COMP2001
Information Management & Retrieval

20 CREDIT MODULE

ASSESSMENT: 100% Coursework **W1: 30% Set Exercises**
W2: 70% Report

MODULE LEADER: Dr Shirley Atkinson

MODULE AIMS

- To introduce students to fundamental principles around graphical representation along with information management, database systems and modelling.
- To consider issues around image compression techniques, how humans can access information and data to support their needs, learn declarative queries and consider common designs for database systems.
- To understand the differences between relational and semi-structured data models and use appropriate data modelling techniques.

ASSESSED LEARNING OUTCOMES (ALO):

1. Demonstrate explicit uses of modelling techniques to gain access to information and data to support a given need.
2. Illustrate an appropriate technical solution to the problems of information privacy, integrity, security and preservation.
3. Illustrate the design of an application of moderate complexity to elicit and visualise information from a data store.

Overview

This document contains all the necessary information pertaining to the assessment of *COMP2001 Information Management & Retrieval*. The module is assessed via **100% coursework**, across two elements: *30% Set Exercises* and *70% Report*.

The sections that follow will detail the assessment tasks that are to be undertaken. The submission and expected feedback dates are presented in Table 1. All assessments are to be submitted electronically via the respective DLE module pages before the stated deadlines.

	Submission Deadline	Feedback
Set Exercises (30%)	15/11/2021	13/12/2021
Report (70%)	17/01/2022	17/01/22

Table 1: Assessment Deadlines

All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. Whilst the assessment information is provided at the start of the module, it is not necessarily expected you will start this immediately – as you will often not have sufficient understanding of the topic. The module leader will provide guidance in this respect.

Set Exercises

There are three set exercises for you to complete during the course of the module. The exercises should be submitted to the module DLE by the deadline shown in the table above. The answers for all three exercises should be submitted on one PDF file. The instructions are given below.

Assessment 1: Set Exercises

CW1 Overview

There are three set exercises outlined below that will allow you to demonstrate your knowledge and understanding around data modelling, information preservation and implementation. Submit a single PDF file that contains the outputs from each of the exercises in the same order as they appear below. The outputs for each exercise are clearly described.

You are to choose ONE of the scenarios in the accompanying document and base your six exercises on your choice. The marks are indicated for each exercise.

Based on the scenario you have chosen, your role is to design and then create the database tables and other associated items requested for this application. You are NOT creating the application but only concentrating on the appropriate data items that will be required.

Carry out an analysis of the scenario, identify the appropriate data items, design and then create an appropriate database layer that will store the data appropriately and preserve the integrity of the data. You are to implement your design by deploying your script to your allocated Microsoft SQL Server database running at socem1.uopnet.plymouth.ac.uk. This is referred to as MSS. Export your sql scripts for the exercises and save into your PDF document. Only the SQL scripts as they are implemented on socem1 are required.

Set Exercise 1: Entity Relationship Diagram (ERD) : 15 marks

You must create a final Entity Relationship Diagram (ERD) for your proposed data layer. This final ERD should only include entities and relationships that can be justified from reasonable assumptions. Make sure that all one-to-one and many-to-many relationships are resolved prior to submission. State any assumptions you have made.

You must name the 3NF relations and draw the ERD using the taught notation of soft boxes and crows feet. Ensure that the ERD offers an accurate representation of the database. Inconsistencies between the ERD and the actual database will mean marks are not earnt.

Set Exercise 2: Normalisation : 20 marks

Produce the Third Normal Form (3NF) from the sample form shown in the scenario above. Normalise the attributes to 3NF showing all intermediate stages, namely: Un-normalised Form (UNF), First Normal Form (1NF) and Second Normal Form (2NF). Be careful to highlight the attributes used for keys.

Additional marks will NOT be gained for optimising the results of 3NF, thus optimising may not be worth your effort.

Set Exercise 3: Hosted Tables : 30 marks

Following your analysis, you are to implement your design on your hosted Microsoft SQL Server database at socem1.uopnet.plymouth.ac.uk. You must use the schema “CW1” for your implementation to differentiate your coursework tables from any others. Once you have

implemented your design, output the tables as SQL Create statements and copy them into your deliverable document.

Set Exercise 4 : View : 10 marks
Identify and implement the view as described in the scenario. You must implement the view under the schema “CW1” on your module hosted Microsoft SQL Server database. Remember that views can be queryable.

