session_id: req.sessionID,
337.          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
338.        });
339.      }
340.
341. });
342. var insert_solution_blob = (req, res, next, data_to_db) => {
343.   return new Promise((resolve, reject) => {
344.     try {
345.
346.       db_connection.query(
347.         'INSERT INTO Solution_File (solution_blob) VALUES (?);',
348.         [data_to_db],
349.         function (err, rows) {
350.           if (err) {
351.             console.log(err);
```

```
                req.flash('error', ' An error occured');
                res.locals.message = req.flash();
                return res.render('student_challenges/view_set_challenges', {
                  title: 'View Set Challenges',
                  menu_id: 'view_set_challenges',
                  role: req.user.role,
                  email: req.user.email,
                  session_id: req.sessionID,
                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                });
                reject(err);
              }
              console.log(rows);
              resolve(rows);
            }
          );

      } catch (error) {
        reject(error);
      }
    });
};
var insert_solution_student = (req, res, next, data_to_db) => {
  return new Promise((resolve, reject) => {
    try {

      db_connection.query(
        'INSERT INTO Solution_Student (solution_file_id, assigned_challenge_id, completion_date) VALUES ((SELECT solution_file_id FROM Solution_File WHERE solution_blob = ?), ?, NOW());',
        [data_to_db, parseInt(req.body.assigned_challenge_id)],
        function (err, rows) {
          if (err) {
            console.log(err);
            req.flash('error', ' An error occured');
            res.locals.message = req.flash();
            return res.render('student_challenges/view_set_challenges', {
              title: 'View Set Challenges',
              menu_id: 'view_set_challenges',
              role: req.user.role,
              email: req.user.email,
              session_id: req.sessionID,
              session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            });
            reject(err);
          }
          console.log(rows);
          resolve(rows);
        }
      );

    } catch (error) {
      reject(error);
    }
  });
};
```

```
406.
407. module.exports = router;
408.
```

This page takes care of all of the backend for the student's side of challenges. Firstly, the GET **/** router takes care of all challenges and data to be displayed on the intial page, similar to the GET **/** method in **~/routes/teacher_challenges/create_challenges.js**.

Looking at the complete challenges POST route, this works in the same way that the creation of challenges works for teachers, using **~/public/text_files/student_complete_challenge.txt** as the text file to be converted into the mediumblob for the database.

Also, the POST request for viewing completed challenges is once again located in the GET **/** route, because each challenge needs to be looped over and a route for each challenge created, doing the same thing, but passing in the specific challenge's data each time, and I wanted to avoid using a global variable.

I will now explain the teacher's side of viewing the students' completed challenge solutions / flagging students who have missed a challenge.

Viewing Assigning Challenges' Solutions as a Teacher

~/views/teacher_challenges/view_student_challenges.ejs

```
 1. <!DOCTYPE html>
 2. <html lang="en">
 3.
 4. <head>
 5.     <% include ../partials/head %>
 6.     <% include ./partials/script_toggle %>
 7. </head>
 8.
 9. <body>
10.     <div class="container-fluid">
11.
12.         <% include ../partials/menu %>
13.
14.         <% if ((typeof all_assigned_challenges === "undefined" || all_assigned_challenges == [] ||
all_assigned_challenges.length == 0) && (typeof all_not_assigned_challenges === "undefined" || all_not_assigned_challenges
== [] || all_not_assigned_challenges.length == 0)) { %>
15.             <h4>No Challenges have been Created yet</h4>
16.         <% } else { %>
17.
18.             <!-- ASSIGNED CHALLENGES -->
19.             <div class="row justify-content-center">
20.                 <div class="col-md-12">
21.                     <div class="card" style="margin-top: 0.5em;">
22.
23.                         <div class="card-header" style="background-color: #E6E6FA;">
```

```
                <div class="row">
                  <div class="col-md-4">
                    <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenges that have been Assigned</h5>
                  </div>
                  <div class="col-md-5"></div>
                  <div class="col-md-3">
                    <button class="btn btn-secondary switch_to_student" onclick="toggle_challenges_have_been_assigned()"
                      style="float: right;" id="display_challenges_have_been_toggle">Hide Challenges</button>
                  </div>
                </div>
              </div>

              <div class="card-body" id="challenges_have_been_assigned" style="background-color: #E6E6FA;">

                <% if (typeof all_assigned_challenges === "undefined" || all_assigned_challenges == [] || all_assigned_challenges.length == 0) { %>
                  <h5>No Challenges have been Assigned yet</h5>
                <% } else { %>

                  <table style="width: 100%;" id="challenges_have_been_assigned_table">

                    <tr>
                      <th>Challenge Title</th>
                      <th>Challenge Description</th>
                      <th></th>
                    </tr>

                    <% all_assigned_challenges.forEach((element, index)=> { %>
                      <tr>
                        <td>
                          <%= element[0].split('\n')[1].toString() %>
                        </td>
                        <td>
                          <%= element[0].split('\n')[3].toString() %>
                        </td>
                        <td>
                          <form
                            action="/view_student_challenges/more_information/<%= element[1].toString() %>" method="post">
                            <button type="submit" value="more_information_<%= element[1].toString() %>" class="btn btn-secondary create_challenges_edit_btn">More Information</button>
                          </form>
                        </td>
                      </tr>
                    <% }); %>

                  </table>

                <% } %>

              </div>

            </div>
```

```
            </div>
        </div>

        <!-- NOT ASSIGNED CHALLENGES -->
        <div class="row justify-content-center">
            <div class="col-md-12">
                <div class="card" style="margin-top: 0.5em;">

                    <div class="card-header" style="background-color: #ADD8E6;">
                        <div class="row">
                            <div class="col-md-4">
                                <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenges that haven't been Assigned</h5>
                            </div>
                            <div class="col-md-5"></div>
                            <div class="col-md-3">
                                <button class="btn btn-secondary switch_to_student" onclick="toggle_challenges_havent_been_assigned()" style="float: right;" id="display_challenges_havent_been_toggle">Show Challenges</button>
                            </div>
                        </div>
                    </div>

                    <div class="card-body" id="challenges_havent_been_assigned" style="background-color: #ADD8E6;">

                        <% if (typeof all_not_assigned_challenges === "undefined" || all_not_assigned_challenges == [] || all_not_assigned_challenges.length == 0) { %>
                            <h5>All Challenges have been Assigned</h5>
                        <% } else { %>

                            <table style="width: 100%;" id="challenges_havent_been_assigned_table">

                                <tr>
                                    <th>Challenge Title</th>
                                    <th>Challenge Description</th>
                                </tr>

                                <% all_not_assigned_challenges.forEach((element, index)=> { %>
                                    <tr>
                                        <td>
                                            <%= element[0].split('\n')[1].toString() %>
                                        </td>
                                        <td>
                                            <%= element[0].split('\n')[3].toString() %>
                                        </td>
                                    </tr>
                                <% }); %>

                            </table>

                        <% } %>

                    </div>

                </div>
```

```
            </div>
        </div>

    <% } %>

  </div>
</body>

<% include ../partials/script %>

</html>
```

This file shows 2 tables. The first is all the challenges that have been assigned to at least 1 student, while the other contains all the challenges that have never been assigned to any challenge. For the assigned challenges, the user can click on the more information button to view the students that have completed or have not completed the assigned challenge.

~/views/teacher_challenges/more_information.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
  <% include ../partials/head %>
  <% include ./partials/script_toggle %>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <!-- CHALLENGE INFO -->
    <div class="row">
      <div class="col-md-12">

        <div class="row justify-content-center">
          <div class="col-md-12">
            <div class="card" style="margin-top: 0.5em;">

              <div class="card-header" style="background-color: #DCDCDC;">
                <div class="row">
                  <div class="col-md-4">
                    <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenge Information</h5>
                  </div>
                  <div class="col-md-5"></div>
                  <div class="col-md-3">
                    <button class="btn btn-secondary switch_to_student" onclick="toggle_challenge_info()" style="float: right;" id="display_challenge_info">Hide Information</button>
                  </div>
                </div>
```

```
                    </div>

                    <div class="card-body" id="challenge_info" style="background-color: #DCDCDC;">

                      <div class="row">
                        <!-- challenge title -->
                        <div class="col-md-4">
                          <label for="challenge_title">CHALLENGE TITLE</label>
                          <textarea id="challenge_title" name="challenge_title" wrap="off" placeholder="Generic Challenge Title" readonly
                            style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_title %></textarea>
                        </div>
                        <!-- challenge description -->
                        <div class="col-md-8">
                          <label for="challenge_description">CHALLENGE DESCRIPTION</label>
                          <textarea id="challenge_description" name="challenge_description" wrap="off"
                            placeholder="Generic Challenge Description" readonly
                            style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
                        </div>
                      </div>

                    </div>

                  </div>
                </div>
              </div>

            </div>
          </div>

          <!-- STUDENTS COMPLETED -->
          <div class="row">
            <div class="col-md-12">

              <div class="row justify-content-center">
                <div class="col-md-12">
                  <div class="card" style="margin-top: 0.5em;">

                    <div class="card-header" style="background-color: #E6E6FA;">
                      <div class="row">
                        <div class="col-md-4">
                          <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Students that have Submitted</h5>
                        </div>
                        <div class="col-md-5"></div>
                        <div class="col-md-3">
                          <button class="btn btn-secondary switch_to_student" onclick="toggle_students_have_submitted()"
                            style="float: right;" id="display_students_have_submitted">Show Students</button>
                        </div>
                      </div>
                    </div>

                    <div class="card-body" id="students_have_submitted" style="background-color: #E6E6FA;">

```

```
                    <% if (typeof students_solved === "undefined" || students_solved == [] || students_solved.length == 0) {
%>
                      <h5>No Students have solved this Challenge yet</h5>
                    <% } else { %>

                      <table style="width: 100%;" id="students_have_submitted_table">

                        <tr>
                          <th>Student ID</th>
                          <th>Due Date</th>
                          <th>Completion Date</th>
                          <th></th>
                        </tr>

                        <% students_solved.forEach((element, index)=> { %>
                          <tr>
                            <td>
                              <%= element[0].toString() %>
                            </td>
                            <td>
                              <%= element[2].toString() %>
                            </td>
                            <td>
                              <%= element[3].toString() %>
                            </td>
                            <td>
                              <form action="/view_student_challenges/more_information/<%=
challenge_teacher_id.toString() %>/view_answer/<%= element[1].toString() %>" method="post">
                                <button type="submit" value="view_answer" class="btn btn-secondary
switch_to_student">See Solution</button>
                              </form>
                            </td>
                          </tr>
                        <% }); %>

                      </table>

                    <% } %>

                  </div>

                </div>
              </div>
            </div>



          </div>
        </div>

        <!-- STUDENTS NOT COMPLETED BUT BEFORE DEADLINE -->
        <div class="row">
          <div class="col-md-12">

            <div class="row justify-content-center">
```

```
            <div class="col-md-12">
              <div class="card" style="margin-top: 0.5em;">

                <div class="card-header" style="background-color: #ADD8E6;">
                  <div class="row">
                    <div class="col-md-4">
                      <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Students yet to Submit</h5>
                    </div>
                    <div class="col-md-5"></div>
                    <div class="col-md-3">
                      <button class="btn btn-secondary switch_to_student" onclick="toggle_students_yet_to()" style="float: right;" id="display_students_yet_to">Show Students</button>
                    </div>
                  </div>
                </div>

                <div class="card-body" id="students_yet_to" style="background-color: #ADD8E6;">
                  <% if (typeof students_not_solved_before_due=="undefined" || students_not_solved_before_due==[] || students_not_solved_before_due.length==0) { %>
                    <h5>No Students left to Submit</h5>
                  <% } else { %>

                    <table style="width: 100%;" id="students_yet_to_table">

                      <tr>
                        <th>Student ID</th>
                        <th>Due Date</th>
                      </tr>

                      <% students_not_solved_before_due.forEach((element, index)=> { %>
                        <tr>
                          <td>
                            <%= element[0].toString() %>
                          </td>
                          <td>
                            <%= element[2].toString() %>
                          </td>
                        </tr>
                      <% }); %>

                    </table>

                  <% } %>
                </div>

              </div>
            </div>
          </div>

        </div>
      </div>

      <!-- STUDENTS NOT COMPLETED AND AFTER DEADLINE -->
      <div class="row">
        <div class="col-md-12">
```

```
189.
190.            <div class="row justify-content-center">
191.              <div class="col-md-12">
192.                <div class="card" style="margin-top: 0.5em;">
193.
194.                  <div class="card-header" style="background-color: #FAA0A0;">
195.                    <div class="row">
196.                      <div class="col-md-5">
197.                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Students that missed the
deadline</h5>
198.                      </div>
199.                      <div class="col-md-4"></div>
200.                      <div class="col-md-3">
201.                        <button class="btn btn-secondary switch_to_student" onclick="toggle_students_missed()"
style="float: right;" id="display_students_missed">Show Students</button>
202.                      </div>
203.                    </div>
204.                  </div>
205.
206.                  <div class="card-body" id="students_missed" style="background-color: #FAA0A0;">
207.
208.                    <% if (typeof students_not_solved_after_due===="undefined" || students_not_solved_after_due==[] ||
students_not_solved_after_due.length==0) { %>
209.                      <h5>No Students have missed the deadline</h5>
210.                    <% } else { %>
211.
212.                      <table style="width: 100%;" id="students_missed_table">
213.
214.                        <tr>
215.                          <th>Student ID</th>
216.                          <th>Due Date</th>
217.                        </tr>
218.
219.                        <% students_not_solved_after_due.forEach((element, index)=> { %>
220.                          <tr>
221.                            <td>
222.                              <%= element[0].toString() %>
223.                            </td>
224.                            <td>
225.                              <%= element[2].toString() %>
226.                            </td>
227.                          </tr>
228.                        <% }); %>
229.
230.                      </table>
231.
232.                    <% } %>
233.
234.                  </div>
235.
236.                </div>
237.              </div>
238.            </div>
239.
240.          </div>
```

```
        </div>

    </div>
</body>

<% include ../partials/script %>

</html>

```

This file contains 3 tables for information about the students who were assigned the current challenge being viewed. First, there is a table for students who have submitted the challenge. Then there is a table for students who have not yet completed it, but it is before the due date. Finally, there is a table for students who have not yet completed the challenge, and it is after the due date, so they can no longer complete the challenge.

For each student who has submitted their solution, there is a button next to their information on the table which allows the teacher to view their solution.

~/views/teacher_challenges/view_answer.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <div class="row">
            <div class="col-md-12">

                <!-- challenge title -->
                <div class="row">
                    <div class="col-md-12">
                        <label for="challenge_title">CHALLENGE TITLE</label>
                        <textarea id="challenge_title" name="challenge_title" wrap="off"
                            placeholder="Generic Challenge Title" readonly
                            style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
                    </div>
                </div>

                <!-- challenge description -->
                <div class="row">
                    <div class="col-md-12">
                        <label for="challenge_description">CHALLENGE DESCRIPTION</label>
```

```
                <textarea id="challenge_description" name="challenge_description" wrap="off"
                  placeholder="Generic Challenge Description" readonly
                  style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
              </div>
            </div>

            <!-- challenge answer -->
            <div class="row">
              <div class="col-md-12">
                <label for="challenge_answer">CHALLENGE ANSWER</label>
                <textarea id="challenge_answer" name="challenge_answer" wrap="off"
                  placeholder="Generic Challenge Answer" readonly
                  style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_answer %></textarea>
              </div>
            </div>

            <button onclick="location.href='../../../';" class="btn btn-primary">Go Back</button>

          </div>
        </div>

    </div>
</body>

<% include ../partials/script %>

</html>

```

This file is basically just the teacher version of **~/views/student_challenges/view_completed_challenge.ejs**.

~/routes/teacher_challenges/view_student_challenges.js

```
var express = require('express');
var router = express.Router();

var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// for writing to files
var fs = require('fs');

// for array/list operations
var _ = require('lodash');

// for DATETIME conversions
var moment = require('moment');
moment.locale('en-gb');
moment.defaultFormat = 'L LT';
```

```
var utc_datetime = moment.utc().format();

// coming to view_student_challenges page
router.get('/', auth.check_authenticated, async function (req, res, next) {

  res.locals.message = req.flash();

  try {

    await get_challenges_assigned_not_assigned(req, res, next, true).then((rows) => {

      var all_assigned_challenges = [];

      rows.forEach(element => {
        all_assigned_challenges.push([element['challenge_blob'].toString(), element['challenge_teacher_id']]);
      });

      console.log('all_assigned_challenges');
      console.log(all_assigned_challenges);

      return {
        'all_assigned_challenges': all_assigned_challenges
      };

    }).then(async (useful_vars) => {

      await get_challenges_assigned_not_assigned(req, res, next, false).then((rows) => {

        var all_not_assigned_challenges = [];

        rows.forEach(element => {
          all_not_assigned_challenges.push([element['challenge_blob'].toString(), element['challenge_teacher_id']]);
        });

        console.log('all_not_assigned_challenges');
        console.log(all_not_assigned_challenges);

        useful_vars['all_not_assigned_challenges'] = all_not_assigned_challenges;

      });

      return useful_vars;

    }).then((useful_vars) => {

      useful_vars['all_assigned_challenges'].forEach(async (element, index) => {

        var route_more_info = '/more_information/' + element[1];

        router.post(route_more_info, async function (req, res, next) {

          res.locals.message = req.flash();

          try {

```

```
                    await get_students_assigned_challenge_teacher_and_solved(req, res, next, parseInt(element[1])).then((rows) => {

                      var students_solved = [];

                      rows.forEach(element_new => {
                        students_solved.push([element_new['CONCAT(Student.student_name, Student.student_number)'].toString(), element_new['student_id'], moment(element_new['due_date']).utc(utc_datetime).local().format(), moment(element_new['completion_date']).utc(utc_datetime).local().format()]);
                      });

                      console.log('students_solved');
                      console.log(students_solved);

                      return {
                        'students_solved': students_solved
                      };

                    }).then(async (useful_vars) => {

                      await get_students_assigned_challenge_teacher_and_not_solved(req, res, next, parseInt(element[1])).then((rows) => {

                        var students_not_solved_before_due = []; // can still solve
                        var students_not_solved_after_due = []; // has not completed assignment and cannot complete it now

                        rows.forEach(element_new => {

                          // past deadline
                          if (new Date() >= new Date(moment(element_new['due_date']).utc(utc_datetime).local().format())) {
                            students_not_solved_before_due.push([element_new['CONCAT(Student.student_name, Student.student_number)'].toString(), element_new['student_id'], moment(element_new['due_date']).utc(utc_datetime).local().format()]);
                          } else {
                            students_not_solved_after_due.push([element_new['CONCAT(Student.student_name, Student.student_number)'].toString(), element_new['student_id'], moment(element_new['due_date']).utc(utc_datetime).local().format()]);
                          };

                        });

                        console.log('students_not_solved_before_due');
                        console.log(students_not_solved_before_due);
                        console.log('students_not_solved_after_due');
                        console.log(students_not_solved_after_due);

                        useful_vars['students_not_solved_before_due'] = students_not_solved_before_due;
                        useful_vars['students_not_solved_after_due'] = students_not_solved_after_due;

                      });

                      return useful_vars;

                    }).then(async (useful_vars) => {
```

```
120.
121.                useful_vars['students_solved'].forEach(async (element_latest, index_latest) => {
122.
123.                  var route_view_solution = route_more_info + '/view_answer/' + element_latest[1];
124.
125.                  router.post(route_view_solution, async function (req, res, next) {
126.
127.                    res.locals.message = req.flash();
128.
129.                    await get_answer_blob(req, res, next, parseInt(element[1]), parseInt(element_latest[1])).then((rows)
=> {
130.
131.                      rows.forEach(first_row => {
132.                        answer_blob = first_row['solution_blob'];
133.                      });
134.
135.                      return res.render('teacher_challenges/view_answer', {
136.                        title: 'View Completed Challenge',
137.                        role: req.user.role,
138.                        email: req.user.email,
139.                        session_id: req.sessionID,
140.                        session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
141.                        challenge_title: element[0].split('\n')[1].toString(),
142.                        challenge_description: element[0].split('\n')[3].toString(),
143.                        challenge_answer: answer_blob.toString()
144.                      });
145.
146.                    });
147.
148.                  });
149.
150.                });
151.
152.                return res.render('teacher_challenges/more_information', {
153.                  title: 'More Information',
154.                  role: req.user.role,
155.                  email: req.user.email,
156.                  session_id: req.sessionID,
157.                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
158.                  challenge_title: element[0].split('\n')[1].toString(),
159.                  challenge_description: element[0].split('\n')[3].toString(),
160.                  challenge_teacher_id: element[1].toString(),
161.                  students_solved: useful_vars['students_solved'],
162.                  students_not_solved_before_due: useful_vars['students_not_solved_before_due'],
163.                  students_not_solved_after_due: useful_vars['students_not_solved_after_due']
164.                });
165.
166.              });
167.
168.            } catch (any_error) {
169.
170.              res.render('teacher_challenges/more_information', {
171.                title: 'More Information',
172.                role: req.user.role,
173.                email: req.user.email,
```

```
                    session_id: req.sessionID,
                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                    challenge_title: element[0].split('\n')[1].toString(),
                    challenge_description: element[0].split('\n')[3].toString(),
                    challenge_teacher_id: element[1].toString()
                  });

                }

              });

            });

            return res.render('teacher_challenges/view_student_challenges', {
              title: 'View Student Challenges',
              menu_id: 'view_student_challenges',
              role: req.user.role,
              email: req.user.email,
              all_assigned_challenges: useful_vars['all_assigned_challenges'],
              all_not_assigned_challenges: useful_vars['all_not_assigned_challenges'],
              session_id: req.sessionID,
              session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            });

          });

        } catch (any_error) {
          console.log(any_error);
          res.render('teacher_challenges/view_student_challenges', {
            title: 'View Student Challenges',
            menu_id: 'view_student_challenges',
            role: req.user.role,
            email: req.user.email,
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
          });
        }

      });

var get_challenges_assigned_not_assigned = (req, res, next, assigned) => {
  return new Promise((resolve, reject) => {
    try {

      if (assigned) {
        // all challenge_teacher_id's that have been assigned to some student
        var query_to_run = 'SELECT * FROM Challenge_Teacher, Challenge_File WHERE Challenge_Teacher.challenge_file_id = Challenge_File.challenge_file_id AND Challenge_Teacher.challenge_teacher_id IN(SELECT challenge_teacher_id FROM Assigned_Challenges) AND Challenge_Teacher.teacher_id = ?;';
      } else {
        // all challenge_teacher_id's that have not been assigned to some student
        var query_to_run = 'SELECT * FROM Challenge_Teacher, Challenge_File WHERE Challenge_Teacher.challenge_file_id = Challenge_File.challenge_file_id AND Challenge_Teacher.challenge_teacher_id NOT IN (SELECT challenge_teacher_id FROM Assigned_Challenges) AND Challenge_Teacher.teacher_id = ?;';
      };
```

```
225.
226.        db_connection.query(
227.          query_to_run,
228.          [req.user.id],
229.          function (err, rows) {
230.
231.            if (err) {
232.              console.log(err);
233.              req.flash('error', ' An error occured');
234.              res.locals.message = req.flash();
235.              return res.render('teacher_challenges/view_student_challenges', {
236.                title: 'View Student Challenges',
237.                menu_id: 'view_student_challenges',
238.                role: req.user.role,
239.                email: req.user.email,
240.                session_id: req.sessionID,
241.                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
242.              });
243.              reject(err);
244.            };
245.
246.            resolve(rows);
247.          }
248.        );
249.
250.      } catch (error) {
251.        reject(error);
252.      }
253.    });
254. };
255.
256. var get_students_assigned_challenge_teacher_and_solved = (req, res, next, challenge_teacher_id) => {
257.    return new Promise((resolve, reject) => {
258.      try {
259.
260.        // get all students who have been assigned challenge_teacher_id and solved
261.        db_connection.query(
262.          'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id,
Assigned_Challenges.due_date, Solution_Student.completion_date FROM Solution_Student, Solution_File, Student,
Assigned_Challenges WHERE Solution_Student.solution_file_id = Solution_File.solution_file_id AND
Assigned_Challenges.assigned_challenge_id = Solution_Student.assigned_challenge_id AND Assigned_Challenges.student_id =
Student.student_id AND Assigned_Challenges.challenge_teacher_id = ? AND Assigned_Challenges.assigned_challenge_id IN
(SELECT assigned_challenge_id FROM Solution_Student);',
263.          [challenge_teacher_id],
264.          function (err, rows) {
265.
266.            if (err) {
267.              console.log(err);
268.              req.flash('error', ' An error occured');
269.              res.locals.message = req.flash();
270.              return res.render('teacher_challenges/view_student_challenges', {
271.                title: 'View Student Challenges',
272.                menu_id: 'view_student_challenges',
273.                role: req.user.role,
274.                email: req.user.email,
```

```
                    session_id: req.sessionID,
                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                  });
                  reject(err);
                };

                resolve(rows);
              }
            );

        } catch (error) {
            reject(error);
        }
    });
};
var get_students_assigned_challenge_teacher_and_not_solved = (req, res, next, challenge_teacher_id) => {
    return new Promise((resolve, reject) => {
        try {

            // get all students who have been assigned challenge_teacher_id and solved
            db_connection.query(
                'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id,
Assigned_Challenges.due_date FROM Student, Assigned_Challenges WHERE Assigned_Challenges.student_id =
Student.student_id AND Assigned_Challenges.challenge_teacher_id = ? AND Assigned_Challenges.assigned_challenge_id NOT
IN (SELECT assigned_challenge_id FROM Solution_Student);',
                [challenge_teacher_id],
                function (err, rows) {

                  if (err) {
                    console.log(err);
                    req.flash('error', ' An error occured');
                    res.locals.message = req.flash();
                    return res.render('teacher_challenges/view_student_challenges', {
                      title: 'View Student Challenges',
                      menu_id: 'view_student_challenges',
                      role: req.user.role,
                      email: req.user.email,
                      session_id: req.sessionID,
                      session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                    });
                    reject(err);
                  };

                  resolve(rows);
                }
            );

        } catch (error) {
            reject(error);
        }
    });
};

var get_answer_blob = (req, res, next, challenge_teacher_id, student_id) => {
    return new Promise((resolve, reject) => {
```

```
    try {

      // get all students who have been assigned challenge_teacher_id and solved
      db_connection.query(
        'SELECT * FROM Solution_File, Solution_Student WHERE Solution_File.solution_file_id = Solution_Student.solution_file_id AND Solution_Student.assigned_challenge_id = (SELECT assigned_challenge_id FROM Assigned_Challenges WHERE challenge_teacher_id = ? AND student_id = ?);',
        [challenge_teacher_id, student_id],
        function (err, rows) {

          if (err) {
            console.log(err);
            req.flash('error', ' An error occured');
            res.locals.message = req.flash();
            return res.render('teacher_challenges/view_student_challenges', {
              title: 'View Student Challenges',
              menu_id: 'view_student_challenges',
              role: req.user.role,
              email: req.user.email,
              session_id: req.sessionID,
              session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            });
            reject(err);
          };

          resolve(rows);
        }
      );

    } catch (error) {
      reject(error);
    }
  });
};

