Final Report

Team 18

- Maryam Abdullaziz
- Saif Albastaki
- Nur Binti Mohd
- Aamir Moh'd
- Daniel Ray
- Gabriel Plume
- Aybike Koyunlu

1.0 Tests and Example Runs

1.1 Commands needed to install and start the system

1. To clone the project repository into your terminal by using the following link in the terminal type in: git clone https://git.shefcompsci.org.uk/com1001-2022-23/team18/project.git
2. Once the folder has been downloaded into the terminal you can then navigate through the individual files to be able to run the sinatra application you will need to follow these steps:
    a. Navigate to the file in your system by using the following (cd project)
    b. Install the gem files by typing in: (bundle install)
    c. Once you have installed all the files needed you can then type in: (sinatra) and the program will run

1.2 Standard Usernames

Account	Username	Email	Password
Manager	manager1	manager1_team18@gmail.com	manager1
Manager	manager2	manager2_team18@gmail.com	manager2
Moderator	moderator1	moderator1_team18@gmail.com	moderator2
Moderator	moderator2	moderator2_team18@gmail.com	moderator1
Administrator	admin1	admin1_team18@gmail.com	admin1
Administrator	admin2	admin2_team18@gmail.com	admin2

Trusted course provider		team18@gmail.com	
Student	learner1	learner1_team18@gmail.com	learner1
Student	learner2	learner2_team18@gmail.com	learner2

2.0 Resubmission of User Stories

- Red: deleted or cant implement it in time
- Green: Added uses stories
- White: implemented

User Stories	Reasoning and How it was Implemented
As a learner, I want to be able to navigate the course itself.	Allow the user to navigate through courses. This was not implemented as we did not need to write any actual courses. Upon entering a course, the user is currently redirected to a video which is where the course would be.
As a learner, I want to be able to find relevant training courses based on my degree.	The course picker allows you to select a degree and a ‘type’ of course
As a learner, I want to be able to choose online courses I’ve already studied.	This is implemented to attempt course quizzes multiple times. Could be done by clicking the link again.
As a learner, I want to be able to choose new, recommended courses.	Once a course has been recommended, the user can select to enrol in it.
As a learner, I want to be able to register an account.	There is a sign-up function for users to create an account.
As a learner, I want to be able to record my background (degree topic, year, other relevant information).	This can be added after sign-up is complete.

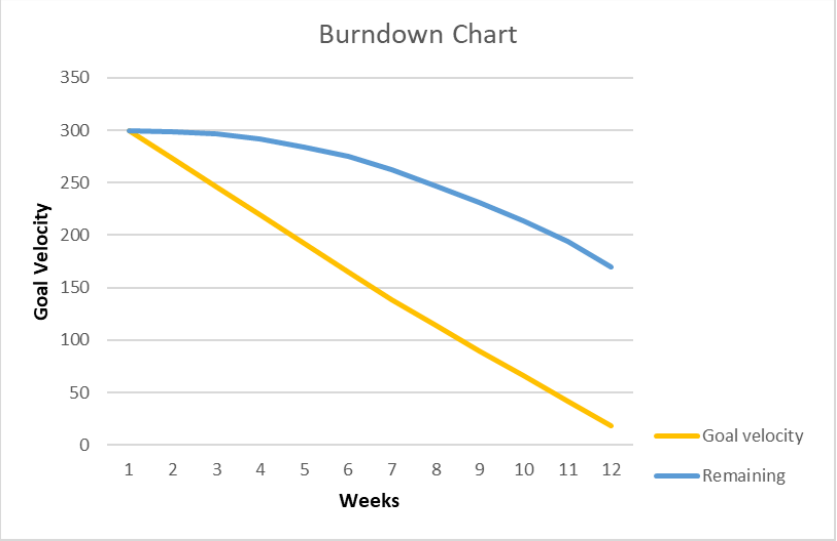
As a learner, I want to be able to rate courses.	So other users can see if a course is good, users can rate courses they've done. Implemented through a linked database
As a learner, I want to be able to log in and out of my account.	Login and logout functions have been added to allow users to change accounts etc…
As an administrator I want to manage the accumulation and consolidation of all financial data necessary for any paid courses.	We were told as a group managing finance is more theoretical so this story has not been physically implemented.
As an administrator I handle the certificate payments.	See above
As an administrator, I want to allow the learner to reset the password if it is forgotten.	The users can request a password reset from their administrator or go through the customer service on the contact page to have their password reset via the edit form.
As an administrator i provide customer services for the learners	The site has a contact page for customer service.
As an administrator we can access the database of the users	The list of students can be seen in the search page.
As an administrator I have reports and list of students suspended such as reasoning and information of the student.	Logs are available to administrators containing student sessions and activities. The list of students can be seen in the search page.
As an administrator, I want to be able to make sure the courses are up to date- can remove or add courses.	Administrators can add/delete courses and edit the information about them.
As a manager I want to monitor my moderators work performance.	This sort of bossware is unethical and wasn't required; thus wasn't implemented.