Set Exercise 5 : Stored procedure : 15 marks
Identify and implement the stored procedure as described in the scenario. You must implement the stored procedure under the schema “CW1” on your module hosted Microsoft SQL Server database.

Set Exercise 6 : Trigger : 10 marks
Identify and implement the trigger as described in the scenario. You must implement the trigger under the schema “CW1” on your module hosted Microsoft SQL Server database.

Outputs
Provide the following details in your PDF document:

- A final Entity Relation diagram drawn using appropriate tools (not hand-drawn).
- Normalisation details to 3rd normal form.
- SQL Script for your tables illustrating constraints.
- SQL Script for your view, stored procedure, and trigger.
- Note your SQL Scripts must match the implementation on the module hosted Microsoft SQL Server.

Assessment 2: Report on tasks

Task:
There are two key parts to this part of the coursework. Part 1 is to create a web service that provides access through a RESTful API interface following the specification provided below. Part 2 is to provide a prototype linked data application where you provide the visualisation of a chosen dataset combined with semantic markup for automatic processing.

The two parts are NOT linked. Part 2 does NOT use Part 1.

Your code must be hosted on the GitHub Classroom link here
https://classroom.github.com/a/zpgEZ5qE

Once the tasks have been completed, you are to write up and report upon your results. Further information on what should be included in the report is provided in the deliverables section.

Part 1
This task requires you to create a Showcase RESTful API that provides functionality as determined by the COMP2001 API specification found in Appendix A of this document. The specification provides information about the basics for the API, you will need to extend and apply your own consideration in some areas.

Your API must be hosted on the web.socem.plymouth.ac.uk server and use the Microsoft SQL Server database on socem1.uopnet.plymouth.ac.uk. Both of these are provided to you for this module. The code for the API must be hosted in the subdirectory of your folder labelled Showcase. This has already been set up for you. In addition, ensure that your database has an appropriate amount of sample data within it to demonstrate the functionality.

Please ensure you have completed Units 1,2,3,4,5 and 7 prior to embarking upon this part of the coursework.

Part 2
This part of the task requires you to create an example linked data application using a coding language of your choice that provides both human readable and machine-readable information about a specific dataset.

- The application you are to create must be based on a dataset of your choice from http://www.dataplymouth.co.uk/datasets.
- The dataset must be in the following formats: csv, JSON or geoJSON. No other format is acceptable.
- Choose one entity from the dataset to expose as linked data. You must create the appropriate URI that uses HTTP protocols to return JSON-LD RDF formatted code.
- You may combine data if you consider it essential to understand the chosen dataset.

- Your application must have a modern look and feel to it therefore you should create the appropriate HTML and CSS to provide this. The use of third-party templates and visualisation tools is permitted but must have attribution in your GitHub readme file.
- Your application should have a minimum of two pages **in addition to the URI**: index and data
    - index should display the following:
        - A link to the original data set.
        - A link to any additional data used to present a meaningful data representation to the end user.
        - The project vision
    - Data should display the data in an appropriate human readable format.
- You should test your URI displays the appropriate RDF output.
- You should test your web application for web accessibility.

You must use the hosting provided on web.socem.plymouth.ac.uk which may well constrain your choice of language. You must check with the module leader that the server will host your language choice before you embark upon coding.

Please refer back to Unit 8 for the instructions on how to set up your application. You must ensure that you have completed the tutorial in that section prior to embarking upon this part of your coursework.

Deliverables

Your source code must be stored in a GitHub repository on GitHub Classroom here https://classroom.github.com/a/zpgEZ5qE. The implementation of the source code must deployed on the web.socem server as instructed in the tasks given above. You MUST adapt your GitHub readme file to represent YOUR work. Failure to adjust the readme file will result in marks being lost.

You must present the work carried out in a report submitted in PDF format. No other format is acceptable. Your report must be approximately 2000 words, please use screenshots and links to code files to illustrate functionality where appropriate and UML diagrams to illustrate the design.