module.exports = router;

```

This file handles all of the backend for teachers viewing information about the challenges that have been assigned to students. I have already explained pretty much all of the code in this file by explaining it in the other router files, since this contains mostly similar information, apart from different SQL queries.

Also, this file uses the **moment** library to convert the dates from the database (which are stored in UTC time) into local time, making it easier for the teacher to actually understand when the due dates were set.

This now completes the explanation for the Implementation of requirement 10.

Having completed all of the explanations for the client requirements and how they have been implemented in my solution, I will now go on to the Testing section, where I will go through my testing plan and link the video where I tested my solution.

Chapter 6 Testing

Overview

In this section, I will demonstrate that I have met my client requirements, as well as other requirements that are necessary for the website to function as it is intended to do. Below, I will provide a table with the tests that I attempt to carry out that will be accompanied by a testing video, in which I will test each component precisely, and note down whether the test passes/fails. If the test passes, the actual outcome(s) column will be left blank, but if the test fails, I will note what happened in the actual outcome(s) tab, and I will detail my process of fixing the error afterwards. The evidence column in the table will contain the timestamp in the video where I conduct that specific test.

The testing video can be found at the following link: https://youtu.be/ZfbiDA_2PHk.

Here is a quick reminder of the colour scheme I will be using for this table:

- ⇒ This colour will be used for all regular text and text that was inputted during the testing video.
- ⇒ This colour will be used to highlight failed tests.
- ⇒ This colour will be used to highlight any tests that passed but had additional comments, as well as any comments that were added after the testing video was completed.

Test Plan/Table

Test No.	Client Requirement	Section/Module	Test Description	Expected Outcome(s)	Actual Outcome(s)	Pass/Fail	Evidence
1	1	No Account > Homepage	Start Database and access (localhost) URL from desktop browser	Homepage loads, displaying page with main visualisation and navbar	-	Pass	1:13
2	1.a, 10.c, 10.c.i, 10.c.ii	No Account > Homepage > Navbar	Hover over red dot to clarify that no one is currently signed in	A small message appears where the mouse is that says “red: no account”	-	Pass	1:53

3	1.a, 6	No Account > Homepage > Navbar	Open the Instruction Set page and return to the Home page	Instruction Set page is displayed, and then Home page is returned to	-	Pass	2:04
4	1.a, 6.a, 9, 9.a	No Account > Homepage > Navbar	Open the “How to use this resource” page and return to the Home page	“How to use this resource” page is displayed, and then Home page is returned to	-	Pass	2:20
5	1.a, 10	No Account > Homepage > Navbar	Open the Login page and return to the Home page	Login page is displayed with the Login form, and then Home page is returned to	-	Pass	2:32
6	1.a, 10	No Account > Homepage > Navbar	Open the Sign Up page and return to the Home page	Sign Up page is displayed with the Sign Up form, and then Home page is returned to	-	Pass	2:44
7	3	No Account > Homepage > Main visualisation > controlbox	Click the “load into RAM” button in the controlbox	RAM memory locations change numbers/values	Also incorrect error appearing in the 2 output boxes	Pass	2:51
8	3	No Account > Homepage > Main visualisation > controlbox	Click the “reset RAM” button in the controlbox	RAM memory locations change numbers/values to all become 0000	-	Pass	3:26
9	3	No Account > Homepage > Main visualisation > controlbox	Click the “step program” button in the controlbox	RAM memory locations change numbers/values, some boxes in the processor and RAM are highlighted blue and	-	Pass	3:38

				change back to grey, an input alert box appears			
10	3	No Account > Homepage > Main visualisation > Assembly code execution	Input a value (5) and the LMC should handle it as the program specifies it to	Input value in controlbox changes to the entered value (5), FDE Cycle Output box has some text in it, and so does Verbose Output box	-	Pass	4:07
11	3, 7, 7.a	No Account > Homepage > Main visualisation > Assembly code execution	Slide the slider (above the controlbox) up to almost full and click the “step program” button again	Some boxes are highlighted blue and return to grey again, but this happens a lot faster now, the FDE Cycle Output box’s text is replaced, and Verbose Output box’s text is appended to	-	Pass	4:39
12	3	No Account > Homepage > Main visualisation > controlbox	Click the “reset RAM” button in the controlbox	RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	-	Pass	5:10
13	3	No Account > Homepage > Main visualisation > controlbox	Click the “clear editor” button in the controlbox	All text disappears from the Assembly Code Editor area	-	Pass	5:27
14	9, 9.a	No Account > Homepage > Main	Click the “how to use this resource” button in the	“How to use this resource” page is displayed in a new	-	Pass	5:43

		visualisation > controlbox	controlbox, and then close the page	tab, and then page is closed			
15	6	No Account > Homepage > Main visualisation > controlbox	Click the "view instruction set" button in the controlbox, and then close the page	Instruction Set page is displayed in a new tab, and then page is closed	-	Pass	5:57
16	2, 2.a, 2.b, 2.c, 2.d, 2.e, 2.f, 8	No Account > Homepage > Main visualisation > controlbox	Click the pre-created programs dropdown list and click on each option 1-by-1	Assembly Code Editor area is replaced with different text for each option in the dropdown	-	Pass	6:13
17	5, 5.a	No Account > Homepage > Main visualisation > controlbox	Click the "upload program" button in the controlbox and select a pre-ready file (max_of_2_nums.txt, then prime_num.py, then back to max_of_2_nums.txt)	Allows the user to select a file from their computer (must only be a text file, other file types are disabled), and after clicking the text file, its contents are put into the Assembly Code Editor area. If prime_num.py is opened, the assembly code area should be blank.	The text file was inputted correctly and displayed, however the python file was also allowed to be inputted and was shown, which should not be allowed	Fail	10:03
18	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	No Account > Homepage > Main visualisation > Assembly code execution	Click the "run program" button in the controlbox	Boxes are highlighted blue and changed back to grey, an input prompt will then be displayed	-	Pass	11:04

19	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	No Account > Homepage > Main visualisation > Assembly code execution	Enter two values (4 and 7) and the LMC will handle it as the program tells it to	The FDE Cycle Output will be overwritten every-so-often, while the Verbose Output will continue to have more text, which can be scrolled through, the input box in the controlbox should display the entered value (4, then 7), and the output box should slowly show 7	-	Pass	11:25
20	2, 10.c.i	No Account > Homepage > Main visualisation > Assembly code execution	Edit the Assembly Code Editor to find the minimum value	The Assembly Code Editor will update the edited text	-	Pass	11:47
21	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	No Account > Homepage > Main visualisation > Assembly code execution	Click the “run program” button in the controlbox	Boxes are highlighted blue and changed back to grey, an input prompt will then be displayed	-	Pass	12:15
22	3, 7, 7.a	No Account > Homepage > Main visualisation > Assembly code execution	Enter the values (4 and 7) and the LMC will handle it as the program tells it to	The FDE Cycle Output will be overwritten every-so-often, while the Verbose Output will continue to have more text, which can be scrolled through, the	-	Pass	12:25

				input box in the controlbox should display the entered values (4, then 7), and the output box should show 4			
23	3, 4, 4.a	No Account > Homepage > Main visualisation > Assembly code execution	Click the “reset RAM” button in the controlbox, then change any variable to a random name, and click the “run program” button	RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor, and the code will ask for an input, which can be any number, and then there will be an alert message which says that there is an error in the code, so all processes will halt, and so, the 2 output boxes stop displaying additional text, and boxes stop changing colours and moving the visualisations	Program still ran completely and halted without displaying an error, using C as an input itself	Fail	12:38
24	3, 4, 4.a	No Account > Homepage > Main visualisation > Assembly code execution	Click the “run program” button, and then the “reset RAM” button	An error message should be displayed saying that the user must first reset the RAM and then run the program, and after resetting the RAM, RAM memory locations change numbers/values to all become 0000, and so do	-	Pass	14:22

				all the other registers in the processor			
25	3, 4, 4.a	No Account > Homepage > Main visualisation > Assembly code execution	Change one of the opcodes to a random sequence of letters, then click “load into RAM”, after which reset the RAM and change the opcode back	An error message should appear in the 2 output boxes, specifying which line the error is at, and this error message should replace the entire text in the 2 boxes, then the RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	Random error additionally displayed saying error at line 14	Pass	15:06
26	1.a	No Account > Instruction Set page > Navbar	Go to the Instruction Set page, and same as test 2	Instruction Set page loads, and same as test 2	-	Pass	16:23
27	1.a	No Account > Instruction Set page > Navbar	Try to open all pages from the navbar after going back to the Instruction Set page each time (similar to tests 3-6)	Each page opens and loads respectively, similar to test 3-6, and each time, the page returns to the same Instruction Set page	-	Pass	16:43
28	1.a	No Account > How to use this resource page > Navbar	Go to the How to use this resource page, and same as test 2	How to use this resource page loads, and same as test 2	-	Pass	18:07
29	1.a	No Account > How to use this	Try to open all pages from the navbar after going back to the	Each page opens and loads respectively, similar to test 3-6, and each time, the	-	Pass	17:33

		resource page > Navbar	How to use this resource page each time (similar to tests 3-6)	page returns to the same How to use this resource page			
30	1.a	No Account > Sign Up page > Navbar	Go to the Sign Up page, and same as test 2	Sign Up page loads, and same as test 2	-	Pass	18:14
31	1.a	No Account > Sign Up page > Navbar	Try to open all pages from the navbar after going back to the Sign Up page each time (similar to tests 3-6)	Each page opens and loads respectively, similar to test 3-6, and each time, the page returns to the same Sign Up page	-	Pass	18:22
32	1.a	No Account > Login page > Navbar	Go to the Login page, and same as test 2	Login page loads, and same as test 2	-	Pass	18:36
33	1.a	No Account > Login page > Navbar	Try to open all pages from the navbar after going back to the Login page each time (similar to tests 3-6), and at the end, go back to the Sign Up page	Each page opens and loads respectively, similar to test 3-6, and each time, the page returns to the same Login page, after which, the Sign Up page finally loads after going back to it	-	Pass	18:43
34	1.a, 10	No Account > Sign Up page > Student/Teacher toggle	Click on the Student/Teacher toggle to switch forms to the teacher sign up form	The teacher sign up form loads, which can be recognised by the placeholder text in the email address box	-	Pass	19:18
35	10	No Account > Sign Up page > Teacher form	Enter the following test data (asdf, ASD, asdf123!, asdf123!) and click sign up button	Error should be displayed, with the error message	-	Pass	19:36

				saying @ is missing in the email address field			
36	10	No Account > Sign Up page > Teacher form	Same as test 35 but change email address to asdf@	Error should be displayed, with the error message saying something is required after @ in the email address field	-	Pass	20:23
37	10	No Account > Sign Up page > Teacher form	Same as test 35 but change email address to asdf@gmail.com	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	20:35
38	10	No Account > Sign Up page > Teacher form	Same as test 35 but change email address to asdf@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	20:51
39	10	No Account > Sign Up page > Teacher form	Same as test 35 but change email address to asdf.1234@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	21:27
40	10	No Account > Sign Up page > Teacher form	Same as test 35 but change email address to asdf.ghjk@dubaicollege.org and change class code to asdf	Error should be displayed, with the error message saying that the class code does not match the required format	-	Pass	22:02
41	10	No Account > Sign Up page > Teacher form	Same as test 40 but change class code to asd	Error should be displayed, with the error message saying that the class code does not match the required format	-	Pass	22:29

42	10	No Account > Sign Up page > Teacher form	Same as test 40 but change class code to ASD, and password and confirm password to hi123!	Error should be displayed, with the error message saying that the password length must be at least 8 characters long	-	Pass	22:41
43	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password and confirm password to thisisapassword	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	23:13
44	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password and confirm password to 12345678	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	23:44
45	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password and confirm password to !@#$%^&*	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	24:04
46	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password and confirm password to this1234	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	24:21

47	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password and confirm password to this!@#$	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	24:34
48	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password and confirm password to 1234!@#$	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	24:54
49	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password to asdf123! and confirm password to ghjk234@	Error should be displayed, with the error message saying that the confirm password does not match the password	-	Pass	25:06
50	10	No Account > Sign Up page > Teacher form	Same as test 42 but change password to asdf123! and confirm password to asdf123!	The user should be redirected to the login page, which indicates that the account has successfully been created	-	Pass	25:33
51	10	No Account > Sign Up page > Student form	Go to the Sign Up page from the Navbar, enter the following test data (qwer, Year 10, ASD, qwer123!, qwer123!) and click sign up button	Error should be displayed, with the error message saying @ is missing in the email address field	-	Pass	26:28

52	10	No Account > Sign Up page > Student form	Same as test 51 but change email address to qwer@	Error should be displayed, with the error message saying something is required after @ in the email address field	-	Pass	27:12
53	10	No Account > Sign Up page > Student form	Same as test 51 but change email address to qwer@gmail.com	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	27:23
54	10	No Account > Sign Up page > Student form	Same as test 51 but change email address to qwer@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	Had to reload the page for the teacher class code to appear	Pass	27:33
55	10	No Account > Sign Up page > Student form	Same as test 51 but change email address to qwer.1234@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	28:18
56	10	No Account > Sign Up page > Student form	Same as test 51 but change email address to qwer8526 @dubaicollege.org and change year group to unselected	Error should be displayed, with the error message saying that the Year Group is a required field	-	Pass	28:42
57	10	No Account > Sign Up page > Student form	Same as test 56 but change year group to Year 10 and leave the class field unselected	Error should be displayed, with the error message saying that the Class field is a required field	-	Pass	29:05
58	10	No Account > Sign Up page > Student form	Same as test 57 but change class field to ASD, and password and confirm password to hi123!	Error should be displayed, with the error message saying that the password	-	Pass	29:25

				length must be at least 8 characters long			
59	10	No Account > Sign Up page > Student form	Same as test 58 but change password and confirm password to thisisapassword	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	29:44
60	10	No Account > Sign Up page > Student form	Same as test 58 but change password and confirm password to 12345678	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	30:09
61	10	No Account > Sign Up page > Student form	Same as test 58 but change password and confirm password to !@#$%^&*	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	30:25
62	10	No Account > Sign Up page > Student form	Same as test 58 but change password and confirm password to this1234	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	30:44
63	10	No Account > Sign Up page > Student form	Same as test 58 but change password and confirm password to this!@#$	Error should be displayed, with the error message saying that the password	-	Pass	31:00

				must contain at least 1 alphabet, 1 integer, and 1 special character			
64	10	No Account > Sign Up page > Student form	Same as test 58 but change password and confirm password to 1234!@#$	Error should be displayed, with the error message saying that the password must contain at least 1 alphabet, 1 integer, and 1 special character	-	Pass	31:17
65	10	No Account > Sign Up page > Student form	Same as test 58 but change password to asdf123! and confirm password to ghjk234@	Error should be displayed, with the error message saying that the confirm password does not match the password	-	Pass	31:36
66	10	No Account > Sign Up page > Student form	Same as test 58 but change password to qwer123! and confirm password to qwer123!	The user should be redirected to the login page, which indicates that the account has successfully been created	-	Pass	31:58
67	1.a, 10	No Account > Login page > Student/Teacher toggle	Click on the Student/Teacher toggle to switch forms to the teacher login form	The teacher login form loads, which can be recognised by the placeholder text in the email address box	-	Pass	32:25
68	10	No Account > Login page > Teacher form	Enter the following test data (asdf, asdf123!) and click login button	Error should be displayed, with the error message saying @ is missing in the email address field	-	Pass	32:37

69	10	No Account > Login page > Teacher form	Same as test 68 but change email address to asdf@	Error should be displayed, with the error message saying something is required after @ in the email address field	-	Pass	32:52
70	10	No Account > Login page > Teacher form	Same as test 68 but change email address to asdf@gmail.com	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	33:02
71	10	No Account > Login page > Teacher form	Same as test 68 but change email address to asdf@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	33:10
72	10	No Account > Login page > Teacher form	Same as test 68 but change email address to asdf.1234@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	33:23
73	10	No Account > Login page > Teacher form	Same as test 68 but change email address to asdf.ghjk@dubaicollege.org and change password to hi123!	Error should be displayed, with the error message saying that the password length must be at least 8 characters long	-	Pass	33:40
74	10	No Account > Login page > Teacher form	Same as test 73 but change password to ghjk234@	Error should be displayed, with the error message saying that the password is incorrect	-	Pass	33:59
75	10	No Account > Login page > Teacher form	Same as test 73 but change password to asdf123!, and then show the console values to	The user should be redirected to the Home page and the dot in the	-	Pass	34:18

			confirm that the password was hashed and remember me was not clicked	top-left corner of the Navbar should now be yellow, indicating that a teacher is logged in, also, the console should show the teacher's data from the database, showing the hashed password, as well as the login max age, which should be ~7200000, indicating that session's cookies should not be remembered for more than 2 hours			
76	10	No Account > Login page > Teacher form	Click the logout button, and then do the same as test 75, but check the remember me box, and then show the console values to confirm that the password was hashed and remember me was clicked	The user should be redirected to the Home page and the dot in the top-left corner of the Navbar should now be yellow, indicating that a teacher is logged in, also, the console should show the teacher's data from the database, showing the hashed password, as well as the login max age, which should be ~2592000000, indicating that session's cookies	-	Pass	35:25

				should be remembered for around 30 days			
77	10	No Account > Login page > Student form	Click the logout button, enter the following test data (qwer, qwer123!) and click login button	Error should be displayed, with the error message saying @ is missing in the email address field	-	Pass	36:12
78	10	No Account > Login page > Student form	Same as test 77 but change email address to qwer@	Error should be displayed, with the error message saying something is required after @ in the email address field	-	Pass	36:32
79	10	No Account > Login page > Student form	Same as test 77 but change email address to qwer@gmail.com	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	36:38
80	10	No Account > Login page > Student form	Same as test 77 but change email address to qwer@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	36:47
81	10	No Account > Login page > Student form	Same as test 77 but change email address to qwer.1234@dubaicollege.org	Error should be displayed, with the error message saying that the email is in an invalid format	-	Pass	37:00
82	10	No Account > Login page > Student form	Same as test 68 but change email address to qwer8526@dubaicollege.org and change password to hi123!	Error should be displayed, with the error message saying that the password length must be at least 8 characters long	-	Pass	37:12

83	10	No Account > Login page > Student form	Same as test 82 but change password to ghjk234@	Error should be displayed, with the error message saying that the password is incorrect	-	Pass	37:25
84	10	No Account > Login page > Student form	Same as test 82 but change password to qwer123!, and then show the console values to confirm that the password was hashed and remember me was not clicked	The user should be redirected to the Home page and the dot in the top-left corner of the Navbar should now be green, indicating that a student is logged in, also, the console should show the student's data from the database, showing the hashed password, as well as the login max age, which should be ~7200000, indicating that session's cookies should not be remembered for more than 2 hours	-	Pass	37:40
85	10	No Account > Login page > Student form	Click the logout button, and then do the same as test 84, but check the remember me box, and then show the console values to confirm that the password was hashed and remember me was clicked	The user should be redirected to the Home page and the dot in the top-left corner of the Navbar should now be green, indicating that a student is logged in, also, the console should show	-	Pass	38:13

				the student's data from the database, showing the hashed password, as well as the login max age, which should be ~2592000000, indicating that session's cookies should be remembered for around 30 days			
86	1.a, 10	No Account > Login page > Sign up link	Click the logout button and click the sign up link that is not in the Navbar	The Sign Up page should be displayed	-	Pass	38:41
87	10	No Account > Forgot password > link to page	Create new account using raghav8526@dubaicollege.org for the email and asdf123; for the password, and login. Click the logout button and then click the forgot password link. Enter the details, doing the same tests as above for the inputs.	The "forgot password" page should be displayed, showing a text field to input an email address and a submit button, which should be submitted, testing for different errors as previously done	Was able to create the account and login, but unable to load and go to the forgot password page, so also unable to test the inputs	Fail	38:57
88	10	No Account > Forgot password > reset password	Open raghav8526@dubaicollege.org inbox and click the link from the email, after which the password in the new page/form should be changed to hjkl123!	There should be an email in the inbox titled "Assembly Code Visualiser Reset Password", and clicking the link opens a new page with the fields new password and confirm password. Entering in the passwords and clicking	No email received from web application, so unable to test changing the password	Fail	40:48

				confirm should display a success message and redirect the user to the login page.			
89	10	No Account > Forgot password > confirm change	Login with raghav8526@dubaicollege.org and hjkl123! And then log back out, then go to the sign up page	The main page indicating a student account should be displayed, then the login page, and then the sign up page	Password was not changed, so unable to login with the new password	Fail	41:44
90	1.a, 10	No Account > Sign Up page > Login link	Click the login link that is not in the Navbar	The Login page should be displayed	-	Pass	42:01
91	1.a, 10, 10.b	Teacher Account > Homepage > Navbar	Login to the teacher account and hover over yellow dot to clarify that a teacher is currently signed in	A small message appears where the mouse is that says “red: no account yellow: teacher account green: student account”	-	Pass	42:11
92	1.a, 6	Teacher Account > Homepage > Navbar	Open the Instruction Set page and return to the Home page	Instruction Set page is displayed, and then Home page is returned to	-	Pass	42:30
93	1.a, 6.a, 9, 9.a	Teacher Account > Homepage > Navbar	Open the “How to use this resource” page and return to the Home page	“How to use this resource” page is displayed, and then Home page is returned to	-	Pass	42:49
94	1.a, 10	Teacher Account > Homepage > Navbar	Open the Create Challenges page and return to the Home page	Create Challenges page is displayed with the Create New Challenge button, text saying no challenges created yet, and an assign	-	Pass	42:58

				form, and then Home page is returned to			
95	1.a, 10	Teacher Account > Homepage > Navbar	Open the View Student Challenges page and return to the Home page	View Student Challenges page is displayed with text saying no challenges have been assigned yet, and then Home page is returned to	-	Pass	43:10
96	3	Teacher Account > Homepage > Main visualisation > controlbox	Click the “load into RAM” button in the controlbox	RAM memory locations change numbers/values	Unnecessary error (previously explained) Random error additionally displayed saying error at line 14	Pass	43:20
97	3	Teacher Account > Homepage > Main visualisation > controlbox	Click the “reset RAM” button in the controlbox	RAM memory locations change numbers/values to all become 0000	-	Pass	43:42
98	3	Teacher Account > Homepage > Main visualisation > controlbox	Click the “step program” button in the controlbox	RAM memory locations change numbers/values, some boxes in the processor and RAM are highlighted blue and change back to grey, an input alert box appears	-	Pass	43:51
99	3	Teacher Account > Homepage > Main	Input a value (5) and the LMC should handle it as the program specifies it to	Input value in controlbox changes to the entered value (5), FDE Cycle Output	-	Pass	44:09

		visualisation > Assembly code execution		box has some text in it, and so does Verbose Output box			
100	3, 7, 7.a	Teacher Account > Homepage > Main visualisation > Assembly code execution	Slide the slider (above the controlbox) up to almost full and click the “step program” button again	Some boxes are highlighted blue and return to grey again, but this happens a lot faster now, the FDE Cycle Output box’s text is replaced, and Verbose Output box’s text is appended to	-	Pass	44:21
101	3	Teacher Account > Homepage > Main visualisation > controlbox	Click the “reset RAM” button in the controlbox	RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	-	Pass	44:53
102	3	Teacher Account > Homepage > Main visualisation > controlbox	Click the “clear editor” button in the controlbox	All text disappears from the Assembly Code Editor area	-	Pass	45:04
103	9, 9.a	Teacher Account > Homepage > Main visualisation > controlbox	Click the “how to use this resource” button in the controlbox, and then close the page	“How to use this resource” page is displayed in a new tab, and then page is closed	-	Pass	45:13
104	6	Teacher Account > Homepage > Main	Click the “view instruction set” button in the controlbox, and then close the page	Instruction Set page is displayed in a new tab, and then page is closed	-	Pass	45:18

		visualisation > controlbox					
105	2, 2.a, 2.b, 2.c, 2.d, 2.e, 2.f, 8	Teacher Account > Homepage > Main visualisation > controlbox	Click the pre-created programs dropdown list and click on each option 1-by-1	Assembly Code Editor aera is replaced with different text for each option in the dropdown	-	Pass	45:29
106	5, 5.a	Teacher Account > Homepage > Main visualisation > controlbox	Click the "upload program" button in the controlbox and select a pre-ready file (max_of_2_nums.txt, then prime_num.py, then back to max_of_2_nums.txt)	Allows the user to select a file from their computer (must only be a text file, other file types are disabled), and after clicking the text file, its contents are put into the Assembly Code Editor area. If prime_num.py is opened, the assembly code area should be blank.	Same as before The text file was inputted correctly and displayed, however the python file was also allowed to be inputted and was shown, which should not be allowed.	Fail	46:16
107	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	Teacher Account > Homepage > Main visualisation > Assembly code execution	Click the "run program" button in the controlbox	Boxes are highlighted blue and changed back to grey, an input prompt will then be displayed	-	Pass	46:59
108	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	Teacher Account > Homepage > Main visualisation >	Enter two values (4 and 7) and the LMC will handle it as the program tells it to	The FDE Cycle Output will be overwritten every-so-often, while the Verbose Output will continue to have more text, which can	-	Pass	47:08

		Assembly code execution		be scrolled through, the input box in the controlbox should display the entered value (4, then 7), and the output box should slowly show 7			
109	2	Teacher Account > Homepage > Main visualisation > Assembly code execution	Edit the Assembly Code Editor to find the minimum value	The Assembly Code Editor will update the edited text	-	Pass	47:21
110	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	Teacher Account > Homepage > Main visualisation > Assembly code execution	Click the “run program” button in the controlbox	Boxes are highlighted blue and changed back to grey, an input prompt will then be displayed	-	Pass	47:33
111	3, 7, 7.a	Teacher Account > Homepage > Main visualisation > Assembly code execution	Enter the values (4 and 7) and the LMC will handle it as the program tells it to	The FDE Cycle Output will be overwritten every-so-often, while the Verbose Output will continue to have more text, which can be scrolled through, the input box in the controlbox should display the entered values (4, then 7), and the output box should show 4	-	Pass	47:39

112	3, 4, 4.a	Teacher Account > Homepage > Main visualisation > Assembly code execution	Click the “reset RAM” button in the controlbox, then change any variable to a random name, and click the “run program” button	RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor, and the code will ask for an input, which can be any number, and then there will be an alert message which says that there is an error in the code, so all processes will halt, and so, the 2 output boxes stop displaying additional text, and boxes stop changing colours and moving the visualisations	Same as above Program still ran completely and halted without displaying an error, using the erroneous variable name as an input itself	Fail	47:51
113	3, 4, 4.a	Teacher Account > Homepage > Main visualisation > Assembly code execution	Click the “run program” button, and then the “reset RAM” button	An error message should be displayed saying that the user must first reset the RAM and then run the program, and after resetting the RAM, RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	Error message does not appear, and program is re-run, but resetting RAM still works	Fail	48:33

114	3, 4, 4.a	Teacher Account > Homepage > Main visualisation > Assembly code execution	Change one of the opcodes to a random sequence of letters, then click "load into RAM", after which reset the RAM and change the opcode back	An error message should appear in the 2 output boxes, specifying which line the error is at, and this error message should replace the entire text in the 2 boxes, then the RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	Error is displayed, but wrong line, and same as above for extra error Random error additionally displayed saying error at line 14	Pass	49:33
115	1.a, 10, 10.a	Student Account > Homepage > Navbar	Logout and login to the student account and hover over green dot to clarify that a student is currently signed in	A small message appears where the mouse is that says "red: no account green: student account"	-	Pass	50:25
116	1.a, 6	Student Account > Homepage > Navbar	Open the Instruction Set page and return to the Home page	Instruction Set page is displayed, and then Home page is returned to	-	Pass	50:52
117	1.a, 6.a, 9, 9.a	Student Account > Homepage > Navbar	Open the "How to use this resource" page and return to the Home page	"How to use this resource" page is displayed, and then Home page is returned to	-	Pass	51:03
118	1.a, 10	Student Account > Homepage > Navbar	Open the View Set Challenges page and return to the Home page	View Set Challenges page is displayed with text saying no challenges set, and then Home page is returned to	-	Pass	51:09
119	3	Student Account > Homepage >	Click the "load into RAM" button in the controlbox	RAM memory locations change numbers/values	Same as above	Pass	51:22

		Main visualisation > controlbox			Random error additionally displayed saying error at line 14		
120	3	Student Account > Homepage > Main visualisation > controlbox	Click the “reset RAM” button in the controlbox	RAM memory locations change numbers/values to all become 0000	-	Pass	51:39
121	3	Student Account > Homepage > Main visualisation > controlbox	Click the “step program” button in the controlbox	RAM memory locations change numbers/values, some boxes in the processor and RAM are highlighted blue and change back to grey, an input alert box appears	-	Pass	51:47
122	3	Student Account > Homepage > Main visualisation > Assembly code execution	Input a value (5) and the LMC should handle it as the program specifies it to	Input value in controlbox changes to the entered value (5), FDE Cycle Output box has some text in it, and so does Verbose Output box	-	Pass	52:05
123	3, 7, 7.a	Student Account > Homepage > Main visualisation > Assembly code execution	Slide the slider (above the controlbox) up to almost full and click the “step program” button again	Some boxes are highlighted blue and return to grey again, but this happens a lot faster now, the FDE Cycle Output box’s text is replaced, and Verbose Output box’s text is appended to	-	Pass	52:17

124	3	Student Account > Homepage > Main visualisation > controlbox	Click the “reset RAM” button in the controlbox	RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	-	Pass	52:52
125	3	Student Account > Homepage > Main visualisation > controlbox	Click the “clear editor” button in the controlbox	All text disappears from the Assembly Code Editor area	-	Pass	53:00
126	9, 9.a	Student Account > Homepage > Main visualisation > controlbox	Click the “how to use this resource” button in the controlbox, and then close the page	“How to use this resource” page is displayed in a new tab, and then page is closed	-	Pass	53:09
127	6	Student Account > Homepage > Main visualisation > controlbox	Click the “view instruction set” button in the controlbox, and then close the page	Instruction Set page is displayed in a new tab, and then page is closed	-	Pass	53:13
128	2, 2.a, 2.b, 2.c, 2.d, 2.e, 2.f, 8	Student Account > Homepage > Main visualisation > controlbox	Click the pre-created programs dropdown list and click on each option 1-by-1	Assembly Code Editor aera is replaced with different text for each option in the dropdown	-	Pass	53:25
129	5, 5.a	Student Account > Homepage > Main	Click the “upload program” button in the controlbox and select a pre-ready file (max_of_2_nums.txt,	Allows the user to select a file from their computer (must only be a text file, other file types are	Same as above The text file was inputted correctly and	Fail	54:36

		visualisation > controlbox	then prime_num.py, then back to max_of_2_nums.txt)	disabled), and after clicking the text file, its contents are put into the Assembly Code Editor area. If prime_num.py is opened, the assembly code area should be blank.	displayed, however the python file was also allowed to be inputted and was shown, which should not be allowed.		
130	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	Student Account > Homepage > Main visualisation > Assembly code execution	Click the “run program” button in the controlbox	Boxes are highlighted blue and changed back to grey, an input prompt will then be displayed	-	Pass	55:08
131	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	Student Account > Homepage > Main visualisation > Assembly code execution	Enter two values (4 and 7) and the LMC will handle it as the program tells it to	The FDE Cycle Output will be overwritten every-so-often, while the Verbose Output will continue to have more text, which can be scrolled through, the input box in the controlbox should display the entered value (4, then 7), and the output box should slowly show 7	-	Pass	55:16
132	2	Student Account > Homepage > Main visualisation >	Edit the Assembly Code Editor to find the minimum value	The Assembly Code Editor will update the edited text	-	Pass	55:27

		Assembly code execution					
133	3, 3.a, 3.b, 3.c, 3.d, 3.e.i, 3.e.ii, 3.e.iii, 3.e.iv, 3.e.v	Student Account > Homepage > Main visualisation > Assembly code execution	Click the "run program" button in the controlbox	Boxes are highlighted blue and changed back to grey, an input prompt will then be displayed	-	Pass	56:02
134	3, 7, 7.a	Student Account > Homepage > Main visualisation > Assembly code execution	Enter the values (4 and 7) and the LMC will handle it as the program tells it to	The FDE Cycle Output will be overwritten every-so-often, while the Verbose Output will continue to have more text, which can be scrolled through, the input box in the controlbox should display the entered values (4, then 7), and the output box should show 4	Still shows 7 instead of 4 – perhaps an issue while editing the assembly code	Fail	56:04
135	3, 4, 4.a	Student Account > Homepage > Main visualisation > Assembly code execution	Click the "reset RAM" button in the controlbox, then change any variable to a random name, and click the "run program" button	RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor, and the code will ask for an input, which can be any number, and then there will be an alert message which says that there is an error in the code, so all processes will	-	Pass	56:52

				halt, and so, the 2 output boxes stop displaying additional text, and boxes stop changing colours and moving the visualisations			
136	3, 4, 4.a	Student Account > Homepage > Main visualisation > Assembly code execution	Click the “run program” button, and then the “reset RAM” button	An error message should be displayed saying that the user must first reset the RAM and then run the program, and after resetting the RAM, RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	-	Pass	58:04
137	3, 4, 4.a	Student Account > Homepage > Main visualisation > Assembly code execution	Change one of the opcodes to a random sequence of letters, then click “load into RAM”, after which reset the RAM and change the opcode back	An error message should appear in the 2 output boxes, specifying which line the error is at, and this error message should replace the entire text in the 2 boxes, then the RAM memory locations change numbers/values to all become 0000, and so do all the other registers in the processor	Same as above Random error additionally displayed saying error at line 14	Pass	58:16

138	1.a, 6	Student Account > Instruction Set page > Navbar	Go to the Instruction Set page, and same as test 115	Instruction Set page loads, and same as test 115	-	Pass	59:28
139	1.a, 6	Student Account > Instruction Set page > Navbar	Try to open all pages from the navbar after going back to the Instruction Set page each time (similar to tests 116-118)	Each page opens and loads respectively, similar to test 116-118, and each time, the page returns to the same Instruction Set page	-	Pass	58:55
140	1.a, 6.a, 9, 9.a	Student Account > How to use this resource page > Navbar	Go to the How to use this resource page, and same as test 115	How to use this resource page loads, and same as test 115	-	Pass	59:33
141	1.a, 6.a, 9, 9.a	Student Account > How to use this resource page > Navbar	Try to open all pages from the navbar after going back to the How to use this resource page each time (similar to tests 116-118)	Each page opens and loads respectively, similar to test 116-118, and each time, the page returns to the same How to use this resource page	-	Pass	59:39
142	1.a, 10.a.i, 10.a.iii	Student Account > View Set Challenges page > Navbar	Go to the View Set Challenges page, and same as test 115	View Set Challenges page loads, and same as test 115	-	Pass	1:00:10
143	1.a, 10.a.i, 10.a.iii	Student Account > View Set Challenges page > Navbar	Try to open all pages from the navbar after going back to the View Set Challenges page each time (similar to tests 116-118)	Each page opens and loads respectively, similar to test 116-118, and each time, the page returns to the same View Set Challenges page	-	Pass	1:00:14

144	1.a, 6	Teacher Account > Instruction Set page > Navbar	Logout and login as the teacher, then go to the Instruction Set page, and same as test 91	Instruction Set page loads, and same as test 91	-	Pass	1:01:28
145	1.a, 6	Teacher Account > Instruction Set page > Navbar	Try to open all pages from the navbar after going back to the Instruction Set page each time (similar to tests 92-95)	Each page opens and loads respectively, similar to test 92-95, and each time, the page returns to the same Instruction Set page	-	Pass	1:01:16
146	1.a, 6.a, 9, 9.a	Teacher Account > How to use this resource page > Navbar	Go to the How to use this resource page, and same as test 91	How to use this resource page loads, and same as test 91	-	Pass	1:02:17
147	1.a, 6.a, 9, 9.a	Teacher Account > How to use this resource page > Navbar	Try to open all pages from the navbar after going back to the How to use this resource page each time (similar to tests 92-95)	Each page opens and loads respectively, similar to test 92-95, and each time, the page returns to the same How to use this resource page	-	Pass	1:01:39
148	1.a, 10.b, 10.b.i, 10.b.ii	Teacher Account > Create Challenges page > Navbar	Go to the Create Challenges page, and same as test 91	Create Challenges page loads, and same as test 91	ToolTip is not displayed on this page, but yellow dot is still shown	Fail	1:02:51
149	1.a, 10.b, 10.b.i, 10.b.ii	Teacher Account > Create Challenges page > Navbar	Try to open all pages from the navbar after going back to the Create Challenges page each time (similar to tests 92-95)	Each page opens and loads respectively, similar to test 92-95, and each time, the page returns to the same Create Challenges page	-	Pass	1:02:23

150	1.a, 10.b, 10.b.iii	Teacher Account > View Student Challenges page > Navbar	Go to the View Student Challenges page, and same as test 91	View Student Challenges page loads, and same as test 91	-	Pass	1:03:52
151	1.a, 10.b, 10.b.iii	Teacher Account > View Student Challenges page > Navbar	Try to open all pages from the navbar after going back to the View Student Challenges page each time (similar to tests 92-95)	Each page opens and loads respectively, similar to test 92-95, and each time, the page returns to the same View Student Challenges page	-	Pass	1:03:27
152	10.b.i, 10.b.ii	Teacher Account > Create Challenges page > Assign Challenge	Go to the Create Challenges page and try to click on the “select your challenge” dropdown, then select the student in the class we want to assign a challenge to (qwer8526), and select a date and time at least 24 hours from the current time, then click the assign button	There should be an error displayed that says a challenge needs to be assigned, but obviously this isn’t possible because a challenge hasn’t been created yet	-	Pass	1:03:58
153	10.b.ii	Teacher Account > Create Challenges page > Create Challenge	Click on the “create new challenge” button	A new page should appear with two titles and two text areas, for Challenge Title and Challenge Description	-	Pass	1:04:36
154	10.b.ii	Teacher Account > Create Challenges page > Create Challenge	Click on the “create new challenge” button	A new page should appear with two titles and two text areas, for Challenge Title and Challenge Description	-	Pass	1:04:36

155	10.b.ii	Teacher Account > Create Challenges page > Create Challenge	Enter the following test data (, This is a test description) and click create button	An error message should appear saying that the Challenge Title is a required field	-	Pass	1:05:07
156	10.b.ii	Teacher Account > Create Challenges page > Create Challenge	Enter the following test data (This is a test title,) and click create button	An error message should appear saying Challenge Description is a required field	-	Pass	1:05:22
157	10.b.ii	Teacher Account > Create Challenges page > Create Challenge	Same as test 155 but change the title from being empty to any characters, totalling 101 or more characters	An error message should appear saying the Challenge Title length must be a maximum of 100 characters	-	Pass	1:05:35
158	10.b.ii	Teacher Account > Create Challenges page > Create Challenge	Same as test 155 but change the title from being empty to any characters, totalling exactly 100 characters	Should be redirected to the original Create Challenges page, and now, instead of text saying that no challenges exist, there should be a table with 1 row, apart from the header, that shows the challenge title, description, and 2 buttons – edit and delete	-	Pass	1:06:13
159	10.b.ii	Teacher Account > Create Challenges page	Click the delete button for the challenge just created	The page should reload and should once again	-	Pass	1:06:46

		> Delete Challenge		show that no challenges have been created yet			
160	10.b.ii	Teacher Account > Create Challenges page > Edit Challenge	Create any 2 new challenges and come back to the original Create Challenge page, then click the edit button for the first challenge	The page should be similar to the “create new challenge” page, but the text boxes should be filled with the text that was just entered for that challenge	-	Pass	1:06:59
161	10.b.ii	Teacher Account > Create Challenges page > Edit Challenge	Go back to the original Create Challenge page, then click the edit button for the second challenge	The page should be similar to the “create new challenge” page, but the text boxes should be filled with the text that was just entered for that challenge	-	Pass	1:07:37
162	10.b.ii	Teacher Account > Create Challenges page > Edit Challenge	Replace the 2 text boxes with the following test data (, This is a test description) and click update button	An error message should appear saying that the Challenge Title is a required field	-	Pass	1:07:57
163	10.b.ii	Teacher Account > Create Challenges page > Edit Challenge	Replace the 2 text boxes with the following test data (This is a test title,) and click update button	An error message should appear saying Challenge Description is a required field	-	Pass	1:08:13
164	10.b.ii	Teacher Account > Create Challenges page > Edit Challenge	Same as test 162 but change the title from being empty to any characters, totalling 101 or more characters	An error message should appear saying the Challenge Title length must be a maximum of 100 characters	Page does not load, error 404 displayed	Fail	1:08:25
165	10.b.ii	Teacher Account > Create	Same as test 162 but change the title from being empty to any	Should be redirected to the original Create		Pass	1:09:18

		Challenges page > Edit Challenge	characters, totalling exactly 100 characters	Challenges page, and the old challenge title should have been changed			
166	10.b.ii	Teacher Account > Create Challenges page > Edit Challenge	Click the edit button for the same challenge, and change the description, then click the update button	Should be redirected to the original Create Challenges page, and the old challenge description should have been changed	Challenge Title was reverted back to the original	Pass	1:10:02
167	10.b.ii	Teacher Account > Create Challenges page > Assign Challenge	Enter the following values (, qwer8526, select a date and time more than 24 hours from now) and click the assign button	Should display an error message saying no challenge has been selected	-	Pass	1:10:47
168	10.b.i, 10.b.ii	Teacher Account > Create Challenges page > Assign Challenge	Same as test 167 but select any of the challenges, and don't select any students	Should display an error message saying no students have been assigned	-	Pass	1:11:20
169	10.b.i	Teacher Account > Create Challenges page > Assign Challenge	Click the select all and deselect all button	qwer8526 should show up in the 10ASD field and then be removed from it	-	Pass	1:11:32
170	10.b.ii	Teacher Account > Create Challenges page > Assign Challenge	Same as test 167 but select qwer8526 and change the date to any time under 24 hours from the current time	Should display an error message saying the due date must be after 24 hours	-	Pass	1:11:51

171	10.b.ii	Teacher Account > Create Challenges page > Assign Challenge	Same as test 167 but select change the date to any time after 24 hours from the current time	The page should reload and say that the challenge has been assigned	-	Pass	1:12:16
172	10.b.ii	Teacher Account > View Student Challenges page > challenge-specific page	Click on the view set challenges button in the navbar and then click on the challenge that was just assigned	Should display a page that shows a list of completed (none) and not completed (qwer8526)	Students yet to submit and student that missed the deadline are titled the wrong way around	Pass	1:12:35
173	10.a.i	Student Account > View Set Challenges page	Log out and login as the student, then open the view set challenges page	Should show a table with 1 row under the "To Complete" section – the assigned challenge that was just set, with its title, description, and due date	-	Pass	1:13:41
174	10.a.i, 10.a.ii	Student Account > View Set Challenges page > submit challenge	Click on the "complete challenge" button	Should show a new page with a text box displaying the challenge title and description, as well as an input area for text and a submit button	-	Pass	1:14:16
175	10.a.ii	Student Account > View Set Challenges page > submit challenge	Enter in the test data (contained in challenge_answer.txt) and click the submit button	The view set challenges should be shown again, but the assigned challenge now comes under the "Completed" section	-	Pass	1:14:27

176	10.a.iii	Student Account > View Set Challenges page > submit challenge	Click the “view answer” button next to the recently completed challenge	A new page should be displayed with the challenge title, challenge description, and the student’s answer, but nothing can be edited	-	Pass	1:15:09
177	10.b.i, 10.b.iii	Teacher Account > View Student Challenges page > challenge-specific page	Log out and login into the teacher account, click on the view set challenges button in the navbar and then click on the challenge that was assigned	Should display a page that shows a list of completed (qwer8526) and not completed (none)	-	Pass	1:15:20
178	10.b.iii	Teacher Account > View Student Challenges page > challenge-specific page	Next to the student’s name, click the “view submission” button	Should display a new page that shows a text area with the challenge description and title, as well as the contents of challenge_answer.txt	-	Pass	1:15:56

Now that I have completed testing the solution, I will conduct some user tests with 2 teachers and 2 students, getting them to give a rating for each task, as well as any comments that they have. I will then look back through this testing table, as well as the user tests to go through and refine small things that didn’t work or had some unexpected behaviour. For larger things that need to be fixed, I will look at them in the Evaluation section.

User Testing

Overview

The purpose of this testing is to establish – once a user has created an account and logged in to the platform – how easy and simple it is for them to complete certain tasks. Is the application user-friendly and intuitive and without bugs? Does everything make logical sense and look appealing enough for the user? To judge, evaluate, and test these sorts of questions, I will use a rating system as outlined below.

Rating System

1 – No support needed, intuitive task, no bugs
2 – Some support needed, slightly intuitive
3 – Some support needed, slightly intuitive OR bugs
4 – Lots of support needed, not intuitive with bugs

Teacher User Tests

Teacher 1

Task No	Task to do:	Expected outcome	Rating	Comments from teacher
1	Create and assign a challenge to at least 2 students for a particular date	The teacher identifies the need to click on CREATE CHALLENGES, clicks on the SHOW CHALLENGES button, clicks on the CREATE NEW CHALLENGE and creates a new challenge, returning to the old page. They then identify the need to click on the SHOW ASSIGN CHALLENGES button, after which they are easily able to select the challenge they just created (from its title), select a student using the drop down menus and/or the SELECT ALL button(s), set a date for the assignment, and then click assign.	1	When I break a validation rule e.g. choosing a date less than today and the page refreshes I would like all my data to have stayed as is i.e. as it currently is built I have to reenter everything. I like the select all and deselect all from a class feature. I also like the auto complete when I am typing in the names of students.
2	View a student submitted challenge	The teacher identifies the need to click on VIEW STUDENT CHALLENGES, clicks on the single challenge they want to see, then identifies and clicks on the student they want to see the solution for.	2	Very intuitive, I like how I can see who has submitted, who has yet to submit and who has missed the deadline. In future maybe from here I could click on these students who have missed a deadline are automatically informed that this has been logged. I love the fact that from the dashboard showing who has completed what challenges I can click on their submission and see it. It would be cool if I could then run it from there e.g. click a button, it loads into the application and I can run it.
3	Edit an existing challenge	The teacher identifies the need to click on CREATE CHALLENGES, followed by SHOW CHALLENGES, followed by the EDIT button next to the specific challenge which they want to change the description or title for. They are then do	2	It wasn't intuitive that I had to go to 'Create Challenges' to edit a challenge, initially I went to 'View Student Challenges'. I suggest calling the 'Create

		the necessary changes and click the UPDATE button to finish editing the challenge.		Challenges' button 'Challenges' instead, from there it was intuitive in terms of how to then edit a challenge.
4	Deleting an existing challenge	The teacher identifies the need to click on CREATE CHALLENGES, followed by SHOW CHALLENGES, and then click the DELETE button next to the challenge which they want to delete. The page should be updated and if SHOW CHALLENGES clicked again, the challenge should no longer appear there, or in the dropdown menu for the Assigning Challenges part.	1	As above BUT I would want a prompt to say "Are you sure you want to delete" first.

In addition to the comments above, the following was a general comment given on the entire project by the same teacher:
I would like an Account page where I can see my class code, maybe change it. Also maybe change my email address for when I forget my password.

I will now conduct the second user test for a teacher.

Teacher 2

Task No	Task to do:	Expected outcome	Rating	Comments from teacher
1	Create and assign a challenge to at least 2 students for a particular date	The teacher identifies the need to click on CREATE CHALLENGES, clicks on the SHOW CHALLENGES button, clicks on the CREATE NEW CHALLENGE and creates a new challenge, returning to the old page. They then identify the need to click on the SHOW ASSIGN CHALLENGES button, after which they are easily able to select the challenge they just created (from its title), select a student using the drop down menus and/or the SELECT ALL button(s), set a date for the assignment, and then click assign.	1	The idea to show/hide different sections with buttons is really good, but perhaps something like a dropdown arrow might be better? I also thought that clicking the create challenges button in the top part (navigation bar) would directly take me to a page that creates a new challenge, rather than viewing all created challenges and assigning them, so perhaps this could be

				labelled better? Everything else was perfect.
2	View a student submitted challenge	The teacher identifies the need to click on VIEW STUDENT CHALLENGES, clicks on the single challenge they want to see, then identifies and clicks on the student they want to see the solution for.	3	The yet to submit students feature did not work, so this needs to be fixed, although this is not really a vital part of the solution I believe. Perhaps I could also have the ability to sort by student rather than challenge? Or maybe even class? Because I wasn't able to see the classes, and I think that more often than not I will be setting assignments to entire classes rather than individual students. Also, perhaps there could be a button that directly loads the students' submitted challenges into the Little Man Computer visualisation, so I can view the visualisation directly, rather than having to copy the code, and then paste it into the editor after navigating back to the home page.
3	Edit an existing challenge	The teacher identifies the need to click on CREATE CHALLENGES, followed by SHOW CHALLENGES, followed by the EDIT button next to the specific challenge which they want to change the description or title for. They are then do the necessary changes and click the UPDATE button to finish editing the challenge.	2	As previously mentioned, perhaps the page to do this from could be more aptly named. I tried multiple edits, and most of them worked, but occasionally, the edit did not save.
4	Deleting an existing challenge	The teacher identifies the need to click on CREATE CHALLENGES, followed by SHOW CHALLENGES, and then click the DELETE button next to the challenge which they want to delete. The page should be updated and if SHOW	1	No comments.

		CHALLENGES clicked again, the challenge should no longer appear there, or in the dropdown menu for the Assigning Challenges part.		

Summary

Both teachers said similar things about the tasks they had to complete, and one of the main things that we could see, even from the testing video that I did, was that sometimes edits did not work, and the "students yet to complete" section did not work either. Therefore, once the student user testing section is completed, I will refine these, although I have a feeling that these simply have to do with the SQL statements that I wrote back in the Documented Design section.

Additionally, it seems as though the label for the page to create new challenges, edit, and delete them, as well as assign new challenges was quite unintuitive, so that could also be changed.

I will now let 2 students conduct their user tests.

Student User Tests

Student 1

Task No	Task to do:	Expected outcome	Rating	Comments from student
1	Access and view an assigned challenge that has already been completed	Student finds the page where they can SEE their completed challenge	1	No comments.
2	Access and view an assigned challenge that needs to be completed	Student finds the page where they can SEE the challenge that they need to complete	1	No comments.

3	Access the page where a challenge can be completed and complete that challenge	Student finds the page where they can SEE the challenge they need to complete, they access the page to enter their solution, and are able to paste in their solution and submit it.	2	Give me the option to confirm my submission. Show the visualization on that page.
4	Complete a challenge. The challenge being: Adding two numbers together and outputting the result.	The student is able to write a solution for adding two numbers together and outputting the result. The visualisation of the solution runs smoothly and without errors. During the creation of the solution the student is able to access the instruction set to identify the necessary instructions required.	2	It would be great to have the challenge that I am completing in a separate popup window so I can see it whilst I am completing the challenge. Give me the option to confirm my submission. Maybe force the instruction manual on new users so I can understand how to use the visualization.
5	The student is able to upload an external program and run it	The student can intuitively identify where to click to upload a file. A file is uploaded and the file (which is a valid file) can be run and the visualisation occurs with no errors.	2	Ensure that user input is only text file (I was able to upload a python program).
6	Resolve an error in the assembly code	A piece of code with an error in is given to the student and they are able to understand what the error is and how to resolve it. The error will be a variable error. The error will be indicated in the FDE and verbose output fields. The student realises the error, checks the FDE and verbose fields and is able to refine the assembly code and fix the error.	3	When program was not supposed to work, it still ran but gave the wrong output. There should be an error message to show that the program broke.

Student 2

Task No	Task to do:	Expected outcome	Rating	Comments from student
1	Access and view an assigned challenge that has already been completed	Student finds the page where they can SEE their completed challenge	1	No comments.
2	Access and view an assigned challenge that needs to be completed	Student finds the page where they can SEE the challenge that they need to complete	2	Perhaps change the button to hide/display the challenges into an arrowhead that points right when hidden and down when shown.
3	Access the page where a challenge can be completed and complete that challenge	Student finds the page where they can SEE the challenge they need to complete, they access the page to enter their solution, and are able to paste in their solution and submit it.	2	Maybe allow editing of the answers in the future.
4	Complete a challenge. The challenge being: Adding two numbers together and outputting the result.	The student is able to write a solution for adding two numbers together and outputting the result. The visualisation of the solution runs smoothly and without errors. During the creation of the solution the student is able to access the instruction set to identify the necessary instructions required.	2	Show the visualization on that page. Maybe allow editing of the answers in the future. Make the user look at the “how to use this resource” page when it is the first time they are using the resource/completing an assignment.
5	The student is able to upload an external program and run it	The student can intuitively identify where to click to upload a file. A file is uploaded and the file (which is a valid file) can be run and the visualisation occurs with no errors.	3	I should only be able to upload a .txt file
6	Resolve an error in the assembly code	A piece of code with an error in is given to the student and they are able to understand what the error is and how to resolve it.	3	Some error messages were shown to highlight the error, but all the messages said they were on the line above where they actually were.

		The error will be a variable error. The error will be indicated in the FDE and verbose output fields. The student realises the error, checks the FDE and verbose fields and is able to refine the assembly code and fix the error.		There were some random errors also displayed. It was quite inconsistent when the program would run if there was an error, or if it would stop. Sometimes, it worked as it's supposed to function, but other times, it completely broke down.

Summary

Both students, like the teachers, said similar things about the tasks they had to complete. One of the main problems that directly relates to a client requirement (Client Requirement 4) is the fact that the debugging of Assembly Code is quite inconsistent and hard. We saw a similar thing during the testing video: although each account uses the exact same code for the Little Man Computer visualisation on the home page, even while using the exact same inputs and processes, we received almost 3 completely different outcomes when it came to debugging the code – finding errors on a line, halting the program if an error is found, identifying the kind of error (only for specific cases), and not being able to run the program again until the problem is fixed.

Similar to what the teachers said, it would be useful to be able to view the main visualisation while completing a challenge. Also, editing answers, and confirmation messages seems to be quite important for the student users, which does make sense, because they could accidentally click a button, and they may realise that their solution is wrong before the deadline, so should be able to edit their response. The idea to force a new user to view the "how to use this resource" page is also a really good idea, as that would also reduce the number of potential errors.

The other things that were flagged during the student user testing are not as important, as they have to do with style of the website (the arrows for hiding/showing), so I am not too worried about those. The important things to look at are the aforementioned items.

I will reference these teacher and student user tests again in the Evaluation section, while contemplating whether each requirement was met or not, but right now, I will go through the failed tests from the Testing Table and try to understand and refine the errors.

Failed Tests and Refinements

Test 17, 106, and 129

Test Description: Click the "upload program" button in the controlbox and select a pre-ready file (max_of_2_nums.txt, then prime_num.py, then back to max_of_2_nums.txt).
Outcome: Fail.

Problems:
Files that contain text but do not end in .txt were allowed to be loaded into the Assembly code text editor area.

Solution:
What needs to be done to fix this is actually quite simple. All that needs to be added is an if statement as follows:

```
if ($('#upload').hasExtension(['.txt'])) {
    // allow upload
}
```

So, the original code for uploading a file:

```
// triggered when a file is chosen for upload program
document.getElementById('upload_program_actual').addEventListener('change', () => {
    let files = document.getElementById('upload_program_actual').files;

    if (files.length == 0) return;

    const file = files[0];

    let reader = new FileReader();

    reader.onload = (element) => {
        const file = element.target.result;

        // regular expression to identify carriage
        // returns and line breaks
        const lines = file.split(/\r\n|\n/);
        document.getElementById('code_area').value = lines.join('\n');
    };

    reader.onerror = (element) => alert(element.target.error.name);
    reader.readAsText(file);

});

```

Becomes the following:

```
// triggered when a file is chosen for upload program
document.getElementById('upload_program_actual').addEventListener('change', () => {
    let files = document.getElementById('upload_program_actual').files;

    if (files.length == 0) return;

```

```
 7.   if ($(files).hasExtension(['.txt'])) {
 8.      const file = files[0];
 9.
10.      let reader = new FileReader();
11.
12.      reader.onload = (element) => {
13.         const file = element.target.result;
14.
15.         // regular expression to identify carriage
16.         // returns and line breaks
17.         const lines = file.split(/\r\n|\n/);
18.         document.getElementById('code_area').value = lines.join('\n');
19.      };
20.
21.      reader.onerror = (element) => alert(element.target.error.name);
22.      reader.readAsText(file);
23.   }
24.
25. });
```

Test 23 and 112

Test Description: Click the “reset RAM” button in the controlbox, then change any variable to a random name, and click the “run program” button.
Outcome: Fail.

Problems:
The Assembly code program still ran completely and stopped without displaying an error, using the erroneous variable name as an input itself.

Refinements:
What can be done to solve this is to check if the next words after the instruction set opcode match the required input type, as this was already determined when initialising each instruction object. So, if the first argument is supposed to be an integer, then the first word after the instruction set operand is an int, and the same done for all the other arguments/operands for each opcode.

In addition, the code would be altered to actually use the **stop** attribute in the Little Man Computer class to check if there are any errors. Hence, if errors exist, then the **HLT** instruction is immediately called, and the error displayed in the 2 output boxes.

Test 25, 96, 114, 119, and 137

Test Description: Change one of the opcodes to a random sequence of letters (25, 114, and 137), then click “load into RAM”, after which reset the RAM and change the opcode back (all).
Outcome: Pass.

Problems:

A random error additionally displayed saying error at line 14 (all), and all of the other errors were displaying the wrong line (25, 114, and 137).

Refinements:
Looking at the incorrect line number for each error, all of them were referencing the line above the actual error, and this means that I must have messed up with my indexing in the for loop, so I simply need to "+ 1" in the code where the error is being displayed.

The random error displaying at line 14 could be due to the fact that I mis-selected the current "active" pre-created challenge. Though the Assembly code that is automatically loaded into the text editor area is actually the "count down" code, I had selected the "add 2 numbers" as the currently active pre-created challenge, and I believe that this was causing an issue with the editor area, as it was adding the latter's code beneath the other, but not showing it, and thus causing the error as the function to read the data and check for errors was encountering an issue. So, all that needs to be done is to change the automatically loaded Assembly code to the code that adds 2 numbers.

The only concern I have with this is why the errors were not consistent. Since it was implemented in the exact same way, in fact in the exact same file, and was referenced the same way, there should be no difference in the tests. I may have to look further into this.

Test 54

Test Description: Go to the Sign Up page and enter the following test data (qwer@dubaicollege.org, Year 10, ASD, qwer123!, qwer123!).
Outcome: Pass.

Problems:
The class code (ASD) did not show up, and the dropdown was empty. The page had to be reloaded, and then everything worked as it should have.

Refinements:
This is probably due to the fact that the SQL query for the dropdown menu was not done asynchronously, meaning that the page loaded before retrieving the data from the database.

This could be solved in 2 different ways. Either 1) the SQL query is implemented using Asnyc/Await, as done with other queries, or 2) a JavaScript function could be called on the page each time it loads in order to reload the page, adding a "#" at the end of the URL while waiting for the response to be received from the database.

Test 87, 88, and 89

Test Description: Create a new account using raghav8526@dubaicollege.org for the email and asdf123; for the password, and login. Click the logout button and then click the forgot password link. Enter the details, and conduct tests to test the input features i.e. required fields and correct email format. Open raghav8526@dubaicollege.org inbox and click the link from the email, after which the password in the new page/form should be changed to "hjkl123!". Login with raghav8526@dubaicollege.org and hjkl123! And then log back out, then go to the sign-up page
Outcome: Fail.

Problems:
Though the account was created, the forgot password link simply did not work, and all tests to do with the forgot password therefore were not conducted.

Refinements:
Looking over my code, I realised that the forgot password link was never actually referenced in the HTML:

```
1. <a href="#" class="btn btn-link">Forgot Your Password?</a>
```

Furthermore, I think I never actually ended up implementing the forgot password functionality using the passport library. However, upon consulting students and teachers about this, it appears as though this is not completely necessary, and so, I will not go further into this until the Evaluation section of this writeup.

Test 24, 113, and 136

Test Description: Click the "run program" button, and then the "reset RAM" button after a program has already been run. An error should appear saying that the RAM needs to be reset before running the program again.
Outcome: Pass (25 and 136). Fail (113).

Problems:
The error message did not appear, and the program was immediately re-run, but resetting RAM still worked.

Refinements:
I could perhaps have a new Boolean variable **has_run** that only turns **true** after the entire Assembly code program has been run, or the HALT opcode has been processed. With this, I could then write an if-else condition that only allows run program to be run if **has_run** is **false**, and the only way to turn it false is to click the reset RAM button.

Again, one of my concerns is the discrepancies between the 3 tests – they should all have the exact same outcomes. Perhaps this could be due to the fact that I may have accidentally done slightly different each time while actually conducting the tests on video, which is a very plausible idea.

Test 22, 111, and 134

Test Description: Enter the values (4 and 7) and the LMC will handle it as the program tells it to (should output 4, as that is the smaller number).
Outcome: Pass (22 and 111). Fail (134).

Problems:
Still shows 7 instead of 4, meaning the code did not work as expected.

Refinements:
I reckon that this is probably a mistake I made while changing the code during the test in order for it to display the smaller number.

However, if it wasn't my fault then, again, I am concerned about the different outcomes for the 3 tests. This will need to be explored further.

Test 148

Test Description: Go to the Create Challenges page, and hover over the yellow dot. A ToolTip should be displayed to clarify that a teacher is currently signed in.
Outcome: Fail.

Problems:
Though the yellow dot was shown, the ToolTip was not.

Refinements:
This is likely due to an error with how much data was being retrieved from the database, so the ToolTip did not load properly.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
    <% include ./partials/script %>
    <% include ./partials/style %>
    <% include ./partials/head %>
    <% include ./partials/script_toggle %>
</head>

<body>
```

```
13.    <div class="container-fluid">
14.
15.       <% include ../partials/menu %>
```

If we look at the code above, which is the beginning of the **views/** EJS file for the create challenges page, all the partial files are linked correctly, so there is no reason for the ToolTip to not work.