As a manager I want to have access to a log of all performed actions.	This can be seen in the same log as previously mentioned for seeing student activity.
As a manager I want to view all courses being added or changed.	This can also be seen in the log
As a manager I want to be able to determine what relevant filtering is.	
As a manager I want to be able to see the filtered and sorted reports.	
As a manager I want to be able to choose a filter and have my list compared to the total users.	
As a manager I want to have access to a list of the users and what courses they are enrolled into.	This has been implemented when you login as a manger you will have access to the list of users with the course they are enrol into
As a moderator, I want to be able add new training courses to the website through uploading.	This has been implemented in a similar way to adding new students
As a moderator, I want to be able to view every student's personalised suggestion list.	This was too complex to be implemented
As a moderator, I want to be able to change the details of any course, such as their status, start date, end date etc.	This is done using a similar edit function as the one used for students.

As a moderator, I want to be able to delete comments in the discussion section of every question.	We did not deem this function as being vital so it was not implemented
As a trusted course provider, I would like to allow to add course details to the system	Course details can be edited in the same way by trusted course providers as they can be by moderators.
As a trusted course provider, I would be able to archive courses	If a course needs updating, it needs to be invisible to users until it has been renovated and is then more up to date. After that, the trusted course provider can have it reuploaded/unhidden.
As a Moderator, I want to be able to hide/reveal trusted courses	This was implemented to ensure, if preferred, users can only see 'official' courses.

3.0 Burndown Chart for Progress of Each Iteration

As a team, for the early stages of starting the project, we did the tasks which were much easier from the lectures and the lab classes. As a group started with the views, databases, forms, sessions and the testings according to the lab classes.



Graph 1: Burndown Chart Goal Velocity against Week

Week	Goal	Done	Goal velocity	Remaining

0	27	0	300	300
1	27	1	273	299
2	27	2	246	297
3	27	5	219	292
4	27	8	192	284
5	27	9	165	275
6	27	13	138	262
7	24	15	114	247
8	24	16	90	231
9	24	17	66	214
10	24	20	42	194
11	24	24	18	170

There were a total of 27 user stories from last semester. Although it was a slow start at first, as soon as we got the hang of it we managed to do more user stories per week. It is important to note that there were a total of 6 and there were an addition of 3 before the easter break. The **goal** is what we intend to complete and **done** is what we have done during the week. **Goal velocity** calculates the target points to zero which is completion of the project. **Remaining** deducts **Done** from **Remaining.** More detailed information can be found in the 2.0 section and in the table below.

Role	**Iteration 1**		**Iteration 2**
	Status	**User Stories**	
Learner	Created	As a learner, I want to be able to navigate the course itself.	As a learner, I want to be able to choose online courses I've already studied.
		As a learner, I want to be able to register an account.	As a learner, I want to be able to choose new, recommended courses.
		As a learner, I want to be able to record my background	As a learner, I want to be able to rate courses.