The report must contain the following sections:

1. Introduction (approximately 2 paragraphs). Introduce the document and signpost the reader to what they will find in it. Provide links to your GitHub repository, your hosted application and the original dataset.
2. Background. Explain here which data set you chose and what information your application provides. Provide the product vision for your linked data application.
3. Legal, Social, Ethical and Professional (LSEP). Discuss here how you addressed issues around information privacy, integrity, security and preserving the data in both the Linked Data application and the API.
4. Design. Present here the appropriate models for both Parts 1 and 2. You should be including a logical ERD for the API having evolved it from the conceptual ERD provided. There is no need to repeat the given diagrams. Provide a data model for the linked data

application, a wireframe to illustrate your interface design, and any other diagrams and textual description you consider to be appropriate.

5. Implementation. Illustrate with screenshots and hyperlinks to your source code in your GitHub repo how you implemented your design. Provide a suitable narrative to help your reader understand your screenshots and diagrams.
6. Evaluation. Show here how you tested your implementation clearly indicating areas for further work and improvement.

Assessment Criteria:

The report, associated diagrams and code are graded in the following way

Category	Criteria
LO1 (30%)	Background clearly justifies a given need. Appropriate literature used to provide evidence. Citations in Harvard style. Logical ERD diagram provided for API Sequence diagram provided for linked data application RDF output graph provided.
LO2 (35%)	SQL code provided to illustrate data integrity SQL code provided to illustrate view, tables and stored procedures Hyperlink provided to code implementation illustrating how SQL injection addressed Description and authoritative literature (Harvard style) provided to justify how LSEP issues addressed. Discussion includes outline of who has access to data, what is in place to stop unauthorised access to data.
LO3 (35%)	Sitemap, wireframes and screenshots provided to illustrate implementation of linked data application Hyperlinks provide to code implementation to illustrate how screenshots implemented. Evidence provided for web-accessibility testing UML package diagram for architecture provided. URL for RDF output provided.

Category	Fail (< 40%)	>= 40%	>= 50%	>= 60%	>= 70%
LO1 (30%)	Evidence vague and hard to follow. Items missing.	Evidence passable. Errors and omissions.	Evidence OK. Some errors and omissions. Little evidence of original thought.	Evidence good. Items provided of good quality. Evidence of original thought.	Evidence excellent and all expected items present. Background clearly articulated and properly justified. Citations appropriate. Clear evidence of original thought
LO2 (35%)	Evidence vague and hard to follow. Items missing. Very little implementation has been constructed, and some of what there is does not work.	Evidence passable. Errors and omissions. Implementation can be followed by expert.	Evidence OK. Some errors and omissions. Implementation has some bugs but runs. Little evidence of original thought.	Evidence good. Items provided of good quality. Complete implementation but room for improvement. Evidence of original thought	Evidence excellent and all expected items present. LSEP issues clearly articulated and properly justified with authoritative literature. Citations appropriate. Privacy discussion in good depth. Clear evidence of original thought

LO3 (35%)	Evidence vague and hard to follow. Items missing. Very little software has been constructed, and what there is does not work.	Evidence passable. Errors and omissions. Implementation can be followed by expert..	Evidence OK. Some errors and omissions. Implementation has some bugs but runs. Little evidence of original thought.	Evidence good. Items provided of good quality. Complete implementation but room for improvement. Evidence of original thought.	Evidence excellent and all expected items present. Clear evidence of original thought.

General Guidance

Extenuating Circumstances

There may be a time during this module where you experience a serious situation which has a significant impact on your ability to complete the assessments. The definition of these can be found in the University Policy on Extenuating Circumstances here:
https://www.plymouth.ac.uk/uploads/production/document/path/15/15317/Extenuating_Circumstances_Policy_and_Procedures.pdf

Plagiarism

All of your work must be of your own words. You must use references for your sources, however you acquire them. Where you wish to use quotations, these must be a very minor part of your overall work.

To copy another person's work is viewed as plagiarism and is not allowed. Any issues of plagiarism and any form of academic dishonesty are treated very seriously. All your work must be your own and other sources must be identified as being theirs, not yours. The copying of another persons' work could result in a penalty being invoked.

Further information on plagiarism policy can be found here:

Plagiarism: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism

Examination Offences: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/examination-offences

Turnitin (http://www.turnitinuk.com/) is an Internet-based 'originality checking tool' which allows documents to be compared with content on the Internet, in journals and in an archive of previously submitted works. It can help to detect unintentional or deliberate plagiarism.