Hence, we can quite confidently conclude that it is due to the sheer volume of information that was being processed by the backend because it was quite a lot: for each challenge and to assign challenges, there were multiple asynchronous SQL statements, and therefore, we could safely assume that the page was still waiting for one of the responses from the database, and the JavaScript partial file with the ToolTip code would only load after receiving that data from the database, simply due to the order of things in the way NodeJS and EJS work.

Test 164 and 166

Test Description: Replace the 2 text boxes for editing a challenge with the following test data (<any characters, totalling 101 or more characters, This is a test description) and click the update button (164). Click the edit button for the same challenge, and change the description, then click the update button (166).
Outcome: Fail (164). Pass (166)

Problems:
Page did not load, and error 404 was displayed (164). Changing the description worked, but the title that was supposed to have changed from above, after being shown as changed, was reverted to the original (166).

Refinements:
Thinking again about the way this data is saved in the database and retrieved, I remembered that I decided to use BLOBs to store this challenge data. However, this may not have been the best idea if I was going to include the functionality to edit challenges and their titles and descriptions, because BLOBs are immutable, and since only an UPDATE SQL command is called, it obviously would not work. Instead, I would either have to save this data as a string in the database, or, instead of doing an UPDATE, simply INSERT a new challenge file, and then override the link between the teacher, the challenge, and the challenge file.

Test 172

Test Description: Click on the view set challenges button in the navbar and then click on the challenge that was just assigned.
Outcome: Pass

Problems:

Students yet to submit and students that missed the deadline are titled the wrong way around.

Refinements:
This is straightforward – the titles just need to be swapped in the HTML/EJS file.

Having now looked at all of the tests that were flagged during the testing video, and having conducted some user testing as well, I will go on to evaluate my entire NEA project.

Chapter 7 Evaluation

Achievement of Objectives

Client Requirement 1

"The resource should be a Web Application which is coded using NodeJS as the framework and MySQL as the database. The website and all of its components should be easy to use, navigate, and understand."

Completely Achieved (Room for Improvement)

After conducting extensive research on various backend frameworks as well as templating engines, I realised the advantages of using NodeJS with the EJS engine, along with using MySQL to connect to the database. Through NodeJS's NPM, I had access to a wide range of dependencies which were able to seamlessly integrate with my server and public JavaScript files. The Express middleware proved very crucial for creating server controllers, managing client routing, and maintaining a view-controller framework, along with routes. Test 1 in the Testing section shows that the resource is a web-based application and how it uses the aforementioned technologies.

However, from the User Testing section above, it was apparent that there were places where things could have been changed for ease of use – for example, the names of the navbar tabs were sometimes misleading, and finding out how to assign new challenges was quite unintuitive. In addition, I got some feedback from my client regarding this requirement, and here are some snippets from the letter:

"Firstly, thank you so much for creating this web application for me. I am really impressed with how well this has been built and I know that it will greatly benefit our students and teachers when it comes to learning and teaching computer architecture and the assembly language modules of the GCSE and A-Level courses. Mostly, this website is easy to use and navigate, so I am sure that this will make it much easier for students to learn."

"However, I won't lie to you, Raghav, I didn't see that there was a page that contained information on how to use this resource until I was actively looking for it. So perhaps this is something to think about?"

From these comments, we can see that the feedback is overall positive, however it seems as though some parts of the website remain relatively obscured or hard to find. For this reason, though the majority of the requirement has been met, I have marked this as completely achieved with room for improvement, because I could definitely make the website easier to navigate, either with a tutorial of where everything is when the website is first opened, or with tooltips or other markers that point the user to the important pages.

Client Requirement 2

"Users should be able to write code in the Assembly Language and be able to run the machine-code operations which do the necessary functions that are required to be learnt by AQA."

Completely Achieved (Room for Improvement)

From the User Testing section, when the students were testing writing their own Assembly code and visualising it, from task 4, they encountered no problems with the correct (syntactically and logically) Assembly code written, though they gave it a score of 2 (where 1 on a scale of 4 was the best). However, this was because they had some suggestions for how the website's functionality could be improved itself to make it easier for students to use, and doesn't necessarily relate to this requirement, so this evidence suggests that this requirement was completed to a high degree of success.

My client had the following things to say about this:

"I was quite intrigued by how you were going to go about allowing users to write their assembly code, and I am extremely impressed with how it has been done – I really like how the line numbers automatically update because it works almost like an IDE itself, which will be extremely beneficial for students. However, though we did speak about it earlier, I do think that having syntax highlighting would be a really cool and useful feature to add to this!"

It is quite evident from the quote above that the client is pleased with how the website can run Assembly code, though he also had some ideas for what I could potentially add in a future version (which we had initially discussed and decided were out of scope for this project). In addition to this, tests 16 and 20 in the Testing section show this requirement to be fully met and working, so this can be considered completely achieved, though I will later be discussing how I could go about doing the improvements and suggestions mentioned, so there is still room for improvement.

Client Requirement 3

"Users should be able to view visualisations of the Assembly code that they have written, with the necessary components included in the visualisation (being smoothly changed and animated)."

Completely Achieved

This client requirement is quite similar to the previous one, and, again, from the student User Testing section, we can see that the student users were able to perfectly see how each component worked with the Assembly code they had written through task 4 again. As before, though they scored it as a 2, that was because they would've liked a slightly different layout, but they were still able to see the components being animated smoothly, which satisfies this

requirement. Also, from the Testing section, tests 8-13, 18, and 19 show that the visualisations work, and the necessary components were discussed during the Analysis and Documented Design sections.

The client had this to say:

"I was also initially a bit concerned with how the visualisation would be displayed and if it would work smoothly but having used this resource during some of my lessons, there have been no crashes, and the animations are easy to follow. In fact, I think that the visualisations of the code are fantastic! They work really well and highlighting the components being used is extremely useful! At the same time, I would really like it if the user could adjust the visualisation speed while it is being run."

From above, we can see that the visualisation works quite seamlessly, as the quote is quite positive. However, the client does bring up a good point about changing the speed while running the visualisation – this could certainly be helpful in different situations but is not an essential feature. Furthermore, it would be an extremely difficult task to do, as the time delay for animations is all done enclosed within a single counter, so it might take a significant amount of work, and I may explore this in the future improvements section. Since all the evidence suggests that the visualisations work as they are supposed to, I have marked this requirement as completely achieved.

Client Requirement 4

"There should be verbose output which displays any and all debugging for erroneous Assembly code that is written."

Partially Achieved

While some errors do indeed get displayed in the verbose output boxes and thus stop the Assembly code and visualisation from running, there were quite a few errors that went undetected. As discussed in the Testing section, there were some discrepancies between the same tests (such as tests 24, 113, and 136), which did not help my confidence in having completed this client requirement, because I am still unsure of why this occurred.

Also, both student users gave this part of their test (task 6 in the User Testing section) a 3, on a scale of 1 to 4, where 4 is the worst possible outcome. This clearly shows that there is more to be achieved for this client requirement. For both of them, they said that the error messages were sometimes random, were on the wrong lines, and overall, just quite inconsistent. However, since the tests, the wrong lines being shown have since been fixed in the Failed Tests and Refinements section above. Additionally, there was a random error that just seemingly appeared out of nowhere, but this has also been fixed.

It is also worthwhile to see what my client said regarding this:

"To be completely honest about the debugging aspect of Assembly code when it is wrong, sometimes, it did not work. I purposefully tried to plant some errors in the code, yet the visualisation kept running and did not inform me of the error. I also do think that some sort of feature to notify the user about logical errors could be beneficial. Of course, having said all this, the main reason for this resource was to be able to view the visualisation to see how the components work, with the debugging aspect as simply an additional feature, though a very useful one."

We can see from above that the client thought that this is one of the major areas that could be greatly improved upon, and I completely agree with him. Though he says that it is not one of the big areas of the project, I think that it is a really important part of the project so that students are able to understand where they went wrong. Since most of these error messages had to be hard-coded, I think it would potentially be a good idea to dig deeper into the logic of the Little Man Computer and consider what other sorts of errors there could be, in addition to considering how best to show these errors to the user.

Since some errors were still displayed and I was able to debug my Assembly code with these errors being displayed, I have marked this requirement as partially achieved, but I will definitely try to refine this aspect in the future, as it is a very important feature.

Client Requirement 5

"Users should have the option to upload text files which contain Assembly Language instructions, using the same Instruction Set as the website."

Completely Achieved

Initially, as shown by tests 17, 106, and 129 in the Testing section, I was able to upload text files, which were inputted correctly and able to run. However, I was still able to upload files that did not end in .txt, which should not have been allowed. This could also be seen from task 5 for the students in the User Testing section, where the task was given a score of 2 and 3 by each tester. However, both their comments said that the only problem was the fact that they were able to upload files apart from .txt, while the rest of the functionality still worked.

This problem of uploading only .txt files was later fixed in the Failed Test and Refinements section, so there are now no problems with this requirement. We can also see this through my client's feedback letter:

"I am also really impressed by how you were able to incorporate the ability to upload and run text files."

From this, we can see that the client had no problems with this requirement, and seems satisfied with this feature, having no criticisms or improvements. From my own analysis, however, I think that I could go one step further into this and ensure that only text files that contain valid text are allowed to be uploaded. In essence, only files that contain Assembly code may be uploaded, and this would have to be checked by ensuring each line contains one of the opcodes from the website's instruction set. I would not immediately check for errors, however, as that is what the visualisation and debugger could be used for – to identify where errors lie in the code. Overall, the website fulfils this requirement as shown by the client, so I have marked it as completely achieved.

Client Requirement 6

"Users should be able to view the Instruction Set on a separate page from the visualisations."
Completely Achieved

This requirement was mostly completed in the Documented Design section of this writeup, because I had to make sure that all of the information for the Instruction Set was relevant and shown to the user. The student users who tested this resource in the User Testing section seemed to concur with the sentiment that this page was done really well with the explanations, however they both mentioned that it would be a good idea to force a first-time user to visit this page. This feature is not essential to meeting the requirement but is something I will explore in the future.

This page worked exactly as it should while I was testing it, which can be seen through tests 3 and 15. The client had very little to say about this, only saying the following:

"Firstly, I must say that I loved the instructions page and the verbose output of the visualisation – the instructions page was really helpful for when I had to present any code to my class because I was able to split my computer screen, and the verbose output helped me explain what was happening."

The above simply shows that the client was happy with this page as it made it easier for him to teach the class. There is not much to dissect within this requirement, so I have marked this as completely achieved.

Client Requirement 7

"Users should be able to see step-by-step outputs of the details of each process being run, in relation to what is happening within the visualisation."
Completely Achieved

This is quite similar to the previous client requirement in terms of the fact that the student users who tested this resource during the User Testing section really liked this feature and it

helped them understand what was happening. This requirement was met during my testing and can be seen via tests 11 and 22 – all of the processes were clearly displayed. The quote from the previous client requirement includes the comments about this one, and simply says that the client really appreciated how well this feature was done. There is not much to expand upon for this requirement either, apart from the fact that maybe some of the text could be colour coded to make it easier to group processes together, which is again not essential, so I can clearly mark this requirement as completely achieved.

Client Requirement 8

"On the same page where users write their Assembly code and view visualisations, there should be a dropdown list of pre-created Assembly codes that can be loaded into the text area, ready for visualisation."

Completely Achieved (Room for Improvement)

When it comes to this requirement, we can see it working through test 16 in the Testing section. All of the given pre-created Assembly codes work, and they give good insight into how to use the visualisation in general – and how the specific instruction set works. My client had this to say for this requirement:

"On the main page, however, the drop-down examples were extremely useful in understanding how your instruction set works."

Clearly the client is quite happy with this part of the project and does not have any faults with it. However, looking at it as the developer myself, I think that there is room for a few more pre-created challenges to be created and put in as options, because there are specific things with the visualisation and the Assembly code instruction set that could be quite ambiguous – a student may want to do something like this, but might not know how to code it, so having more examples could never hurt. Since the client likes this and the tests passed, I have marked it as completely achieved, but I personally feel like there could be more examples, so I have marked it with room for improvement.

Client Requirement 9

"There should be an introductory tutorial for new users of the resource who have never visited the website."

Partially Achieved

Although there is a comprehensive tab in the navigation menu that explains how to use the resource, what it is for, and how the main visualisation works, one of the student testers in the User Testing section brought up a very interesting point during their test: that every user should be forced to view the "how to use this resource" page for the first time that they decide to use the visualiser or try to complete an assignment. I think that doing this would be much more

helpful for all users to perfectly understand exactly how the website actually works, and what they should expect from it.

For the requirement being met via the page, from tests 4 and 14 in the Testing section, we can see that the page works and displays all the necessary information. However, it is important to note what my client said about this, which was discussed a little bit earlier:

"However, I won't lie to you, Raghav, I didn't see that there was a page that contained information on how to use this resource until I was actively looking for it. So perhaps this is something to think about?"

From this we can see that it was still somewhat hard to find the necessary introductory material to the website to understand how it works and could be improved upon. In addition, though the page itself exists with explanations, it only contains static images and text, and something like a video which explains everything in further depth and detail, showing exactly where to click and how to get to each page, would probably be a lot more helpful for a first-time user of the resource. Also, to make the page more identifiable / brought to the attention of the user quicker, perhaps I could change the order of the tabs in the navbar. Or, I could even have this as a little button that floats at the bottom of the screen, and instead, make it a pop-up window, to draw attention to it, because that seems to be what the students in the User Testing section thought was lacking on the website with regards to this requirement.

Since I think that an additional video or walkthrough would have been much more helpful, and more attention could have been drawn to the existing explanations, which was basically the point of this requirement, I have marked it as partially achieved.

Client Requirement 10

"Users should be able to login and sign-up for a student or teacher account, depending on whether they are a student or teacher at Dubai College, using their Dubai College email address. In addition, teachers should be able to create challenges, which can be completed by students, and teachers should then be able to view all the students' answers."
Completely Achieved (Room for Improvement)

Firstly, logging in and signing up took a lot of time to figure out – even though I was using the passport library, it was quite a complicated and new task to me, as I had only used PHP in the past. Now, however, this process is completely seamless and smooth, and quite intuitive. This can be seen through tests 34-86, where no tests failed, and all the possible kinds of errors were handled. I did not get the users to test this because I had extensively tested it myself, and in any case, the users would not be focusing too much on logging in or signing up as compared to actually using the features that those things allow.

It is, however, important to note that the “forgot my password” feature was never implemented, even though this was never an explicit requirement that the client asked for. Having said that, it wouldn’t really be necessary for a school environment, especially when no one would ever really want to access another person’s account for this resource, so teachers could keep a list of students’ passwords in case they forget them.

When it comes to the actual challenges, from the teachers’ side during the User Testing section, tasks 1-4 were all about this requirement. Overall, I received good scores, getting 1s and 2s, however, there were a few small errors, and one score of 3, where the “yet to submit” students feature did not work. This was later addressed in the Failed Tests and Refinements section and is now working. From the students’ side, tasks 1, 2, and 3 were about this requirement. I received half 1s and half 2s for this, but for the 2s, the students simply stated that they would have preferred a slightly different design, though it still worked as it was supposed to, without any errors. Since the functionality was fine, I think that this part of the requirement can be marked as fully achieved, although there is still something left to be desired when it comes to ease of use for the users.

My client did not have much to say about this, apart from the fact that everything worked exactly as he would have liked it to. Here is exactly what was stated in his letter:

“In addition, I really like the challenges and how they work. The fact that I do not have to write the solution to what I give the students as challenges is really helpful, because this way I can just run their solutions in the visualisation and view the outputs to see if their code works.”

From the above, we can see clearly that the client was happy with how the challenges feature works and did not have any complaints or improvements that he thought of. However, I have a few things that I would like to mention (some of which were discussed in the Scope section of the Analysis).

Firstly, I think it would be better if it were clearer what is meant by Teacher Class Code for when they sign up – this could potentially be put into the “how to use this resource” page so that teachers know for certain what this is. Also, a point which was briefly touched upon by the teachers in the User Testing section, I think that having a confirmation message when trying to delete challenges or submit them would be helpful to ensure that nothing major is accidentally done in the website through a mis-click. Also, I could have perhaps added some way to sort students by classes instead of challenges. This would make navigation for teachers much easier – I think that teachers would probably use that feature more often than what is currently available. Finally, I think that it would be a good idea to have some sort of way where the user can immediately view the visualisation, instead of having to copy and paste a code from the challenge response into the home page, after navigating there. Something similar was also

mentioned by a student in the User Testing section, where pages could essentially be pop-ups, to make the process of copy-pasting code into the visualisation easier.

Having said all this, the main point of this requirement was for the challenges to be working so that teacher could assign them and view responses after students responded, which the website is able to do, which can be seen from tests 152-178 in the Testing section. Overall, I think that the challenges aspect was implemented really well, save for a few exceptions. The project meets all of the sub-divisions laid out by this requirement, and hence, it has been completely achieved, but as previously mentioned, there is certainly room for some improvements to make it easier for users to actually use the website.

Having now analysed each client requirement separately, I will go on to evaluate the overall effectiveness of my solution.

Overall Effectiveness of Solution

The overall effectiveness of the solution could only really be tested by my client and other end-users, because, obviously, the developer knows how each system and module in the resource is supposed to work. Through meetings with my client and getting other users to test my resource, I was able to identify the inherent problems with the resource, as well as the struggles they faced while using the resource. By thoroughly exploring these problems during the Analysis section, I was able to recognise the vital objectives that would be required for this project, and investigated the necessary functionalities, eventually creating my final requirements.

Throughout this project, I kept in touch with my client to make sure that what I was designing suited his needs to the best possible standard, and so that no decisions were made uninformed. During the Design section, I ended up having a lot of visual ideas of how certain pages would look, as well as prototyping significant chunks of the actual project, because I still had to make some decisions and do so research. Essentially, the Design section proved to be a sort of extension of the Analysis. I dedicated most of my time to working on the Analysis and Design sections of this project, and I think that that helped me ensure that I would be sticking to only what the client wants, and not go overboard, but at the same time, develop all the core features as fully as possible.

As a reminder, the main project idea was to create an Assembly code visualiser, where students could write their own Assembly code, and view how the internal components of a computer work while the Assembly code is running. In addition to this, it ended up being a resource that could be used by teachers to assign students specific challenges (to solve in Assembly code) so that the students could have a better understanding of 1) how Assembly code works, in general, and 2) how the components of the CPU work together, and in tandem with the RAM.

I focused on making sure that these 2 core features remained at the heart of the project, and also tried to ensure that it would be as easy to use as possible, because otherwise, Dubai College teachers simply would not use it. Through the tests I conducted in the Testing section and the tests that some users conducted in the User Testing section (both teachers and students), the resource does indeed appear to be extremely useful, and after the I refined some of the problems that were identified in the video and by the users themselves, I definitely think that this website is now at a level where it could be extremely useful and helpful to teach both GCSE and A-Level students about Assembly code and the internal components of a computer.

In fact, my client said that this project “will greatly benefit our students and teachers when it comes to learning and teaching computer architecture and the assembly language modules of the GCSE and A-Level courses”, which was the main objective of the website, so I can confidently say that I was able to make a resource that helps students learn about these topics. However, of course there were limitations with how much I was able to achieve. Namely the

fact that there were quite a few places where the website could have been a lot easier to use, which have been discussed in previous sections, so I do feel as though perhaps this resource could have been slightly more successful if I was able to make it easier to use.

Although I accomplished most of my primary objectives, seen from how much of each client requirement I achieved above – albeit to different degrees – I can always continue to improve on the project and add more features. Again, a lot of important features that could be helpful, that had not been identified earlier, have been recently brought up and thought about, and these could be explored in further detail in the future. For example, one of the best ideas was to allow a user to see the visualisation at the same time as viewing a completed challenge.

Though I have conducted some testing, I feel as though testing on a wider scale, with more users, would still be required in order to quantify exactly how effective this solution is, and if it is indeed more useful than any similar ideas that are already available online. Furthermore, if the school were to use it, they would need to conduct more tests of their own to ensure that all data is secure, especially when dealing with the personal data of students. Overall, though I do believe that my website was quite good after having tested it myself, showing it to a couple of students and teachers, and from my client's feedback letter, I think that actually using this resource in more lessons would allow me to identify even more niche errors that may occur to evaluate how resilient the website is, and accurately come to a decision of how useful and effective it is for students and teachers.

Client's (Mr. Wood's) Feedback Letter

Dear Raghav,

Firstly, thank you so much for creating this web application for me. I am really impressed with how well this has been built and I know that it will greatly benefit our students and teachers when it comes to learning and teaching computer architecture and the assembly language modules of the GCSE and A-Level courses. Mostly, this website is easy to use and navigate, so I am sure that this will make it much easier for students to learn.

With this resource, our students will finally be able to test their written assembly code, which will without a doubt help them answer questions about it when it comes to the exams. Typically, our students have struggled with these types of questions, and this resource brings a new way of practicing – especially with the ability for teachers to set challenges.

I was quite intrigued by how you were going to go about allowing users to write their assembly code, and I am extremely impressed with how it has been done – I really like how the line numbers automatically update because it works almost like an IDE itself, which will be extremely beneficial for students. However, though we did speak about it earlier, I do think that having syntax highlighting would be a really cool and useful feature to add to this!

I was also initially a bit concerned with how the visualisation would be displayed and if it would work smoothly but having used this resource during some of my lessons, there have been no crashes, and the animations are easy to follow. In fact, I think that the visualisations of the code are fantastic! They work really well and highlighting the components being used is extremely useful! At the same time, I would really like it if the user could adjust the visualisation speed while it is being run.

To be completely honest about the debugging aspect of Assembly code when it is wrong, sometimes, it did not work. I purposefully tried to plant some errors in the code, yet the visualisation kept running and did not inform me of the error. I also do think that some sort of feature to notify the user about logical errors could be beneficial. Of course, having said all this, the main reason for this resource was to be able to view the visualisation to see how the components work, with the debugging aspect as simply an additional feature, though a very useful one.

When it comes to some of the auxiliary features of the website that you asked me for specific feedback on, I have a few comments and some potential improvements.

Firstly, I must say that I loved the instructions page and the verbose output of the visualisation – the instructions page was really helpful for when I had to present any code to my class

because I was able to split my computer screen, and the verbose output helped me explain what was happening. However, I won't lie to you, Raghav, I didn't see that there was a page that contained information on how to use this resource until I was actively looking for it. So perhaps this is something to think about?

On the main page, however, the drop-down examples were extremely useful in understanding how your instruction set works. I am also really impressed by how you were able to incorporate the ability to upload and run text files. In addition, I really like the challenges and how they work. The fact that I do not have to write the solution to what I give the students as challenges is really helpful, because this way I can just run their solutions in the visualisation and view the outputs to see if their code works.

Overall, I would like to thank you for giving us this amazing resource to use! I cannot wait for this to be used by more people and I really appreciate all the effort that has gone into this project. Though I have mentioned some things to think about above, on the whole, I think that you have far exceeded my expectations. I knew that this was going to be a difficult undertaking from the beginning, and having created such a fantastic resource, I think that congratulations are in order. I am sure that many schools and teachers will be asking for access to this resource, so you should be proud, because this project will help so many people who struggle with these topics.

Once again, thank you and great work,
Mark Wood

Future Improvements

In this section, I will analyse and think about how I would go about adding certain features in the future as potential improvements to make my solution better and more effectively achieve my client requirements. These have been taken from the Achievement of Objectives sections earlier in the Evaluation, as well as from the Scope section in the Analysis. I will give a simple description of the improvement and talk a little about how I could achieve each improvement, but I will not be fully implementing these.

Syntax Highlighting

This feature automatically changes the text colour of certain words/commands in any written Assembly code and is done while this code is being written.

Firstly, for this feature, it is important to note that the current Assembly code editor area is a **<textarea>** HTML tag. This means that the style of its contents cannot be changed. Therefore, to achieve this feature, I would need to make this editor area a ** ** instead of a **<textarea>** so that I could alter the style. Within this ** ** tag, I could then use **** or **<p>** tags to separate words, and then assign colours to specific words/commands with the use of some JavaScript.

An example of some code to alter a word is shown below:

```
var keywords = ["INP","OUT"]; // words to change colour for
$("#assembly_code_editor_area").on("keyup", function(e){
  if (e.keyCode == 32) { // space key pressed
    var newHTML = "";
    $(this).text().replace(/[\s]+/g, " ").trim().split(" ").forEach(function (val) { // loop through words
    if (keywords.indexOf(val.trim().toUpperCase()) > -1)
      newHTML += "<span style='color: red'>" + val + " </span>";
    else
      newHTML += "<span style='color: black'>" + val + " </span>";
  });
  $(this).html(newHTML);
 }
});
```

The basic idea of this code is that there should be a pre-defined array or dictionary of words which should be a different colour. Then, whenever an edit is made in the editor area (or when the space key is pressed), the entire contents of the ** ** should be retrieved and split into words. After that, if we iterate through the words and find a word contained in the pre-defined array, we replace that text with the text within its own **** tag, which has a different colour.

Check Format of .txt File Uploaded

This is the idea that only .txt files with Assembly code contained within it should be allowed to be uploaded to the Assembly code text editor area.

To achieve this, first, I would need to read the data from the text file. From there, we can think about common characteristics of Assembly code. For one, I could check if there is any line within the file that has more than 4 words (anything separated by spaces), because if so, then it is more than likely not Assembly code. However, the only exception to this rule would be if there is a **//** at the beginning of the line, which indicates a comment, so I would need to check for this. Also, I could scan the code for the opcodes, and if the proportion of opcodes in the text file to the other words is quite large (say, approximately 20%), then it is likely Assembly code. However, the only problem with this approach is that it is not guaranteed to be 100% successful all the time.

An alternative approach could be to try to run the contents in the file as Assembly code for the visualisation in the background, and if there is an error, then the file does not contain Assembly code. However, the problem with this idea is that the user's Assembly code could simply be faulty. So, I could alter this to only check for syntax errors. However, a similar reason could cause a flaw in this idea.

I do not think there is a completely accurate way to determine if it is Assembly code, hence, a mixture of the 2 approaches mentioned above could be applied for this feature.

Force Instructions Page on First Login

My idea for this feature is to redirect a logged in user to the "how to use this resource" page if it is the first time that they are logging in. Otherwise, they should be redirected to the home page.

The logic for this is actually quite simple. First, I would need to create a new field in the Student/Teacher tables in the database called "has_logged_in". This will have a default value of False when an account is created.

Then, when logging in, I could check if has_logged_in is True. If it is not True (it is False), then the user is redirected to the "how to use this resource" page. Otherwise, they are redirected to the home page. After they have logged in and has_logged_in has been checked, I could use a SQL statement to change the has_logged_in to True, and this would now never return to False, since the user has logged in at least once before.

The SQL statement would look something like the following:

```
1. UPDATE Student SET has_logged_in = True WHERE student_id = ?; // for a student
2. UPDATE Teacher SET has_logged_in = True WHERE teacher_id = ?; // for a teacher
```

Colour Code Text in Verbose Output

This is quite similar to the syntax highlighting mentioned above. The difference, however, is the fact that I already know all of the things that could be outputted here, so I would simply need

to alter the code to have colour styles for specific words in the HTML and CSS files. This is quite a self-explanatory improvement.

Display Students' Challenges by Class

This code would be very similar to what already exists in **~/routes/teacher_challenges/view_student_challenges.js**. However, the SQL query would need to be changed so that I am instead searching for class. Something like the following could be used:

```
1. SELECT student_id FROM Students_In_Classes WHERE Students_In_Classes.class_id = Class.class_id AND Class.teacher_id = Teacher.teacher_id AND teacher_id = ? AND year_group = ?; // get all students in the class
2. SELECT challenge_teacher_id FROM Assigned_Challenges WHERE student_id = ?; // get the assigned challenges for each student in the class
```

The first SQL statement above would be used to get all of the students in the currently signed in teacher's class, after they specify a year group. After this, if the results of this command are stored in a list, this can be iterated over, and the second SQL statement could be used to get the assigned challenges for each student in that list (class). Once this is done, the challenges can be separated into to do, completed, and missed using the same logic as the SQL statements in the aforementioned directory.

Forgot My Password

In order to accomplish this, since each account is linked with a DC email, a user would simply need to enter their own email into a form. After this, an email would be sent to their inbox with a link to a page that allows them to reset their password. Once this is done, and the password meets all the conditions for a strong password, an UPDATE SQL query could be used to change the password in the database, after being hashed of course. Then, the user would be redirected to the login page, where they could enter their new password.

This is quite a standard way for "forgot my password" functionality to work, and there is no reason for me to differ from the standard. The only thing is that I would have to do some more research about what library/package I could use with NodeJS's Passport library to send emails and reset passwords, because coding all of that from scratch would be too much. I have not done this research yet, but it is something I could do in the future.

Confirmation Message

This is for when a student user wants to submit a challenge, or for when a teacher user wants to create, edit, delete, or assign a challenge.

This is actually quite simple to accomplish with JavaScript. By using the **window.confirm()** method, I could simply pass a text message into the function, and it would produce a popup with the options of "OK" and "Cancel". I could simply use an if statement with this method, and only go through with the changes if OK is clicked. The method is used in a similar way to the following:

```
if (window.confirm('Are you sure?')) {
    // clicked OK
    // do the necessary function
} else {
    // clicked cancel
    return
}
```

Button to Copy Assembly code and Redirect to Visualisation Page

As discussed before, this would be very useful for teachers to test students' submitted answers, and for students to view their previously submitted answers.

Firstly, a new button would need to be created, and this could be called something like "view visualisation of code". Immediately, the text in the text area above (the only **<textarea>** tag on the page) would need to be copied to the clipboard. After this, the page should be redirected to the home (main visualisation) page, where the Assembly code text editor is overwritten with the text/code copied to the clipboard.

Redirection is a common function, with the **href** attribute in HTML, and is easy to code. However, in order to copy text to the computer's clipboard, some JavaScript code would need to be used. Something akin to the following would be quite useful:

```
function copy_text() {
    var copy_text = document.getElementById("assembly_code"); // get text

    copy_text.select();
    copy_text.setSelectionRange(0, 99999); // for mobile devices

    navigator.clipboard.writeText(copy_text.value); // copy the text inside the text field

    alert("Copied the text: " + copy_text.value); // let user know text has been copied
}
```

This is just a simple example of how to copy text, but the basic principles remain the same, and could be implemented by changing a few of the variables to work with my code, as is generally true for almost all of the displayed code in this section.

Having looked at some of the potential future improvements I could make and discussing how they would work, I have now finished my evaluation of this project.

Chapter 8 Appendix I: All Project Code

~/bin/www

```
#!/usr/bin/env node

/**
```

```
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('assembly-code-visualiser:server');
var http = require('http');

require('dotenv').config()

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT);
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
```

```
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}

```

~/lib/auth.js

```
function check_authenticated(req, res, next) {
    if (req.isAuthenticated()) {
        return next();
    } else {
        res.redirect('/login');
    }
};

function check_not_authenticated(req, res, next) {
    if (req.isAuthenticated()) {
        return res.redirect('/')
    } else {
        next();
    }
}

module.exports = {
```

```
    check_authenticated,
    check_not_authenticated
};

```

~/lib/db.js

```
var mysql = require('mysql');

require('dotenv').config()

// from .env
var DB_HOST = process.env.DB_HOST
var DB_USER = process.env.DB_USER
var DB_PASSWORD = process.env.DB_PASSWORD
var DB_DATABASE = process.env.DB_DATABASE
var DB_PORT = process.env.DB_PORT

// connect to db, creation of tables done in phpMyAdmin UI
var connection = mysql.createConnection({
    host: DB_HOST,
    user: DB_USER,
    password: DB_PASSWORD,
    database: DB_DATABASE,
    port: DB_PORT
})

connection.connect(function (error) {
    if (!!error) {
        console.log('Could not establish a connection with the database:' + error);
    } else {
        console.log('Connected to Database..!');
    }
});

module.exports = connection;

```

~/lib/passport-config.js

```
var LocalStrategy = require('passport-local').Strategy;
var bcrypt = require('bcrypt');

// for connecting to database
var db_connection = require('./db');

var get_student_by_email = (email) => {
    return new Promise((resolve, reject) => {
        try {
            db_connection.query(
                'SELECT * FROM Student WHERE student_email = ?;', [email],
                function (err, rows) {

                    if (err) {
```

```
                console.log(err);
                reject(err);
            }

            console.log(rows);

            resolve(rows);
        }
    );
  } catch (error) {
    reject(error);
  }
});
};

var get_teacher_by_email = (email) => {
  return new Promise((resolve, reject) => {
    try {
      db_connection.query(
        'SELECT * FROM Teacher WHERE teacher_email = ?;', [email],
        function (err, rows) {

            if (err) {
                console.log('1:', err);
                reject(err);
            }

            console.log('2:', rows);

            resolve(rows);
        }
      );
    } catch (error) {
      reject('3:', error);
    }
  });
};

function initialize(passport) {
  console.log('Initialized Passport..!');

  var authenticate_user_student = async (req, email, password, done) => {

    // all user-side scripting for validation

    // ensure everything filled out
    if (email.length === 0) {
      return done(null, false, { message: ' Email is required ' });
    }
    if (password.length === 0) {
      return done(null, false, { message: ' Password is required' });
    }

    // check email matches regex for student email
    var regex_student = /^[A-Za-z]+\d{4}\@dubaicollege.org$/;
```

```
    if (email.match(regex_student) === null) {
      return done(null, false, { message: ' Invalid email format (DC email required)' });
    }

    // ensure password length is >= 8 characters
    if (password.length < 8) {
      return done(null, false, { message: ' Password length must be at least 8 characters' });
    }

    // all server-side scripting for validation
    try {

      await get_student_by_email(email).then(async (rows) => {

        if (!rows || rows.length <= 0) {
          return done(null, false, { message: ' No user with that email' });
        }

        try {

          if (await bcrypt.compare(password, rows[0].student_password)) {
            // everything correct from login
            rows[0].role = 'student';
            return done(null, rows[0])
          } else {
            return done(null, false, { message: ' Password incorrect' })
          }

        } catch (err) {
          console.log(err);
           return done(err)
         }

      });

    } catch (error) {
      console.log(error);
      return done(null, false, { message: error });
    }

  };

  passport.use('local_student', new LocalStrategy({
    usernameField: 'email',
    passwordField: 'password',
    passReqToCallback: true
  },
  authenticate_user_student));

  var authenticate_user_teacher = async (req, email, password, done) => {

    // all user-side scripting for validation

    // ensure everything filled out
    if (email.length === 0) {
```

```
        return done(null, false, { message: ' Email is required ' });
    }
    if (password.length === 0) {
        return done(null, false, { message: ' Password is required' });
    }

    // check email matches regex for teacher email
    var regex_teacher = /^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$/;
    if (email.match(regex_teacher) === null) {
        return done(null, false, { message: ' Invalid email format (DC email required)' });
    }

    // ensure password length is >= 8 characters
    if (password.length < 8) {
        return done(null, false, { message: ' Password length must be at least 8 characters' });
    }

    // all server-side scripting for validation
    try {

        await get_teacher_by_email(email).then(async (rows) => {

            if (!rows || rows.length <= 0) {
                return done(null, false, { message: ' No user with that email' });
            }

            try {

                if (await bcrypt.compare(password, rows[0].teacher_password)) {
                    // everything correct from login
                    rows[0].role = 'teacher';
                    return done(null, rows[0])
                } else {
                    return done(null, false, { message: ' Password incorrect' })
                }

            } catch (err) {
                console.log('4:', err);
                return done(err)
            }

        });

    } catch (error) {
        console.log('5:', error);
        return done(null, false, { message: error });
    }

};

passport.use('local_teacher', new LocalStrategy({
    usernameField: 'email',
    passwordField: 'password',
    passReqToCallback: true
},
```

```
    authenticate_user_teacher));

  passport.serializeUser(function(user, done) {
    process.nextTick(function() {
      if (user.role === 'student') {
        done(null, { id: user.student_id, email: user.student_email, role: user.role });
      } else {
        done(null, { id: user.teacher_id, email: user.teacher_email, role: user.role });
      }
    });
  });

  passport.deserializeUser(function(user, done) {
    process.nextTick(function() {
      return done(null, user);
    })
  });
};

module.exports = initialize

```

~/lib/redis.js

```
// redis for cookie and session management
var { createClient } = require('redis');

require('dotenv').config()

// from .env
var REDIS_HOST = process.env.REDIS_HOST
var REDIS_PORT = process.env.REDIS_PORT
var REDIS_DATABASE = process.env.REDIS_DATABASE

// redis configuration
var redisClient = createClient({
  legacyMode: true,
  host: REDIS_HOST,
  port: REDIS_PORT,
  database: REDIS_DATABASE
});

redisClient.connect().catch(console.error);

redisClient.on('error', function (err) {
  console.log('Could not establish a connection with redis:' + err);
});
redisClient.on('connect', function (err) {
  console.log('Connected to Redis..!');
});

module.exports = redisClient;

```

~/public/javascripts/active_challenges_display_toggler.js

```
1. $(document).ready(function () {
2.    $('#active_challenges_card_body').hide();
3.    $('#active_challenges').hide();
4.    $('#challenges_havent_been_assigned').hide();
5.    $('#students_have_submitted').hide();
6.    $('#students_have_submitted_table').hide();
7.    $('#students_yet_to').hide();
8.    $('#students_yet_to_table').hide();
9.    $('#students_missed').hide();
10.    $('#students_missed_table').hide();
11. });
12.
13. function toggle_display_challenges() {
14.    $('#active_challenges_card_body').toggle();
15.    $('#active_challenges').toggle();
16.    if ($('#display_challenges_toggle').text() == "Show Challenges" || $('#display_challenges_toggle').text() === "Show
Challenges") {
17.       $('#display_challenges_toggle').text("Hide Challenges");
18.    } else {
19.       $('#display_challenges_toggle').text("Show Challenges");
20.    };
21. };
22.
23. function toggle_assign_challenges() {
24.    $('#assign_challenges_card_body').toggle();
25.    $('#assign_challenge_form').toggle();
26.    if ($('#assign_challenges_toggle').text() == "Show Assign Challenges" || $('#assign_challenges_toggle').text() === "Show
Assign Challenges") {
27.       $('#assign_challenges_toggle').text("Hide Assign Challenges");
28.    } else {
29.       $('#assign_challenges_toggle').text("Show Assign Challenges");
30.    };
31. };
32.
33. function toggle_challenges_have_been_assigned() {
34.    $('#challenges_have_been_assigned').toggle();
35.    if ($('#display_challenges_have_been_toggle').text() == "Show Challenges" ||
$('#display_challenges_have_been_toggle').text() === "Show Challenges") {
36.       $('#display_challenges_have_been_toggle').text("Hide Challenges");
37.    } else {
38.       $('#display_challenges_have_been_toggle').text("Show Challenges");
39.    };
40. };
41.
42. function toggle_challenges_havent_been_assigned() {
43.    $('#challenges_havent_been_assigned').toggle();
44.    if ($('#display_challenges_havent_been_toggle').text() == "Show Challenges" ||
$('#display_challenges_havent_been_toggle').text() === "Show Challenges") {
45.       $('#display_challenges_havent_been_toggle').text("Hide Challenges");
46.    } else {
47.       $('#display_challenges_havent_been_toggle').text("Show Challenges");
48.    };
```

```
49. };
50.
51. function toggle_challenge_info() {
52.     $('#challenge_info').toggle();
53.     if ($('#display_challenge_info').text() == "Show Information" || $('#display_challenge_info').text() === "Show
Information") {
54.         $('#display_challenge_info').text("Hide Information");
55.     } else {
56.         $('#display_challenge_info').text("Show Information");
57.     };
58. }
59.
60. function toggle_students_have_submitted() {
61.     $('#students_have_submitted').toggle();
62.     $('#students_have_submitted_table').toggle();
63.     if ($('#display_students_have_submitted').text() == "Show Students" || $('#display_students_have_submitted').text() ===
"Show Students") {
64.         $('#display_students_have_submitted').text("Hide Students");
65.     } else {
66.         $('#display_students_have_submitted').text("Show Students");
67.     };
68. }
69.
70. function toggle_students_yet_to() {
71.     $('#students_yet_to').toggle();
72.     $('#students_yet_to_table').toggle();
73.     if ($('#display_students_yet_to').text() == "Show Students" || $('#display_students_yet_to').text() === "Show Students") {
74.         $('#display_students_yet_to').text("Hide Students");
75.     } else {
76.         $('#display_students_yet_to').text("Show Students");
77.     };
78. }
79.
80. function toggle_students_missed() {
81.     $('#students_missed').toggle();
82.     $('#students_missed_table').toggle();
83.     if ($('#display_students_missed').text() == "Show Students" || $('#display_students_missed').text() === "Show Students")
{
84.         $('#display_students_missed').text("Hide Students");
85.     } else {
86.         $('#display_students_missed').text("Show Students");
87.     };
88. }
89.
```