		(degree topic, year, other relevant information).	
	Haven’t been fully implemented	As a learner, I want to be able to choose new, recommended courses.	
Administrator	Created	As an administrator we can access the database of the users	As an administrator I have reports and list of students suspended such as reasoning and information of the student.
	Haven’t been fully Implemented	As an administrator, I want to allow the learner to reset the password if it is forgotten.	
		As an administrator i provide customer services for the learners	
		As an administrator, I want to be able to make sure the courses are up to date- can remove or add courses.	
Manager	Created	As a manager I want to view all courses being added or changed.	As a manager I want to have access to a log of all performed actions.
		As a manager I want to have access to a list of the users and what courses they are enrolled into.	
		As a manager I want to be able to determine what relevant filtering is.	
		As a manager I want to have access to a list of the users and	

		what courses they are enrolled into.	
Moderator	Created	As a moderator, I want to be able add new training courses to the website through uploading.	
		As a moderator, I want to be able to change the details of any course, such as their status, start date, end date etc.	

4.0 Exemplar Code

```ruby
get "/recommendation" do
  #variable assigned to a field to put in the form to query
  @current_degree = params.fetch("current_degree", "").strip
  @course_certificate = params.fetch("course_certificate", "").strip
  @interest = params.fetch("interest" , "").strip
  @interests = @interest.split(",")

  #Query through the two field to give a recommendation
  @courses = if @current_degree.empty? && @interest.empty? then
                puts "No courses found"
                []

             else
               Course.where(Sequel.like(:degree, "%#{@current_degree}%")).where(Sequel.like(:interest, "%#{@interest}%"))
             end
  erb :recommendation
end
```

Figure 1: The controller for the recommendation page at project/controllers/learner-form-controllers.rb

```erb
<div class="navigation">
  <form>
    Degree Options
    <li>Maths</li>
    <li> Computer Science</li>
    <li> Biology </li>
    <p>Enter your current degree: <input type="text" name="current_degree" value="<%= h @current_degree %>"/>
    <br/>
    What are your interest?:
    <li> Applying Knowledge</li>
    <li> Critical Thinking </li>
    <li> Personal Devolepment </li>
    <input type="text" name="interest" value="<%= h @interest %>"/>
    <input type="submit" value="Submit" /></p>
  </form>
</div>
<br />

<p class = "recommendation">
<% if @courses.count > 0 %>
  <% @courses.each do |course| %>

      <h1><%= h course.name %></h1>
      <h3>Description :<%= h course.description %></h3>
      <h3>Year Group: <%= h course.yeargroup %></h3>
      <h3>Degree: <%= h course.degree %></h3>
      <h3>Payment: <%= h course.paid %></h3>
      <h3>Certificate: <%= h course.certificate %></h3>
      <h3>Interest: <%= h course.interest %></h3>
      <a href="/learner-form"><button> ENROLL IN THIS COURSE NOW </button></a>
  <% end %>
<% else %>
  <p>You have no recommendations</p>
<% end %>
</p>
```

Figure 2: The ERB file for the recommendation page that only shows the courses that is recommended to the user at the directory project/views/recommendations.erb

```ruby
get '/account' do
  erb :account
end

# This route clears the session and redirects the user to the home page
get '/logout' do
  session.clear
  puts "You logged out ."
  redirect '/'
end

# This route handles the login form submission
post '/account' do
  username = params[:username] # gets the username from the login form parameter
  password = params[:password] # gets the password from the login form parameter
  user = User.where(username: username).single_record # gets the user record from the database

  # If the user is an authorized user, sets the session username and renders a page based on the user role
  if user && ['staff', 'moderator', 'admin', 'manager','trustedUser'].include?(user.role) && password == user.passwordHash
    session[:username] = username
    case user.role
    when 'staff'
      puts "User is a staff member"
      erb :staff_only
    when 'moderator'
      puts "User is a moderator"
      erb :moderator_only
    when 'admin'
      puts "User is an administrator"
      erb :admin_only
    when 'manager'
      puts "User is a manager"
      erb :manager_only
    when 'trustedUser'
      puts "User is a trustedUser"
      erb :trustedUser
    end
  # If the user is a learner, sets the session username and redirects to the home page
  elsif user && password == user.passwordHash
    session[:username] = username
    puts "User is a learner"
    redirect '/'
  # If the user is suspended, sets the session username and renders the suspension page
  elsif user && 1 == user.suspended && password == user.passwordHash
    session[:username] = username
    erb :suspension
  # If the user is not found in the database or the password is incorrect, renders the login page with an error message
  else
    @var = "Invalid username or password."
    puts @var
    puts "Invalid username or password."
    erb :account
  end
end