It is a formative tool that makes it easy for students to review their citations and referencing as an aid to learning good academic practice. Turnitin produces an 'originality report' to help guide you. To learn more about Turnitin go to:
https://guides.turnitin.com/01_Manuals_and_Guides/Student/Student_User_Manual

Referencing

The University of Plymouth Library has produced an online support referencing guide which is available here: http://plymouth.libguides.com/referencing.

Another recommended referencing resource is Cite Them Right Online; this is an online resource which provides you with specific guidance about how to reference lots of different types of materials.

The Learn Higher Network has also provided a number of documents to support students with referencing:
References and Bibliographies Booklet:
http://www.learnhigher.ac.uk/writing-for-university/referencing/references-and-bibliographies-booklet/
Checking your assignments' references:
http://www.learnhigher.ac.uk/writing-for-university/academic-writing/checking-your-assigments-references/

Appendix A

COMP2001 API specification (The Showcase)
Task 1 requires you to create an API to handle the showcase database as a RESTful web service. The following details specify how this API should be designed.

Product Vision
For students that need to enter their project details, and administrators who need to manage the project information, the Showcase is a RESTful API web service providing simple and robust access to the data store via the HTTP infrastructure.

Functional Requirements
The application is to be a machine-to-machine communication mechanism, therefore it has no visual requirements. The following user stories are the starting point for the product backlog.

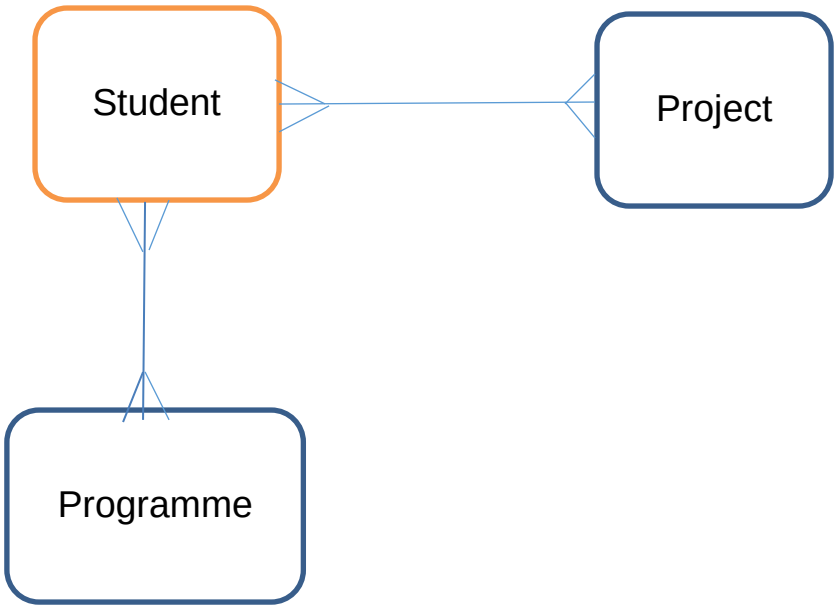
- As an administrator I wish to enter Programme Details
- As an administrator I wish to check that a Programme has been uploaded.
- As an administrator I wish to edit an existing Programme
- As an administrator I wish to delete an existing Programme

Non-Functional Requirements
- The data must be stored in a Microsoft SQL Server database hosted on socem1.uopnet.plymouth.ac.uk
- The API must be hosted on the web server web.socem.plymouth.ac.uk
- The API must be written in C# and ASP.NET
- Response times are not within the scope of this application
- There are no interfaces for humans to interact with therefore usability requirements are outside of the scope of this application
- Reliability requirements are outside of the scope of this application
- All appropriate attempts to mitigate the OWASP top 10 vulnerabilities are to be taken where possible.
- Authentication of the client application to the API is to be considered.

Data Layer
You are to create your data structure using the Microsoft SQL Server database hosted on socem1. You will have been allocated this early on in the module and the log in details can be found in Unit 1.

Below is the initial ERD. As you can see there are only three tables required. Using the other information in this specification you are to determine what columns and keys are required and if any link tables are required.

Ensure you apply appropriate integrity measures and transactions.



A student can belong to more than one programme but not at any one time (eg: a student can progress through programmes such as from a BSc to a Masters, or change their degree). A student can belong to one or more group projects.

Add optionality to your final ERD and include any assumptions you make.

Students store information about the Name and Student ID.
Programmes store the Programme Title and Programme Code
Projects store the Title, Description, Year, thumbnail image file, poster image file.