~/public/javascripts/form_toggler.js

```
1. // immediately hide teacher form when page loads
2. $(document).ready(function () {
3.     $('#teacher_form').hide();
4.     $('#teacher_card').hide();
5. });
6.
7. function change_form_login() {
8.     $('#teacher_form').toggle();
```

```
    $('#student_form').toggle();
    $('#teacher_card').toggle();
    $('#student_card').toggle();
}

```

~/public/javascripts/lmc.js

```
// define classes

// AQA A level instruction template
class Instruction {

  constructor(args) {

    this.name = args.name;
    this.numerical_value = args.numerical_value;

    this.operands = args.operands; // list of 'R's, 'M's, 'B's and/or 'L', where R = Register, M = Memory, B = Both/Either, L = Label (for Branch)

  };

  no_of_operands() {
    return this.operands.length;
  };

  get_operand_type(operand_num) {
    return this.operands[operand_num - 1];
  }; // e.g. for LDR instruction, operand_num = 1 would return R because the first operand is Rd

  process_instruction_opcode() {
    return;
  };

};

class HLT_Instruction extends Instruction {

  process_instruction_opcode(LMC) {

    return new Promise((resolve, reject) => {
      var i = 0;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        if (i == 0) {
          LMC.log_output('HLT');
          LMC.log_output('Executing instruction...');
        } else if (i == 1) {
          LMC.deactivate_wrapper('control_unit_wrapper');
          LMC.deactivate_wrapper('decode_unit_wrapper');
          control_bus.setOptions({
            color: '#AAAAAA'
          });
```

```
                LMC.log_output('Program stopped');
                LMC.cycles++;
            } else {
                LMC.log_output('####################');
                LMC.log_output('Program executed in ' + LMC.cycles + ' FDE cycles');
                LMC.paused = true;
                LMC.stop = true;
                document.getElementById('clock').value = LMC.cycles;
                LMC.deactivate_wrapper('clock_wrapper');
                LMC.cycles = 0;
                resolve();
                clearInterval(LMC.carry_on);
            };
            i++;
        }, LMC.animation_interval);
    });

  };

};

class INP_Instruction extends Instruction {

  process_instruction_opcode(LMC) {

    return new Promise((resolve, reject) => {
        var i = 0;
        LMC.carry_on = setInterval(() => {
            // fetch instruction
            LMC.check_error();

            if (LMC.assembly_code_error == true) {
                resolve();
                clearInterval(LMC.carry_on);
            } else {
                if (i == 0) {
                    LMC.log_output('INP');
                    LMC.log_output('Executing instruction...');
                } else if (i == 1) {
                    LMC.deactivate_wrapper('control_unit_wrapper');
                    LMC.deactivate_wrapper('decode_unit_wrapper');
                    control_bus.setOptions({
                        color: '#AAAAAA'
                    });
                    LMC.log_output('Waiting for user input...');
                    LMC.cycles++;
                } else if (i == 2) {
                    LMC.inp = prompt('User input:');
                    document.getElementById('input').value = LMC.inp;
                    document.getElementById('input').scrollTop = document.getElementById('input').scrollHeight;
                } else {
                    LMC.log_output('Store user input in Accumulator: ' + LMC.inp);
                    document.getElementById('accumulator').value = LMC.inp;
                    document.getElementById('clock').value = LMC.cycles;
                    LMC.deactivate_wrapper('clock_wrapper');
```

```
                resolve();
                clearInterval(LMC.carry_on);
            };
            i++;
        };
    }, LMC.animation_interval);
  });

};

};

class OUT_Instruction extends Instruction {

  process_instruction_opcode(LMC) {

    return new Promise((resolve, reject) => {
      var i = 0;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve();
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('OUT');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.activate_wrapper('accumulator_wrapper');
          } else if (i == 2) {
            LMC.deactivate_wrapper('accumulator_wrapper');
            LMC.log_output('Output value held in Accumulator: ' + document.getElementById('accumulator').value);
            LMC.cycles++;
            document.getElementById('output').value = document.getElementById('accumulator').value;
            document.getElementById('output').scrollTop = document.getElementById('output').scrollHeight;
          } else {
            document.getElementById('clock').value = LMC.cycles;
            LMC.deactivate_wrapper('clock_wrapper');
            resolve();
            clearInterval(LMC.carry_on);
          };
          i++;
        };
      }, LMC.animation_interval);
    });

  };

```

```
};

class ADD_Instruction extends Instruction {

  process_instruction_opcode(LMC, direct, operand, accumulator) {

    return new Promise((resolve, reject) => {
      var i = 0;
      var MBR;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve(MBR, accumulator);
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('ADD');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.activate_wrapper('ALU_wrapper');
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.cycles++;
          } else if (i == 2) {
            if (direct) {
              MBR = parseInt(operand);
              LMC.activate_wrapper('MBR_wrapper');
            } else {
              MBR = parseInt(document.getElementById('memory_location_' + operand).value);
              LMC.activate_wrapper('MAR_wrapper');
              LMC.deactivate_wrapper('MBR_wrapper');
            };
          } else if (i == 3) {
            if (direct) {
              LMC.activate_wrapper('MBR_wrapper');
              LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
              document.getElementById('MBR').value = MBR;
            } else {
              LMC.deactivate_wrapper('MAR_wrapper');
              LMC.deactivate_wrapper('MBR_wrapper');
              LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
              LMC.log_output('Fetch data at location held by MAR and store it in MBR: ' + MBR);
              document.getElementById('MAR').value = operand;
              document.getElementById('MBR').value = MBR;
            }
          } else if (i == 4) {
            LMC.activate_wrapper('MBR_wrapper');
            LMC.activate_wrapper('accumulator_wrapper');
          } else if (i == 5) {
            LMC.deactivate_wrapper('MBR_wrapper');
```

```
                    LMC.deactivate_wrapper('accumulator_wrapper');
                    LMC.log_output('Add MBR value to Accumulator and store the result in Accumulator: ' + accumulator + '+' +
MBR + '=' + (accumulator + MBR));
                    accumulator += MBR;
                    document.getElementById('accumulator').value = accumulator;
                } else {
                    document.getElementById('clock').value = LMC.cycles;
                    LMC.deactivate_wrapper('clock_wrapper');
                    LMC.deactivate_wrapper('ALU_wrapper');
                    resolve(MBR, accumulator);
                    clearInterval(LMC.carry_on);
                };
                i++;
            };
        }, LMC.animation_interval);
    });

  };

};

class SUB_Instruction extends Instruction {

  process_instruction_opcode(LMC, direct, operand, accumulator) {

    return new Promise((resolve, reject) => {
      var i = 0;
      var MBR;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve(MBR, accumulator);
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('SUB');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.activate_wrapper('ALU_wrapper');
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.cycles++;
          } else if (i == 2) {
            if (direct) {
              MBR = parseInt(operand);
              LMC.activate_wrapper('MBR_wrapper');
            } else {
              MBR = parseInt(document.getElementById('memory_location_' + operand).value);
              LMC.activate_wrapper('MAR_wrapper');
              LMC.deactivate_wrapper('MBR_wrapper');
```

```
                    };
                } else if (i == 3) {
                    if (direct) {
                        LMC.activate_wrapper('MBR_wrapper');
                        LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
                        document.getElementById('MBR').value = MBR;
                    } else {
                        LMC.deactivate_wrapper('MAR_wrapper');
                        LMC.deactivate_wrapper('MBR_wrapper');
                        LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
                        LMC.log_output('Fetch data at location held by MAR and store it in MBR: ' + MBR);
                        document.getElementById('MAR').value = operand;
                        document.getElementById('MBR').value = MBR;
                    }
                } else if (i == 4) {
                    LMC.activate_wrapper('MBR_wrapper');
                    LMC.activate_wrapper('accumulator_wrapper');
                } else if (i == 5) {
                    LMC.deactivate_wrapper('MBR_wrapper');
                    LMC.deactivate_wrapper('accumulator_wrapper');
                    LMC.log_output('Subtract MBR value from Accumulator and store the result in Accumulator: ' + accumulator
+ '-' + MBR + '=' + (accumulator - MBR));
                    accumulator -= MBR;
                    document.getElementById('accumulator').value = accumulator;
                } else {
                    document.getElementById('clock').value = LMC.cycles;
                    LMC.deactivate_wrapper('clock_wrapper');
                    LMC.deactivate_wrapper('ALU_wrapper');
                    resolve(MBR, accumulator);
                    clearInterval(LMC.carry_on);
                };
                i++;
            };
        }, LMC.animation_interval);
    });

  };

};

class LDA_Instruction extends Instruction {

  process_instruction_opcode(LMC, direct, operand) {

    return new Promise((resolve, reject) => {
        var i = 0;
        var MBR;
        LMC.carry_on = setInterval(() => {
            // fetch instruction
            LMC.check_error();

            if (LMC.assembly_code_error == true) {
                resolve(MBR);
                clearInterval(LMC.carry_on);
            } else {
```

```
            if (i == 0) {
              LMC.log_output('LDA');
              LMC.log_output('Executing instruction...');
            } else if (i == 1) {
              LMC.deactivate_wrapper('control_unit_wrapper');
              LMC.deactivate_wrapper('decode_unit_wrapper');
              control_bus.setOptions({
                color: '#AAAAAA'
              });
              LMC.cycles++;
            } else if (i == 2) {
              if (direct) {
                MBR = parseInt(operand);
                LMC.activate_wrapper('MBR_wrapper');
              } else {
                MBR = parseInt(document.getElementById('memory_location_' + operand).value);
                LMC.activate_wrapper('MAR_wrapper');
                LMC.deactivate_wrapper('MBR_wrapper');
              };
            } else if (i == 3) {
              if (direct) {
                LMC.activate_wrapper('MBR_wrapper');
                LMC.log_output('Direct addressing: set MBR to operand of current instruction: ' + operand);
                document.getElementById('MBR').value = MBR;
              } else {
                LMC.deactivate_wrapper('MAR_wrapper');
                LMC.deactivate_wrapper('MBR_wrapper');
                MBR = parseInt(document.getElementById('memory_location_' + operand).value);
                LMC.log_output('Set MAR to operand of the current instruction: ' + operand);
                LMC.log_output('Fetch data at location held by MAR (' + operand + ') and store it in MBR: ' + MBR);
                document.getElementById('MAR').value = operand;
                document.getElementById('MBR').value = MBR;
              }
            } else if (i == 4) {
              LMC.activate_wrapper('MBR_wrapper');
              LMC.activate_wrapper('accumulator_wrapper');
            } else if (i == 5) {
              LMC.deactivate_wrapper('MBR_wrapper');
              LMC.deactivate_wrapper('accumulator_wrapper');
              document.getElementById('accumulator').value = MBR;
              LMC.log_output('Store MBR value in Accumulator: ' + MBR);
            } else {
              document.getElementById('clock').value = LMC.cycles;
              LMC.deactivate_wrapper('clock_wrapper');
              resolve(MBR);
              clearInterval(LMC.carry_on);
            };
            i++;
          };
        }, LMC.animation_interval);
      });

    };

};
```

```
class STA_Instruction extends Instruction {

  process_instruction_opcode(LMC, operand) {

    return new Promise((resolve, reject) => {
      var i = 0;
      var MBR;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve(MBR);
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('STA');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.cycles++;
          } else if (i == 2) {
            MBR = parseInt(document.getElementById('accumulator').value);
            LMC.activate_wrapper('MBR_wrapper');
            LMC.activate_wrapper('accumulator_wrapper');
          } else if (i == 3) {
            LMC.deactivate_wrapper('accumulator_wrapper');
            LMC.log_output('Set MBR to value held in Accumulator: ' + MBR);
            document.getElementById('MBR').value = MBR;
          } else if (i == 4) {
            LMC.deactivate_wrapper('MBR_wrapper');
            LMC.activate_wrapper('MAR_wrapper');
            LMC.log_output('Store MBR value ' + MBR + ' at memory location held in MAR: ' + operand);
            document.getElementById('MAR').value = operand;
            document.getElementById('memory_location_' + operand).value = MBR;
          } else {
            LMC.deactivate_wrapper('MAR_wrapper');
            document.getElementById('clock').value = LMC.cycles;
            LMC.deactivate_wrapper('clock_wrapper');
            resolve(MBR);
            clearInterval(LMC.carry_on);
          };
          i++;
        };
      }, LMC.animation_interval);
    });

  };

};
```

```
class BRA_Instruction extends Instruction {

  process_instruction_opcode(LMC, operand) {

    return new Promise((resolve, reject) => {
      var i = 0;
      LMC.carry_on = setInterval(() => {
        // fetch instruction
        LMC.check_error();

        if (LMC.assembly_code_error == true) {
          resolve();
          clearInterval(LMC.carry_on);
        } else {
          if (i == 0) {
            LMC.log_output('BRA');
            LMC.log_output('Executing instruction...');
          } else if (i == 1) {
            LMC.activate_wrapper('status_register_wrapper');
            LMC.deactivate_wrapper('control_unit_wrapper');
            LMC.deactivate_wrapper('decode_unit_wrapper');
            control_bus.setOptions({
              color: '#AAAAAA'
            });
            LMC.cycles++;
          } else if (i == 2) {
            LMC.activate_wrapper('PC_wrapper');
          } else if (i == 3) {
            document.getElementById('PC').value = operand;
            LMC.deactivate_wrapper('PC_wrapper');
            LMC.program_counter = parseInt(operand);
            LMC.log_output('Set PC to operand of instruction: ' + operand);
          } else {
            document.getElementById('clock').value = LMC.cycles;
            LMC.deactivate_wrapper('clock_wrapper');
            LMC.deactivate_wrapper('status_register_wrapper');
            resolve();
            clearInterval(LMC.carry_on);
          };
          i++;
        };
      }, LMC.animation_interval);
    });

  };

};

class BRZ_Instruction extends Instruction {

  process_instruction_opcode(LMC, operand, accumulator) {

    return new Promise((resolve, reject) => {
      var i = 0;
```

```
        LMC.carry_on = setInterval(() => {
          // fetch instruction
          LMC.check_error();

          if (LMC.assembly_code_error == true) {
            resolve();
            clearInterval(LMC.carry_on);
          } else {
            if (i == 0) {
              LMC.log_output('BRZ');
              LMC.log_output('Executing instruction...');
            } else if (i == 1) {
              LMC.activate_wrapper('status_register_wrapper');
              LMC.deactivate_wrapper('control_unit_wrapper');
              LMC.deactivate_wrapper('decode_unit_wrapper');
              control_bus.setOptions({
                color: '#AAAAAA'
              });
              LMC.cycles++;
            } else if (i == 2) {
              LMC.log_output('Check if value held in Accumulator is 0');
            } else if (i == 3) {
              if (accumulator == 0) {
                LMC.log_output('0 == 0 - true');
                LMC.activate_wrapper('PC_wrapper');
              } else {
                LMC.log_output(accumulator + ' == 0 - false');
              };
            } else if (i == 4) {
              if (accumulator == 0) {
                LMC.deactivate_wrapper('PC_wrapper');
                LMC.log_output('Set PC to operand of instruction: ' + operand);
                document.getElementById('PC').value = operand;
                LMC.program_counter = parseInt(operand);
              } else {
                document.getElementById('clock').value = LMC.cycles;
                LMC.deactivate_wrapper('clock_wrapper');
                LMC.deactivate_wrapper('status_register_wrapper');
                resolve();
                clearInterval(LMC.carry_on);
              };
            } else {
              document.getElementById('clock').value = LMC.cycles;
              LMC.deactivate_wrapper('clock_wrapper');
              LMC.deactivate_wrapper('status_register_wrapper');
              resolve();
              clearInterval(LMC.carry_on);
            };
            i++;
          };
        }, LMC.animation_interval);
      });

    };

```

```
539. };
540.
541. class BRP_Instruction extends Instruction {
542.
543.    process_instruction_opcode(LMC, operand, accumulator) {
544.
545.       return new Promise((resolve, reject) => {
546.          var i = 0;
547.          LMC.carry_on = setInterval(() => {
548.             // fetch instruction
549.             LMC.check_error();
550.
551.             if (LMC.assembly_code_error == true) {
552.                resolve();
553.                clearInterval(LMC.carry_on);
554.             } else {
555.                if (i == 0) {
556.                   LMC.log_output('BRP');
557.                   LMC.log_output('Executing instruction...');
558.                } else if (i == 1) {
559.                   LMC.activate_wrapper('status_register_wrapper');
560.                   LMC.deactivate_wrapper('control_unit_wrapper');
561.                   LMC.deactivate_wrapper('decode_unit_wrapper');
562.                   control_bus.setOptions({
563.                      color: '#AAAAAA'
564.                   });
565.                   LMC.cycles++;
566.                } else if (i == 2) {
567.                   LMC.log_output('Check if value held in Accumulator is positive (>= 0)');
568.                } else if (i == 3) {
569.                   if (accumulator >= 0) {
570.                      LMC.log_output(accumulator + ' >= 0 - true');
571.                      LMC.activate_wrapper('PC_wrapper');
572.                   } else {
573.                      LMC.log_output(accumulator + ' >= 0 - false');
574.                   };
575.                } else if (i == 4) {
576.                   if (accumulator >= 0) {
577.                      LMC.deactivate_wrapper('PC_wrapper');
578.                      LMC.log_output('Set PC to operand of instruction: ' + operand);
579.                      document.getElementById('PC').value = operand;
580.                      LMC.program_counter = parseInt(operand);
581.                   } else {
582.                      document.getElementById('clock').value = LMC.cycles;
583.                      LMC.deactivate_wrapper('status_register_wrapper');
584.                      LMC.deactivate_wrapper('clock_wrapper');
585.                      resolve();
586.                      clearInterval(LMC.carry_on);
587.                   };
588.                } else {
589.                   document.getElementById('clock').value = LMC.cycles;
590.                   LMC.deactivate_wrapper('clock_wrapper');
591.                   LMC.deactivate_wrapper('status_register_wrapper');
592.                   resolve();
593.                   clearInterval(LMC.carry_on);
```

```
                };
                i++;
            };
        }, LMC.animation_interval);
    });

  };

};

// LMC template w/ methods used later
class Little_Man_Computer {

  constructor(args) {

    this.outputs = [];
    this.inputs = args.inputs || [];

    this.RAM = args.RAM || [];
    this.RAM_value_length = args.RAM_value_length || 3; // same as general_registers_value_length

    this.general_registers = args.general_registers || [];
    this.program_counter = 0;

    this.instruction_set = args.instruction_set || {};
    this.instruction_set_objs = args.instruction_set_objs || {};

    // this.clock = 25;
    // this.time_lapse = 10000;
    this.time_numerator = 10000;
    this.paused = true;

    this.labels = {};
    this.errors = [];
    this.inp;
    this.carry_on;
    this.cycles = 0;

    this.stop = false;

    this.animation_interval = 300; // milliseconds

    this.assembly_code_error = false;

  };

  // ensure backend RAM values are 2 digits long
  RAM_backend_to_RAM_value_length_digits(location) {

    // ensure 4 digits long
    if (this.RAM[location].length < this.RAM_value_length) {

      while (this.RAM[location].length < this.RAM_value_length) {
        this.RAM[location] = '0' + this.RAM[location];
      }
```

```

    } else if (this.RAM[location].length > this.RAM_value_length) {

        var location_as_string = location.toString();

        while (location_as_string.length < 2) {
            location_as_string = '0' + location_as_string;
        }

        alert('RAM value at memory location ' + location_as_string + ' must be ' + this.RAM_value_length.toString() + ' or less digits long');

        this.reset_specific_RAM(location);

    } else {
        // do nothing - correct length
    };

};

// populate frontend visualisation RAM locations with the values stored in this.RAM
load_RAM_from_backend() {
    for (var i = 0; i < this.RAM.length; i++) {
        // will always be 2 digits long because backend is always 2 digits long
        document.getElementById("memory_location_" + i.toString()).value = this.RAM[i];
    };
};

// populate this.RAM with the values in the frontent visualisation RAM locations
load_RAM_from_frontend() {

    for (var i = 0; i < this.RAM.length; i++) {
        this.RAM[i] = document.getElementById("memory_location_" + i.toString()).value;
        // this.RAM_backend_to_RAM_value_length_digits(i);
    };

    this.load_RAM_from_backend();

};

// reset all values in RAM location in frontend + backend to 00
reset_RAM() {
    for (var i = 0; i < this.RAM.length; i++) {
        this.RAM[i] = '0'.repeat(this.RAM_value_length);
    };
    this.load_RAM_from_backend();
};

// reset the value at a specific RAM location in frontend + backend to 00
reset_specific_RAM(location) {
    this.RAM[location] = '0'.repeat(this.RAM_value_length);
    document.getElementById("memory_location_" + location.toString()).value = this.RAM[location];
};

// ensure backend general registers' values are 2 digits long
```

```
general_registers_backend_to_RAM_value_length_digits(location) {

  if (this.general_registers[location].length < this.RAM_value_length) {

    while (this.general_registers[location].length < this.RAM_value_length) {
      this.general_registers[location] = '0' + this.general_registers[location];
    }

  } else if (this.general_registers[location].length > this.RAM_value_length) {

    var location_as_string = location.toString();

    while (location_as_string.length < this.RAM_value_length) {
      location_as_string = '0' + location_as_string;
    }

    alert('General Purpose Register ' + location_as_string + '\'s value must be ' + this.RAM_value_length.toString() + ' or less digits long');

    this.reset_specific_general_register(location);

  } else {
    // do nothing - correct length
  };

};

// populate frontend visualisation general purpose registers with the values stored in this.general_registers
load_general_registers_from_backend() {
  for (var i = 0; i < this.general_registers.length; i++) {
    // will always be 2 digits long because backend is always 2 digits long
    document.getElementById("general_register_" + i.toString()).value = this.general_registers[i];
  };
};

// reset all values in the general registers in backend (automatically update to frontend because read-only)
reset_general_registers() {
  for (var i = 0; i < this.general_registers.length; i++) {
    this.general_registers[i] = '0'.repeat(this.RAM_value_length);
  };
  this.load_general_registers_from_backend();
};

// reset the value of a specific general register in frontend + backend to 00
reset_specific_general_register(location) {
  this.general_registers[location] = '0'.repeat(this.RAM_value_length);
  document.getElementById("general_register_" + location.toString()).value = this.general_registers[location];
};

reset_all_registers() {
  this.reset_general_registers();

  document.getElementById('MAR').value = '00';
  document.getElementById('accumulator').value = '00';
  document.getElementById('status_register').value = '00';
```

```
    document.getElementById('CIR').value = '00';
    document.getElementById('PC').value = '00';
    document.getElementById('clock').value = '0';
    document.getElementById('MBR').value = '0'.repeat(this.RAM_value_length);

    this.program_counter = 0;

    this.deactivate_wrapper('ALU_wrapper');
    this.deactivate_wrapper('accumulator_wrapper');
    this.deactivate_wrapper('PC_wrapper');
    this.deactivate_wrapper('CIR_wrapper');
    this.deactivate_wrapper('MAR_wrapper');
    this.deactivate_wrapper('control_unit_wrapper');
    this.deactivate_wrapper('clock_wrapper');
    this.deactivate_wrapper('MBR_wrapper');
    this.deactivate_wrapper('status_register_wrapper');

    // leaderlines + wrappers back to normal
    address_bus.setOptions({
      end: LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
        x: 0,
        y: '23%'
      }),
      path: 'straight',
      color: '#AAAAAA'
    });
    control_bus.setOptions({
      color: '#AAAAAA'
    });
    data_bus.setOptions({
      end: LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
        x: 0,
        y: '86%'
      }),
      path: 'straight',
      color: '#AAAAAA'
    });
  };

  reset_verbose_output() {
    document.getElementById('verbose_output').value = 'Verbose Output';
  };

  reset_all_inp_out() {
    document.getElementById('input').value = '00';
    document.getElementById('output').value = '00';
    document.getElementById('FDE_output').value = 'Output of FDE cycle';
    this.reset_verbose_output();
  };

  // change border color of element to blue if gray, and vice-versa
  activate_deactivate_wrapper(element) {

    var element_info = document.getElementById(element);

```

```
    if ($(element_info).css('border-color') !== 'rgb(64, 64, 255)') {
        $(element_info).css('border-color', '#4040FF');
    } else {
        $(element_info).css('border-color', '#AAAAAA');
    };
};

activate_wrapper(element) {
    var element_info = document.getElementById(element);
    $(element_info).css('border-color', '#4040FF');
};

deactivate_wrapper(element) {
    var element_info = document.getElementById(element);
    $(element_info).css('border-color', '#AAAAAA');
};

// log to FDE Cycle Output (based on reset), and to Verbose Output
log_output(text, reset = false) {

    if (reset) {
        document.getElementById('FDE_output').value = text.replace('   ', '   ') + '\r\n';
    } else {
        document.getElementById('FDE_output').value = document.getElementById('FDE_output').value + text.replace('   ', '   ') + '\r\n';
    }

    document.getElementById('verbose_output').value = document.getElementById('verbose_output').value + text.replace('   ', '   ') + '\r\n';
    document.getElementById('verbose_output').scrollTop = document.getElementById('verbose_output').scrollHeight;

};

// get code from Assembly Code text area
scan_code() {

    this.labels = {};
    var counter = 0;
    var assembly_code = document.getElementById('code_area').value.toUpperCase();
    var lines = assembly_code.split('\n');

    for (var i = 0; i < lines.length; i++) {

        var line = lines[i].trim();

        if (line != '') {

            var instruction = line.split(/\s+/);

            // variable that stores input
            if (instruction.length == 2) {

                if (instruction[1] == 'DAT') {
                    this.labels[instruction[0]] = counter;
                    this.reset_specific_RAM(counter);
```

```
            } else if (!(instruction[0] in this.instruction_set)) {
                this.labels[instruction[0]] = counter;
            };

        };

        // variable that stores value after DAT
        if (instruction.length == 3) {

            if (instruction[1] == 'DAT') {
                this.labels[instruction[0]] = counter;
                this.RAM[counter] = instruction[2].toString();
            } else if (!(instruction[0] in this.instruction_set)) {
                this.labels[instruction[0]] = counter;
            }

        };

        counter++;

    };
  };
};

// into RAM from Assembly Code after converting to opcode + operand form
load() {

  this.load_RAM_from_frontend();

  this.paused = true;
  this.cycles = 0;
  this.assembly_code_error = false;

  // reset all registers (general purpose and special purpose)
  this.reset_all_registers();
  // reset all inputs / outputs / displays
  this.reset_all_inp_out();
  // reset RAM
  this.reset_RAM();
  // reset general registers
  this.reset_general_registers();

  // scan code for labels
  this.scan_code();

  // load program in RAM
  var counter = 0;
  var assembly_code = document.getElementById('code_area').value.toUpperCase();
  var lines = assembly_code.split('\n');

  for (var i = 0; i < lines.length; i++) {

    var line = lines[i].trim();

    if (line != '') {
```

```

        var instruction = line.split(/\s+/);

        if (instruction.length == 3) {

          var label = instruction[0];
          var opcode = instruction[1];
          var operand = instruction[2];

          this.labels[label] = counter;

          if (operand in this.labels) {
            operand = this.labels[operand];
          };

          if (operand.length > 1) {

            if (operand.substring(0, 1) == '@') {

              operand = operand.substring(1);

              if (isNaN(operand)) {
                if (operand in this.labels) {
                  operand = '@' + this.labels[operand];
                };
              } else {
                operand = '@' + operand;
              };

            } else if (operand.substring(0, 1) == '#') { // immediate addressing

              operand = operand.substring(1);

              if (isNaN(operand)) {
                if (operand in this.labels) {
                  operand = '#' + this.labels[operand];
                };
              } else {
                operand = '#' + operand;
              };

            };

          };

          if (opcode in this.instruction_set) {

            var encode = this.instruction_set[opcode].numerical_value + operand;
            this.RAM[counter] = encode.toString();

            counter++;

          } else if (opcode == 'DAT') {

            counter++;
```

```

            } else {

                this.log_output('Error at line ' + counter + ': Invalid opcode');

            };

        } else if (instruction.length == 2) {

            if (instruction[0] in this.instruction_set) {

                var opcode = instruction[0];
                var operand = instruction[1];

                if (operand in this.labels) {
                    operand = this.labels[operand];
                };

                // Indirect Addressing
                if (operand.length > 1) {

                    if (operand.substring(0, 1) == '@') {

                        operand = operand.substring(1);

                        if (isNaN(operand)) {
                            if (operand in this.labels) {
                                operand = '@' + this.labels[operand];
                            };
                        } else {
                            operand = '@' + operand;
                        };

                    } else if (operand.substring(0, 1) == '#') {

                        operand = operand.substring(1);

                        if (isNaN(operand)) {
                            if (operand in this.labels) {
                                operand = '#' + this.labels[operand];
                            };
                        } else {
                            operand = '#' + operand;
                        };

                    };

                };

                var encode = this.instruction_set[opcode].numerical_value + operand;
                this.RAM[counter] = encode.toString();

                counter++;

            } else {
```

```
1030.
1031.                 this.labels[instruction[0]] = counter;
1032.                 var opcode = instruction[1];
1033.
1034.                 if (opcode in this.instruction_set) {
1035.
1036.                    var encode = this.instruction_set[opcode].numerical_value;
1037.                    this.RAM[counter] = encode.toString();
1038.
1039.                    counter++;
1040.
1041.                 } else if (instruction[1] == 'DAT') {
1042.
1043.                    counter++;
1044.
1045.                 } else {
1046.
1047.                    this.log_output('Error at line ' + counter + ': Invalid instruction');
1048.
1049.                 };
1050.
1051.              };
1052.
1053.           } else {
1054.
1055.              if (line in this.instruction_set) {
1056.                 this.RAM[counter] = this.instruction_set[line].numerical_value;
1057.                 counter++;
1058.              } else {
1059.                 this.log_output('Error at line ' + counter + ': Invalid instruction');
1060.              };
1061.
1062.           };
1063.
1064.        };
1065.
1066.        this.load_RAM_from_backend();
1067.
1068.     };
1069.
1070.     for (var i = 0; i < this.RAM.length; i++) {
1071.        if (this.RAM[i] == '') {
1072.           this.reset_specific_RAM(i);
1073.        };
1074.     };
1075.     this.load_RAM_from_backend();
1076.  };
1077.
1078.  check_error() {
1079.     let FDE = document.getElementById('FDE_output').value;
1080.     // alert(FDE);
1081.     // alert(FDE.split('\n')[(FDE.split('\n').length - 1) - 1]);
1082.     // alert(FDE.split('\n')[(FDE.split('\n').length - 1) - 2]);
1083.     let verbose = document.getElementById('verbose_output').value;
1084.     // alert(verbose);
```

```
      // alert(verbose.split('\n')[(verbose.split('\n').length - 1) - 1]);
      // alert(verbose.split('\n')[(verbose.split('\n').length - 1) - 2]);

      let FDE_list = FDE.split('\n');
      let verbose_list = verbose.split('\n');

      // dealing with errors of log_outputs
      if (FDE_list.length >= 3 && verbose_list.length >= 3) {
        if (FDE_list[(FDE_list.length - 1) - 1] == FDE_list[(FDE_list.length - 1) - 2] || verbose_list[(verbose_list.length - 1) - 1] == verbose_list[(verbose_list.length - 1) - 2]) {
          this.assembly_code_error = true;
        } else {
          this.assembly_code_error = false;
        };
      } else {
        this.assembly_code_error = false;
      };
    }

    process_instruction_main_beginning() {
      return new Promise((resolve, reject) => {
        var i = 0;
        var instruction, k_str;
        this.carry_on = setInterval(() => {
          // fetch instruction
          this.check_error();

          if (this.assembly_code_error == true) {
            resolve(instruction);
            clearInterval(this.carry_on);
          } else {
            if (i == 0) {
              this.program_counter = parseInt(document.getElementById('PC').value);
              this.log_output('####################', true);
              this.log_output('Fetching instruction...');
              this.activate_wrapper('control_unit_wrapper');
              this.activate_wrapper('clock_wrapper');
              control_bus.setOptions({
                color: '#4040FF'
              });
            } else if (i == 1) {
              this.activate_wrapper('PC_wrapper');
            } else if (i == 2) {
              instruction = document.getElementById('memory_location_' + this.program_counter).value;
              this.activate_wrapper('MAR_wrapper');
              document.getElementById('MAR').value = this.program_counter;
            } else if (i == 3) {
              this.deactivate_wrapper('PC_wrapper');
              this.deactivate_wrapper('MAR_wrapper');
              this.log_output('Set MAR to value of PC: ' + this.program_counter);
            } else if (i == 4) {
              this.activate_wrapper('PC_wrapper');
              this.program_counter++;
              document.getElementById('PC').value = this.program_counter;
            } else if (i == 5) {
```

```
            this.deactivate_wrapper('PC_wrapper');
            this.log_output('Increment PC by 1');
        } else if (i == 6) {
            this.log_output('Fetch instruction from address stored in MAR');
        } else if (i == 7) {
            this.activate_wrapper('MAR_wrapper');
        } else if (i == 8) {

            // get memory wrapper location as 2 digits e.g. 00, 01, 92, etc.
            k_str = document.getElementById('MAR').value.toString();
            while (k_str.length < 2) {
                k_str = '0' + k_str;
            };

            // move address bus + change color
            address_bus.setOptions({
                end: document.getElementById('memory_' + k_str + '_wrapper'),
                path: 'fluid',
                color: '#4040FF'
            });

        } else if (i == 9) {
            this.activate_wrapper('memory_location_' + document.getElementById('MAR').value.toString());
        } else if (i == 10) {
            // move data bus + change color
            data_bus.setOptions({
                end: document.getElementById('memory_' + k_str + '_wrapper'),
                path: 'fluid',
                color: '#4040FF'
            });
        } else if (i == 11) {
            this.activate_wrapper('MBR_wrapper');
            document.getElementById('MBR').value = instruction;
        } else if (i == 12) {

            // leaderlines + wrappers back to normal
            address_bus.setOptions({
                end: LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
                    x: 0,
                    y: '23%'
                }),
                path: 'straight',
                color: '#AAAAAA'
            });
            data_bus.setOptions({
                end: LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
                    x: 0,
                    y: '86%'
                }),
                path: 'straight',
                color: '#AAAAAA'
            });

            this.deactivate_wrapper('MAR_wrapper');
            this.deactivate_wrapper('memory_location_' + document.getElementById('MAR').value.toString());
```

```
                this.deactivate_wrapper('MBR_wrapper');

                this.log_output('Fetched instruction ' + instruction + ' stored in MBR');

            } else if (i == 13) {
                this.deactivate_wrapper('control_unit_wrapper');
                control_bus.setOptions({
                    color: '#AAAAAA'
                });
                this.activate_wrapper('MBR_wrapper');
            } else if (i == 14) {
                this.activate_wrapper('CIR_wrapper');
                document.getElementById('CIR').value = instruction;
            } else if (i == 15) {
                this.deactivate_wrapper('MBR_wrapper');
                this.deactivate_wrapper('CIR_wrapper');
                this.log_output('Copied instruction from MBR to CIR');
            } else {
                this.log_output('Decoding instruction stored in CIR...');
                this.activate_wrapper('decode_unit_wrapper');
                this.activate_wrapper('control_unit_wrapper');
                control_bus.setOptions({
                    color: '#4040FF'
                });
                resolve(instruction);
                clearInterval(this.carry_on);
            }
            i++;
        };
    }, this.animation_interval);
  });
};

process_instruction_HLT() {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['HLT'].process_instruction_opcode(LMC_Obj);
    resolve();
  });
};

process_instruction_INP() {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['INP'].process_instruction_opcode(LMC_Obj);
    resolve();
  });
};

process_instruction_OUT() {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['OUT'].process_instruction_opcode(LMC_Obj);
    resolve();
  });
```

```
};

process_instruction_ADD(direct, operand, accumulator) {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['ADD'].process_instruction_opcode(LMC_Obj, direct, operand, accumulator);
    resolve();
  });
};

process_instruction_SUB(direct, operand, accumulator) {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['SUB'].process_instruction_opcode(LMC_Obj, direct, operand, accumulator);
    resolve();
  });
};

process_instruction_LDA(direct, operand) {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['LDA'].process_instruction_opcode(LMC_Obj, direct, operand);
    resolve();
  });
};

process_instruction_STA(operand) {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['STA'].process_instruction_opcode(LMC_Obj, operand);
    resolve();
  });
};

process_instruction_BRA(operand) {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['BRA'].process_instruction_opcode(LMC_Obj, operand);
    resolve();
  });
};

process_instruction_BRZ(operand, accumulator) {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['BRZ'].process_instruction_opcode(LMC_Obj, operand, accumulator);
    resolve();
  });
};

process_instruction_BRP(operand, accumulator) {
  return new Promise(async (resolve, reject) => {
    var LMC_Obj = this;
    await this.instruction_set_objs['BRP'].process_instruction_opcode(LMC_Obj, operand, accumulator);
    resolve();
```

```
    });
};

// asynchronous so order of events is followed
async process_instruction() {

    if (this.assembly_code_error == false) {

        // must be done in order, so await is used
        var instruction = await this.process_instruction_main_beginning();

        // halt instruction
        if (instruction == '000') {

            await this.process_instruction_HLT();

        } else if (instruction == '901') { // input instruction

            await this.process_instruction_INP();

        } else if (instruction == '902') { // output instruction

            await this.process_instruction_OUT();

        } else {

            var opcode = instruction.substr(0, 1);
            var operand_address = instruction.substr(1);
            var operand;
            var direct = false;

            if (operand_address.substr(0, 1) == '@') { // indirect addressing
                operand = parseInt(document.getElementById('memory_location_' + operand_address.substr(1)).value);
            } else if (operand_address.substr(0, 1) == '#') { // direct addressing
                direct = true;
                operand = operand_address.substr(1);
            } else {
                operand = operand_address;
            };

            var accumulator = parseInt(document.getElementById('accumulator').value);

            if (opcode == '1') { // add instruction
                var MBR, accumulator = await this.process_instruction_ADD(direct, operand, accumulator);
            } else if (opcode == '2') { // subtract instruction
                var MBR, accumulator = await this.process_instruction_SUB(direct, operand, accumulator);
            } else if (opcode == '5') { // load to accumulator
                var MBR = await this.process_instruction_LDA(direct, operand);
            } else if (opcode == '3') { // store accumulator value in memory
                var MBR = await this.process_instruction_STA(operand);
            } else if (opcode == '6') { // always branch
                await this.process_instruction_BRA(operand);
            } else if (opcode == '7') { // branch if zero
                await this.process_instruction_BRZ(operand, accumulator);
            } else if (opcode == '8') { // branch if positive or 0
```

```
                    await this.process_instruction_BRP(operand, accumulator);
                };
            };
        } else { // invalid instruction - not recongnised
            alert('Error in assembly code! Executing HLT instruction!');
            this.activate_wrapper('status_register_wrapper');
            document.getElementById('status_register').value = '-1';
            await this.process_instruction_HLT();
            this.paused = true;
            this.stop = true;
        };
    };

    step() {

        this.paused = true;
        this.animation_interval = parseInt(this.time_numerator / document.getElementById('clock_speed').value);
        this.process_instruction();

    };

    async run() {

        this.cycles = 0;
        this.stop = false;

        this.reset_all_registers();
        this.reset_all_inp_out();

        this.scan_code();

        this.paused = false;
        this.animation_interval = parseInt(this.time_numerator / document.getElementById('clock_speed').value);

        while (!this.stop) {
            await this.process_instruction();
        };

    };

};

// functions to initialize values

// create instruction and LMC objects
function initialise_LMC() {

    var INP = new Instruction({ // get user input + store in accumulator
        name: 'INP',
        numerical_value: '901',
        operands: []
    });
    var OUT = new Instruction({ // output value stored in accumulator
        name: 'OUT',
        numerical_value: '902',
```

```
    operands: []
});
var LDA = new Instruction({ // load accumulator w/ contents of given RAM address
    name: 'LDA',
    numerical_value: '5',
    operands: ['M']
});
var STA = new Instruction({ // store value in accumulator to RAM address OR register
    name: 'STA',
    numerical_value: '3',
    operands: ['B']
});
var ADD = new Instruction({ // add accumulator value to RAM address value OR register value
    name: 'ADD',
    numerical_value: '1',
    operands: ['B']
});
var SUB = new Instruction({ // subtract contents of RAM address OR register contents from accumulator value
    name: 'SUB',
    numerical_value: '2',
    operands: ['B']
});
var BRP = new Instruction({ // branch to RAM address if accumulator value is zero or positive
    name: 'BRP',
    numerical_value: '8',
    operands: ['M']
});
var BRZ = new Instruction({ // branch to RAM address if accumulator value is zero
    name: 'BRZ',
    numerical_value: '7',
    operands: ['M']
});
var BRA = new Instruction({ // branch to RAM address
    name: 'BRA',
    numerical_value: '6',
    operands: ['M']
});
var HLT = new Instruction({ // stop assembly code
    name: 'HLT',
    numerical_value: '000',
    operands: []
});
var DAT = new Instruction({ // associates a label to a free RAM address. optional value used to store at RAM address
    name: 'DAT',
    numerical_value: '', // will be 4 or something like that
    operands: [] // will be number, RAM address
});

// dictionary of Instruction objects
var instruction_set = {
    'INP': INP,
    'OUT': OUT,
    'LDA': LDA,
    'STA': STA,
    'ADD': ADD,
```

```
    'SUB': SUB,
    'BRP': BRP,
    'BRZ': BRZ,
    'BRA': BRA,
    'HLT': HLT,
    'DAT': DAT
};

var INP_Obj = new INP_Instruction({
    name: 'INP',
    numerical_value: '901',
    operands: []
});
var OUT_Obj = new OUT_Instruction({
    name: 'OUT',
    numerical_value: '902',
    operands: []
});
var LDA_Obj = new LDA_Instruction({
    name: 'LDA',
    numerical_value: '5',
    operands: ['M']
});
var STA_Obj = new STA_Instruction({
    name: 'STA',
    numerical_value: '3',
    operands: ['B']
});
var ADD_Obj = new ADD_Instruction({
    name: 'ADD',
    numerical_value: '1',
    operands: ['B']
});
var SUB_Obj = new SUB_Instruction({
    name: 'SUB',
    numerical_value: '2',
    operands: ['B']
});
var BRP_Obj = new BRP_Instruction({
    name: 'BRP',
    numerical_value: '8',
    operands: ['M']
});
var BRZ_Obj = new BRZ_Instruction({
    name: 'BRZ',
    numerical_value: '7',
    operands: ['M']
});
var BRA_Obj = new BRA_Instruction({
    name: 'BRA',
    numerical_value: '6',
    operands: ['M']
});
var HLT_Obj = new HLT_Instruction({
    name: 'HLT',
```

```
    numerical_value: '000',
    operands: []
  });

  // similar dictionary but different use (only for process_instruction)
  var instruction_set_objs = {
    'INP': INP_Obj,
    'OUT': OUT_Obj,
    'LDA': LDA_Obj,
    'STA': STA_Obj,
    'ADD': ADD_Obj,
    'SUB': SUB_Obj,
    'BRP': BRP_Obj,
    'BRZ': BRZ_Obj,
    'BRA': BRA_Obj,
    'HLT': HLT_Obj
  }

  RAM = [];
  var RAM_value_length = 4;

  for (var i = 0; i < 100; i++) {
    RAM.push('0'.repeat(RAM_value_length));
  };

  general_registers = [];

  for (var i = 0; i < 10; i++) {
    general_registers.push('0'.repeat(RAM_value_length));
  }

  // LMC object with all processing
  var LMC = new Little_Man_Computer({
    instruction_set: instruction_set,
    instruction_set_objs: instruction_set_objs,
    RAM: RAM,
    RAM_value_length: RAM_value_length,
    general_registers: general_registers
  });

  return LMC
};

// create leader-lines for address, control, & data buses
function create_buses() {

  address_bus = new LeaderLine(
    document.getElementById('MAR_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_10_wrapper'), {
      x: 0,
      y: '23%'
    }),
    {
      color: '#AAAAAA',
      size: 8,
```

```
        // outline: true,
        // endPlugOutline: true,
        // outlineColor: '#AAAAAA',
        // endPlugSize: 0.75,
        middleLabel: LeaderLine.pathLabel({
            text: 'Address Bus',
            color: 'black'
        }),
        path: 'straight',
        startSocket: 'right',
        endSocket: 'left',
        endPlug: 'arrow3'
    }
);

control_bus = new LeaderLine(
    document.getElementById('control_unit_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_50_wrapper'), {
        x: 0,
        y: '14%'
    }),
    {
        color: '#AAAAAA',
        size: 8,
        // outline: true,
        // endPlugOutline: true,
        // outlineColor: '#AAAAAA',
        // endPlugSize: 0.75,
        middleLabel: LeaderLine.pathLabel({
            text: 'Control Bus',
            color: 'black'
        }),
        path: 'straight',
        startSocket: 'right',
        endSocket: 'left',
        startPlug: 'arrow3',
        endPlug: 'arrow3'
    }
);

data_bus = new LeaderLine(
    document.getElementById('MBR_wrapper'),
    LeaderLine.pointAnchor(document.getElementById('memory_80_wrapper'), {
        x: 0,
        y: '86%'
    }),
    {
        color: '#AAAAAA',
        size: 8,
        // outline: true,
        // endPlugOutline: true,
        // outlineColor: '#AAAAAA',
        // endPlugSize: 0.75,
        middleLabel: LeaderLine.pathLabel({
            text: 'Data Bus',
```

```
            color: 'black'
        },
        path: 'straight',
        startSocket: 'right',
        endSocket: 'left',
        startPlug: 'arrow3',
        endPlug: 'arrow3',
    }
  );

};

LMC = initialise_LMC();

// all window event listeners from HTML

// on loading the page
window.addEventListener('load', function () {
  // draw leaderlines for 3 buses
  create_buses();
  // disable spellcheck for all input/output fields
  $('body').attr("spellcheck", false);
});

// run everytime anything on the page is clicked
window.addEventListener('click', function () {

  LMC.load_RAM_from_frontend();
  LMC.load_general_registers_from_backend();

});

// all other initialisations / scripts

// create line numbers for assembly code area
var assembly_code_area = document.getElementById('code_area');
var line_numbers = document.querySelector('.line-numbers');
assembly_code_area.addEventListener('keyup', event => {
  var number_of_lines = event.target.value.split('\n').length;
  line_numbers.innerHTML = Array(number_of_lines).fill('<span></span>').join('');
});
$(document).ready(function () {
  var number_of_lines = assembly_code_area.value.split('\n').length;
  line_numbers.innerHTML = Array(number_of_lines).fill('<span></span>').join('');
});

// make line numbers scroll with assembly code area
(function ($) {

  var methods = {

    /**
     * @var options default options
     */
    options: {
```

```
        /**
         * @var eventTimeOut time in milliseconds
         */
        eventTimeOut: 200,

        /**
         * @var scrollSpeed time in milliseconds the other light boxes should take to scroll to their relevant point.
         */
        scrollSpeed: 100
    },

    /**
     * init
     *
     * method that applies brings everything together
     *
     * @param options object an object that overwrites the default options
     * @return null
     */
    init: function (options) {
        // overwrite defaults
        $.extend(methods.options, options);

        /**
         * @var $this object cached global jQuery variable
         */
        var $this = $(this);

        // Assign IDs. Used to prevent jittering of element that is calling.
        var id_num = 0;
        $(this).each(function () {
            $(this).data("scrollid", id_num);
            id_num++;
        });

        // bind on scroll
        var callback = function (el) {

            var
                /**
                 * @var scroll_percent number percentage of top of element scrolled
                 */
                scroll_percent = methods.convertScrollToPercent.call(el),
                /**
                 * @var me number scroll id of element calling the scroll method
                 */
                me = el.data("scrollid");

            // Loop through each
            $.each($this, function () {
                // Set scroll to false to prevent this element from calling same event
                $(this).data("scroll", false);
                // if it's the same element, skip
                if (me !== $(this).data("scrollid")) {
                    // Scroll the element to percent
```

```
                methods.scrollToPercent.call(this, scroll_percent);
            }
            // Turn scroll back on
            $(this).data("scroll", true);
        });

    }

    // Assign scroll handler
    $(this).scroll(function (event) {
        /**
         * @var scroll_on boolean get scroll data value to see if it's safe to scroll
         */
        var scroll_on = $(this).data("scroll");

        // cancel call if it's not safe
        if (scroll_on == false) return;

        // Call method callback
        callback($(this));
    });

},

/**
 * convertScrollToPercent
 *
 * converts the scroll position to percent and returns it.
 *
 * @return number the percentage (duh)
 */
convertScrollToPercent: function () {
    // Defaults

    var
        horrizontal = $(this).data("horizontal") || false,
        scrollStyle = (horrizontal != false ? "scrollLeft" : "scrollTop"),
        scrollWidth = (horrizontal != false ? "scrollWidth" : "scrollHeight");
    // get top and scroll position

    var
        top = $(this)[scrollStyle](),
        height = (horrizontal ? $(this)[0].scrollWidth - $(this).width() : $(this)[0].scrollHeight - $(this).height());

    // console.log("top :" + top + ", height :" + height);
    // height == 100 then what does top equal? ()
    var
        answer = 100 * top,
        answer = answer / height

    //   return answer;
    return Math.round(answer);

},