```

Figure 3: The log in controller that allows different roles to access the website differently at the directory project/controllers/login_controller.rb

```ruby
get "/add" do
    @user = User.new #Create a new User
    erb :"manager_forms_views/add"
  end

post "/add" do
  @user = User.new  #Create a new user oject
  @user.load(params) # Load the parameters to the user object

  if @user.valid? #Check if the user is valid
    @user.save_changes #Save changes if it is
    redirect "/search"
  end

  erb :"manager_forms_views/add"
end
```

Figure 4:The add.rb allows us to add users to the db. This is only accessible by staff members in the directory project/controllers/managers_form_controllers/login_controller.rb

```ruby
post "/delete" do
    id = params["id"]

    @user = User[id] #Retrieve the value of id
    puts id

    if User.id_exists?(id) #Check if the id exists
      @user.delete
      redirect "/search"
    end

    erb :"manager_forms_views/delete"
  end
```

Figure 5:The delete.rb allows us to delete the users. This is only accessible by staff members in the project/controllers/managers_form_controllers/login_controller.rb

```ruby
def expect_args(ords, args, catch_nil=true)

    #loads null values into new args array
    new_args = Array.new(args.length() -1)
    for i in (0...args.length) do
      if ords.include?(i)
        new_args[i] = args[i]
      else
        new_args[i] = nil
      end
    end
    test = Comment.new()
    #the method being tested
    test.load [new_args[0],new_args[1],new_args[2],new_args[3],new_args[4]]

    vals = [test.id,
            test.courseID,
            test.userID,
            test.stars,
            test.comment]
    #expects all specified values to match args
    #doesn't check other values if catch_nil is specified false
    for i in (0...vals.length) do
      expect (vals[i]).to eq(new_args[i]) unless !catch_nil && new_args[i] == nil
    end
    #checking the new_args element value is faster than running include again
end
```

Figure 6: extract from spec/unit/model_spec/courses_spec.rb

This is an example of a subroutine used in model testing, for Comment.rb, which is designed to streamline the testing of the properties of the Comment object. It takes an array of “ordinates”, an array of “arguments” and a boolean flag to determine whether the “arguments” include null values which need testing.

The “ordinates” are used to determine which attributes are being tested, and the “arguments” provide values to test against.

Multiple of these methods appear across the model tests, with variations on the names and number of attributes. Here is an example of the method being called in a test:

```ruby
describe "When given valid parameters to load" do
    it "loads successfully" do
      expect_args([0,1,2,3,4], args_normal)
    end
end
```

Figure 7: extract from spec/unit/model_spec/courses_spec.rb

This uses the array args_normal (defined as a standard set of inputs for a new Comment) and tests that all of the attributes match. The catch_null parameter defaults to False.