Stored procedures are to handle the CRUD functionality. These should be as below.

Operation Name	**Create_Programme**
Involved Tables	Programmes
Overview	Enter Programme details The programme code is used to check if the programme exists prior to inserting.
Pre-condition	Programmes tables have been created
Post-condition	A new record is added into Programmes table and an appropriate HTTP status number is returned. See interface for more details.
Input parameters	@Programme_Code, @Title
Output parameters	@ResponseMessage
Return Value	None

Operation Name	**Update_Programme**
Involved Tables	Programmes
Overview	To allow an edit of the programme record.

Pre-condition	Programmes table has been created and the programme record is in existence
Post-condition	The programme record will have been updated with new values. The Programme Code however can not be edited.
Input parameters	@Programme_Code, @Title
Output parameters	None
Return Value	None

Operation Name	**Delete_Programme**
Involved Tables	Programmes
Overview	To allow to remove Programme record.
Pre-condition	Programmes table has been created and the programme record is in existence
Post-condition	The Programmes record will have been deleted
Input parameters	@Programme_Code
Output parameters	None
Return Value	None

One trigger is to be created that on updating a programme the old programme details are saved into an audit table. This audit table will not be exposed by the API but is to be put into place for future work.

One view is to be created that presents the administrator with the details of the students and the programmes they are registered on. This is not to be exposed as an endpoint but is to be considered as expansion capability for future work.

Business Layer

Not all aspects of the system will be specified. Only those of most importance are represented here. Please use the feedback sessions provided during the seminars to clarify aspects you are unclear of.

The patterns in use are: MVC, Repository.

The class diagram below illustrates the three core classes. The ProjectsController should be in the Controllers folder, the DataAccess and Project classes should be in the Models folder. The DataAccess class is representing the Repository pattern with a singleton class containing the operations to interact with the database. This ensures that the data remains throughout the lifecycle of the application.

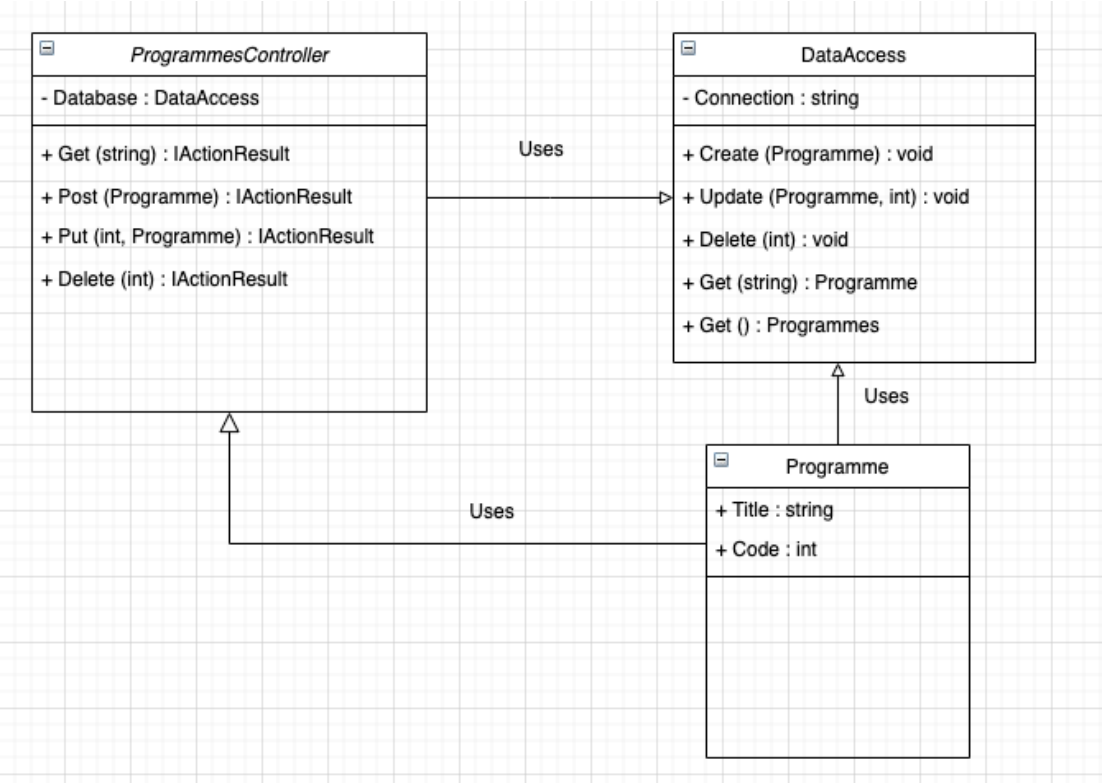


Figure 1: Class diagram

The sequence diagram below models the high level information regarding the incoming request to view all existing projects for a particular programme. The Actor represented here is a client application NOT a human user. Remember this is a web service representing machine-to-machine communication. The interface/boundary is represented by the TCP/IP network providing Internet connectivity.

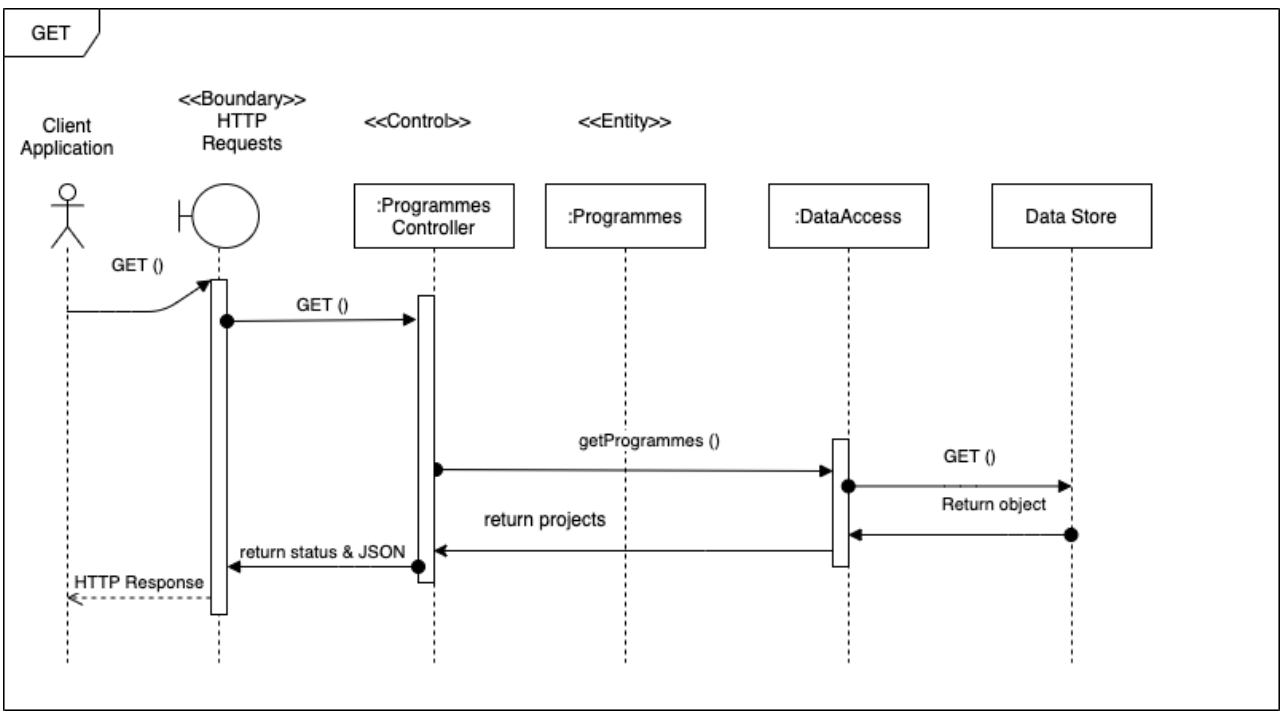


Figure 2: GET Sequence

In Figure 2 above, the flow of activity is as follows:

1. The client application constructs an HTTP packet as per the interface instructions. (See Interface Layer : API Endpoints for more information).
2. The ProgrammesController receives the HTTP request using the GET verb.
3. The ProgrammesController calls getProgrammes on the DataAccess object.
    a. IF you were using an incoming Programme Code to obtain just one programme record, this is where you would pass it in.
4. The DataAccess object calls the GET method, passing in the appropriate parameter and receiving the appropriate return.
5. The ProgrammesController interprets the response, sends back the appropriate HTTP Status code along with the appropriate JSON format.

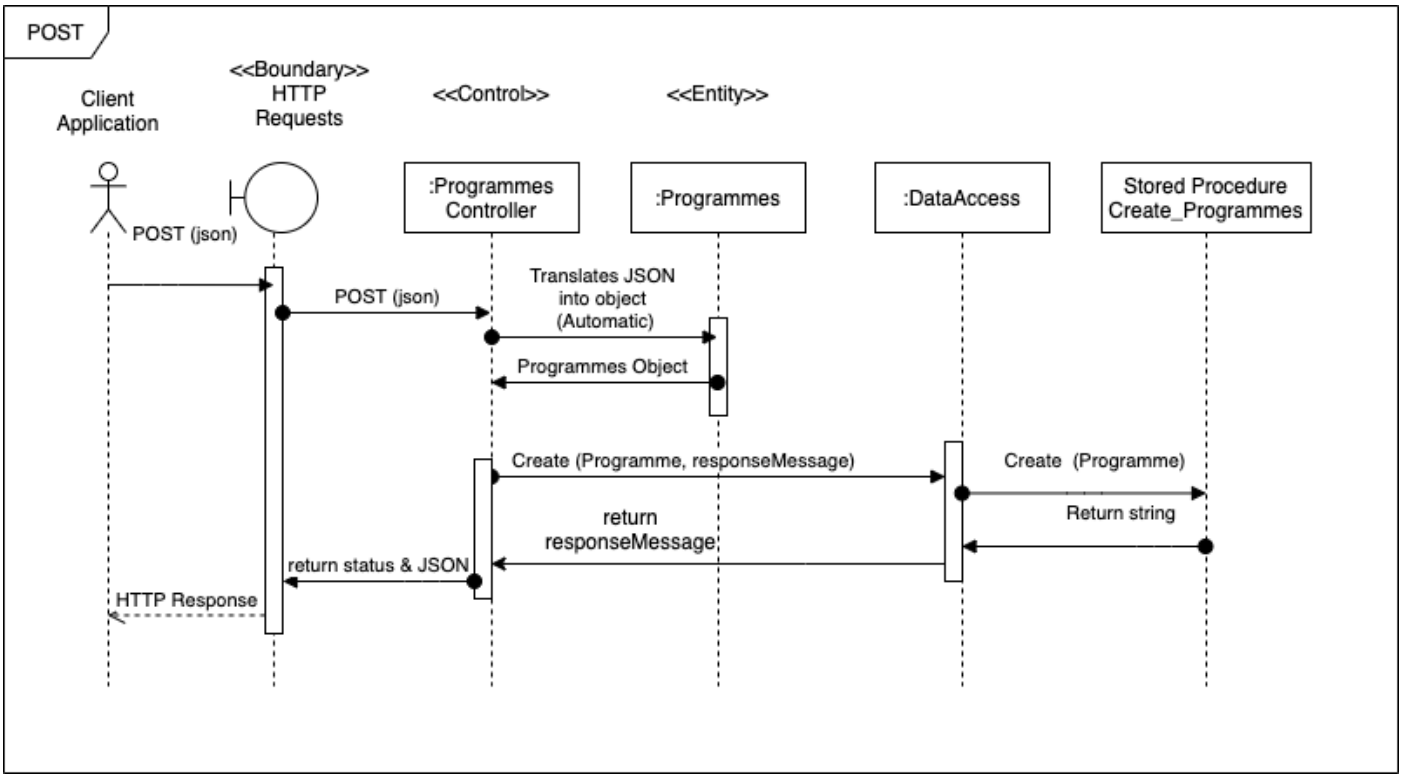


Figure 3: POST Sequence

In figure 3 above the sequence is very similar for handling the POST verb. PUT and Delete will again be similar and thus will not be represented here. Just to note that POST will return the Programme Code of the programme on a successful entry. This code can then be used within the client system as they wish.

Interface Layer : API Endpoints

The API description for the required endpoints, operations (HTTP verbs), authentication mechanism and expected input and output are all provided by the accompany yaml file found with this specification. The yaml file is designed as an Open API document and can be read by a Swagger editor. Using a browser, navigate to the online version of the Swagger editor at https://editor.swagger.io/ and open up the comp2001_api.yaml file.

Figure 4: Endpoints from Swagger file