```

```
    /**
     * scrollToPercent
     *
     * This method takes a percent and scrolls an element to that given point.
     *
     * @param percent numher the percent to which the element will be scrolled to.
     */
    scrollToPercent: function (percent) {

      var
        /**
         * @var horrizontal boolean what type of scroll to use. Horizontal or vertical
         */
        horrizontal = $(this).data("horizontal") || false,
        /**
         * @var height number the height or width of the element
         */
        height = (horrizontal ? $(this)[0].scrollWidth - $(this).width() : $(this)[0].scrollHeight - $(this).height()),
        // Perform equation to determin relative scroll positon
        answer = percent * height,
        /**
         * @var answer number the top or left position
         */
        answer = answer / 100;

      // scroll element to answer position
      if (horrizontal == true) {
        $(this).scrollLeft(answer);
      }
      else {
        $(this).scrollTop(answer);
      }
    }
  }

  /**
   * bindScroll
   *
   * a jQuery plugin that will bind the scroll to the given elements.
   * Each element should have a data attribute called scrollid. This
   * will let the plugin know not to apply the scrolling bind to that active
   * element. If element is to be a horizontal scroll, specify by setting
   * data-horizontal attribute to true.
   *
   * @param options object object with options (currently no options exist);
   * @return object returns the jQuery DOM object to maintain chainability
   */
  $.fn.bindScroll = function (options) {
    methods.init.call(this, options);
    return this;
  }

})($);
$(document).ready(function () {
  // apply bindScroll to all elements with .scroll.
```

```
    $(".scroll_together").bindScroll();
});

// allow tabs in assembly code area
document.getElementById('code_area').addEventListener('keydown', function (key) {
    if (key.keyCode == 9) {

        // get caret position / selection
        var start = this.selectionStart;
        var end = this.selectionEnd;

        var target = key.target;
        var value = target.value;

        // set textarea value to `text before caret` + tab + `text after caret`
        target.value = value.substring(0, start) + '\t' + value.substring(end);

        // put caret at right position again (add 1 for tab)
        this.selectionStart = this.selectionEnd = start + 1;

        // prevent losing focus
        key.preventDefault();

    };
}, false);

// triggered when a file is chosen for upload program
document.getElementById('upload_program_actual').addEventListener('change', () => {
    let files = document.getElementById('upload_program_actual').files;

    if (files.length == 0) return;

    const file = files[0];

    let reader = new FileReader();

    reader.onload = (element) => {
        const file = element.target.result;

        // regular expression to identify carriage
        // returns and line breaks
        const lines = file.split(/\r\n|\n/);
        document.getElementById('code_area').value = lines.join('\n');
    };

    reader.onerror = (element) => alert(element.target.error.name);
    reader.readAsText(file);

});

// functions called from HTML buttons

// remove all text froma assembly code editor
function clear_editor() {
    document.getElementById('code_area').value = '';
```

```
};

// set all values in RAM back to 00
function reset_RAM(LMC) {
    LMC.reset_RAM();
    LMC.reset_all_registers();
    LMC.reset_all_inp_out();
    LMC.stop = false;
    LMC.assembly_code_error = false;
    LMC = initialise_LMC();
};

// open new tab with /instruction_set
function open_link_new_tab(location) {

    // ternary operator used in case window.location doesn't include '/', works as follows:
    // bool_expr ? result_if_bool_expr_true : result_if_bool_expr_false;
    var redirect_to = window.location.toString() + (window.location.toString().includes('/') ? location : ('/' + location));

    window.open(redirect_to, '_blank');
};

function load_into_RAM(LMC) {
    LMC.load();
};
function run_program(LMC) {

    if (LMC.RAM[0] == '0000') {
        load_into_RAM(LMC);
    };

    if (!LMC.stop) {
        LMC.run();
    } else {
        alert('Program finished. To re-run, please reset RAM and load the program into RAM again.');
    };

};
function step_program(LMC) {

    if (LMC.RAM[0] == '0000') {
        load_into_RAM(LMC);
    };

    if (!LMC.stop) {
        LMC.step();
    } else {
        alert('Program finished. To re-run, please reset RAM and load the program into RAM again.');
    };

};

// for pre-created programs loading into assembly code area
$("#pre-created_programs").on("change", function () {

```

```
    try {

        var selected_option = $("#pre-created_programs").val().toString();
        var directory = '../text_files/pre_created_programs/' + selected_option + '.txt';

        // get file with same name as pre-created_programs selection and put that into text area
        fetch(directory).then(response => response.text()).then(text => {
            document.getElementById("code_area").value = text;
        });

    } catch (error) {
        document.getElementById("code_area").value = "does not exist";
    }

});

```

~/public/javascripts/menu.js

```
// set active tab in the navbar automatically
$(document).ready(function () {
    var element = $('meta[name="active-menu"]').attr('content');
    $('#' + element).addClass('active');
});

// activate tooltips (the thing that shows up underneath the navbar user indicator)
$(document).ready(function () {
    $('[data-toggle="tooltip"]').tooltip();
});

// post logout method when logout clicked on navbar (because links can't post, only get)
function post_logout() {
    document.forms['logout_form'].submit();
};

```

~/public/javascripts/student_toggles.js

```
$(document).ready(function () {
    $('#completed_card_body').hide();
    $('#completed').hide();
    $('#missed_card_body').hide();
    $('#missed').hide();
});

function toggle_display_challenges() {
    $('#active_challenges_card_body').toggle();
    $('#active_challenges').toggle();
    if ($('#display_challenges_toggle').text() == "Show Challenges" || $('#display_challenges_toggle').text() === "Show Challenges") {
        $('#display_challenges_toggle').text("Hide Challenges");
    } else {
        $('#display_challenges_toggle').text("Show Challenges");
    };
```

```
16. };
17.
18. function toggle_display_completed() {
19.     $('#completed_card_body').toggle();
20.     $('#completed').toggle();
21.     if ($('#display_completed_toggle').text() == "Show Challenges" || $('#display_completed_toggle').text() === "Show
Challenges") {
22.         $('#display_completed_toggle').text("Hide Challenges");
23.     } else {
24.         $('#display_completed_toggle').text("Show Challenges");
25.     };
26. };
27.
28. function toggle_display_missed() {
29.     $('#missed_card_body').toggle();
30.     $('#missed').toggle();
31.     if ($('#display_missed_toggle').text() == "Show Challenges" || $('#display_missed_toggle').text() === "Show Challenges") {
32.         $('#display_missed_toggle').text("Hide Challenges");
33.     } else {
34.         $('#display_missed_toggle').text("Show Challenges");
35.     };
36. };
37.
```

~/public/stylesheets/main_visualisation_style.css

```
 1. /* main visualisation stuff */
 2. #main_visualisation_wrapper_row_1, #main_visualisation_wrapper_row_2 {
 3.     display: flex;
 4.     flex-direction: row;
 5. }
 6.
 7. #code_wrapper, #RAM_wrapper, #CPU_wrapper, #general_register_wrapper, #ALU_wrapper, #accumulator_wrapper,
#PC_wrapper, #decode_unit_wrapper, #CIR_wrapper, #MAR_wrapper, #status_register_wrapper, #control_unit_wrapper,
#clock_wrapper, #MBR_wrapper, #controlbox_wrapper, #input_wrapper, #output_wrapper, #FDE_output_wrapper,
#verbose_output_wrapper {
 8.     border: 0.2em #AAAAAA solid;
 9. }
10. #code_wrapper:hover, #RAM_wrapper:hover, #CPU_wrapper:hover, #general_register_wrapper:hover,
#ALU_wrapper:hover, #accumulator_wrapper:hover, #PC_wrapper:hover, #decode_unit_wrapper:hover, #CIR_wrapper:hover,
#MAR_wrapper:hover, #status_register_wrapper:hover, #control_unit_wrapper:hover, #clock_wrapper:hover,
#MBR_wrapper:hover, #controlbox_wrapper:hover, #input_wrapper:hover, #output_wrapper:hover,
#FDE_output_wrapper:hover, #verbose_output_wrapper:hover {
11.     border: 0.2em #FF4040 solid !important;
12. }
13.
14. #code_wrapper {
15.     height: 41.5em;
16.     display: inline-flex;
17.     /* flex-direction: column; */
18.     /* float: left; */
19.     flex-grow: 1;
20. }
21. #code_area {
22.     resize: none;
```

```
    margin-left: 1em;
    float: right;
    overflow-y: scroll;
    height: 41em;
    width: 14em;
    white-space: nowrap;
    margin-right: 0.05em;
}
.line-numbers {
    text-align: left;
    float: left;
    height: 41em;
    overflow-y: scroll;
    /* below for Firefox */
    scrollbar-width: none;
    margin-right: -2em;
}
/* below for Safari & Chrome */
.line-numbers::-webkit-scrollbar {
    display: none;
}
.line-numbers span {
    counter-increment: linenumber;
}
.line-numbers span::before {
    content: counter(linenumber);
    display: block;
}

#RAM_wrapper {
    height: 41.5em;
    display: flex;
    text-align: center;
    flex-grow: 1;
    margin-left: -0.75em;
}
/* attributes with ids starting with "memory_location_" */
[id^="memory_location_"] {
    text-align: center;
    font-size: medium;
    border-color: #AAAAAA;
    border-style: solid;
}
/* attributes with ids starting with "memory_" AND ending with "_wrapper" */
[id^="memory_"][id$="_wrapper"] {
    border: 0.05em #AAAAAA solid;
    padding: 0.1em;
    margin: 0.2em 0;
}
/* same as above but when mouse hovering over it & change for second expression only */
[id^="memory_"][id$="_wrapper"]:hover > [id^="memory_location_"] {
    border-color: #FF4040;
}

#CPU_wrapper {
```

```
    flex-grow: 1;
    height: 41.5em;
    margin-right: -0.725em;
    /* TODO changed this */
    margin-left: 0.5em;
}
#CPU_column_2, #CPU_column_3 {
    margin-left: -1em;
    text-align: center;
    display: flex;
    flex-direction: column;
}

#general_register_wrapper {
    text-align: center;
    padding: 0.5em;
    margin: 0.175em 0.9em;
    float: left;
}
[id^="general_register_"] {
    text-align: center;
    border-color: #AAAAAA;
    border-style: solid;
}
[id^="register_"][id$="_wrapper"] {
    border: 0.05em #AAAAAA solid;
    padding: 0.15em;
}
[id^="register_"][id$="_wrapper"]:hover > [id^="general_register_"] {
    border-color: #FF4040;
}
/* attributes with ids starting with "register_" AND ending with "_wrapper", but not attributes with id "register_0_wrapper" or "register_9_wrapper" */
[id^="register_"][id$="_wrapper"]:not(#register_0_wrapper, #register_9_wrapper) {
    margin: 0.15em 0;
}

#ALU_wrapper, #accumulator_wrapper, #PC_wrapper, #decode_unit_wrapper, #CIR_wrapper, #MAR_wrapper, #status_register_wrapper, #control_unit_wrapper, #clock_wrapper, #MBR_wrapper {
    text-align: center;
    padding: 0.5em;
    margin: 0.075em 0 0.5em 0;
    flex-grow: 1;
    display: flex;
    flex-direction: column;
}
#ALU, #accumulator, #PC, #decode_unit, #CIR, #MAR, #status_register, #control_unit, #clock, #MBR {
    text-align: center;
    flex-grow: 1;
}
#ALU_wrapper, #MAR_wrapper {
    margin-top: -1.32em;
}
#CIR_wrapper, #MBR_wrapper {
    margin-bottom: 0.18em;
```

```
}

/* change color of leader-lines when hovered over */
/* any events defined in css take precedence over js */
svg.leader-line {
    pointer-events: auto !important;
}
/* for any element with class "leader-line", when hovered, go to the g child element, and then the first use element */
.leader-line:hover g > use:first-child {
    stroke: #FF4040 !important;
}
/* for any element with class "leader-line", when hovered, go to the element with id "#leader-line-x-plug-marker-y", then the g child element, then the first use element */
.leader-line:hover #leader-line-1-plug-marker-0 > g > use:first-child, .leader-line:hover #leader-line-1-plug-marker-1 > g > use:first-child, .leader-line:hover #leader-line-2-plug-marker-0 > g > use:first-child, .leader-line:hover #leader-line-2-plug-marker-1 > g > use:first-child, .leader-line:hover #leader-line-3-plug-marker-0 > g > use:first-child, .leader-line:hover #leader-line-3-plug-marker-1 > g > use:first-child {
    fill: #FF4040 !important;
}

#controlbox_wrapper {
    flex-grow: 1;
    float: left;
    margin-top: 1em;
}
#controlbox_column_1 {
    text-align: center;
    display: flex;
    flex-direction: column;
}
#controlbox_top_two_rows_column_2, #controlbox_bottom_two_rows_column_2 {
    text-align: center;
    display: flex;
    flex-direction: column;
}
#how_to_use_this_resource:hover, #view_instruction_set:hover, #upload_program:hover, #reset_RAM:hover, #load_into_RAM:hover, #run_program:hover, #clear_editor:hover, #step_program:hover {
    border-color: #FF4040 !important;
    border-style: solid;
}

#input_wrapper, #output_wrapper {
    text-align: center;
    padding: 0.5em;
    margin: 0.075em 0 0.5em 0;
    flex-grow: 1;
    display: flex;
    flex-direction: column;
    height: 10em;
}
#input_wrapper {
    margin-top: 0.5em;
}
#output_wrapper {
    margin-bottom: 0.5em;
```

```
}
#input, #output {
    text-align: center;
    flex-grow: 1;
}

#controlbox_row_1_column_2, #controlbox_row_2_column_2, #upload_program_wrapper, #reset_RAM_wrapper, #load_into_RAM_wrapper, #run_program_wrapper {
    display: flex;
    flex-direction: row;
    padding: 0.5em;
    flex-grow: 1;
}
#clear_editor_wrapper, #pre-created_programs_wrapper, #step_program_wrapper {
    display: flex;
    flex-direction: column;
    flex-grow: 1;
}
#upload_program, #reset_RAM, #load_into_RAM, #run_program, #clear_editor, #pre-created_programs, #step_program {
    flex-grow: 1;
    padding: 0.5em;
    margin-top: -0.5em;
}
#upload_program {
    margin-left: -0.5em;
}
#clear_editor {
    margin-left: -0.75em;
    margin-right: 2.5em;
}
#run_program {
    margin-right: -0.5em;
}
#step_program {
    margin-left: 2.5em;
    margin-right: -0.75em;
}
#pre-created_programs {
    margin-left: -3em;
    margin-right: -3em;
    text-align: center;
}
#pre-created_programs:hover {
    border-color: #FF4040 !important;
}
#how_to_use_this_resource_wrapper, #view_instruction_set_wrapper {
    flex-grow: 1;
    padding: 0.5em;
}
#how_to_use_this_resource {
    margin-top: -0.375em;
}

#FDE_output_wrapper, #verbose_output_wrapper {
    height: 22em;
```

```
    display: flex;
    flex-direction: column;
    flex-grow: 1;
    margin-top: 1em;
}
#FDE_output, #verbose_output {
    resize: none;
    flex-grow: 1;
}
#verbose_output_wrapper {
    margin-left: -0.75em;
}

#second_title_text_row {
    margin-top: 0.5em;
    margin-bottom: -1em;
}
#verbose_output_title {
    margin-left: -0.7em;
}
#general_registers_title {
    white-space: nowrap;
    font-size: small;
}

#instruction_set_viewer {
    height: 40em;
}

#clock_speed {
    margin-top: 0.3em;
    display: flex;
    flex-direction: row;
    flex-grow: 1;
    margin-left: -1em;
}

```

~/public/stylesheets/style.css

```
/* main overall attribute stuff */
* {
    box-sizing: border-box;
    padding: 0;
    margin: 0;
}

body {
    font-size: 1em;
    font-family: "Lucida Grande", Helvetica, Arial, sans-serif;
    /* TODO: UNCOMMENT BELOW LINE AFTER DONE TESTING STUFF FOR MAIN VISUALISATION */
    overflow-x: hidden;
}

```

```
.container-fluid {
    margin: 0;
    padding: 0;
}

/* all navbar stuff */
.navbar {
    display: flex;
    flex-flow: row nowrap;
    justify-content: space-around;
    position: relative;
    width: 100%;
    font-size: 1em;
    border-radius: 0.75em;
    padding-top: 0.25em;
    padding-bottom: 0.25em;
    text-align: center;
}

.navbar_wrapper {
    display: flex;
    flex-flow: row nowrap;
    width: 100%;
    justify-content: space-around;
    align-items: center;
    vertical-align: middle;
}

.navbar_left, .navbar_right {
    display: flex;
    flex-flow: row nowrap;
}

.navbar_left {
    justify-content: flex-start;
    align-items: center;
}

.navbar_right {
    margin-left: auto;
    align-items: center;
}

/* so
 is a newline for the data of a tooltip */
.tooltip-inner {
    white-space: pre-wrap;
}

#navbar_user_indicator {
    position: absolute;
    max-height: 1.5em;
    display: flex;
    flex-flow: row nowrap;
    justify-content: flex-start;
```

```
    padding-top: 0.35em;
    padding-bottom: 0.35em;
    padding-left: 0.5em;
}

#navbar_title {
    color: whitesmoke;
    padding-left: 2em;
    padding-right: 1.25em;
}

/* form stuff */
input {
    width: 100%;
}

#logout_form {
    display: none;
}

/* instruction set page */
#instruction_set_table {
    margin-top: 0.25em;
    border: 0.25em double black;
}
#instruction_set_table > tr, th, td {
    text-align: center;
    border: 0.25em double black;
    padding: 1em;
}
#instruction_set_viewer {
    height: 45em;
}

/* how to use this resource page */
.centered_header {
    text-align: center;
}
.sub_header {
    margin-top: 0.25em;
    border: 0.25em double black;
    margin-bottom: 0;
}
#image_explanation_table > tr, th, td {
    text-align: center;
    border: 0.25em double black;
    border-top: 0;
    padding: 1em;
}
#sub_header_2_t, #sub_header_2_s {
    border: 0.25em double black;
    border-top: 0;
    padding-top: 0.25em;
    margin-bottom: 0;
}
```

```
#sub_header_2_t {
    margin-left: 0.14em;
}
#sub_header_2_s {
    border-left: 0;
    margin-right: -0.14em;
}
#pages_info_table {
    table-layout: fixed;
}
#pages_info_table > tr, th, td {
    text-align: center;
    border: 0.25em double black;
    border-top: 0;
    padding: 1em;
}

/* login and sign up pages */
.remember_me_float {
    float: left;
}
.form-group {
    padding: 0.5em;
}
#student_card, #teacher_card {
    margin-top: 7.5%;
}
.switch_to_student:hover {
    background-color: #93c572;
    color: black;
}
.switch_to_teacher:hover {
    background-color: #cccc00;
    color: black;
}

/* teacher challenges pages */
#active_challenges {
    margin-top: 0.25em;
    border: 0.25em double black;
}
#create_challenges_create_btn:hover {
    background-color: #93c572;
    color: black;
}
.create_challenges_edit_btn:hover {
    background-color: #cccc00;
    color: black;
}
.create_challenges_delete_btn:hover {
    background-color: #aa4a44;
}
#challenges_have_been_assigned_table, #challenges_havent_been_assigned_table, #students_have_submitted_table, #students_yet_to_table, #students_missed_table {
    border: 0.25em double black;
```

```
}

/* student challenges pages */
#challenges_todo_table, #completed, #missed {
    border: 0.25em double black;
}

```

~/routes/login_sign_up/login.js

```
var express = require('express');
var router = express.Router();

// set up passport
var passport = require('passport');
var initializePassport = require('../../lib/passport-config');
var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// coming from index page -> login
router.get('/', auth.check_not_authenticated, function (req, res, next) {
    res.locals.message = req.flash();
    res.render('login_sign_up/login', { title: 'Login', menu_id: 'login' });
});

// login form - student
router.post('/student', auth.check_not_authenticated, passport.authenticate('local_student', {
    failureRedirect: '/login',
    failureFlash: true,
    session: true
}), function(req, res) {
    // on successful authentication
    // if remember me box not ticked
    if (typeof req.body.remember_me === "undefined") {
        console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
        res.redirect('/');
    } else {
        req.session.cookie.maxAge = 2592000000; // 30 days
        console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
        res.redirect('/');
    }
});

// login form - teacher
router.post('/teacher', auth.check_not_authenticated, passport.authenticate('local_teacher', {
    failureRedirect: '/login',
    failureFlash: true,
    session: true
}), function (req, res) {
    // on successful authentication
    // if remember me box not ticked
    if (typeof req.body.remember_me === "undefined") {
```

```
45.         console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
46.         res.redirect('/');
47.     } else {
48.         req.session.cookie.maxAge = 2592000000; // 30 days
49.         console.log('login max age (milliseconds): ' + req.session.cookie.maxAge.toString());
50.         res.redirect('/');
51.     }
52. });
53.
54. module.exports = router;
55.
```

~/routes/login_sign_up/sign_up.js

```
 1. var express = require('express');
 2. var router = express.Router();
 3. var passport = require('passport');
 4.
 5. var auth = require('../../lib/auth');
 6. // for connecting to database
 7. var db_connection = require('../../lib/db');
 8.
 9. // for decrypting password
10. var bcrypt = require('bcrypt');
11. const { format } = require('../../lib/db');
12.
13. // coming from index page -> sign_up
14. router.get('/', auth.check_not_authenticated, function (req, res, next) {
15.
16.     try {
17.
18.         // want all class_codes from the Teacher table to fill out the options for the student sign up
19.         db_connection.query(
20.             'SELECT class_code FROM Teacher;',
21.             function (err, rows) {
22.
23.                 if (err) {
24.                     console.log(err);
25.                     req.flash('error', ' An error occured');
26.                     res.locals.message = req.flash();
27.                     res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
28.                 }
29.
30.                 if (rows.length === 0) {
31.                     res.locals.message = req.flash();
32.                     res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
33.                 } else {
34.                     // everything works
35.                     res.locals.message = req.flash();
36.                     res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up', class_codes: rows });
37.                 }
38.             }
39.         );
40.
41.     } catch (error) {
```

```
42.
43.         res.locals.message = req.flash();
44.         res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
45.
46.     }
47. });
48.
49. // sign up form - student
50. router.post('/student', auth.check_not_authenticated, async function(req, res) {
51.     try {
52.
53.         // all user-side scripting for validation
54.         var error_message = false;
55.
56.         // ensure everything filled out
57.         if (req.body.email.length === 0) {
58.             error_message = true;
59.             req.flash('error', ' Email is required');
60.         }
61.         if (req.body.year_group.length === 0) {
62.             error_message = true;
63.             req.flash('error', ' Year Group is required');
64.         }
65.         if (req.body.teacher_initials.length === 0) {
66.             error_message = true;
67.             req.flash('error', ' Class is required');
68.         }
69.         if (req.body.password.length === 0) {
70.             error_message = true;
71.             req.flash('error', ' Password is required');
72.         }
73.         if (req.body.password_confirmation.length === 0) {
74.             error_message = true;
75.             req.flash('error', ' Confirm password is required');
76.         }
77.
78.         // check email matches regex for student email
79.         var regex_student = /^[A-Za-z]+\d{4}\@dubaicollege.org$/;
80.         if (req.body.email.match(regex_student) === null) {
81.             error_message = true;
82.             req.flash('error', ' Invalid email format (DC student email required)');
83.         }
84.
85.         // ensure a teacher has signed up
86.         if (req.body.teacher_initials === 'no_teacher') {
87.             error_message = true;
88.             req.flash('error', ' A teacher needs to sign up first');
89.         }
90.
91.         // ensure password length is >= 8 characters
92.         if (req.body.password.length < 8) {
93.             error_message = true;
94.             req.flash('error', ' Password length must be at least 8 characters');
95.         }
96.
```

```
    // ensure password contains at least 1 alphabet, 1 number, and 1 special character
    var special_chars = /[ `!@#$%^&*()_+\-=\[\]{};':"\\|,.<>\/?~]/; // regex for special characters
    var alphabet_chars = /[a-zA-Z]/; // regex for alphabets
    var number_chars = /\d/; // regex for digit
    if (!(special_chars.test(req.body.password) && alphabet_chars.test(req.body.password) &&
number_chars.test(req.body.password))) {
      error_message = true;
      req.flash('error', ' Password must contain at least 1 alphabet, 1 integer, and 1 special character');
    }

    // ensure confirm password is the same as password
    if (req.body.password !== req.body.password_confirmation) {
      error_message = true;
      req.flash('error', ' Confirm password does not match password');
    }

    // return with the error message
    if (error_message) {
      res.locals.message = req.flash();
      return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
    }

    // all server-side scripting for validation

    // ensure email is not already being used
    db_connection.query(
      'SELECT * FROM Student WHERE student_email = ?;', [req.body.email],
      function (err, rows) {
        if (!rows || rows.length <= 0) {
          // good, do not want a result
          // pass - nothing needs to happen
        } else {
          // bad, 1 email cannot be used multiple times
          req.flash('error', ' Email is already being used');
          res.locals.message = req.flash();
          return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
        }
      }
    );

    // validation complete, so add all data to the database
    var hashed_password = await bcrypt.hash(req.body.password, 10);
    db_connection.query(
      'INSERT INTO Student (student_email, student_name, student_number, student_password) VALUES (?, ?, ?, ?);',
      [req.body.email, req.body.email.split('@')[0].slice(0, -4), req.body.email.split('@')[0].substr(-4), hashed_password],
      function (err, rows) {
        if (err) {
          console.log(err);
          req.flash('error', ' An error occured');
          res.locals.message = req.flash();
          return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
        }
        console.log(rows);
      }
    );
```

```
    // cross-parameterised SQL (tells MySQL exactly where the parameters are supposed to go (where the ? marks are))
    db_connection.query(
      'INSERT INTO Students_In_Classes (student_id, class_id) VALUES ((SELECT student_id FROM Student WHERE student_email = ?), (SELECT class_id FROM Class WHERE year_group = ? AND teacher_id = (SELECT teacher_id FROM Teacher WHERE class_code = ?)));',
      [req.body.email, req.body.year_group, req.body.teacher_initials],
      function (err, rows) {
        if (err) {
          console.log(err);
          req.flash('error', ' An error occured');
          res.locals.message = req.flash();
          return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
        }
        console.log(rows);
      }
    );

    return res.redirect('/login');

  } catch {
    return res.redirect('/sign_up');
  }
});

// sign up form - teacher
router.post('/teacher', auth.check_not_authenticated, async function(req, res) {
  try {

    // all user-side scripting for validation
    var error_message = false;

    // ensure everything filled out
    if (req.body.email.length === 0) {
      error_message = true;
      req.flash('error', ' Email is required');
    }
    if (req.body.teacher_initials.length === 0) {
      error_message = true;
      req.flash('error', ' Class code is required');
    }
    if (req.body.password.length === 0) {
      error_message = true;
      req.flash('error', ' Password is required');
    }
    if (req.body.password_confirmation.length === 0) {
      error_message = true;
      req.flash('error', ' Confirm password is required');
    }

    // check email matches regex for teacher email
    var regex_teacher = /^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$/;
    if (req.body.email.match(regex_teacher) === null) {
      error_message = true;
      req.flash('error', ' Invalid email format (DC teacher email required)');
    }
```

```
204.
205.        // ensure class code is 3 letters
206.        var regex_class_code = /^[A-Z]{3}$/;
207.        if (!(req.body.teacher_initials.match(regex_class_code))) {
208.          error_message = true;
209.          req.flash('error', ' Invalid class code format (3 capital letters)');
210.        }
211.
212.        // ensure password length is >= 8 characters
213.        if (req.body.password.length < 8) {
214.          error_message = true;
215.          req.flash('error', ' Password length must be at least 8 characters');
216.        }
217.
218.        // ensure password contains at least 1 alphabet, 1 number, and 1 special character
219.        var special_chars = /[ `!@#$%^&*()_+\-=\[\]{};':"\\|,.<>\/?~]/; // regex for special characters
220.        var alphabet_chars = /[a-zA-Z]/; // regex for alphabets
221.        var number_chars = /\d/; // regex for digit
222.        if (!(special_chars.test(req.body.password) && alphabet_chars.test(req.body.password) &&
number_chars.test(req.body.password))) {
223.          error_message = true;
224.          req.flash('error', ' Password must contain at least 1 alphabet, 1 integer, and 1 special character');
225.        }
226.
227.        // ensure confirm password is the same as password
228.        if (req.body.password !== req.body.password_confirmation) {
229.          error_message = true;
230.          req.flash('error', ' Confirm password does not match password');
231.        }
232.
233.        // return with the error message
234.        if (error_message) {
235.          res.locals.message = req.flash();
236.          return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
237.        }
238.
239.        // all server-side scripting for validation
240.
241.        // ensure email is not already being used
242.        db_connection.query(
243.          'SELECT * FROM Teacher WHERE teacher_email = ?;', [req.body.email],
244.          function (err, rows) {
245.            if (!rows || rows.length <= 0) {
246.              // good, do not want a result
247.              // pass - nothing needs to happen
248.            } else {
249.              // bad, 1 email cannot be used multiple times
250.              req.flash('error', ' Email is already being used');
251.              res.locals.message = req.flash();
252.              return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
253.            }
254.          }
255.        );
256.
257.        // ensure class code is not already being used
```

```
    db_connection.query(
      'SELECT * FROM Teacher WHERE class_code = ?;', [req.body.teacher_initials],
      function (err, rows) {
        if (!rows || rows.length <= 0) {
          // good, do not want a result
          // pass - nothing needs to happen
        } else {
          // bad, 1 email cannot be used multiple times
          req.flash('error', ' Class code is already being used');
          res.locals.message = req.flash();
          return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
        }
      }
    );

    // validation complete, so add all data to the database
    // add teacher info to Teacher table
    var hashed_password = await bcrypt.hash(req.body.password, 10);
    db_connection.query(
      'INSERT INTO Teacher (teacher_email, teacher_first_name, teacher_last_name, class_code, teacher_password) VALUES (?, ?, ?, ?, ?);',
      [req.body.email, req.body.email.split('@')[0].split('.')[0].charAt(0).toUpperCase() + req.body.email.split('@')[0].split('.')[0].slice(1), req.body.email.split('@')[0].split('.')[1].charAt(0).toUpperCase() + req.body.email.split('@')[0].split('.')[1].slice(1), req.body.teacher_initials, hashed_password],
      function (err, rows) {
        if (err) {
          console.log(err);
          req.flash('error', ' An error occured');
          res.locals.message = req.flash();
          return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
        }
        console.log(rows);
      }
    );
    // add 4 new classes (year 10-13 inclusive), with the teacher
    for (var year = 10; year <= 13; year++) {
      db_connection.query(
        'INSERT INTO Class (year_group, teacher_id) VALUES (?, (SELECT teacher_id FROM Teacher WHERE teacher_email = ?));',
        [year, req.body.email],
        function (err, rows) {
          if (err) {
            console.log(err);
            req.flash('error', ' An error occured');
            res.locals.message = req.flash();
            return res.render('login_sign_up/sign_up', { title: 'Sign Up', menu_id: 'sign_up' });
          }
          console.log(rows);
        }
      );
    }

    return res.redirect('/login');

  } catch {
```

```
        return res.redirect('/sign_up');
    }
});

module.exports = router;

```

~/routes/student_challenges/view_set_challenges.js

```
var express = require('express');
var router = express.Router();

var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// for writing to files
var fs = require('fs');

// for array/list operations
var _ = require('lodash');

// for DATETIME conversions
var moment = require('moment');
moment.locale('en-gb');
moment.defaultFormat = 'L LT';
var utc_datetime = moment.utc().format();

// coming to create_challenge page
router.get('/', auth.check_authenticated, async function (req, res, next) {

    console.log(req.user.id);

    try {

        res.locals.message = req.flash();

        await get_all_solved_unsolved_challenges(req, res, next, true, req.user.id).then((rows) => {

            var all_completed_challenges = [];

            // from db to list
            rows.forEach(element => {
                all_completed_challenges.push([element['challenge_blob'].toString(), element['assigned_challenge_id'],
moment(element['due_date']).utc(utc_datetime).local().format(),
moment(element['completion_date']).utc(utc_datetime).local().format(), element['solution_blob'].toString()]);
            });

            console.log('all_completed_challenges');
            console.log(all_completed_challenges);

            return {
                'all_completed_challenges': all_completed_challenges
            };
```

```

    }).then(async (useful_vars) => {

        await get_all_unsolved_due_before_after_curr_date_challenges(req, res, next, true, req.user.id).then((rows) => {

            var all_missed_challenges = [];

            // from db to list
            rows.forEach(element => {
                all_missed_challenges.push([element['challenge_blob'].toString(), element['assigned_challenge_id'], moment(element['due_date']).utc(utc_datetime).local().format()]);
            });

            console.log('all_missed_challenges');
            console.log(all_missed_challenges);

            useful_vars['all_missed_challenges'] = all_missed_challenges;

        });

        return useful_vars;

    }).then(async (useful_vars) => {

        await get_all_unsolved_due_before_after_curr_date_challenges(req, res, next, false, req.user.id).then((rows) => {

            var all_challenges_to_do = [];

            // from db to list
            rows.forEach(element => {
                all_challenges_to_do.push([element['challenge_blob'].toString(), element['assigned_challenge_id'], moment(element['due_date']).utc(utc_datetime).local().format()]);
            });

            console.log('all_challenges_to_do');
            console.log(all_challenges_to_do);

            useful_vars['all_challenges_to_do'] = all_challenges_to_do;

        });

        return useful_vars;

    }).then((useful_vars) => {

        useful_vars['all_challenges_to_do'].forEach(async (element, index) => {

            var route_complete = '/complete_challenge/' + element[1];

            router.post(route_complete, function (req, res, next) {

                res.locals.message = req.flash();

                res.render('student_challenges/complete_challenge', {
                    title: 'Complete Challenge',
```

```
                role: req.user.role,
                email: req.user.email,
                 session_id: req.sessionID,
                 session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                 challenge_title: element[0].split('\n')[1].toString(),
                 challenge_description: element[0].split('\n')[3].toString(),
                 assigned_challenge_id: element[1].toString()
              });

           });

        });

        useful_vars['all_completed_challenges'].forEach(async (element, index) => {

           var route_view_completed = '/view_completed/' + element[1];

           router.post(route_view_completed, async function (req, res, next) {

              res.locals.message = req.flash();

              res.render('student_challenges/view_completed_challenge', {
                 title: 'View Completed Challenge',
                 role: req.user.role,
                 email: req.user.email,
                 session_id: req.sessionID,
                 session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                 challenge_title: element[0].split('\n')[1].toString(),
                 challenge_description: element[0].split('\n')[3].toString(),
                 challenge_answer: element[4].toString()
              });

           });

        });

        return res.render('student_challenges/view_set_challenges', {
           title: 'View Set Challenges',
           menu_id: 'view_set_challenges',
           role: req.user.role,
           email: req.user.email,
           all_completed_challenges: useful_vars['all_completed_challenges'],
           all_missed_challenges: useful_vars['all_missed_challenges'],
           all_challenges_to_do: useful_vars['all_challenges_to_do'],
           session_id: req.sessionID,
           session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });

     });

  } catch (any_error) {
     console.log(any_error);
     res.render('student_challenges/view_set_challenges', {
        title: 'View Set Challenges',
        menu_id: 'view_set_challenges',
```

```
153.          role: req.user.role,
154.          email: req.user.email,
155.          session_id: req.sessionID,
156.          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
157.        });
158.    }
159.
160. });
161.
162. var get_all_unsolved_due_before_after_curr_date_challenges = (req, res, next, before, user_id) => {
163.    return new Promise((resolve, reject) => {
164.        try {
165.
166.          // get all challenges that were due before the current date - i.e. missed
167.          if (before) {
168.            var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE
Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id =
Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND
Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student,
Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) AND
Assigned_Challenges.due_date < NOW() ORDER BY Assigned_Challenges.due_date ASC;';
169.          } else { // all challenges that can still be solved by the student, the due date is after the current time
170.            var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE
Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id =
Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND
Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student,
Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) AND
Assigned_Challenges.due_date >= NOW() ORDER BY Assigned_Challenges.due_date ASC;';
171.          };
172.
173.          db_connection.query(
174.            query_to_run,
175.            [user_id],
176.            function (err, rows) {
177.
178.              if (err) {
179.                console.log(err);
180.                req.flash('error', ' An error occured');
181.                res.locals.message = req.flash();
182.                return res.render('student_challenges/view_set_challenges', {
183.                  title: 'View Set Challenges',
184.                  menu_id: 'view_set_challenges',
185.                  role: req.user.role,
186.                  email: req.user.email,
187.                  session_id: req.sessionID,
188.                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
189.                });
190.                reject(err);
191.              };
192.
193.              resolve(rows);
194.            }
195.          );
196.
197.        } catch (error) {
```

```
        reject(error);
      }
    });
};
var get_all_solved_unsolved_challenges = (req, res, next, solved, user_id) => {
  return new Promise((resolve, reject) => {
    try {

        if (solved) {
          var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges, Solution_Student, Solution_File WHERE Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id = Assigned_Challenges.challenge_teacher_id AND Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id AND Solution_File.solution_file_id = Solution_Student.solution_file_id AND Assigned_Challenges.student_id = ? AND Assigned_Challenges.assigned_challenge_id IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student, Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) ORDER BY Assigned_Challenges.due_date ASC;';
        } else {
          var query_to_run = 'SELECT * FROM Challenge_File, Challenge_Teacher, Assigned_Challenges WHERE Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.challenge_teacher_id = Assigned_Challenges.challenge_teacher_id AND Assigned_Challenges.student_id = ? AND Assigned_Challenges.assigned_challenge_id NOT IN (SELECT Solution_Student.assigned_challenge_id FROM Solution_Student, Assigned_Challenges WHERE Solution_Student.assigned_challenge_id = Assigned_Challenges.assigned_challenge_id) ORDER BY Assigned_Challenges.due_date ASC;';
        };

        db_connection.query(
          query_to_run,
          [user_id],
          function (err, rows) {

            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              return res.render('student_challenges/view_set_challenges', {
                title: 'View Set Challenges',
                menu_id: 'view_set_challenges',
                role: req.user.role,
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
              });
              reject(err);
            };

            resolve(rows);
          }
        );

    } catch (error) {
        reject(error);
    }
  });
};
```

```
241.
242. // create new challenge and put in database
243. router.post('/complete_challenge/complete', async function (req, res, next) {
244.
245.    console.log('attempted completion');
246.
247.    try {
248.
249.       var complete_challenge_file_path = './public/text_files/student_complete_challenge.txt';
250.
251.       // check if file exists
252.
253.       // ASYNCHRONOUS
254.       // fs.access(complete_challenge_file_path, fs.F_OK, (err) => {
255.       //  if (err) {
256.       //      console.error(err);
257.       //      return;
258.       //  };
259.       //  // file exists
260.       //  console.log('Teacher Create Challenge file already exists');
261.       // });
262.
263.       // SYNCHRONOUS
264.       try {
265.          if (fs.existsSync(complete_challenge_file_path)) {
266.             // file exists
267.             console.log('Student Complete Challenge file already exists');
268.
269.             try {
270.                fs.unlinkSync(complete_challenge_file_path);
271.                // file removed
272.                console.log('Student Complete Challenge file deleted');
273.             } catch (error) {
274.                console.error(error);
275.             };
276.          }
277.       } catch (err) {
278.          console.error(err);
279.       };
280.
281.       console.log('student_complete_challenge.txt updated');
282.
283.       // all user-side scripting for validation
284.       var error_message = false;
285.
286.       // ensure everything filled out
287.       if (req.body.challenge_answer.length === 0) {
288.          error_message = true;
289.          req.flash('error', ' Challenge Answer is required');
290.       }
291.
292.       // return with the error message
293.       if (error_message) {
294.          res.locals.message = req.flash();
295.          return res.render('student_challenges/view_set_challenges', {
```

```
            title: 'View Set Challenges',
            menu_id: 'view_set_challenges',
            role: req.user.role,
            email: req.user.email,
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
          });
        }

        var file_data = req.body.challenge_answer.toString();
        fs.writeFileSync(complete_challenge_file_path, file_data,
          // callback function that is called after writing file is done
          function (err) {
            if (err) {
              console.log(err);
            };
            console.log('Data written to Student Complete Challenge file successfully')
          });

        var data_to_db = fs.readFileSync(complete_challenge_file_path);

        console.log('before query 1');

        // put file as MEDIUMBLOB in db
        await insert_solution_blob(req, res, next, data_to_db).then(async (rows) => {
          console.log('query 1 done');
          await insert_solution_student(req, res, next, data_to_db).then(() => {
            console.log('query 2 done');
            // redirect to page to create a new challenge
            return res.redirect('/view_set_challenges');
          });
        });

      } catch (any_error) {
        res.locals.message = req.flash();
        return res.render('student_challenges/view_set_challenges', {
          title: 'View Set Challenges',
          menu_id: 'view_set_challenges',
          role: req.user.role,
          email: req.user.email,
          session_id: req.sessionID,
          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });
      }

  });
var insert_solution_blob = (req, res, next, data_to_db) => {
  return new Promise((resolve, reject) => {
    try {

      db_connection.query(
        'INSERT INTO Solution_File (solution_blob) VALUES (?);',
        [data_to_db],
        function (err, rows) {
          if (err) {
```

```
                console.log(err);
                req.flash('error', ' An error occured');
                res.locals.message = req.flash();
                return res.render('student_challenges/view_set_challenges', {
                  title: 'View Set Challenges',
                  menu_id: 'view_set_challenges',
                  role: req.user.role,
                  email: req.user.email,
                  session_id: req.sessionID,
                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                });
                reject(err);
              }
              console.log(rows);
              resolve(rows);
            }
          );

      } catch (error) {
        reject(error);
      }
    });
};
var insert_solution_student = (req, res, next, data_to_db) => {
  return new Promise((resolve, reject) => {
    try {

      db_connection.query(
        'INSERT INTO Solution_Student (solution_file_id, assigned_challenge_id, completion_date) VALUES ((SELECT solution_file_id FROM Solution_File WHERE solution_blob = ?), ?, NOW());',
        [data_to_db, parseInt(req.body.assigned_challenge_id)],
        function (err, rows) {
          if (err) {
            console.log(err);
            req.flash('error', ' An error occured');
            res.locals.message = req.flash();
            return res.render('student_challenges/view_set_challenges', {
              title: 'View Set Challenges',
              menu_id: 'view_set_challenges',
              role: req.user.role,
              email: req.user.email,
              session_id: req.sessionID,
              session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            });
            reject(err);
          }
          console.log(rows);
          resolve(rows);
        }
      );

    } catch (error) {
      reject(error);
    }
  });
```

```
};

module.exports = router;

```

~/routes/teacher_challenges/create_challenges.js

```
var express = require('express');
var router = express.Router();

var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// for writing to files
var fs = require('fs');

// for array/list operations
var _ = require('lodash');

// coming to create_challenge page
router.get('/', auth.check_authenticated, async function (req, res, next) {

  try {

    res.locals.message = req.flash();

    await get_challenge_txt_and_id(req, res, next).then((row_challenge_txt_id) => {

      var challenges_to_display = [];
      // var challenges_only = [];

      row_challenge_txt_id.forEach(element => {
        challenges_to_display.push([element['challenge_blob'].toString(), element['challenge_teacher_id']]);
        // challenges_only.push(element['challenge_blob'].toString());
      });

      var challenge_title_match_id = [];

      challenges_to_display.forEach(element => {
        // populate list of [challenge_title, challenge_teacher_id]
        challenge_title_match_id.push([element[0].split('\n')[1], element[1]]);
      });

      return {
        'challenges_to_display': challenges_to_display,
        // 'challenges_only': challenges_only,
        'challenge_title_match_id': challenge_title_match_id
      };

    }).then(async (useful_vars) => {

      // store 2d list of [class (year + teacher id, e.g. 10RAW), class_id]
```

```
        var classes = [];

        await get_classes_of_teacher(req, res, next).then((row_classes_of_teacher) => {
          row_classes_of_teacher.forEach((element, index) => {
            classes.push([element['CONCAT(year_group, class_code)'], element['class_id']]);
          });
        });

        useful_vars['classes'] = classes;

        return useful_vars;

      }).then(async (useful_vars) => {

        await get_students_in_teacher_classes(req, res, next, useful_vars).then((row_class_students) => {

          var list_of_class_students = [];

          useful_vars['classes'].forEach((element, index) => {
            list_of_class_students.push([useful_vars['classes'][index][0], useful_vars['classes'][index][1],
row_class_students[index][1]]);
          });

          useful_vars['list_of_class_students'] = list_of_class_students;

        });

        console.log('challenges_to_display');
        console.log(useful_vars['challenges_to_display']);
        console.log('challenge_title_match_id');
        console.log(useful_vars['challenge_title_match_id']);
        console.log('classes');
        console.log(useful_vars['classes']);
        console.log('list_of_class_students');
        console.log(useful_vars['list_of_class_students']);

        return useful_vars;

      }).then(async (useful_vars) => {

        useful_vars['challenges_to_display'].forEach(async (element, index) => {

          // delete route for each challenge using the challenge_teacher_id
          var route_delete = '/delete/' + element[1];

          router.post(route_delete, async function (req, res, next) {

            await delete_challenge(req, res, next, element).then(() => {
              return res.redirect('/create_challenges');
            });

          });

          // edit route for each challenge using the challenge_teacher_id
          var route_edit = '/edit/' + element[1];
```

```

            router.post(route_edit, async function (req, res, next) {

              res.locals.message = req.flash();

              res.render('teacher_challenges/edit_challenge', {
                title: 'Edit Challenge',
                role: req.user.role,
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                challenge_title: element[0].split('\n')[1].toString(),
                challenge_description: element[0].split('\n')[3].toString(),
                challenge_teacher_id: element[1].toString(),
              });

            });

          });

          res.render('teacher_challenges/create_challenges', {
            title: 'Create Challenges',
            menu_id: 'create_challenges',
            role: req.user.role,
            email: req.user.email,
            // challenge_titles: challenge_titles,
            // challenge_descriptions: challenge_descriptions,
            challenges_to_display: useful_vars['challenges_to_display'],
            challenge_title_match_id: useful_vars['challenge_title_match_id'],
            class_students: useful_vars['list_of_class_students'],
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
          });

        });

      } catch (any_error) {
        console.log(any_error);
        res.render('teacher_challenges/create_challenges', {
          title: 'Create Challenges',
          menu_id: 'create_challenges',
          role: req.user.role,
          email: req.user.email,
          session_id: req.sessionID,
          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });
      }

  });

var get_challenge_txt_and_id = (req, res, next) => {
  return new Promise((resolve, reject) => {
    try {

      // get challenge text from teacher_id
```

```
        db_connection.query(
          'SELECT * FROM Challenge_File, Challenge_Teacher WHERE Challenge_File.challenge_file_id = Challenge_Teacher.challenge_file_id AND Challenge_Teacher.teacher_id = ?;',
          [req.user.id],
          function (err, rows) {

            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              reject(err);
            };

            resolve(rows);

          }
        );

      } catch (error) {
        reject(error);
      }
    });
};

var get_classes_of_teacher = (req, res, next) => {
  return new Promise((resolve, reject) => {
    try {

      // get classes of currently signed in teacher
      db_connection.query(
        'SELECT CONCAT(year_group, class_code), class_id FROM Class, Teacher WHERE Class.teacher_id = Teacher.teacher_id AND Class.teacher_id = ?;',
        [req.user.id],
        function (err, rows) {

          if (err) {
            console.log(err);
            req.flash('error', ' An error occured');
            res.locals.message = req.flash();
            return res.redirect('/create_challenges');
            reject(err)
          };

          resolve(rows);

        }
      );

    } catch (error) {
      reject(error);
    }
  });
};

var get_students_in_teacher_classes = async (req, res, next, useful_vars) => {
```

```
  return new Promise(async (resolve, reject) => {
    try {

      // 2/3d list of [[class_id, [student1, student2, etc.]]]
      var class_students = [];

      await useful_vars['classes'].forEach((list_element, list_index) => {

        get_student_in_specific_class(req, res, next, list_element[1]).then((row_students) => {

          var students = [];

          row_students.forEach((element, index) => {
            students.push(element['CONCAT(student_name, student_number)']);
          });

          class_students.push([list_element[1], students]);

          return class_students;

        }).then((class_students) => {
          if (list_index == useful_vars['classes'].length - 1) {
            resolve(class_students);
          };
        });

      });

    } catch (error) {
      reject(error);
    }
  });
};
var get_student_in_specific_class = async (req, res, next, teacher_class) => {
  return new Promise((resolve, reject) => {
    try {

      // get classes of currently signed in teacher
      db_connection.query(
        'SELECT CONCAT(student_name, student_number) FROM Student, Students_In_Classes WHERE Student.student_id = Students_In_Classes.student_id AND class_id = ?;',
        [teacher_class],
        function (err, rows) {

          if (err) {
            console.log(err);
            req.flash('error', ' An error occured');
            res.locals.message = req.flash();
            return res.redirect('/create_challenges');
            reject(err)
          };

          resolve(rows);

        }
```

```
            );

        } catch (error) {
            reject(error);
        }
    });
};

var delete_challenge = (req, res, next, challenges_to_display_element) => {
    return new Promise((resolve, reject) => {
        try {

            // delete challenge
            db_connection.query(
                'DELETE FROM Challenge_File WHERE challenge_file_id = (SELECT challenge_file_id FROM Challenge_Teacher WHERE challenge_teacher_id = ?);',
                [challenges_to_display_element[1]],
                function (err, rows) {

                    if (err) {
                        console.log(err);
                        req.flash('error', ' An error occured');
                        res.locals.message = req.flash();
                        return res.redirect('/create_challenges');
                        reject(err)
                    };

                    resolve(rows);

                }
            );

        } catch (error) {
            reject(error);
        }
    });
};

// redirect to new_challenge page creation - testing stuffs
router.post('/new_challenge', function (req, res, next) {

    // code stuffs here
    console.log('new_challenge button clicked');

    // redirect to page to create a new challenge
    res.redirect('/create_challenges/new_challenge');

});

// coming to create_challenge page
router.get('/new_challenge', auth.check_authenticated, function (req, res, next) {

    res.locals.message = req.flash();

    res.render('teacher_challenges/new_challenge', {
```

```
318.        title: 'New Challenge',
319.        role: req.user.role,
320.        email: req.user.email,
321.        session_id: req.sessionID,
322.        session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
323.      });
324.
325. });
326.
327. // create new challenge and put in database
328. router.post('/new_challenge/create', function (req, res, next) {
329.
330.      try {
331.
332.        var create_challenge_file_path = './public/text_files/teacher_create_challenge.txt';
333.
334.        // check if file exists
335.
336.        // ASYNCHRONOUS
337.        // fs.access(create_challenge_file_path, fs.F_OK, (err) => {
338.        //  if (err) {
339.        //      console.error(err);
340.        //      return;
341.        //  };
342.        //  // file exists
343.        //  console.log('Teacher Create Challenge file already exists');
344.        // });
345.
346.        // SYNCHRONOUS
347.        try {
348.          if (fs.existsSync(create_challenge_file_path)) {
349.            // file exists
350.            console.log('Teacher Create Challenge file already exists');
351.
352.            try {
353.              fs.unlinkSync(create_challenge_file_path);
354.              // file removed
355.              console.log('Teacher Create Challenge file deleted');
356.            } catch (error) {
357.              console.error(error);
358.            };
359.          }
360.        } catch (err) {
361.          console.error(err);
362.        };
363.
364.        // all user-side scripting for validation
365.        var error_message = false;
366.
367.        // ensure everything filled out
368.        if (req.body.challenge_title.length === 0) {
369.          error_message = true;
370.          req.flash('error', ' Challenge Title is required');
371.        }
372.        if (req.body.challenge_description.length === 0) {
```

```
        error_message = true;
        req.flash('error', ' Challenge Description is required');
    }

    // ensure challenge title length is <= 100 characters
    if (req.body.challenge_title.length > 100) {
        error_message = true;
        req.flash('error', ' Challenge Title length must be max 100 characters');
    }

    // return with the error message
    if (error_message) {
        res.locals.message = req.flash();
        return res.render('teacher_challenges/new_challenge', {
            title: 'New Challenge',
            role: req.user.role,
            email: req.user.email,
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });
    }

    var file_data = '###CHALLENGE TITLE###\n' + req.body.challenge_title.toString() + '\n###CHALLENGE
DESCRIPTION###\n' + req.body.challenge_description.toString();
    fs.writeFileSync(create_challenge_file_path, file_data,
    // callback function that is called after writing file is done
    function (err) {
        if (err) {
            console.log(err);
        };
        console.log('Data written to Teacher Create Challenge file successfully')
    });

    var data_to_db = fs.readFileSync(create_challenge_file_path);

    // put file as MEDIUMBLOB in db
    db_connection.query(
        'INSERT INTO Challenge_File (challenge_blob) VALUES (?);',
        [data_to_db],
        function (err, rows) {
            if (err) {
                console.log(err);
                req.flash('error', ' An error occured');
                res.locals.message = req.flash();
                return res.render('teacher_challenges/new_challenge', {
                    title: 'New Challenge',
                    role: req.user.role,
                    email: req.user.email,
                    session_id: req.sessionID,
                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                });
            }
            console.log(rows);
        }
    );
```

```
427.
428.        // record relation b/w teacher and challenge
429.        db_connection.query(
430.          'INSERT INTO Challenge_Teacher (challenge_file_id, teacher_id) VALUES ((SELECT challenge_file_id FROM
Challenge_File WHERE challenge_blob = ?), ?);',
431.          [data_to_db, req.user.id],
432.          function (err, rows) {
433.            if (err) {
434.              console.log(err);
435.              req.flash('error', ' An error occured');
436.              res.locals.message = req.flash();
437.              return res.render('teacher_challenges/new_challenge', {
438.                title: 'New Challenge',
439.                role: req.user.role,
440.                email: req.user.email,
441.                session_id: req.sessionID,
442.                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
443.              });
444.            }
445.            console.log(rows);
446.          }
447.        );
448.
449.        // redirect to page to create a new challenge
450.        return res.redirect('/create_challenges');
451.
452.      } catch (any_error) {
453.        res.locals.message = req.flash();
454.        return res.render('teacher_challenges/new_challenge', {
455.          title: 'New Challenge',
456.          role: req.user.role,
457.          email: req.user.email,
458.          session_id: req.sessionID,
459.          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
460.        });
461.      }
462.
463. });
464.
465. // update the database for edited challenge
466. router.post('/edit/update', function (req, res, next) {
467.
468.      try {
469.
470.        var edit_challenge_file_path = './public/text_files/teacher_edit_challenge.txt';
471.
472.        // check if file exists
473.
474.        // ASYNCHRONOUS
475.        // fs.access(create_challenge_file_path, fs.F_OK, (err) => {
476.        //  if (err) {
477.        //      console.error(err);
478.        //      return;
479.        //  };
480.        //  // file exists
```

```
481.    //  console.log('Teacher Create Challenge file already exists');
482.    // });
483.
484.    // SYNCHRONOUS
485.    try {
486.      if (fs.existsSync(edit_challenge_file_path)) {
487.        // file exists
488.        console.log('Teacher Edit Challenge file already exists');
489.
490.        try {
491.          fs.unlinkSync(edit_challenge_file_path);
492.          // file removed
493.          console.log('Teacher Edit Challenge file deleted');
494.        } catch (error) {
495.          console.error(error);
496.        };
497.      }
498.    } catch (err) {
499.      console.error(err);
500.    };
501.
502.    // all user-side scripting for validation
503.    var error_message = false;
504.
505.    // ensure everything filled out
506.    if (req.body.challenge_title.length === 0) {
507.      error_message = true;
508.      req.flash('error', ' Challenge Title is required');
509.    }
510.    if (req.body.challenge_description.length === 0) {
511.      error_message = true;
512.      req.flash('error', ' Challenge Description is required');
513.    }
514.
515.    // ensure challenge title length is <= 100 characters
516.    if (req.body.challenge_title.length > 100) {
517.      error_message = true;
518.      req.flash('error', ' Challenge Title length must be max 100 characters');
519.    }
520.
521.    // return with the error message
522.    if (error_message) {
523.      res.locals.message = req.flash();
524.      res.render('teacher_challenges/edit_challenge', {
525.        title: 'Edit Challenge',
526.        role: req.user.role,
527.        email: req.user.email,
528.        session_id: req.sessionID,
529.        session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
530.        challenge_title: req.body.challenge_title_og.toString(),
531.        challenge_description: req.body.challenge_description_og.toString(),
532.        challenge_teacher_id: req.body.challenge_teacher_id.toString(),
533.      });
534.    }
535.
```

```
    var file_data = '###CHALLENGE TITLE###\n' + req.body.challenge_title.toString() + '\n###CHALLENGE
DESCRIPTION###\n' + req.body.challenge_description.toString();
    fs.writeFileSync(edit_challenge_file_path, file_data,
      // callback function that is called after writing file is done
      function (err) {
        if (err) {
          console.log(err);
        };
        console.log('Data written to Teacher Create Challenge file successfully')
      });

    var data_to_db = fs.readFileSync(edit_challenge_file_path);

    // put file as MEDIUMBLOB in db
    db_connection.query(
      'UPDATE Challenge_File SET challenge_blob = ? WHERE challenge_file_id = (SELECT challenge_file_id FROM
Challenge_Teacher WHERE challenge_teacher_id = ?);',
      [data_to_db, req.body.challenge_teacher_id],
      function (err, rows) {
        if (err) {
          console.log(err);
          req.flash('error', ' An error occured');
          res.locals.message = req.flash();
          return res.render('teacher_challenges/edit_challenge', {
            title: 'Edit Challenge',
            role: req.user.role,
            email: req.user.email,
            session_id: req.sessionID,
            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            challenge_title: req.body.challenge_title_og.toString(),
            challenge_description: req.body.challenge_description_og.toString(),
            challenge_teacher_id: req.body.challenge_teacher_id.toString(),
          });
        }
        console.log(rows);
      }
    );

    // redirect to page to create a new challenge
    return res.redirect('/create_challenges');

  } catch (any_error) {
    res.locals.message = req.flash();
    return res.render('teacher_challenges/edit_challenge', {
      title: 'Edit Challenge',
      role: req.user.role,
      email: req.user.email,
      session_id: req.sessionID,
      session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
      challenge_title: req.body.challenge_title_og.toString(),
      challenge_description: req.body.challenge_description_og.toString(),
      challenge_teacher_id: req.body.challenge_teacher_id.toString(),
    });
  }

```

```
589. });
590.
591. // assign challenges to students/classes
592. router.post('/assign', async function (req, res, next) {
593.
594.    console.log('assign button clicked');
595.
596.    var challenges_to_display_temp = req.body.challenges_to_display.toString().split('###CHALLENGE
TITLE###').map(element => element.replace(/[\r]/gm, '')).filter(element => element !== '');
597.    // all vars to keep challenges + students updated
598.    var challenges_to_display = [];
599.    challenges_to_display_temp.forEach((element, index) => {
600.       if (index !== challenges_to_display_temp.length - 1) {
601.          challenges_to_display.push(['###CHALLENGE TITLE###' + element.slice(0, - 2 - element.split(',').reverse()[1].length),
parseInt(element.split(',').reverse()[1])]);
602.       } else {
603.          challenges_to_display.push(['###CHALLENGE TITLE###' + element.slice(0, - 1 - element.split(',').reverse()[0].length),
parseInt(element.split(',').reverse()[0])]);
604.       };
605.    });
606.
607.    var challenge_title_match_id = req.body.challenge_title_match_id.filter(element => element !== '')[0].split(',');
608.    challenge_title_match_id.forEach((element, index) => {
609.       if (index % 2 === 0) {
610.          // console.log(element, index)
611.          challenge_title_match_id[index] = [element.toString(), parseInt(challenge_title_match_id[index + 1])];
612.       } else {
613.          challenge_title_match_id[index] = '';
614.       };
615.    });
616.    challenge_title_match_id = challenge_title_match_id.filter(element => element !== '');
617.
618.    var class_students = req.body.class_students.split('\n').filter(element => element !== '');
619.    class_students.filter(element => element !== '').forEach((element, index) => {
620.       class_students[index] = element.replace('\r', '').split(',');
621.       class_students[index].forEach((element_2, index_2) => {
622.
623.          if (index_2 == 1) {
624.             class_students[index][index_2] = parseInt(element_2);
625.          }
626.
627.          if (index_2 == 2) {
628.             if (element_2 == '') {
629.                class_students[index][index_2] = [];
630.             } else {
631.                class_students[index][index_2] = class_students[index].slice(2, class_students[index].length + 1);
632.             }
633.          }
634.       });
635.
636.       class_students[index] = class_students[index].slice(0, 3);
637.    });
638.
639.    // var students_to_assign = req.body.students_selected.split(' | ').filter(element => element !== '');
640.    var students_to_assign = req.body.students_selected_2.split(',').filter(element => element !== '');
```

```
641.    var due_date = new Date(req.body.due_date);
642.    var today = new Date();
643.
644.    try {
645.
646.      // all user-side scripting for validation
647.      var error_message = false;
648.
649.      // ensure at least 1 student selected
650.      if (students_to_assign.length === 0 || students_to_assign === [] || typeof students_to_assign === "undefined") {
651.        error_message = true;
652.        req.flash('error', ' Select at least 1 student');
653.      }
654.
655.      // due date must be > 24 hours away from current time
656.      if (due_date.getTime() <= new Date(today.getTime() + 60 * 60 * 24 * 1000).getTime()) {
657.        error_message = true;
658.        req.flash('error', ' Due date must be at least 24 hours after current time');
659.      }
660.
661.      // return with the error message
662.      if (error_message) {
663.        res.locals.message = req.flash();
664.        return res.render('teacher_challenges/create_challenges', {
665.          title: 'Create Challenges',
666.          menu_id: 'create_challenges',
667.          role: req.user.role,
668.          email: req.user.email,
669.          challenges_to_display: challenges_to_display,
670.          challenge_title_match_id: challenge_title_match_id,
671.          class_students: class_students,
672.          session_id: req.sessionID,
673.          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
674.        });
675.      }
676.
677.      var student_ids_assigned_this_challenge_already = [];
678.      var student_names_assigned_this_challenge_already = [];
679.
680.      await get_students_already_assigned_challenge(req, res, next, challenges_to_display, challenge_title_match_id,
class_students).then((rows) => {
681.        rows.forEach((element, index) => {
682.          student_ids_assigned_this_challenge_already.push([element['CONCAT(Student.student_name,
Student.student_number)'], element['student_id']]);
683.        });
684.      }).then(() => {
685.
686.        student_ids_assigned_this_challenge_already.forEach((element, index) => {
687.          student_names_assigned_this_challenge_already.push(element[0]);
688.        });
689.
690.        var contains_1 = student_names_assigned_this_challenge_already.some(element => {
691.          return students_to_assign.includes(element);
692.        });
693.        var contains_2 = students_to_assign.some(element => {
```

```
            return student_names_assigned_this_challenge_already.includes(element);
        });

        if (contains_1 == true || contains_2 == true) {
            error_message = true;
            req.flash('error', ' At least 1 student has already been assigned this challenge!');
        };

        // return with the error message
        if (error_message) {
            res.locals.message = req.flash();
            return res.render('teacher_challenges/create_challenges', {
                title: 'Create Challenges',
                menu_id: 'create_challenges',
                role: req.user.role,
                email: req.user.email,
                challenges_to_display: challenges_to_display,
                challenge_title_match_id: challenge_title_match_id,
                class_students: class_students,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            });
        } else {

            students_to_assign.forEach((element, index) => {
                // record relation b/w teacher and challenge
                db_connection.query(
                    'INSERT INTO Assigned_Challenges (challenge_teacher_id, student_id, due_date) VALUES (?, (SELECT student_id FROM Student WHERE student_name = ? AND student_number = ?), ?);',
                    [challenge_title_match_id[_.findIndex(challenge_title_match_id, function (el) { return el[0] == req.body.assign_challenge_dd })][1], element.slice(0, -4), element.slice(-4), due_date.toISOString().slice(0, 19).replace('T', ' ')],
                    function (err, rows) {
                        if (err) {
                            console.log(err);
                            req.flash('error', ' An error occured');
                            res.locals.message = req.flash();
                            return res.render('teacher_challenges/create_challenges', {
                                title: 'Create Challenges',
                                menu_id: 'create_challenges',
                                role: req.user.role,
                                email: req.user.email,
                                challenges_to_display: challenges_to_display,
                                challenge_title_match_id: challenge_title_match_id,
                                class_students: class_students,
                                session_id: req.sessionID,
                                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                            });
                        }
                        // console.log(rows);
                    }
                );
            });

            req.flash('success', ' Assigned challenge to student(s)');
            res.locals.message = req.flash();
```

```
            return res.render('teacher_challenges/create_challenges', {
              title: 'Create Challenges',
              menu_id: 'create_challenges',
              role: req.user.role,
              email: req.user.email,
              challenges_to_display: challenges_to_display,
              challenge_title_match_id: challenge_title_match_id,
              class_students: class_students,
              session_id: req.sessionID,
              session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
            });

          }

        });

      } catch (error) {
        console.log(error);
        // redirect to page to create a new challenge
        return res.render('teacher_challenges/create_challenges', {
          title: 'Create Challenges',
          menu_id: 'create_challenges',
          role: req.user.role,
          email: req.user.email,
          challenges_to_display: challenges_to_display,
          challenge_title_match_id: challenge_title_match_id,
          class_students: class_students,
          session_id: req.sessionID,
          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
        });
      };
  });

var get_students_already_assigned_challenge = (req, res, next, challenges_to_display, challenge_title_match_id, class_students) => {
    return new Promise((resolve, reject) => {
        try {

            // get the students who have already been assigned the specific challenge
            db_connection.query(
                'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id FROM Assigned_Challenges, Challenge_Teacher, Student WHERE Assigned_Challenges.challenge_teacher_id = Challenge_Teacher.challenge_teacher_id AND Assigned_Challenges.student_id = Student.student_id AND Assigned_Challenges.challenge_teacher_id = (SELECT challenge_teacher_id FROM Challenge_Teacher WHERE challenge_teacher_id = ?);',
                [challenge_title_match_id[_.findIndex(challenge_title_match_id, function (el) { return el[0] == req.body.assign_challenge_dd })][1]],
                function (err, rows) {

                    if (err) {
                        console.log(err);
                        req.flash('error', ' An error occured');
                        res.locals.message = req.flash();
                        return res.render('teacher_challenges/create_challenges', {
                            title: 'Create Challenges',
```

```
                    menu_id: 'create_challenges',
                    role: req.user.role,
                    email: req.user.email,
                    challenges_to_display: challenges_to_display,
                    challenge_title_match_id: challenge_title_match_id,
                    class_students: class_students,
                    session_id: req.sessionID,
                    session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                  });
                  reject(err)
                }

                resolve(rows);

              }
            );

        } catch (error) {
            reject(error);
        }
    });
};

module.exports = router;

```

~/routes/teacher_challenges/view_student_challenges.js

```
var express = require('express');
var router = express.Router();

var auth = require('../../lib/auth');

// for connecting to database
var db_connection = require('../../lib/db');

// for writing to files
var fs = require('fs');

// for array/list operations
var _ = require('lodash');

// for DATETIME conversions
var moment = require('moment');
moment.locale('en-gb');
moment.defaultFormat = 'L LT';
var utc_datetime = moment.utc().format();

// coming to view_student_challenges page
router.get('/', auth.check_authenticated, async function (req, res, next) {

    res.locals.message = req.flash();

    try {
```

```
27.
28.     await get_challenges_assigned_not_assigned(req, res, next, true).then((rows) => {
29.
30.       var all_assigned_challenges = [];
31.
32.       rows.forEach(element => {
33.         all_assigned_challenges.push([element['challenge_blob'].toString(), element['challenge_teacher_id']]);
34.       });
35.
36.       console.log('all_assigned_challenges');
37.       console.log(all_assigned_challenges);
38.
39.       return {
40.         'all_assigned_challenges': all_assigned_challenges
41.       };
42.
43.     }).then(async (useful_vars) => {
44.
45.       await get_challenges_assigned_not_assigned(req, res, next, false).then((rows) => {
46.
47.         var all_not_assigned_challenges = [];
48.
49.         rows.forEach(element => {
50.           all_not_assigned_challenges.push([element['challenge_blob'].toString(), element['challenge_teacher_id']]);
51.         });
52.
53.         console.log('all_not_assigned_challenges');
54.         console.log(all_not_assigned_challenges);
55.
56.         useful_vars['all_not_assigned_challenges'] = all_not_assigned_challenges;
57.
58.       });
59.
60.       return useful_vars;
61.
62.     }).then((useful_vars) => {
63.
64.       useful_vars['all_assigned_challenges'].forEach(async (element, index) => {
65.
66.         var route_more_info = '/more_information/' + element[1];
67.
68.         router.post(route_more_info, async function (req, res, next) {
69.
70.           res.locals.message = req.flash();
71.
72.           try {
73.
74.             await get_students_assigned_challenge_teacher_and_solved(req, res, next, parseInt(element[1])).then((rows)
=> {
75.
76.               var students_solved = [];
77.
78.               rows.forEach(element_new => {
79.                 students_solved.push([element_new['CONCAT(Student.student_name,
Student.student_number)'].toString(), element_new['student_id'],
```

```
moment(element_new['due_date']).utc(utc_datetime).local().format(),
moment(element_new['completion_date']).utc(utc_datetime).local().format()]);
                    });

                    console.log('students_solved');
                    console.log(students_solved);

                    return {
                        'students_solved': students_solved
                    };

                }).then(async (useful_vars) => {

                    await get_students_assigned_challenge_teacher_and_not_solved(req, res, next,
parseInt(element[1])).then((rows) => {

                        var students_not_solved_before_due = []; // can still solve
                        var students_not_solved_after_due = []; // has not completed assignment and cannot complete it now

                        rows.forEach(element_new => {

                            // past deadline
                            if (new Date() >= new Date(moment(element_new['due_date']).utc(utc_datetime).local().format())) {
                                students_not_solved_before_due.push([element_new['CONCAT(Student.student_name,
Student.student_number)'].toString(), element_new['student_id'],
moment(element_new['due_date']).utc(utc_datetime).local().format()]);
                            } else {
                                students_not_solved_after_due.push([element_new['CONCAT(Student.student_name,
Student.student_number)'].toString(), element_new['student_id'],
moment(element_new['due_date']).utc(utc_datetime).local().format()]);
                            };

                        });

                        console.log('students_not_solved_before_due');
                        console.log(students_not_solved_before_due);
                        console.log('students_not_solved_after_due');
                        console.log(students_not_solved_after_due);

                        useful_vars['students_not_solved_before_due'] = students_not_solved_before_due;
                        useful_vars['students_not_solved_after_due'] = students_not_solved_after_due;

                    });

                    return useful_vars;

                }).then(async (useful_vars) => {

                    useful_vars['students_solved'].forEach(async (element_latest, index_latest) => {

                        var route_view_solution = route_more_info + '/view_answer/' + element_latest[1];

                        router.post(route_view_solution, async function (req, res, next) {

                            res.locals.message = req.flash();
```

```
                    await get_answer_blob(req, res, next, parseInt(element[1]), parseInt(element_latest[1])).then((rows) => {

                      rows.forEach(first_row => {
                        answer_blob = first_row['solution_blob'];
                      });

                      return res.render('teacher_challenges/view_answer', {
                        title: 'View Completed Challenge',
                        role: req.user.role,
                        email: req.user.email,
                        session_id: req.sessionID,
                        session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                        challenge_title: element[0].split('\n')[1].toString(),
                        challenge_description: element[0].split('\n')[3].toString(),
                        challenge_answer: answer_blob.toString()
                      });

                    });

                  });

                return res.render('teacher_challenges/more_information', {
                  title: 'More Information',
                  role: req.user.role,
                  email: req.user.email,
                  session_id: req.sessionID,
                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                  challenge_title: element[0].split('\n')[1].toString(),
                  challenge_description: element[0].split('\n')[3].toString(),
                  challenge_teacher_id: element[1].toString(),
                  students_solved: useful_vars['students_solved'],
                  students_not_solved_before_due: useful_vars['students_not_solved_before_due'],
                  students_not_solved_after_due: useful_vars['students_not_solved_after_due']
                });

              });

            } catch (any_error) {

              res.render('teacher_challenges/more_information', {
                title: 'More Information',
                role: req.user.role,
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                challenge_title: element[0].split('\n')[1].toString(),
                challenge_description: element[0].split('\n')[3].toString(),
                challenge_teacher_id: element[1].toString()
              });

            }
```

```
182.
183.            });
184.
185.        });
186.
187.        return res.render('teacher_challenges/view_student_challenges', {
188.            title: 'View Student Challenges',
189.            menu_id: 'view_student_challenges',
190.            role: req.user.role,
191.            email: req.user.email,
192.            all_assigned_challenges: useful_vars['all_assigned_challenges'],
193.            all_not_assigned_challenges: useful_vars['all_not_assigned_challenges'],
194.            session_id: req.sessionID,
195.            session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
196.        });
197.
198.    });
199.
200.  } catch (any_error) {
201.      console.log(any_error);
202.      res.render('teacher_challenges/view_student_challenges', {
203.          title: 'View Student Challenges',
204.          menu_id: 'view_student_challenges',
205.          role: req.user.role,
206.          email: req.user.email,
207.          session_id: req.sessionID,
208.          session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
209.      });
210.  }
211.
212. });
213.
214. var get_challenges_assigned_not_assigned = (req, res, next, assigned) => {
215.    return new Promise((resolve, reject) => {
216.      try {
217.
218.          if (assigned) {
219.              // all challenge_teacher_id's that have been assigned to some student
220.              var query_to_run = 'SELECT * FROM Challenge_Teacher, Challenge_File WHERE
Challenge_Teacher.challenge_file_id = Challenge_File.challenge_file_id AND Challenge_Teacher.challenge_teacher_id
IN(SELECT challenge_teacher_id FROM Assigned_Challenges) AND Challenge_Teacher.teacher_id = ?;';
221.          } else {
222.              // all challenge_teacher_id's that have not been assigned to some student
223.              var query_to_run = 'SELECT * FROM Challenge_Teacher, Challenge_File WHERE
Challenge_Teacher.challenge_file_id = Challenge_File.challenge_file_id AND Challenge_Teacher.challenge_teacher_id NOT IN
(SELECT challenge_teacher_id FROM Assigned_Challenges) AND Challenge_Teacher.teacher_id = ?;';
224.          };
225.
226.          db_connection.query(
227.              query_to_run,
228.              [req.user.id],
229.              function (err, rows) {
230.
231.                  if (err) {
232.                      console.log(err);
```

```
                req.flash('error', ' An error occured');
                res.locals.message = req.flash();
                return res.render('teacher_challenges/view_student_challenges', {
                  title: 'View Student Challenges',
                  menu_id: 'view_student_challenges',
                  role: req.user.role,
                  email: req.user.email,
                  session_id: req.sessionID,
                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                });
                reject(err);
              };

              resolve(rows);
            }
          );

        } catch (error) {
          reject(error);
        }
      });
};

var get_students_assigned_challenge_teacher_and_solved = (req, res, next, challenge_teacher_id) => {
  return new Promise((resolve, reject) => {
    try {

      // get all students who have been assigned challenge_teacher_id and solved
      db_connection.query(
        'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id, Assigned_Challenges.due_date, Solution_Student.completion_date FROM Solution_Student, Solution_File, Student, Assigned_Challenges WHERE Solution_Student.solution_file_id = Solution_File.solution_file_id AND Assigned_Challenges.assigned_challenge_id = Solution_Student.assigned_challenge_id AND Assigned_Challenges.student_id = Student.student_id AND Assigned_Challenges.challenge_teacher_id = ? AND Assigned_Challenges.assigned_challenge_id IN (SELECT assigned_challenge_id FROM Solution_Student);',
          [challenge_teacher_id],
          function (err, rows) {

            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              return res.render('teacher_challenges/view_student_challenges', {
                title: 'View Student Challenges',
                menu_id: 'view_student_challenges',
                role: req.user.role,
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
              });
              reject(err);
            };

            resolve(rows);
          }
```

```
        );

      } catch (error) {
        reject(error);
      }
    });
};
var get_students_assigned_challenge_teacher_and_not_solved = (req, res, next, challenge_teacher_id) => {
    return new Promise((resolve, reject) => {
      try {

        // get all students who have been assigned challenge_teacher_id and solved
        db_connection.query(
          'SELECT CONCAT(Student.student_name, Student.student_number), Student.student_id, Assigned_Challenges.due_date FROM Student, Assigned_Challenges WHERE Assigned_Challenges.student_id = Student.student_id AND Assigned_Challenges.challenge_teacher_id = ? AND Assigned_Challenges.assigned_challenge_id NOT IN (SELECT assigned_challenge_id FROM Solution_Student);',
          [challenge_teacher_id],
          function (err, rows) {

            if (err) {
              console.log(err);
              req.flash('error', ' An error occured');
              res.locals.message = req.flash();
              return res.render('teacher_challenges/view_student_challenges', {
                title: 'View Student Challenges',
                menu_id: 'view_student_challenges',
                role: req.user.role,
                email: req.user.email,
                session_id: req.sessionID,
                session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
              });
              reject(err);
            };

            resolve(rows);
          }
        );

      } catch (error) {
        reject(error);
      }
    });
};

var get_answer_blob = (req, res, next, challenge_teacher_id, student_id) => {
    return new Promise((resolve, reject) => {
      try {

        // get all students who have been assigned challenge_teacher_id and solved
        db_connection.query(
          'SELECT * FROM Solution_File, Solution_Student WHERE Solution_File.solution_file_id = Solution_Student.solution_file_id AND Solution_Student.assigned_challenge_id = (SELECT assigned_challenge_id FROM Assigned_Challenges WHERE challenge_teacher_id = ? AND student_id = ?);',
          [challenge_teacher_id, student_id],
```

```
            function (err, rows) {

              if (err) {
                console.log(err);
                req.flash('error', ' An error occured');
                res.locals.message = req.flash();
                return res.render('teacher_challenges/view_student_challenges', {
                  title: 'View Student Challenges',
                  menu_id: 'view_student_challenges',
                  role: req.user.role,
                  email: req.user.email,
                  session_id: req.sessionID,
                  session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
                });
                reject(err);
              };

              resolve(rows);
            }
          );

        } catch (error) {
          reject(error);
        }
      });
  };

module.exports = router;

```

~/routes/index.js

```
// all libs/modules
var express = require('express');
var router = express.Router();

var auth = require('../lib/auth');

/* GET home page. */
router.get('/', function(req, res, next) {

  console.log('req.user:', req.user);

  res.locals.message = req.flash();

  // no user
  if (typeof req.user == 'undefined') {
    res.render('index', {
      title: 'Home',
      menu_id: 'home',
      session_id: req.sessionID,
      session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
    });
  } else { // user exists
```

```
23.         res.render('index', {
24.           title: 'Home',
25.           menu_id: 'home',
26.           role: req.user.role,
27.           email: req.user.email,
28.           session_id: req.sessionID,
29.           session_expiry_time: new Date(req.session.cookie.expires) - new Date(),
30.         });
31.     }
32.
33. });
34.
35. /* GET instruction set page */
36. router.get('/instruction_set', function (req, res, next) {
37.
38.     res.locals.message = req.flash();
39.
40.     if (typeof req.user == 'undefined') {
41.         res.render('instructions/instruction_set', {
42.           title: 'Instruction Set',
43.           menu_id: 'instruction_set',
44.         });
45.     } else {
46.         res.render('instructions/instruction_set', {
47.           title: 'Instruction Set',
48.           menu_id: 'instruction_set',
49.           role: req.user.role,
50.         });
51.     }
52. });
53.
54. /* GET how to use page */
55. router.get('/how_to_use', function (req, res, next) {
56.
57.     res.locals.message = req.flash();
58.
59.     if (typeof req.user == 'undefined') {
60.         res.render('instructions/how_to_use', {
61.           title: 'How To Use This Resource',
62.           menu_id: 'how_to_use',
63.         });
64.     } else {
65.         res.render('instructions/how_to_use', {
66.           title: 'How To Use This Resource',
67.           menu_id: 'how_to_use',
68.           role: req.user.role,
69.         });
70.     }
71. });
72.
73. /* POST logout button request */
74. router.post('/logout', function(req, res) {
75.     res.locals.message = req.flash();
76.     req.logout(function(err) {
77.
```

```
    if (err) {
      console.log(err);
      return next(err);
    }

    // remove cookies and destroy session
    req.session.destroy((error) => {

      if (error) {
        console.log(error);
        return next(error);
      }

      res.redirect('/login');
    });
  });
});

module.exports = router;

```

~/views/instructions/how_to_use.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
  <% include ../partials/head %>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <!-- HOW TO USE THIS WEBSITE/RESOURCE INFO -->

    <div class="row">

      <h4 class="centered_header sub_header">LMC:</h4>
      <table id="image_explanation_table">
        <tr>
          <td>
            <a href="/images/LMC_top_half_explanation.png" target="_blank"><img style="width: 100%; height: 100%; object-fit: scale-down;" src="/images/LMC_top_half_explanation.png" alt="Explanation of Top Half of Home Page" /></a>
          </td>
          <td>
            <a href="/images/LMC_top_half_explanation.png" target="_blank"><img style="width: 100%; height: 100%; object-fit: scale-down;" src="/images/LMC_bottom_half_explanation.png" alt="Explanation of Top Half of Home Page" /></a>
          </td>
        </tr>
      </table>

    </div>
```

```
30.
31.       <br />
32.
33.       <div class="row centered_header">
34.
35.          <h4 class="sub_header">Challenges:</h4>
36.
37.          <div class="col-md-6" id="sub_header_2_t">
38.             <h5>Teachers:</h5>
39.          </div>
40.          <div class="col-md-6" id="sub_header_2_s">
41.             <h5>Students:</h5>
42.          </div>
43.
44.          <table class="centered_header" id="pages_info_table">
45.
46.             <tr>
47.                <td>
48.                   <p>From the <b>CREATE CHALLENGES</b> page, Teachers have the ability to create, edit, and delete
challenges. In addition to this, they can also assign any of these challenges to specific Students, or to a whole class by clicking
the "Select All" button next to the respective class.</p>
49.                </td>
50.                <td rowspan="2">
51.                   <p>From the <b>VIEW SET CHALLENGES</b> page, Students are able to view their pending challenges to
complete, their completed challenges, and the challenges that they have missed, along with all the available information - due
data, and completion date if possible.<br /><br />For the challenges left to complete, the Student is able to submit their
solution by clicking on the button next to the challenge, and this will take them to a new page which displays the Challenge Title
and Challenge Description, and has an input field for the Student to paste in their Assembly Code solution to the problem.<br
/><br />For completed challenges, Students can also see what their answer was by clicking the "View Answer" button next to
the challenge.</p>
52.                </td>
53.             </tr>
54.
55.             <tr>
56.                <td>
57.                   <p>From the <b>VIEW STUDENT CHALLENGES</b> page, Teachers can first view their challenges that have
been assigned to at least 1 Student, and those that simply haven't been assigned yet.<br /><br />After clicking on the "More
Information" button for a challenge that has been assigned already, the Teacher will be able to see the Challenge Title and
Challenge Description, as well as all the Students who have submitted, have yet to submit, or have failed to submit the
challenge - with information about the due date, and completion date, if available.<br /><br />The Teacher can then view the
Student's solution if the Student has already submitted this assignment.</p>
58.                </td>
59.             </tr>
60.
61.          </table>
62.
63.       </div>
64.
65.    </div>
66. </body>
67.
68. <% include ../partials/script %>
69.
70. </html>
71.
```

~/views/instructions/instruction_set.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <!-- INSTRUCTION SET STUFF (PROBABLY TABLE)

        <div class="row">
            <div class="col-md-2"></div>

            <div class="col-md-8">
                <object id="instruction_set_viewer" data="https://filestore.aqa.org.uk/resources/computing/AQA-75162-75172-ALI.PDF" type="application/pdf" width="100%">
                    You can also view the pdf from the AQA website <a href="https://filestore.aqa.org.uk/resources/computing/AQA-75162-75172-ALI.PDF">here</a>.
                </object>
            </div>

            <div class="col-md-2"></div>
        </div> -->

        <div class="row">
            <div class="col-md-12">

                <table id="instruction_set_table">
                    <caption>Above is an explanation of the Instruction Set for the LMC. This has been taken from <a
                        href="https://en.wikipedia.org/wiki/Little_man_computer#Instructions" target="_blank">here</a>.</caption>
                    <tbody>
                        <tr>
                            <th>Numeric code
                            </th>
                            <th>Mnemonic code
                            </th>
                            <th>Instruction
                            </th>
                            <th>Description
                            </th>
                        </tr>
                        <tr>
                            <td>1xx</td>
                            <td>ADD</td>
                            <td>ADD
                            </td>
                            <td>Add the value stored in mailbox xx to whatever value is currently on the accumulator (calculator).
```

```
            <dl>
              <dd>Note: the contents of the mailbox are not changed, and the actions of the accumulator
                (calculator) are not defined for add instructions that cause sums larger than 3 digits.
                Similarly to SUBTRACT, one could set the negative flag on overflow.</dd>
            </dl>
          </td>
        </tr>
        <tr>
          <td>2xx</td>
          <td>SUB</td>
          <td>SUBTRACT
          </td>
          <td>Subtract the value stored in mailbox xx from whatever value is currently on the accumulator
            (calculator).
            <dl>
              <dd>Note: the contents of the mailbox are not changed, and the actions of the accumulator are not
                defined for subtract instructions that cause negative results - however, a negative flag will be
                set so that <b>7xx (BRZ)</b> and <b>8xx (BRP)</b> can be used properly.</dd>
            </dl>
          </td>
        </tr>
        <tr>
          <td>3xx</td>
          <td>STA</td>
          <td>STORE
          </td>
          <td>Store the contents of the accumulator in mailbox xx (destructive).
            <dl>
              <dd>Note: the contents of the accumulator (calculator) are not changed (non-destructive), but
                contents of mailbox are replaced regardless of what was in there (destructive)</dd>
            </dl>
          </td>
        </tr>
        <tr>
          <td>5xx</td>
          <td>LDA</td>
          <td>LOAD
          </td>
          <td>Load the value from mailbox xx (non-destructive) and enter it in the accumulator (destructive).
          </td>
        </tr>
        <tr>
          <td>6xx</td>
          <td>BRA</td>
          <td>BRANCH (unconditional)</td>
          <td>Set the program counter to the given address (value xx). That is, the value in mailbox xx will be the
            next instruction executed.
          </td>
        </tr>
        <tr>
          <td>7xx</td>
          <td>BRZ</td>
          <td>BRANCH IF ZERO (conditional)
          </td>
          <td>If the accumulator (calculator) contains the value 000, set the program counter to the value xx.
```

```
105.                Otherwise, do nothing. Whether the negative flag is taken into account is undefined. When a SUBTRACT
106.                underflows the accumulator, this flag is set, after which the accumulator is undefined, potentially
107.                zero, causing behavior of BRZ to be undefined on underflow. Suggested behavior would be to branch if
108.                accumulator is zero and negative flag is not set.
109.                <dl>
110.                  <dd>Note: since the program is stored in memory, data and program instructions all have the same
111.                    address/location format.</dd>
112.                </dl>
113.              </td>
114.            </tr>
115.            <tr>
116.              <td>8xx</td>
117.              <td>BRP</td>
118.              <td>BRANCH IF POSITIVE (conditional)
119.              </td>
120.              <td>If the accumulator (calculator) is 0 or positive, set the program counter to the value xx. Otherwise, do
121.                nothing. As LMC memory cells can only hold values between 0 and 999, this instruction depends solely
on
122.                the negative flag set by an underflow on SUBTRACT and potentially on an overflow on ADD (undefined).
123.                <dl>
124.                  <dd>Note: since the program is stored in memory, data and program instructions all have the same
125.                    address/location format.</dd>
126.                </dl>
127.              </td>
128.            </tr>
129.            <tr>
130.              <td>901</td>
131.              <td>INP</td>
132.              <td>INPUT
133.              </td>
134.              <td>Go to the INBOX, fetch the value from the user, and put it in the accumulator (calculator)
135.                <dl>
136.                  <dd>Note: this will overwrite whatever value was in the accumulator (destructive)</dd>
137.                </dl>
138.              </td>
139.            </tr>
140.            <tr>
141.              <td>902</td>
142.              <td>OUT</td>
143.              <td>OUTPUT
144.              </td>
145.              <td>Copy the value from the accumulator (calculator) to the OUTBOX.
146.                <dl>
147.                  <dd>Note: the contents of the accumulator are not changed (non-destructive).</dd>
148.                </dl>
149.              </td>
150.            </tr>
151.            <tr>
152.              <td>000</td>
153.              <td>HLT</td>
154.              <td>HALT</td>
155.              <td>Stop working/end the program.</td>
156.            </tr>
157.            <tr>
158.              <td>N/A</td>
```

```
                    <td>DAT</td>
                    <td>DATA</td>
                    <td>This is an assembler instruction which
                      simply loads the value into the next available mailbox. DAT can also be used in conjunction with labels
                      to declare variables. For example, DAT 984 will store the value 984 into a mailbox at the address of the
                      DAT instruction.
                    </td>
                  </tr>
                </tbody>
              </table>

            </div>
          </div>

          <div class="row">
            <div class="col-md-12">

              <p>Note that xx in the table refers to any memory location in RAM - from 00 up to and including 99. It may also be useful to look at the AQA Instruction Set. Even though we do not use it for this visualisation, it may prove helpful to understand direct and immediate addressing.</p>

            </div>
          </div>

          <div class="row">
            <div class="col-md-12">
              <object id="instruction_set_viewer" data="https://filestore.aqa.org.uk/resources/computing/AQA-75162-75172-ALI.PDF"
                type="application/pdf" width="100%">
                You can also view the pdf from the AQA website <a href="https://filestore.aqa.org.uk/resources/computing/AQA-75162-75172-ALI.PDF">here</a>.
              </object>
            </div>
          </div>

        </div>
</body>

<% include ../partials/script %>

</html>

```

~/views/login_sign_up/partials/script.ejs

```
<!-- Local files -->
<script type="text/javascript" src="/javascripts/form_toggler.js"></script>

```

~/views/login_sign_up/login.ejs

```
<!DOCTYPE html>
<html lang="en">

```

```
<head>
  <% include ../partials/head %>
</head>

<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <div class="row justify-content-center" id="student_card">
      <div class="col-md-8">
        <div class="card">

          <div class="card-header" style="background-color: #93c572;">
            <div class="row">
              <div class="col-md-3">
                <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Student Login</h5>
              </div>
              <div class="col-md-6"></div>
              <div class="col-md-3">
                <button class="btn btn-secondary switch_to_teacher" onclick="change_form_login()">Switch to Teacher Login</button>
              </div>
            </div>
          </div>

          <div class="card-body">

            <!-- student form -->
            <form action="/login/student" method="post" id="student_form">

              <!-- student's email -->
              <div class="form-group row">
                <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                <div class="col-md-8">
                  <!-- automatically check for correctly formatted student dc email (regex in JavaScript: '\d{4}' is the same as '[0-9][0-9][0-9][0-9]' ) -->
                  <input type="email" id="email" name="email" class="form-control" pattern="^[A-Za-z]+\d{4}\@dubaicollege.org$"
                    placeholder="student_name_and_number@dubaicollege.org" required autofocus />
                </div>
              </div>

              <!-- student's password -->
              <div class="form-group row">
                <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
                <div class="col-md-8">
                  <input type="password" id="password" name="password" class="form-control" placeholder="enter a password" required />
                </div>
              </div>

              <!-- remember me button -->
              <div class="form-group row">
                <div class="col-md-8 offset-md-3">
```

```
                        <label for="remember_me" style="margin-left: 0.5em;">REMEMBER ME</label>
                        <div class="remember_me_float">
                          <input type="checkbox" id="remember_me" name="remember_me" value="1" />
                        </div>
                      </div>
                    </div>

                    <!-- submit button -->
                    <div class="row">
                      <div class="col-md-8 offset-md-3">
                        <button type="submit" class="btn btn-primary" value="Login">LOGIN</button>
                        <a href="#" class="btn btn-link">Forgot Your Password?</a>
                      </div>
                    </div>

                    <% if (message.error) { %>
                      <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
                        <strong>
                          <%= message.error %>
                        </strong>
                      </div>
                    <% } %>

                    <% if (message.success) { %>
                      <div class="alert alert-success" role="alert" style="margin-top: 1em;">
                        <strong>
                          <%= message.success %>
                        </strong>
                      </div>
                    <% } %>

                  </form>

                  <h5 style="margin-top: 1em;">Don't have an account? <a href="/sign_up">Sign up...</a></h5>

                </div>

              </div>
            </div>
          </div>

          <div class="row justify-content-center" id="teacher_card">
            <div class="col-md-8">
              <div class="card">

                <div class="card-header" style="background-color: #cccc00;">
                  <div class="row">
                    <div class="col-md-3">
                      <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Teacher Login</h5>
                    </div>
                    <div class="col-md-6"></div>
                    <div class="col-md-3">
                      <button class="btn btn-secondary switch_to_student" onclick="change_form_login()">Switch to Student Login</button>
                    </div>
```

```
            </div>
        </div>

        <div class="card-body">

            <!-- teacher form -->
            <form action="/login/teacher" method="post" id="teacher_form">

                <!-- teacher's email -->
                <div class="form-group row">
                    <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                    <div class="col-md-8">
                        <!-- automatically check for correctly formatted teacher dc email -->
                        <input type="email" id="email" name="email" class="form-control" pattern="^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$" placeholder="teacher_first_name.teacher_last_name@dubaicollege.org" required />
                    </div>
                </div>

                <!-- teacher's password -->
                <div class="form-group row">
                    <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
                    <div class="col-md-8">
                        <input type="password" id="password" name="password" class="form-control" placeholder="enter a password" required />
                    </div>
                </div>

                <!-- remember me button -->
                <div class="form-group row">
                    <div class="col-md-8 offset-md-3">
                        <label for="remember_me" style="margin-left: 0.5em;">REMEMBER ME</label>
                        <div class="remember_me_float">
                            <input type="checkbox" id="remember_me" name="remember_me" value="1" />
                        </div>
                    </div>
                </div>

                <!-- submit button -->
                <div class="row">
                    <div class="col-md-8 offset-md-3">
                        <button type="submit" class="btn btn-primary" value="Login">LOGIN</button>
                        <a href="#" class="btn btn-link">Forgot Your Password?</a>
                    </div>
                </div>

                <% if (message.error) { %>
                    <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
                        <strong>
                            <%= message.error %>
                        </strong>
                    </div>
                <% } %>

                <% if (message.success) { %>
                    <div class="alert alert-success" role="alert" style="margin-top: 1em;">
```

```
                        <strong>
                          <%= message.success %>
                        </strong>
                      </div>
                    <% } %>

                  </form>

                  <h5 style="margin-top: 1em;">Don't have an account? <a href="/sign_up">Sign up...</a></h5>

                </div>

              </div>
            </div>
        </div>

    </div>
</body>

<% include ../partials/script %>
<% include partials/script %>

</html>

```

~/views/login_sign_up/sign_up.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
    <!-- ensure that the page reloads on first redirect so all data from db is up-to-date -->
    <script type="text/javascript">
        $(document).ready(function () {
            //Check if the current URL contains '#'

            //if (document.URL.slice(-8, document.URL.length - 1) == '/assign' || document.URL.slice(-8, document.URL.length - 1) === '/assign') {
            //  url = document.URL.slice(0, document.URL.length - 8);
            //  location.replace(url);
            //};

            if (document.URL.indexOf("#") == -1) {
                // Set the URL to whatever it was plus "#".
                url = document.URL + "#";
                location = "#";

                //Reload the page
                location.reload(true);
            }
        });
    </script>
</head>
```

```
<body>
  <div class="container-fluid">

    <% include ../partials/menu %>

    <div class="row justify-content-center" id="student_card">
      <div class="col-md-8">
        <div class="card">

          <div class="card-header" style="background-color: #93c572;">
            <div class="row">
              <div class="col-md-3">
                <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Student Sign-up</h5>
              </div>
              <div class="col-md-9">
                <button class="btn btn-secondary switch_to_teacher" onclick="change_form_login()" style="float: right;">Switch to Teacher Sign-up</button>
              </div>
            </div>
          </div>

          <div class="card-body">

            <!-- student form -->
            <form action="/sign_up/student" method="POST" id="student_form">

              <!-- student's email -->
              <div class="form-group row">

                <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                <div class="col-md-8">
                  <!-- automatically check for correctly formatted student dc email (regex in JavaScript: '\d{4}' is the same as '[0-9][0-9][0-9][0-9]' ) -->
                  <input type="email" id="email" name="email" pattern="^[A-Za-z]+\d{4}\@dubaicollege.org$" class="form-control" placeholder="student_name_and_number@dubaicollege.org" required autofocus />
                </div>
              </div>

              <!-- student's year group -->
              <div class="form-group row">
                <label for="year_group" class="col-md-3 col-form-label">YEAR GROUP</label>
                <div class="col-md-8">
                  <!-- drop-down menu options -->
                  <!-- NOTE: ASSUMPTION THAT EVERY TEACHER TEACHES EVERY YEAR (THIS MAKES IT EASIER TO SIGN UP, EVEN IF THOSE CLASSES DON'T ACTUALLY HAVE STUDENTS IN THEM) -->
                  <select id="year_group" name="year_group" class="form-control" required>
                    <option value="" selected hidden>select your year group</option>
                    <option value="10">Year 10</option>
                    <option value="11">Year 11</option>
                    <option value="12">Year 12</option>
                    <option value="13">Year 13</option>
                  </select>
                </div>
              </div>
```

```
78.
79.                 <!-- student's teacher (3-letter code) -->
80.                 <div class="form-group row">
81.                   <label for="teacher_initials" class="col-md-3 col-form-label">CLASS (TEACHER CODE)</label>
82.                   <div class="col-md-8">
83.                     <!-- drop-down menu options -->
84.                     <select id="teacher_initials" name="teacher_initials" class="form-control" required>
85.                       <option value="" selected hidden>select your class</option>
86.                       <% if (typeof class_codes=="undefined" ) { %>
87.                         <option value="no_teacher">no teacher has signed up yet</option>
88.                       <% } else { %>
89.                         <% for (var class_code_counter=0; class_code_counter < class_codes.length;
class_code_counter++) { %>
90.                           <option value="<%= class_codes[class_code_counter].class_code %>">
91.                             <%= class_codes[class_code_counter].class_code %>
92.                           </option>
93.                         <% } %>
94.                       <% } %>
95.                     </select>
96.                   </div>
97.                 </div>
98.
99.                 <!-- student's password -->
100.                <div class="form-group row">
101.                  <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
102.                  <div class="col-md-8">
103.                    <input type="password" id="password" name="password" class="form-control" placeholder="enter a
password" required />
104.                  </div>
105.                </div>
106.
107.                <!-- student's password confirmation -->
108.                <div class="form-group row">
109.                  <label for="password_confirmation" class="col-md-3 col-form-label">CONFIRM PASSWORD</label>
110.                  <div class="col-md-8">
111.                    <input type="password" id="password_confirmation" name="password_confirmation" class="form-
control" placeholder="enter the same password again" required />
112.                  </div>
113.                </div>
114.
115.                <!-- submit button -->
116.                <div class="row">
117.                  <div class="col-md-8 offset-md-3">
118.                    <button type="submit" class="btn btn-primary">SIGN UP</button>
119.                  </div>
120.                </div>
121.
122.                <% if (message.error) { %>
123.                  <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
124.                    <strong>
125.                      <%= message.error %>
126.                    </strong>
127.                  </div>
128.                <% } %>
129.
```

```
                <% if (message.success) { %>
                  <div class="alert alert-success" role="alert" style="margin-top: 1em;">
                    <strong>
                      <%= message.success %>
                    </strong>
                  </div>
                <% } %>

              </form>

              <h5 style="margin-top: 1em;">Already have an account? <a href="/login">Login...</a></h5>

            </div>

          </div>
        </div>
      </div>

      <div class="row justify-content-center" id="teacher_card">
        <div class="col-md-8">
          <div class="card">

            <div class="card-header" style="background-color: #cccc00;">
              <div class="row">
                <div class="col-md-3">
                  <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Teacher Sign-up</h5>
                </div>
                <div class="col-md-9">
                  <button class="btn btn-secondary switch_to_student" onclick="change_form_login()" style="float: right;">Switch to Student Sign-up</button>
                </div>
              </div>
            </div>

            <div class="card-body">

              <!-- teacher form -->
              <form action="/sign_up/teacher" method="POST" id="teacher_form">

                <!-- teacher's email -->
                <div class="form-group row">
                  <label for="email" class="col-md-3 col-form-label">EMAIL ADDRESS</label>
                  <div class="col-md-8">
                    <!-- automatically check for correctly formatted teacher dc email -->
                    <input type="email" id="email" name="email" pattern="^[A-Za-z]+\.[A-Za-z]+\@dubaicollege.org$" class="form-control" placeholder="teacher_first_name.teacher_last_name@dubaicollege.org" required />
                  </div>
                </div>

                <!-- teacher's class code (3-letter code) -->
                <div class="form-group row">
                  <label for="teacher_initials" class="col-md-3 col-form-label">CLASS CODE</label>
                  <div class="col-md-8">
                    <input type="text" id="teacher_initials" name="teacher_initials" class="form-control" pattern="^[A-Z]{3}$" placeholder="3-letter DC teacher identification" required />
```

```
                    </div>
                  </div>

                  <!-- teacher's password -->
                  <div class="form-group row">
                    <label for="password" class="col-md-3 col-form-label">PASSWORD</label>
                    <div class="col-md-8">
                      <input type="password" id="password" name="password" class="form-control" placeholder="enter a
password" required />
                    </div>
                  </div>

                  <!-- teacher's password confirmation -->
                  <div class="form-group row">
                    <label for="password_confirmation" class="col-md-3 col-form-label">CONFIRM PASSWORD</label>
                    <div class="col-md-8">
                      <input type="password" id="password_confirmation" name="password_confirmation" class="form-
control" placeholder="enter the same password again" required />
                    </div>
                  </div>

                  <!-- submit button -->
                  <div class="row">
                    <div class="col-md-8 offset-md-3">
                      <button type="submit" class="btn btn-primary">SIGN UP</button>
                    </div>
                  </div>

                  <% if (message.error) { %>
                    <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
                      <strong>
                        <%= message.error %>
                      </strong>
                    </div>
                  <% } %>

                  <% if (message.success) { %>
                    <div class="alert alert-success" role="alert" style="margin-top: 1em;">
                      <strong>
                        <%= message.success %>
                      </strong>
                    </div>
                  <% } %>

                </form>

                <h5 style="margin-top: 1em;">Already have an account? <a href="/login">Login...</a></h5>

              </div>

            </div>
          </div>
        </div>

      </div>
```

```
</body>

<% include ../partials/script %>
<% include partials/script %>

</html>

```

~/views/partials/head.ejs

```
<title><%= (typeof title != "undefined" ? title : "") %></title>

<meta charset="utf-8">
<meta name="active-menu" content="<%= (typeof menu_id != "undefined" ? menu_id : "") %>">
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">

<!-- Bootstrap - latest compiled and minified CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">

<!-- jQuery -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<!-- Local files -->
<link rel="stylesheet" href="/stylesheets/style.css">

```

~/views/partials/lmc_script.ejs

```
<!-- Local files -->
<!-- general js file for main visualisation -->
<script type="text/javascript" src="/javascripts/lmc.js"></script>
<!-- leader_line plugin for creating arrows b/w divs -->
<script src="/plugins/leader_line/leader-line.min.js"></script>

```

~/views/partials/main_visualisation_head.ejs

```
<!-- Local files -->
<link rel="stylesheet" href="/stylesheets/main_visualisation_style.css">

```

~/views/partials/main_visualisation.ejs

```
<div class="row">

  <div class="col-md-2">
    <span>Assembly Code Editor</span>
  </div>

  <div class="col-md-4">
    <span>  Processor</span>
```

```
 9.    </div>
10.
11.    <div class="col-md-1"></div>
12.
13.    <div class="col-md-5">
14.        <span>RAM</span>
15.    </div>
16.
17. </div>
18.
19. <div class="row" id="main_visualisation_wrapper_row_1">
20.
21.    <div class="col-md-2">
22.        <div id="code_wrapper">
23.            <div class="line-numbers col-md-2 scroll_together">
24.                <span></span>
25.            </div>
26.            <textarea class="scroll_together" id="code_area" rows=27 wrap="off">
27.        INP
28. loop    OUT
29.        STA var1
30.        SUB var2
31.        STA var1
32.        BRP loop
33.        HLT
34.
35. var2    DAT 1
36. var1    DAT
37.
38. // Counts down in decrements of var2 (1), starting with the inputted number</textarea>
39.        </div>
40.    </div>
41.
42.    <div class="col-md-4">
43.        <div id="CPU_wrapper">
44.
45.            <div class="row">
46.                <div class="col-md-4">
47.                    <span id="general_registers_title">General Purpose Registers</span>
48.                </div>
49.            </div>
50.
51.            <div class="row">
52.
53.                <div class="col-md-4" id="general_register_wrapper">
54.                    <!-- create General Purpose Registers (from 0 to 9) -->
55.                    <% for (var i = 0; i < 10; i++) { %>
56.                        <div id="register_<%= i %>_wrapper">
57.                            <tr>
58.                                <td>Register <%= i.toString() %><input readonly id="general_register_<%= i %>" type="text"
value="0000" /></td>
59.                            </tr>
60.                        </div>
61.                    <% } %>
62.                </div>
```

```

        <div class="col-md-4" id="CPU_column_2">

          <div id="ALU_wrapper">
            Arithemtic Logic Unit
            <input readonly id="ALU" type="text" value="" />
          </div>

          <div id="accumulator_wrapper">
            Accumulator
            <input readonly id="accumulator" type="text" value="00" />
          </div>

          <div id="PC_wrapper">
            Program Counter
            <input readonly id="PC" type="text" value="00" />
          </div>

          <div id="decode_unit_wrapper">
            Decode Unit
            <input readonly id="decode_unit" type="text" value="" />
          </div>

          <div id="CIR_wrapper">
            Current Instruction Register
            <input readonly id="CIR" type="text" value="00" />
          </div>

        </div>

        <div class="col-md-4" id="CPU_column_3">

          <div id="MAR_wrapper">
            Memory Address Register
            <input readonly id="MAR" type="text" value="00" />
          </div>

          <div id="status_register_wrapper">
            Status Register
            <input readonly id="status_register" type="text" value="00" />
          </div>

          <div id="control_unit_wrapper">
            Control Unit
            <input readonly id="control_unit" type="text" value="" />
          </div>

          <div id="clock_wrapper">
            Clock (cycles)
            <input readonly id="clock" type="text" value="0" />
          </div>

          <div id="MBR_wrapper">
            Memory Buffer Register
            <input readonly id="MBR" type="text" value="0000" />
```

```
                </div>

            </div>

            </div>
        </div>
    </div>

    <div class="col-md-1"></div>

    <div class="col-md-5">
        <div id="RAM_wrapper">

            <table id="RAM">
                <!-- create RAM (memory locations from 00 to 99) -->
                <% for (var i = 0; i < 10; i++) { %>
                    <tr>
                        <% for (var j = 0; j < 10; j++) { %>

                            <% var k = (i * 10) + j %>
                            <% var k_str = k.toString() %>

                            <%# for memory locations 0-9, we want to display 00, 01, etc. %>
                            <% while (k_str.length < 2) { %>
                                <% k_str = '0' + k_str %>
                            <% } %>

                            <td id="memory_<%= k_str %>_wrapper"><%= k_str %><input id="memory_location_<%= k %>"
type="text" value="0000" /></td>

                        <% } %>
                    </tr>
                <% } %>
            </table>
        </div>

    </div>
</div>

<div class="row" id="second_title_text_row">

    <div class="col-md-6">
        <span>
            <div class="row">
                <div class="col-md-5">User Controlbox for Visualisation</div>
                <div class="col-md-7"><input type="range" id="clock_speed" min="5" max="5000" value="25"></div>
            </div>
        </span>
    </div>

    <!-- log outputs -->
    <div class="col-md-3">
        <span>FDE Cycle Output</span>
    </div>

```

```
  <div class="col-md-3" id="verbose_output_title">
    <span>Verbose Output</span>
  </div>

</div>

<div class="row" id="main_visualisation_wrapper_row_2">

  <div class="col-md-6" id="controlbox_wrapper">
    <div class="row">

      <div class="col-md-3">
        <div id="input_wrapper">
          <tr>
            <td>Input<input id="input" type="text" value="00" /></td>
          </tr>
        </div>
      </div>

      <div class="col-md-9" id="controlbox_top_two_rows_column_2">
        <div class="row" id="controlbox_row_1_column_2">

          <div class="col-md-3" id="upload_program_wrapper">
            <!-- button acts as label for hidden input file field -->
            <input hidden type="file" id="upload_program_actual" name="Upload Program" accept="text/plain"/>
            <button id="upload_program"><label for="upload_program_actual">Upload Program</label></button>
          </div>

          <div class="col-md-3" id="reset_RAM_wrapper">
            <button id="reset_RAM" onclick="reset_RAM(LMC)">Reset RAM</button>
          </div>

          <div class="col-md-3" id="load_into_RAM_wrapper">
            <button id="load_into_RAM" onclick="load_into_RAM(LMC)">Load into RAM</button>
          </div>

          <div class="col-md-3" id="run_program_wrapper">
            <button id="run_program" onclick="run_program(LMC)">Run Program</button>
          </div>

        </div>

        <div class="row" id="controlbox_row_2_column_2">

          <div class="col-md-4" id="clear_editor_wrapper">
            <button id="clear_editor" onclick="clear_editor()">Clear Editor</button>
          </div>

          <!-- pre-created dropdown logic in lmc.js -->
          <div class="col-md-4" id="pre-created_programs_wrapper">
            <select id="pre-created_programs" name="pre-created_programs">
              <option value="" selected hidden>Pre-created Programs</option>
              <option value="add_2_nums" id="add_2_nums">add 2 nums</option>
              <option value="sub_2_nums" id="sub_2_nums">sub 2 nums</option>
              <option value="mult_2_nums" id="mult_2_nums">mult 2 nums</option>
```

```
                    <option value="count_down" id="count_down">count down</option>
                    <option value="immediate_addressing" id="immediate_addressing">immediate addressing</option>
                    <option value="square_inp_until_0" id="square_inp_until_0">square inp until 0</option>
                    <option value="bubble_sort" id="bubble_sort">bubble sort</option>
                  </select>
                </div>

                <div class="col-md-4" id="step_program_wrapper">
                  <button id="step_program" onclick="step_program(LMC)">Step Program</button>
                </div>

              </div>
            </div>

        </div>

        <div class="row">

            <div class="col-md-3">
              <div id="output_wrapper">
                <tr>
                  <td>Output<input readonly id="output" type="text" value="00" /></td>
                </tr>
              </div>
            </div>

            <div class="col-md-9" id="controlbox_bottom_two_rows_column_2">

              <div class="row" id="how_to_use_this_resource_wrapper">
                <button id="how_to_use_this_resource" onclick="open_link_new_tab('how_to_use')">How to Use this Resource</button>
              </div>

              <div class="row" id="view_instruction_set_wrapper">
                <button id="view_instruction_set" onclick="open_link_new_tab('instruction_set')">View Instruction Set</button>
              </div>

            </div>

        </div>
    </div>

    <div class="col-md-3">
        <div id="FDE_output_wrapper">
            <textarea id="FDE_output" wrap="on">Output of FDE cycle</textarea>
        </div>
    </div>

    <div class="col-md-3">
        <div id="verbose_output_wrapper">
            <textarea id="verbose_output" wrap="on">Verbose Output</textarea>
        </div>
    </div>

```

```
</div>

<% include ./lmc_script %>

```

~/views/partials/menu.ejs

```
<div class="row" id="navbar_row">
  <div class="col-md-12">
    <nav class="navbar navbar-default navbar-expand-sm navbar-dark bg-dark fixed-top">
      <div class="container-fluid navbar_wrapper">

        <!-- hidden form for POST method for logout tab in navbar -->
        <form id="logout_form" name="logout_form" action="/logout" method="post">
          <button type="submit">LOG OUT</button>
        </form>

         <div class="navbar_left">
           <ul class="navbar-nav nav_items_left">
             <li>
               <!-- coloured dot icon (3 diff colours for different possible accounts / no account) - needs to be coded -->
               <!-- tooltip should change depending on account type - no account: red; teacher: red, yellow, green; student: red, green; -->
               <span>
                 <% if (typeof role === "undefined" ) { %>
                   <img class="tooltip-inner" id="navbar_user_indicator" src="/images/red_dot_indicator.png" alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account" />
                 <% } else { %>
                   <% if (role === 'student') { %>
                     <img class="tooltip-inner" id="navbar_user_indicator" src="/images/green_dot_indicator.png" alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account
green: student account" />
                   <% } else { %>
                     <img class="tooltip-inner" id="navbar_user_indicator" src="/images/yellow_dot_indicator.png" alt="User Indicator" data-toggle="tooltip" data-placement="auto" title="red: no account
yellow: teacher account
green: student account" />
                   <% } %>
                 <% } %>
               </span>
             </li>
             <li>
               <span id="navbar_title">Assembly Code Visualiser</span>
             </li>
           </ul>
         </div>

         <div class="collapse navbar-collapse" id="main_navbar">

           <div class="navbar_left">
             <!-- main options on the left -->
             <ul class="navbar-nav nav_items_left" id="navbar_options">
               <li class="nav-item">
                 <a class="nav-link" id="home" href="/">HOME</a>
```

```
                </li>
                <li class="nav-item">
                  <a class="nav-link" id="instruction_set" href="/instruction_set">INSTRUCTION SET</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" id="how_to_use" href="/how_to_use">HOW TO USE THIS RESOURCE</a>
                </li>
                <% if (typeof role==="undefined" ) { %>
                <% } else { %>
                  <% if (role===='student' ) { %>
                    <li class="nav-item">
                      <a class="nav-link" id="view_set_challenges" href="/view_set_challenges">VIEW SET
CHALLENGES</a>
                    </li>
                  <% } else { %>
                    <li class="nav-item">
                      <a class="nav-link" id="create_challenges" href="/create_challenges">CREATE CHALLENGES</a>
                    </li>
                    <li class="nav-item">
                      <a class="nav-link" id="view_student_challenges" href="/view_student_challenges">VIEW STUDENT
CHALLENGES</a>
                    </li>
                  <% } %>
                <% } %>
              </ul>
            </div>

            <div class="navbar_right">
              <!-- login / sign up button on the right -->
              <ul class="navbar-nav nav_items_right">
                <% if (typeof role === "undefined" ) { %>
                  <li class="navbar-item">
                    <a class="nav-link navbar_right" id="login" href="/login">LOGIN</a>
                  </li>
                  <li class="navbar-item">
                    <a class="nav-link navbar_right" id="sign_up" href="/sign_up">SIGN UP</a>
                  </li>
                <% } else { %>
                  <li class="navbar-item">
                    <a class="nav-link navbar_right" href="#" onclick="post_logout()">LOG OUT</a>
                  </li>
                <% } %>
              </ul>
            </div>
          </div>

          <!-- collapsible menu w/ 3 horizontal lines for when screen too small -->
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#main_navbar"
            aria-controls="main_navbar" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>

        </div>
      </nav>
    </div>
```

```
</div>

```

~/views/partials/script.ejs

```
<!-- Bootstrap - latest compiled JavaScript -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-A3rJD856KowSb7dwlZdYEkO39Gagi7vIsF0jrRAoQmDKKtQBHUuLZ9AsSv4jD4Xa"
    crossorigin="anonymous"></script>

<!-- Local files -->
<script type="text/javascript" src="/javascripts/menu.js"></script>

```

~/views/student_challenges/partials/script_toggle.ejs

```
<!-- Local files -->
<script type="text/javascript" src="/javascripts/student_toggles.js"></script>

```

~/views/student_challenges/complete_challenges.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <div class="row">
            <div class="col-md-12">

                <form action="/view_set_challenges/complete_challenge/complete" method="post">

                    <!-- challenge title -->
                    <div class="row">
                        <div class="col-md-12">
                            <label for="challenge_title">CHALLENGE TITLE</label>
                            <textarea id="challenge_title" name="challenge_title" wrap="off"
                                placeholder="Generic Challenge Title" readonly
                                style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
                        </div>
                    </div>

                    <!-- challenge description -->
                    <div class="row">
                        <div class="col-md-12">
                            <label for="challenge_description">CHALLENGE DESCRIPTION</label>
```

```
                <textarea id="challenge_description" name="challenge_description" wrap="off"
                  placeholder="Generic Challenge Description" readonly
                  style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
              </div>
            </div>

            <!-- challenge answer -->
            <div class="row">
              <div class="col-md-12">
                <label for="challenge_answer">CHALLENGE ANSWER</label>
                <textarea id="challenge_answer" name="challenge_answer" wrap="off"
                  placeholder="Generic Challenge Answer" required
                  style="overflow-y: scroll; height: 20em; width: 100%;"></textarea>
              </div>
            </div>

            <!-- submit button -->
            <button type="submit" value="Complete" class="btn btn-primary">Complete</button>

            <!-- hidden to be retrievable in js file -->
            <div class="row" hidden aria-hidden="true">
              <div class="col-md-12">
                <label for="assigned_challenge_id">ASSIGNED CHALLENGE ID</label>
                <textarea id="assigned_challenge_id"
                  name="assigned_challenge_id"><%= assigned_challenge_id %></textarea>
              </div>
            </div>

          </form>

          <% if (message.error) { %>
            <div class="alert alert-danger" role="alert">
              <strong>
                <%= message.error %>
              </strong>
            </div>
          <% } %>

          <% if (message.success) { %>
            <div class="alert alert-success" role="alert">
              <strong>
                <%= message.success %>
              </strong>
            </div>
          <% } %>

        </div>
      </div>

    </div>
  </body>

  <% include ../partials/script %>

  </html>
```

```
87.
```

~/views/student_challenges/view_completed_challenge.ejs

```
1. <!DOCTYPE html>
2. <html lang="en">
3.
4. <head>
5.    <% include ../partials/head %>
6. </head>
7.
8. <body>
9.    <div class="container-fluid">
10.
11.       <% include ../partials/menu %>
12.
13.       <div class="row">
14.          <div class="col-md-12">
15.
16.                <!-- challenge title -->
17.                <div class="row">
18.                   <div class="col-md-12">
19.                      <label for="challenge_title">CHALLENGE TITLE</label>
20.                      <textarea id="challenge_title" name="challenge_title" wrap="off"
21.                         placeholder="Generic Challenge Title" readonly
22.                         style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
23.                   </div>
24.                </div>
25.
26.                <!-- challenge description -->
27.                <div class="row">
28.                   <div class="col-md-12">
29.                      <label for="challenge_description">CHALLENGE DESCRIPTION</label>
30.                      <textarea id="challenge_description" name="challenge_description" wrap="off"
31.                         placeholder="Generic Challenge Description" readonly
32.                         style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
33.                   </div>
34.                </div>
35.
36.                <!-- challenge answer -->
37.                <div class="row">
38.                   <div class="col-md-12">
39.                      <label for="challenge_answer">CHALLENGE ANSWER</label>
40.                      <textarea id="challenge_answer" name="challenge_answer" wrap="off" placeholder="Generic Challenge
Answer"
41.                         readonly style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_answer %></textarea>
42.                   </div>
43.                </div>
44.
45.                <button onclick="location.href='../';" class="btn btn-primary">Go Back</button>
46.
47.          </div>
48.       </div>
49.
```

```
    </div>
</body>

<% include ../partials/script %>

</html>

```

~/views/student_challenges/view_set_challenges.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
    <% include ./partials/script_toggle %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <% if (typeof all_completed_challenges === "undefined" && typeof all_missed_challenges === "undefined" && typeof all_challenges_to_do === "undefined") { %>
            <h4>Go back to the <a href="../">Home</a> Page</h4>
        <% } else { %>

            <!-- CHALLENGES TO DO -->
            <div class="row">
                <div class="col-md-12">

                    <div class="row justify-content-center">
                        <div class="col-md-12">
                            <div class="card" style="margin-top: 0.5em;">

                                <div class="card-header" style="background-color: #ADD8E6;">
                                    <div class="row">
                                        <div class="col-md-4">
                                            <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenges to Complete</h5>
                                        </div>
                                        <div class="col-md-5"></div>
                                        <div class="col-md-3">
                                            <button class="btn btn-secondary switch_to_student" onclick="toggle_display_challenges()" style="float: right;" id="display_challenges_toggle">Hide Challenges</button>
                                        </div>
                                    </div>
                                </div>

                                <div class="card-body" id="active_challenges_card_body" style="background-color: #ADD8E6;">

                                    <% if (typeof all_challenges_to_do === "undefined" || all_challenges_to_do.length == 0 || all_challenges_to_do == []) { %>
```

```
                        <h5>No Challenges set</h5>
                    <% } else { %>

                        <table style="width: 100%;" id="active_challenges">

                            <tr>
                                <th>Challenge Title</th>
                                <th>Challenge Description</th>
                                <th>Due Date</th>
                                <th></th>
                            </tr>

                            <% all_challenges_to_do.forEach((element, index)=> { %>
                                <tr>
                                    <td style="word-wrap: break-word;">
                                        <%= element[0].split('\n')[1].toString() %>
                                    </td>
                                    <td style="word-wrap: break-word;">
                                        <%= element[0].split('\n')[3].toString() %>
                                    </td>
                                    <td>
                                        <%= element[2].toString() %>
                                    </td>
                                    <td>
                                        <form action="/view_set_challenges/complete_challenge/<%= element[1].toString() %>"
method="post">
                                            <button type="submit" value="complete_challenge_<%= element[1].toString() %>"
class="btn btn-secondary switch_to_teacher">Complete Challenge</button>
                                        </form>
                                    </td>
                                </tr>
                            <% }); %>

                        </table>

                    <% } %>

                </div>

            </div>
        </div>
    </div>

    </div>
</div>

<!-- CHALLENGES FINISHED IN TIME -->
<div class="row">
    <div class="col-md-12">

        <div class="row justify-content-center">
            <div class="col-md-12">
                <div class="card" style="margin-top: 0.5em;">

                    <div class="card-header" style="background-color: #E6E6FA;">
```

```
                    <div class="row">
                      <div class="col-md-4">
                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Completed Challenges</h5>
                      </div>
                      <div class="col-md-5"></div>
                      <div class="col-md-3">
                          <button class="btn btn-secondary switch_to_student" onclick="toggle_display_completed()"
                            style="float: right;" id="display_completed_toggle">Show Challenges</button>
                        </div>
                    </div>
                  </div>

                  <div class="card-body" id="completed_card_body" style="background-color: #E6E6FA;">

                    <% if (typeof all_completed_challenges===="undefined" || all_completed_challenges.length==0 || all_completed_challenges==[]) { %>
                        <h5>No Completed Challenges</h5>
                    <% } else { %>

                      <table style="width: 100%;" id="completed">

                        <tr>
                          <th style="width: 25%;">Challenge Title</th>
                          <th style="width: 45%;">Challenge Description</th>
                          <th style="width: 10%;">Due Date</th>
                          <th style="width: 10%;">Completion Date</th>
                          <th style="width: 10%;"></th>
                        </tr>

                        <% all_completed_challenges.forEach((element, index)=> { %>
                          <tr>
                            <td style="word-wrap: break-word;">
                              <%= element[0].split('\n')[1].toString() %>
                            </td>
                            <td style="word-wrap: break-word;">
                              <%= element[0].split('\n')[3].toString() %>
                            </td>
                            <td>
                              <%= element[2].toString() %>
                            </td>
                            <td>
                              <%= element[3].toString() %>
                            </td>
                            <td>
                              <form action="/view_set_challenges/view_completed/<%= element[1].toString() %>" method="post">
                                <button type="submit" value="view_completed_<%= element[1].toString() %>" class="btn btn-secondary switch_to_student">View Answer</button>
                              </form>
                            </td>
                          </tr>
                        <% }); %>

                      </table>
```

```
145.
146.                        <% } %>
147.
148.                    </div>
149.
150.                </div>
151.            </div>
152.        </div>
153.
154.        </div>
155.    </div>
156.
157.    <!-- CHALLENGES NOT FINISHED IN TIME -->
158.    <div class="row">
159.        <div class="col-md-12">
160.
161.            <div class="row justify-content-center">
162.                <div class="col-md-12">
163.                    <div class="card" style="margin-top: 0.5em;">
164.
165.                        <div class="card-header" style="background-color: #FAA0A0;">
166.                            <div class="row">
167.                                <div class="col-md-5">
168.                                    <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Missed Challenges</h5>
169.                                </div>
170.                                <div class="col-md-4"></div>
171.                                <div class="col-md-3">
172.                                    <button class="btn btn-secondary switch_to_student" onclick="toggle_display_missed()"
style="float: right;" id="display_missed_toggle">Show Challenges</button>
173.                                </div>
174.                            </div>
175.                        </div>
176.
177.                        <div class="card-body" id="missed_card_body" style="background-color: #FAA0A0;">
178.
179.                            <% if (typeof all_missed_challenges===="undefined" || all_missed_challenges.length==0 ||
all_missed_challenges==[]) { %>
180.                                <h5>No Completed Challenges</h5>
181.                            <% } else { %>
182.
183.                                <table style="width: 100%;" id="missed">
184.
185.                                    <tr>
186.                                        <th>Challenge Title</th>
187.                                        <th>Challenge Description</th>
188.                                        <th>Due Date</th>
189.                                    </tr>
190.
191.                                    <% all_missed_challenges.forEach((element, index)=> { %>
192.                                        <tr>
193.                                            <td style="word-wrap: break-word;">
194.                                                <%= element[0].split('\n')[1].toString() %>
195.                                            </td>
196.                                            <td style="word-wrap: break-word;">
197.                                                <%= element[0].split('\n')[3].toString() %>
```

```
                                    </td>
                                    <td>
                                      <%= element[2].toString() %>
                                    </td>
                                  </tr>
                                <% }); %>

                              </table>

                            <% } %>

                          </div>

                        </div>
                      </div>
                    </div>

                  </div>
                </div>

            <% } %>

        </div>
</body>

<% include ../partials/script %>

</html>

```

~/views/teacher_challenges/partials/head.ejs

```
<!-- <link href="plugins/chosen/docsupport/style.css" rel="stylesheet" /> -->
<link href="/plugins/chosen/chosen.css" rel="stylesheet" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.10.2/jquery.js"></script>
<script src="/plugins/chosen/chosen.jquery.js"></script>

```

~/views/teacher_challenges/script_toggle.ejs

```
<!-- Local files -->
<script type="text/javascript" src="/javascripts/active_challenges_display_toggler.js"></script>

```

~/views/teacher_challenges/script.ejs

```
<!-- ensure that the page reloads on first redirect so all data from db is up-to-date -->
<!-- <script type="text/javascript">
  $(document).ready(function () {
    //Check if the current URL contains '#'

    //if (document.URL.slice(-8, document.URL.length - 1) == '/assign' || document.URL.slice(-8, document.URL.length - 1) === '/assign') {
```

```
7.      //  url = document.URL.slice(0, document.URL.length - 8);
8.      //  location.replace(url);
9.      //};
10.
11.      if (document.URL.indexOf("#") == -1) {
12.         // Set the URL to whatever it was plus "#".
13.         url = document.URL + "#";
14.         location = "#";
15.
16.         //Reload the page
17.         location.reload(true);
18.      }
19.   });
20. </script> -->
21.
22. <!-- dynamically update hidden textarea for selected students checked to assign a challenge to -->
23. <!-- <script type="text/javascript">
24.
25.   window.addEventListener('click', function () {
26.
27.      var checkboxes = document.getElementsByName('student');
28.      var result = '';
29.
30.      for (var i = 0; i < checkboxes.length; i++) {
31.         if (checkboxes[i].checked) {
32.            result += checkboxes[i].value + " | ";
33.         };
34.      };
35.
36.      document.getElementById('students_selected').value = result;
37.
38.   });
39.
40. </script> -->
41.
42. <!-- nicer multi-select + add selected vals to hidden textarea with students to assign to -->
43. <script type="text/javascript">
44.   $(document).ready(function () {
45.
46.      $(".chosen-select").chosen({
47.         no_results_text: "no student with the name:",
48.         width: "100%"
49.      });
50.
51.      $(".chosen-select").on('change', function(evt, params) {
52.
53.         if (params.selected) {
54.            document.getElementById('students_selected_2').value += ',' + params.selected.toString() + ',';
55.         } else if (params.deselected) {
56.            document.getElementById('students_selected_2').value =
document.getElementById('students_selected_2').value.split(',').filter(element => element !== '').filter(element => element !==
params.deselected.toString());
57.         } else {};
58.
59.         //document.getElementById('students_selected_2').value += Object.keys(params);
```

```

        });

    });
</script>

<!-- select all / deselect all buttons -->
<script type="text/javascript">
    $(document).ready(function () {
        $('.chosen-toggle').each(function (index) {
            $(this).on('click', function () {
                // alert($(this).parent().parent().find('.students').text());
                $(this).parent().parent().find('.students').prop('selected', $(this).hasClass('select')).parent().trigger('chosen:updated');
            });
        });
    });
</script>

<!-- add/remove from hidden textarea for students to assign to when select/deselect all clicked -->
<script>

    function select_all(index) {

        var class_students_list = JSON.parse('<%- JSON.stringify(class_students) %>');

        var students_to_select = class_students_list[index][2];
        var students_already = document.getElementById('students_selected_2').value.split(',').filter(element => element !== '');

        var unique_students = [...new Set([...students_to_select.toString().split(','), ...students_already.toString().split(',')])];

        document.getElementById('students_selected_2').value = ',' + unique_students.toString() + ',';

    };

    function deselect_all(index) {

        var class_students_list = JSON.parse('<%- JSON.stringify(class_students) %>');

        var students_to_deselect = class_students_list[index][2];
        var students_already = document.getElementById('students_selected_2').value.split(',').filter(element => element !== '');

        var remaining_students = students_already.toString().split(',').filter(element => !students_to_deselect.toString().split(',').includes(element));

        document.getElementById('students_selected_2').value = ',' + remaining_students.toString() + ',';

    };

</script>

```

~/views/teacher_challenges/style.ejs

```
<style>

  .select,
  .chosen-select,
  .chosen-select-no-single,
  .chosen-select-no-results,
  .chosen-select-deselect,
  .chosen-select-rtl,
  .chosen-select-width {
      width: 350px !important;
   }

</style>

```

~/views/teacher_challenges/create_challenges.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
   <% include ../partials/head %>
   <% include ./partials/script %>
   <% include ./partials/style %>
   <% include ./partials/head %>
   <% include ./partials/script_toggle %>
</head>

<body>
   <div class="container-fluid">

      <% include ../partials/menu %>

      <!-- display created challenges -->
      <div class="row justify-content-center">
         <div class="col-md-12">
            <div class="card" style="margin-top: 0.5em;">

               <div class="card-header" style="background-color: #E6E6FA;">
                  <div class="row">
                     <div class="col-md-3">
                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Created Challenges</h5>
                     </div>
                     <div class="col-md-6"></div>
                     <div class="col-md-3">
                        <button class="btn btn-secondary switch_to_student" onclick="toggle_display_challenges()" style="float: right;" id="display_challenges_toggle">Show Challenges</button>
                     </div>
                  </div>
               </div>

```

```
            <div class="card-body" id="active_challenges_card_body" style="background-color: #E6E6FA;">

              <% if (typeof challenges_to_display === "undefined") { %>
                    <h4><a href="../create_challenges">Reload</a> Page</h4>
              <% } else  { %>
                <% if (challenges_to_display.length == 0 || challenges_to_display == []) { %>
                  <h4>No Challenges Created Yet</h4>
                  <!-- create a new challenge -->
                  <form action="/create_challenges/new_challenge" method="post">
                    <!-- submit button -->
                    <button type="submit" value="Create New Challenge" class="btn btn-secondary"
                      id="create_challenges_create_btn">CREATE NEW CHALLENGE</button>
                  </form>
                <% } else { %>

                  <table id="active_challenges" style="width: 100%; table-layout: fixed;">

                    <tr>
                      <th style="width: 25%;">Challenge Title</th>
                      <th style="width: 60%;">Challenge Description</th>
                      <th colspan="2" style="width: 15%;">
                        <!-- create a new challenge -->
                        <form action="/create_challenges/new_challenge" method="post">
                          <!-- submit button -->
                          <button type="submit" value="Create New Challenge" class="btn btn-secondary"
id="create_challenges_create_btn">CREATE NEW CHALLENGE</button>
                        </form>
                      </th>
                    </tr>

                    <% challenges_to_display.forEach((element, index) => { %>
                      <tr>
                        <td style="word-wrap: break-word;">
                          <%= element[0].split('\n')[1].toString() %>
                        </td>
                        <td style="word-wrap: break-word;">
                          <%= element[0].split('\n')[3].toString() %>
                        </td>
                        <!-- edit challenge -->
                        <td>
                          <form action="/create_challenges/edit/<%= element[1].toString() %>" method="post">
                            <button type="submit" value="edit_<%= element[1].toString() %>" class="btn btn-secondary
create_challenges_edit_btn">Edit</button>
                          </form>
                        </td>
                        <!-- create challenge -->
                        <td>
                          <form action="/create_challenges/delete/<%= element[1].toString() %>" method="post">
                            <button type="submit" value="delete_<%= element[1].toString() %>" class="btn btn-
secondary create_challenges_delete_btn">Delete</button>
                          </form>
                        </td>
                      </tr>
                    <% }); %>
                  </table>
```

```
86.
87.                     <% } %>
88.                  <% } %>
89.
90.               </div>
91.
92.            </div>
93.         </div>
94.      </div>
95.
96.      <!-- assign challenges -->
97.      <div class="row justify-content-center">
98.         <div class="col-md-12">
99.            <div class="card" style="margin-top: 0.5em;">
100.
101.               <div class="card-header" style="background-color: #ADD8E6;">
102.                  <div class="row">
103.                     <div class="col-md-3">
104.                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Assign Challenges</h5>
105.                     </div>
106.                     <div class="col-md-6"></div>
107.                     <div class="col-md-3">
108.                        <button class="btn btn-secondary switch_to_student" onclick="toggle_assign_challenges()"
109.                           style="float: right;" id="assign_challenges_toggle">Hide Assign Challenges</button>
110.                     </div>
111.                  </div>
112.               </div>
113.
114.               <div class="card-body" id="assign_challenges_card_body" style="background-color: #ADD8E6;">
115.
116.                  <form action="/create_challenges/assign" method="POST" id="assign_challenge_form">
117.
118.                     <% if (typeof challenges_to_display===="undefined" ) { %>
119.
120.                     <% } else { %>
121.
122.                        <div class="form-group row">
123.                           <label class="col-md-3 col-form-label">CHALLENGE TITLE</label>
124.                           <div class="col-md-8">
125.
126.                              <select id="assign_challenge_dd" name="assign_challenge_dd" class="form-control" required>
127.                                 <option value="" selected hidden>select your challenge</option>
128.                                 <% var challenge_titles=[]; %>
129.                                 <% challenges_to_display.forEach((element, index)=> { %>
130.                                    <% challenge_titles.push(element[0].split('\n')[1].toString()); %>
131.                                 <% }); %>
132.
133.                                 <% if (typeof challenge_titles===="undefined" ) { %>
134.
135.                                    <option value="no_challenge">you haven't created any challenges</option>
136.
137.                                 <% } else { %>
138.
139.                                    <% for (var challenge_titles_counter=0; challenge_titles_counter < challenge_titles.length;
challenge_titles_counter++) { %>
```

```
                                <option value="<%= challenge_titles[challenge_titles_counter] %>">
                                  <%= challenge_titles[challenge_titles_counter] %>
                                </option>
                              <% } %>

                            <% } %>
                          </select>

                        </div>
                      </div>

                    <% } %>

                    <% if (typeof class_students===="undefined" ) { %>

                    <% } else { %>
                      <!-- DROPDOWN FOR CLASSES WITH STUDENTS -->
                      <% class_students.forEach((element, index)=> { %>

                        <div class="form-group row">

                          <label class="col-md-3 col-form-label"><%= element[0].toString() %></label>

                          <div class="col-md-5">
                            <select id="class_<%= element[0].toString() %>" data-placeholder="select <%= element[0].toString() %> students" multiple class="form-control chosen-select">
                              <% if (element[2].length===0 || !element[2].length) { %>
                                <option value="" disabled>no students in this class</option>
                              <% } else { %>
                                <% element[2].toString().split(',').forEach((element_2, index_2)=> { %>
                                  <option value="<%= element_2 %>" class="students">
                                    <%= element_2 %>
                                  </option>
                                <%}); %>
                              <% } %>
                            </select>
                          </div>


                          <!-- only show select all / deselect all buttons if there are students in that class -->
                          <% if (element[2].length===0 || !element[2].length) { %>
                            <div class="col-md-2"></div>
                            <div class="col-md-2"></div>
                          <% } else { %>
                              <div class="col-md-2">
                                <button type="button" id="select_all_<%= element[0].toString() %>" class="chosen-toggle select btn btn-secondary switch_to_student" onclick="select_all('<%- index.toString() %>')" style="width: 100% !important;">Select all</button>
                              </div>
                              <div class="col-md-2">
                                <button type="button" class="chosen-toggle deselect btn btn-secondary create_challenges_delete_btn" onclick="deselect_all('<%- index.toString() %>')" style="width: 100% !important;">Deselect all</button>
                              </div>
                          <% } %>
```

```
190.
191.                        </div>
192.
193.                    <% }); %>
194.                <% } %>
195.
196.                <div class="form-group row">
197.                    <label class="col-md-3 col-form-label">DUE DATE</label>
198.                    <div class="col-md-8">
199.                        <input id="due_date" type="datetime-local" name="due_date" class="form-control" required />
200.                    </div>
201.
202.                    <div class="col-md-6"></div>
203.
204.                </div>
205.
206.                <!-- submit button -->
207.                <div class="row">
208.                    <div class="col-md-8 offset-md-3" style="margin-top: 0.75em;">
209.                        <button type="submit" class="btn btn-primary">ASSIGN</button>
210.                    </div>
211.                </div>
212.
213.                <!-- error message -->
214.                <% if (message.error) { %>
215.                    <div class="alert alert-danger" role="alert" style="margin-top: 1em;">
216.                        <strong>
217.                            <%= message.error %>
218.                        </strong>
219.                    </div>
220.                <% } %>
221.
222.                <% if (message.success) { %>
223.                    <div class="alert alert-success" role="alert" style="margin-top: 1em;">
224.                        <strong>
225.                            <%= message.success %>
226.                        </strong>
227.                    </div>
228.                <% } %>
229.
230.                <div class="row" hidden aria-hidden="true">
231.                    <div class="col-md-12">
232.                        <label for="class_students">class_students</label>
233.                        <textarea id="class_students" name="class_students"><% class_students.forEach((element, index) => {
%><%= element + '\n'%><%}); %></textarea>
234.                    </div>
235.                </div>
236.
237.                <div class="row" hidden aria-hidden="true">
238.                    <div class="col-md-12">
239.                        <label for="challenge_title_match_id">challenge_title_match_id</label>
240.                        <textarea id="challenge_title_match_id" name="challenge_title_match_id"><%-
challenge_title_match_id %></textarea>
241.                    </div>
242.                </div>
```

```
243.
244.                    <div class="row" hidden aria-hidden="true">
245.                      <div class="col-md-12">
246.                        <label for="challenges_to_display">challenges_to_display</label>
247.                        <textarea id="challenges_to_display" name="challenges_to_display"><%- challenges_to_display
%></textarea>
248.                      </div>
249.                    </div>
250.
251.                    <div class="row" hidden aria-hidden="true">
252.                      <div class="col-md-12">
253.                        <label for="students_selected_2">SELECTED STUDENTS TO ASSIGN CHALLENGE TO</label>
254.                        <textarea id="students_selected_2" name="students_selected_2"></textarea>
255.                      </div>
256.                    </div>
257.
258.                    <!-- hidden to be retrievable in js file -->
259.                    <div class="row" hidden aria-hidden="true">
260.                      <div class="col-md-12">
261.                        <label for="challenge_title_match_id">SELECTED CHALLENGE TO ASSIGN ID</label>
262.                        <% if (typeof challenges_title_match_id=="undefined" ) { %>
263.                          <textarea id="challenge_title_match_id" name="challenge_title_match_id"></textarea>
264.                        <% } else { %>
265.                          <textarea id="challenge_title_match_id" name="challenge_title_match_id"><%
challenge_title_match_id.forEach(element => { %><%= element[0] + 'thisisaveryspeficbreak' + element[1] +
'thisisaveryspeficbreak' %><% }); %></textarea>
266.                        <% } %>
267.                      </div>
268.                    </div>
269.
270.                  </form>
271.
272.                </div>
273.
274.              </div>
275.            </div>
276.          </div>
277.
278.        </div>
279. </body>
280.
281. <% include ../partials/script %>
282.
283. </html>
284.
```

~/views/teacher_challenges/edit_challenge.ejs

```
1. <!DOCTYPE html>
2. <html lang="en">
3.
4. <head>
5.     <% include ../partials/head %>
6. </head>
```

```
7.
8. <body>
9.    <div class="container-fluid">
10.
11.       <% include ../partials/menu %>
12.
13.          <div class="row">
14.             <div class="col-md-12">
15.
16.                <form action="/create_challenges/edit/update" method="post">
17.
18.                   <!-- challenge title -->
19.                   <div class="row">
20.                      <div class="col-md-12">
21.                         <label for="challenge_title">CHALLENGE TITLE</label>
22.                         <textarea id="challenge_title" name="challenge_title" wrap="off"
23.                            placeholder="Generic Challenge Title" required
24.                            style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
25.                      </div>
26.                   </div>
27.
28.                   <!-- challenge description -->
29.                   <div class="row">
30.                      <div class="col-md-12">
31.                         <label for="challenge_description">CHALLENGE DESCRIPTION</label>
32.                         <textarea id="challenge_description" name="challenge_description" wrap="off"
33.                            placeholder="Generic Challenge Description" required
34.                            style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
35.                      </div>
36.                   </div>
37.
38.                   <!-- submit button -->
39.                   <button type="submit" value="Update" class="btn btn-primary">UPDATE</button>
40.
41.                   <!-- hidden to be retrievable in js file -->
42.                   <div class="row" hidden aria-hidden="true">
43.                      <div class="col-md-12">
44.                         <label for="challenge_teacher_id">CHALLENGE TEACHER ID</label>
45.                         <textarea id="challenge_teacher_id" name="challenge_teacher_id"><%= challenge_teacher_id
%></textarea>
46.                      </div>
47.                   </div>
48.
49.                   <!-- hidden to be retrievable in js file -->
50.                   <div class="row" hidden aria-hidden="true">
51.                      <div class="col-md-12">
52.                         <label for="challenge_title_og">CHALLENGE TITLE</label>
53.                         <textarea id="challenge_title_og" name="challenge_title_og" wrap="off" placeholder="Generic Challenge
Title" required
54.                            style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
55.                      </div>
56.                   </div>
57.
58.                   <!-- hidden to be retrievable in js file -->
59.                   <div class="row" hidden aria-hidden="true">
```

```
                    <div class="col-md-12">
                      <label for="challenge_description_og">CHALLENGE DESCRIPTION</label>
                      <textarea id="challenge_description_og" name="challenge_description_og" wrap="off"
                        placeholder="Generic Challenge Description" required
                        style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
                    </div>
                  </div>

                </form>

                <% if (message.error) { %>
                  <div class="alert alert-danger" role="alert">
                    <strong>
                      <%= message.error %>
                    </strong>
                  </div>
                <% } %>

                <% if (message.success) { %>
                  <div class="alert alert-success" role="alert">
                    <strong>
                      <%= message.success %>
                    </strong>
                  </div>
                <% } %>

              </div>
            </div>
</body>

<% include ../partials/script %>

</html>

```

~/views/teacher_challenges/more_information.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
    <% include ./partials/script_toggle %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <!-- CHALLENGE INFO -->
        <div class="row">
            <div class="col-md-12">

```

```
18.             <div class="row justify-content-center">
19.               <div class="col-md-12">
20.                 <div class="card" style="margin-top: 0.5em;">
21.
22.                   <div class="card-header" style="background-color: #DCDCDC;">
23.                     <div class="row">
24.                       <div class="col-md-4">
25.                         <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenge Information</h5>
26.                       </div>
27.                       <div class="col-md-5"></div>
28.                       <div class="col-md-3">
29.                         <button class="btn btn-secondary switch_to_student" onclick="toggle_challenge_info()" style="float:
right;" id="display_challenge_info">Hide Information</button>
30.                       </div>
31.                     </div>
32.                   </div>
33.
34.                   <div class="card-body" id="challenge_info" style="background-color: #DCDCDC;">
35.
36.                     <div class="row">
37.                       <!-- challenge title -->
38.                       <div class="col-md-4">
39.                         <label for="challenge_title">CHALLENGE TITLE</label>
40.                         <textarea id="challenge_title" name="challenge_title" wrap="off" placeholder="Generic Challenge
Title" readonly
41.                           style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_title %></textarea>
42.                       </div>
43.                       <!-- challenge description -->
44.                       <div class="col-md-8">
45.                         <label for="challenge_description">CHALLENGE DESCRIPTION</label>
46.                         <textarea id="challenge_description" name="challenge_description" wrap="off"
47.                           placeholder="Generic Challenge Description" readonly
48.                           style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
49.                       </div>
50.                     </div>
51.
52.                   </div>
53.
54.                 </div>
55.               </div>
56.             </div>
57.
58.           </div>
59.         </div>
60.
61.         <!-- STUDENTS COMPLETED -->
62.         <div class="row">
63.           <div class="col-md-12">
64.
65.             <div class="row justify-content-center">
66.               <div class="col-md-12">
67.                 <div class="card" style="margin-top: 0.5em;">
68.
69.                   <div class="card-header" style="background-color: #E6E6FA;">
70.                     <div class="row">
```

```
                    <div class="col-md-4">
                      <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Students that have Submitted</h5>
                    </div>
                    <div class="col-md-5"></div>
                    <div class="col-md-3">
                      <button class="btn btn-secondary switch_to_student" onclick="toggle_students_have_submitted()"
                        style="float: right;" id="display_students_have_submitted">Show Students</button>
                    </div>
                  </div>
                </div>

                <div class="card-body" id="students_have_submitted" style="background-color: #E6E6FA;">

                  <% if (typeof students_solved === "undefined" || students_solved == [] || students_solved.length == 0) { %>
                    <h5>No Students have solved this Challenge yet</h5>
                  <% } else { %>

                    <table style="width: 100%;" id="students_have_submitted_table">

                      <tr>
                        <th>Student ID</th>
                        <th>Due Date</th>
                        <th>Completion Date</th>
                        <th></th>
                      </tr>

                      <% students_solved.forEach((element, index)=> { %>
                        <tr>
                          <td>
                            <%= element[0].toString() %>
                          </td>
                          <td>
                            <%= element[2].toString() %>
                          </td>
                          <td>
                            <%= element[3].toString() %>
                          </td>
                          <td>
                            <form action="/view_student_challenges/more_information/<%= challenge_teacher_id.toString() %>/view_answer/<%= element[1].toString() %>" method="post">
                              <button type="submit" value="view_answer" class="btn btn-secondary switch_to_student">See Solution</button>
                            </form>
                          </td>
                        </tr>
                      <% }); %>

                    </table>

                  <% } %>

                </div>

```

```
                </div>
              </div>
            </div>



        </div>
      </div>

      <!-- STUDENTS NOT COMPLETED BUT BEFORE DEADLINE -->
      <div class="row">
        <div class="col-md-12">

            <div class="row justify-content-center">
              <div class="col-md-12">
                <div class="card" style="margin-top: 0.5em;">

                  <div class="card-header" style="background-color: #ADD8E6;">
                    <div class="row">
                      <div class="col-md-4">
                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Students yet to Submit</h5>
                      </div>
                      <div class="col-md-5"></div>
                      <div class="col-md-3">
                        <button class="btn btn-secondary switch_to_student" onclick="toggle_students_yet_to()"
style="float: right;" id="display_students_yet_to">Show Students</button>
                      </div>
                    </div>
                  </div>

                  <div class="card-body" id="students_yet_to" style="background-color: #ADD8E6;">
                    <% if (typeof students_not_solved_before_due===="undefined" || students_not_solved_before_due==[]
|| students_not_solved_before_due.length==0) { %>
                      <h5>No Students left to Submit</h5>
                    <% } else { %>

                      <table style="width: 100%;" id="students_yet_to_table">

                        <tr>
                          <th>Student ID</th>
                          <th>Due Date</th>
                        </tr>

                        <% students_not_solved_before_due.forEach((element, index)=> { %>
                          <tr>
                            <td>
                              <%= element[0].toString() %>
                            </td>
                            <td>
                              <%= element[2].toString() %>
                            </td>
                          </tr>
                        <% }); %>

                      </table>
```

```
175.
176.                    <% } %>
177.                  </div>
178.
179.                </div>
180.              </div>
181.            </div>
182.
183.          </div>
184.        </div>
185.
186.        <!-- STUDENTS NOT COMPLETED AND AFTER DEADLINE -->
187.        <div class="row">
188.          <div class="col-md-12">
189.
190.            <div class="row justify-content-center">
191.              <div class="col-md-12">
192.                <div class="card" style="margin-top: 0.5em;">
193.
194.                  <div class="card-header" style="background-color: #FAA0A0;">
195.                    <div class="row">
196.                      <div class="col-md-5">
197.                        <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Students that missed the
deadline</h5>
198.                      </div>
199.                      <div class="col-md-4"></div>
200.                      <div class="col-md-3">
201.                        <button class="btn btn-secondary switch_to_student" onclick="toggle_students_missed()"
style="float: right;" id="display_students_missed">Show Students</button>
202.                      </div>
203.                    </div>
204.                  </div>
205.
206.                  <div class="card-body" id="students_missed" style="background-color: #FAA0A0;">
207.
208.                    <% if (typeof students_not_solved_after_due=="undefined" || students_not_solved_after_due==[] ||
students_not_solved_after_due.length==0) { %>
209.                      <h5>No Students have missed the deadline</h5>
210.                    <% } else { %>
211.
212.                      <table style="width: 100%;" id="students_missed_table">
213.
214.                        <tr>
215.                          <th>Student ID</th>
216.                          <th>Due Date</th>
217.                        </tr>
218.
219.                        <% students_not_solved_after_due.forEach((element, index)=> { %>
220.                          <tr>
221.                            <td>
222.                              <%= element[0].toString() %>
223.                            </td>
224.                            <td>
225.                              <%= element[2].toString() %>
226.                            </td>
```

```
                                    </tr>
                                <% }); %>

                            </table>

                        <% } %>

                    </div>

                </div>
            </div>
        </div>

    </div>
  </div>

</div>
</body>

<% include ../partials/script %>

</html>

```

~/views/teacher_challenges/new_challenge.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

            <div class="row">
                <div class="col-md-12">

                    <form action="/create_challenges/new_challenge/create" method="post">

                        <!-- challenge title -->
                        <div class="row">
                            <div class="col-md-12">
                                <label for="challenge_title">CHALLENGE TITLE</label>
                                <textarea id="challenge_title" name="challenge_title" wrap="off" placeholder="Generic Challenge Title" required style="overflow-y: scroll; height: 1.5em; width: 100%;"></textarea>
                            </div>
                        </div>

                        <!-- challenge description -->
                        <div class="row">
```

```
                    <div class="col-md-12">
                        <label for="challenge_description">CHALLENGE DESCRIPTION</label>
                        <textarea id="challenge_description" name="challenge_description" wrap="off" placeholder="Generic Challenge Description" required style="overflow-y: scroll; height: 20em; width: 100%;"></textarea>
                    </div>
                </div>

                <!-- submit button -->
                <button type="submit" value="Create" class="btn btn-primary">CREATE</button>
            </form>

            <% if (message.error) { %>
                <div class="alert alert-danger" role="alert">
                    <strong>
                        <%= message.error %>
                    </strong>
                </div>
            <% } %>

            <% if (message.success) { %>
                <div class="alert alert-success" role="alert">
                    <strong>
                        <%= message.success %>
                    </strong>
                </div>
            <% } %>

        </div>
    </div>
</body>

<% include ../partials/script %>

</html>

```

~/views/teacher_challenges/view_answer.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include ../partials/menu %>

        <div class="row">
            <div class="col-md-12">

                <!-- challenge title -->
```

```
            <div class="row">
              <div class="col-md-12">
                <label for="challenge_title">CHALLENGE TITLE</label>
                <textarea id="challenge_title" name="challenge_title" wrap="off"
                  placeholder="Generic Challenge Title" readonly
                  style="overflow-y: scroll; height: 1.5em; width: 100%;"><%= challenge_title %></textarea>
              </div>
            </div>

            <!-- challenge description -->
            <div class="row">
              <div class="col-md-12">
                <label for="challenge_description">CHALLENGE DESCRIPTION</label>
                <textarea id="challenge_description" name="challenge_description" wrap="off"
                  placeholder="Generic Challenge Description" readonly
                  style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_description %></textarea>
              </div>
            </div>

            <!-- challenge answer -->
            <div class="row">
              <div class="col-md-12">
                <label for="challenge_answer">CHALLENGE ANSWER</label>
                <textarea id="challenge_answer" name="challenge_answer" wrap="off"
                  placeholder="Generic Challenge Answer" readonly
                  style="overflow-y: scroll; height: 20em; width: 100%;"><%= challenge_answer %></textarea>
              </div>
            </div>

            <button onclick="location.href='../../../';" class="btn btn-primary">Go Back</button>

          </div>
        </div>

    </div>
</body>

<% include ../partials/script %>

</html>

```

~/views/teacher_challenges/view_student_challenges.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include ../partials/head %>
    <% include ./partials/script_toggle %>
</head>

<body>
    <div class="container-fluid">
```

```
11.
12.     <% include ../partials/menu %>
13.
14.     <% if ((typeof all_assigned_challenges === "undefined" || all_assigned_challenges == [] ||
all_assigned_challenges.length == 0) && (typeof all_not_assigned_challenges === "undefined" || all_not_assigned_challenges
== [] || all_not_assigned_challenges.length == 0)) { %>
15.         <h4>No Challenges have been Created yet</h4>
16.     <% } else { %>
17.
18.         <!-- ASSIGNED CHALLENGES -->
19.         <div class="row justify-content-center">
20.           <div class="col-md-12">
21.             <div class="card" style="margin-top: 0.5em;">
22.
23.               <div class="card-header" style="background-color: #E6E6FA;">
24.                 <div class="row">
25.                   <div class="col-md-4">
26.                     <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenges that have been
Assigned</h5>
27.                   </div>
28.                   <div class="col-md-5"></div>
29.                   <div class="col-md-3">
30.                     <button class="btn btn-secondary switch_to_student"
onclick="toggle_challenges_have_been_assigned()"
31.                       style="float: right;" id="display_challenges_have_been_toggle">Hide Challenges</button>
32.                   </div>
33.                 </div>
34.               </div>
35.
36.               <div class="card-body" id="challenges_have_been_assigned" style="background-color: #E6E6FA;">
37.
38.                 <% if (typeof all_assigned_challenges === "undefined" || all_assigned_challenges == [] ||
all_assigned_challenges.length == 0) { %>
39.                   <h5>No Challenges have been Assigned yet</h5>
40.                 <% } else { %>
41.
42.                   <table style="width: 100%;" id="challenges_have_been_assigned_table">
43.
44.                     <tr>
45.                       <th>Challenge Title</th>
46.                       <th>Challenge Description</th>
47.                       <th></th>
48.                     </tr>
49.
50.                     <% all_assigned_challenges.forEach((element, index)=> { %>
51.                       <tr>
52.                         <td>
53.                           <%= element[0].split('\n')[1].toString() %>
54.                         </td>
55.                         <td>
56.                           <%= element[0].split('\n')[3].toString() %>
57.                         </td>
58.                         <td>
59.                           <form
```

```
                                    action="/view_student_challenges/more_information/<%= element[1].toString() %>"
method="post">
                                    <button type="submit" value="more_information_<%= element[1].toString() %>" class="btn
btn-secondary create_challenges_edit_btn">More Information</button>
                                </form>
                              </td>
                            </tr>
                          <% }); %>

                        </table>

                      <% } %>

                    </div>

                  </div>
                </div>
              </div>

              <!-- NOT ASSIGNED CHALLENGES -->
              <div class="row justify-content-center">
                <div class="col-md-12">
                  <div class="card" style="margin-top: 0.5em;">

                    <div class="card-header" style="background-color: #ADD8E6;">
                      <div class="row">
                        <div class="col-md-4">
                          <h5 style="vertical-align: middle; line-height: 1.5; margin-top: 0.1em;">Challenges that haven't been
Assigned</h5>
                        </div>
                        <div class="col-md-5"></div>
                        <div class="col-md-3">
                          <button class="btn btn-secondary switch_to_student"
onclick="toggle_challenges_havent_been_assigned()" style="float: right;" id="display_challenges_havent_been_toggle">Show
Challenges</button>
                        </div>
                      </div>
                    </div>

                    <div class="card-body" id="challenges_havent_been_assigned" style="background-color: #ADD8E6;">

                      <% if (typeof all_not_assigned_challenges === "undefined" || all_not_assigned_challenges == [] ||
all_not_assigned_challenges.length == 0) { %>
                        <h5>All Challenges have been Assigned</h5>
                      <% } else { %>

                        <table style="width: 100%;" id="challenges_havent_been_assigned_table">

                          <tr>
                            <th>Challenge Title</th>
                            <th>Challenge Description</th>
                          </tr>

                          <% all_not_assigned_challenges.forEach((element, index)=> { %>
                            <tr>
```

```
                                <td>
                                  <%= element[0].split('\n')[1].toString() %>
                                </td>
                                <td>
                                  <%= element[0].split('\n')[3].toString() %>
                                </td>
                              </tr>
                            <% }); %>

                          </table>

                        <% } %>

                      </div>

                    </div>
                  </div>
                </div>

          <% } %>

      </div>
  </body>

  <% include ../partials/script %>

  </html>

```

~/views/error.ejs

```
<!-- default error message template -->
<!DOCTYPE html>
<html lang="en">

<head>
    <% include partials/head %>
</head>

<body>
    <div class="container-fluid">

        <% include partials/menu %>

        <div class="row">
            <div class="col-md-12">

                <h1><%= message %></h1>
                <h2><%= error.status %></h2>
                <pre><%= error.stack %></pre>

            </div>
        </div>

```

```
    </div>
</body>

<% include partials/script %>

</html>

```

~/views/index.ejs

```
<!DOCTYPE html>
<html lang="en">

<head>
    <% include partials/head %>
    <% include partials/main_visualisation_head %>
</head>

<body>
    <div class="container-fluid">

        <% include partials/menu %>
        <!-- route to ejs template with main assembly visualisation -->
        <% include partials/main_visualisation %>

    </div>
</body>

<% include partials/script %>

</html>

```

~/.env

```
DB_HOST = '127.0.0.1'
DB_USER = 'hidden for security reasons'
DB_PASSWORD = 'hidden for security reasons'
DB_DATABASE = 'assembly_code_visualiser_db'
DB_PORT = 8889

REDIS_HOST = '127.0.0.1'
REDIS_PORT = 6379
REDIS_DATABASE = 0

SESSION_SECRET_KEY = 'hidden for security reasons'
PORT = 3000

NODE_ENV = 'development'

```

~/app.js

```
// all packages
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var flash = require('express-flash');
var session = require('express-session');

var passport = require('passport');
var method_override = require('method-override');

// redis for cookie and session management
var connectRedis = require('connect-redis');
var redisClient = require('./lib/redis');

// all routes
var index_router = require('./routes/index');
var login_router = require('./routes/login_sign_up/login');
var sign_up_router = require('./routes/login_sign_up/sign_up');
var teacher_challenges_router = require('./routes/teacher_challenges/create_challenges');
var student_challenges_router = require('./routes/student_challenges/view_set_challenges');
var teacher_challenges_view_student_challenges_router =
require('./routes/teacher_challenges/view_student_challenges');

// use dotenv (local configurations)
require('dotenv').config()

// initialise express
var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

// initialise all packages
app.use(logger('dev'));
app.use(express.json());

// allow getting information from forms (sign up & login)
app.use(express.urlencoded({ extended: true }));

// set static directory
app.use(express.static(path.join(__dirname, 'public')));

app.use(cookieParser());

// redis setup
var RedisStore = connectRedis(session);
// session
app.use(session({
```

```
  secret: process.env.SESSION_SECRET_KEY,
  cookie: {
    secure: false, // if true, would only transmit cookies over https
    httpOnly: false, // if true, would prevent client-side JS from reading cookies
    maxAge: 7200000, // session max age in milliseconds (2 hours)
  },
  store: new RedisStore({ client: redisClient }),
  saveUninitialized: false,
  resave: false,
}));

app.use(flash());

// set up passport
app.use(passport.initialize());
app.use(passport.session());
var initializePassport = require('./lib/passport-config');
initializePassport(passport);

app.use(method_override('_method'))

// initialise all routes
app.use('/', index_router);
app.use('/login', login_router);
app.use('/sign_up', sign_up_router);
app.use('/logout', index_router);
app.use('/instruction_set', index_router);
app.use('/how_to_use', index_router);
app.use('/create_challenges', teacher_challenges_router);
app.use('/create_challenges/delete', teacher_challenges_router);
app.use('/create_challenges/assign', teacher_challenges_router);
app.use('/create_challenges/edit', teacher_challenges_router);
app.use('/view_set_challenges', student_challenges_router);
app.use('/view_student_challenges', teacher_challenges_view_student_challenges_router);

// catch 404 and forward to error handler
app.use(function (req, res, next) {
  next(createError(404));
});

// error handler
app.use(function (err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = process.env.NODE_ENV === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;

```

Chapter 9 Bibliography

AQA. (2019, January 21). *AS and A-level Computer Science Assembly Language Instruction Set [Screenshot].* Retrieved March 2022, from AQA – education charity providing GCSEs, A-levels and support: https://filestore.aqa.org.uk/resources/computing/AQA-75162-75172-ALI.PDF

AQA. (2019, January 21). *AS and A-level Computer Science Specification [Screenshot].* Retrieved March 2022, from AQA – education charity providing GCSEs, A-levels and support: https://filestore.aqa.org.uk/resources/computing/specifications/AQA-7516-7517-SP-2015.PDF

AQA. (2020, January 29). *Specification for first teaching in 2020: Specification [Screenshot].* Retrieved March 2022, from AQA – education charity providing GCSEs, A-levels and support: https://filestore.aqa.org.uk/resources/computing/specifications/AQA-8525-SP-2020.PDF

Hankin, P. (2016, April 20). *Little Man Computer Emulator [Screenshot]*. Retrieved from Paul's Blog - Tasty brain snacks: https://blog.paulhankin.net/lmc/lmc.html

Higginson, P. L. (2014). *Little Man Computer - CPU Simulator [Screenshot]*. Retrieved from Peter Higginson's Personal Site: https://www.peterhigginson.co.uk/LMC/

Kamani, S. (2022, February 22). *Session Cookie Authentication in Node.js (With Complete Examples) [Screenshot].* Retrieved from Soham Kamani: https://www.sohamkamani.com/nodejs/session-cookie-authentication/

Manning Publications. (2018, October 22). *Routing and Analyzing Request Data with Node.js.* Retrieved from Manning Free Content Center: https://freecontent.manning.com/get-programming-with-node-js-routing-and-analyzing-request-data/

Wikimedia Foundation. (2022, February 18). *Wikipedia - Little man computer [Screenshot]*. Retrieved March 2022, from Wikipedia: https://en.wikipedia.org/wiki/Little_man_computer