5.0 Testing

In this project, testing is really important to assure the quality and to check if we achieved the requirements.Testing for models consists primarily of TDD tests, with refactoring in mind. Specifically, the methods within the models currently take a “params” hash as parameters. Refactoring is planned wherein the controllers extract the elements of this hash, and pass the relevant information to the models. These parameters are given by the model tests, to represent this goal.

```
45 examples, 30 failures

Failed examples:

rspec ./spec/unit/model_spec/Comment_spec.rb:39 # Comment model When given blank parameters to load has blank attributes
rspec ./spec/unit/model_spec/Comment_spec.rb:45 # Comment model When given valid parameters to load loads successfully
rspec ./spec/unit/model_spec/Comment_spec.rb:51 # Comment model When given invalid/incomplete parameters to load raises the correct error for id
rspec ./spec/unit/model_spec/Comment_spec.rb:56 # Comment model When given invalid/incomplete parameters to load raises the correct error for courseID
rspec ./spec/unit/model_spec/Comment_spec.rb:61 # Comment model When given invalid/incomplete parameters to load raises the correct error for userID
rspec ./spec/unit/model_spec/Comment_spec.rb:66 # Comment model When given invalid/incomplete parameters to load raises the correct error for stars
rspec ./spec/unit/model_spec/Comment_spec.rb:71 # Comment model When given invalid/incomplete parameters to load raises the correct error for comment
rspec ./spec/unit/model_spec/Course_spec.rb:43 # Course model When given blank parameters to load has blank attributes
rspec ./spec/unit/model_spec/Course_spec.rb:46 # Course model When given blank parameters to load is not valid to load
rspec ./spec/unit/model_spec/Course_spec.rb:52 # Course model When given valid parameters to load loads successfully
rspec ./spec/unit/model_spec/Course_spec.rb:59 # Course model When given invalid/incomplete parameters to load raises an error for name
rspec ./spec/unit/model_spec/Course_spec.rb:64 # Course model When given invalid/incomplete parameters to load raises an error for description
rspec ./spec/unit/model_spec/Course_spec.rb:69 # Course model When given invalid/incomplete parameters to load raises an error for yeargroup
rspec ./spec/unit/model_spec/Course_spec.rb:74 # Course model When given invalid/incomplete parameters to load raises an error for degree
rspec ./spec/unit/model_spec/Course_spec.rb:79 # Course model When given invalid/incomplete parameters to load raises an error for paid
rspec ./spec/unit/model_spec/Course_spec.rb:84 # Course model When given invalid/incomplete parameters to load raises an error for certificate
rspec ./spec/unit/model_spec/Mark_spec.rb:43 # Mark model When given blank parameters to load has blank attributes
rspec ./spec/unit/model_spec/Mark_spec.rb:49 # Mark model When given valid parameters to load loads successfully
rspec ./spec/unit/model_spec/Mark_spec.rb:55 # Mark model When given invalid/incomplete parameters to load raises an error for id
rspec ./spec/unit/model_spec/Mark_spec.rb:60 # Mark model When given invalid/incomplete parameters to load raises an error for userID
rspec ./spec/unit/model_spec/Mark_spec.rb:65 # Mark model When given invalid/incomplete parameters to load raises an error for courseID
rspec ./spec/unit/model_spec/Mark_spec.rb:71 # Mark model When given invalid/incomplete parameters to load raises an error for percentage
rspec ./spec/unit/model_spec/Mark_spec.rb:76 # Mark model When given invalid/incomplete parameters to load raises an error for pass
rspec ./spec/unit/model_spec/Mark_spec.rb:81 # Mark model When given invalid/incomplete parameters to load raises an error for attempt
rspec ./spec/unit/model_spec/User_spec.rb:48 # User model When given valid (full) parameters to load successfully loads
rspec ./spec/unit/model_spec/User_spec.rb:67 # User model When given valid (initial) parameters to load successfully loads
rspec ./spec/unit/model_spec/User_spec.rb:87 # User model When given no parameters to load has blank attributes
rspec ./spec/unit/model_spec/User_spec.rb:108 # User model When given invalid/incomplete parameters to load raises an error for username
rspec ./spec/unit/model_spec/User_spec.rb:113 # User model When given invalid/incomplete parameters to load raises an error for passwordHash
rspec ./spec/unit/model_spec/User_spec.rb:118 # User model When given invalid/incomplete parameters to load raises an error for email
```

Figure 8: