University of Central Lancashire
UCLan Cyprus

Course: BSc Computing

Module Code: CO3411
Module Title: Distributed Enterprise Systems

Assessment Title: CO3411 Coursework

Type of assessment: Coursework

This assessment is worth **50%** of the overall module mark.

HOW, WHEN, AND WHERE TO SUBMIT:

The assessment release date is: 19/11/2024.

The assessment deadline date and time is: **14/03/2025 at 17:00.**

You should aim to submit your assessment in advance of the deadline.

Feedback will be provided by: 30/03/2025.

Note: If you have any valid mitigating circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you will need to apply online, via MyUCLan with your evidence **prior to the deadline**. Further information on Mitigating Circumstances via this link.

SUBMISSION DETAILS

You need to submit:

1) A .zip file containing:
    a. Your implementation of the RESTful API as a Node.js project along with any dependencies which are needed to run your code. This should be bundled in a folder with the following name: `username_events_app` (where username is your username).
    b. Postman script(s) that implement your testing efforts, bundled in a folder with the name `username_tests` (where username is your username).
2) The .zip file should be named in the following format: `username_CO3411.zip`
3) Your answers to the two questions. There will be a separate Turnitin link for this submission. You can submit either a Word or PDF file. Note that your submission will be checked for plagiarism.

Read this guidance carefully, and any questions, please discuss with your Module Leader.

Good luck!

Additional Support available:

All links are available through the online Student Hub

1. Academic support for this assessment will be provided by contacting the Module Leader – **Nicos Kasenides** (nkasenides2@uclan.ac.uk)

2. Our **Library resources** link can be found in the library area of the Student Hub or via your subject librarian at CyprusLibrary@uclan.ac.uk

3. For help with Turnitin, see Blackboard and Turnitin Support on the Student Hub

4. If you have a disability, specific learning difficulty, long-term health or mental health condition, and not yet advised us, or would like to review your support, **Student Support Officers (cyprusstudentsupport@uclancyprus.ac.cy)** can assist with reasonable adjustments and support. To find out more, you can visit the **Student Support Service** page of the Student Hub. You can also call +357 24694026 or +357 24694108 or +357 24694073.

5. For mental health and wellbeing support, please complete our online referral form or email the Psychological Wellbeing and Counselling Centre at UCLan Cyprus at wellbeing@uclancyprus.ac.cy.

6. For any other support queries, please contact **Student Support** via CyprusStudentSupport@uclan.ac.uk.

7. For consideration of Academic Integrity, please refer to detailed guidelines in our policy document . All assessed work should be genuinely your own work, and all resources fully cited.

For this assignment, you are not permitted to use any category of AI tools.

PREPARING FOR YOUR ASSIGNMENT.

Before completing the assessment, you should ensure you are up to date with all the practical lab exercises on Blackboard. These exercises are designed to give you working with the tools and techniques required to complete the assignment. Refer to the Module Information Pack to understand the Learning Outcomes and Marking Criteria.

University of Central Lancashire
UCLan Cyprus

Learning outcomes

This assessment has been designed to assess the following learning outcomes:

> **LO1** – Critically evaluate the role of patterns and frameworks in the design of an enterprise application architecture.
>
> **LO2** – Compare potential technologies for the development of an enterprise application.
>
> **LO3** – Implement a distributed application using appropriate technology and frameworks.

You should spend approximately 20 hours on this assignment. Don't leave this work until the last couple of days before the submission date. Make sure you identify problems early so they can be addressed.

Instructions

This assignment is divided into two parts. In the first part you will have to develop a RESTful API for an event planning system. In the second part you will have to answer questions that assess your knowledge of distributed enterprise systems and your critical thinking.

Part 1 – Implementation (80% of the coursework)

You need to develop a RESTful API for an event planning system. You will be responsible for creating the services which will manage several entities such as users, organizers, events, and more. The specifications for these services are provided to you, and you will need to develop these services using a modern backend development runtime: Node.js.

Part 2 - Questions (20% of the coursework - each question with equal weight)

Answer the following two questions:

1. For part 1 you used an asynchronous event driven architecture based on Node.js. Briefly discuss how the use of an alternative architecture and/or framework could have helped implement your backend architecture from Part 1. Where appropriate use citations and references to justify your answer. (Suggested answer length: 150-300 words.)
2. For part 1 you were asked to use Node.js with Express and possibly EJS. List at least two more alternative technologies you could have used for implementing this API. Pick one of these two and compare it with Node/Express/EJS. What are the advantages and disadvantages of each technology? (Suggested answer length: 150-300 words.)

For each of the two questions, you need to answer not only from your own experience, but also from looking up the literature, finding and citing proper academic sources. Consult the marking scheme for more details.

Before you begin

Before attempting this assessment, it is highly recommended that you revisit the lecture slides and lab worksheets, and any notes you may have taken during these sessions. These provide

all the necessary information for you to successfully complete this assessment. All resources are available on the CO3411 Blackboard area under Module Materials.

Implementation

The following specification must be followed in a way that demonstrates your understanding of the RESTful approach. You **must** use the following tools for this assignment:

- The services must be implemented using **Node.js** and **Express.js**.
- For persistence, you need to use either **SQLite** or **Sequelize**.

You must also make the following assumptions:

- Services may be used concurrently. Therefore, you must use **transactions** where necessary to ensure that any checks and subsequent update operations are carried out atomically to maintain consistency.
- The quality of the code must also be inspected using an appropriate testing strategy. You must provide evidence of **testing** using automated scripts for some aspects of your implementation.

System model

The event planning backend comprises of the following data entities. You are expected to create the database structure and map this model to a valid set of types and constraints in your database tables.

User
id (Number, PK, AI)
username (Text, Unique)
firstname (Text)
lastname (Text)

Reservation
id (Number, PK, AI)
eventID (Number, FK)
userID (Number, FK)

Event
id (Number, PK, AI)
eventTypeID (Number, FK)
organizerID (Number, FK)
name (Text)
price (Number)
dateTime* (Number)
locationLatitude (Number)
locationLongitude (Number)
maxParticipants (Number)

EventType
id (Number, PK, AI)
name (Text)

Organizer
id (Number, PK, AI)
name (Text)

* Dates/times must be represented as Unix timestamps.

University of Central Lancashire
UCLan Cyprus

API Overview

The following diagram provides an overview of the event planning API. In this system, each resource contains several services which must be defined following the RESTful specifications and implemented as web services.

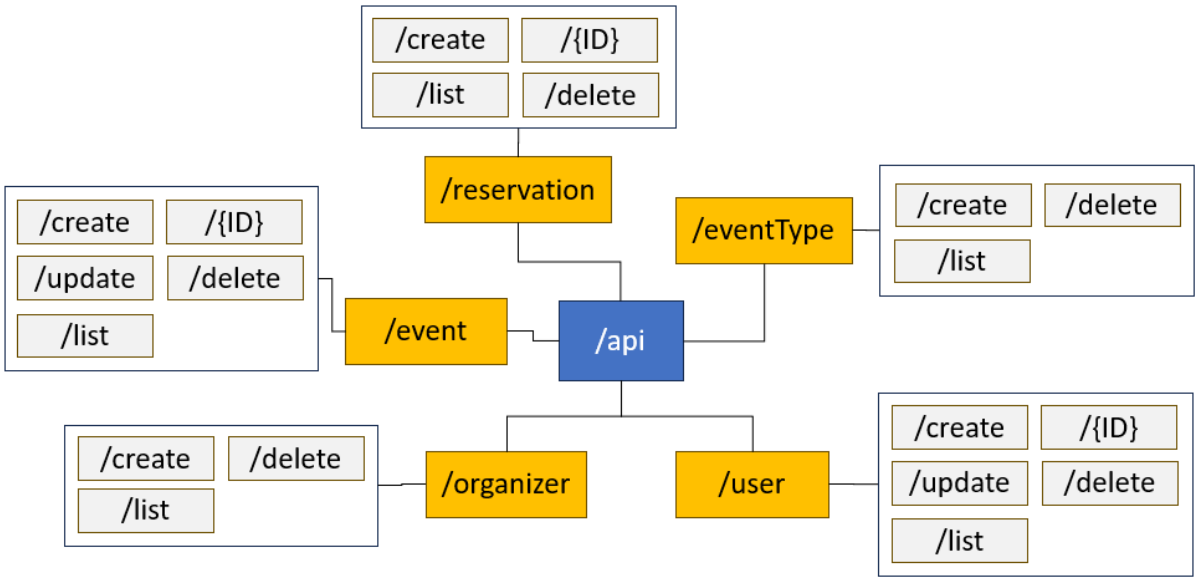

Figure 1: An overview of the event planning system.

University of Central Lancashire
UCLan Cyprus

Service specifications

The event planning API you develop must conform to the constraints of the RESTful architecture style. It should include the following services:

Users

Create user	**40%+**
Description	Creates a new user.
URI	…/api/user/create
HTTP Method	POST
Parameters	-

Body	JSON object containing information used to create a student.
	`{`
	`  "username": "jsmith",`
	`  "firstname": "John",`
	`  "lastname": "Smith"`
	`}`

Constraints	• All parameters must be provided and must be non-empty strings. • The username must contain only alphanumeric characters • The username must be unique. • The firstname and lastname must contain at least 2 characters.
Success return	Status 200 A JSON object containing the details for the newly created item.
Error return	• Status 422 if any of the parameters are invalid or missing. • Status 409 if a student with the specified username already exists. *All error returns must be accompanied by a relevant error message.*

Get user	**40%+**
Description	Retrieves data for a user.
URI	…/api/user/**{ID}**
HTTP Method	GET
Parameters	-
Body	-
Constraints	• The ID must be provided, and must be a valid ID. • A user with this ID must exist.
Success return	Status 200 The details of a user object.
Error return	• Status 422 if the ID is invalid. • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

Delete user	50%+
Description	Deletes a user.
URI	…/api/user/delete
HTTP Method	DELETE
Parameters	ID (Integer)
Body	-
Constraints	• The ID must be provided, and must be a valid ID. • A user with this ID must exist. • The user must not be referenced in any other entities.
Success return	Status 200 The text "OK".
Error return	• Status 422 if the ID is invalid or the item could not be deleted. (e.g., when another item is referencing it). • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

Update user	50%+
Description	Updates a user's details.
URI	…/api/user/update
HTTP Method	PUT
Parameters	-

Body	A JSON object containing the data for the updated student.
	Example:
	`{`
	`  "id": 1,`
	`  "username": "jsmith",`
	`  "firstname": "John",`
	`  "lastname": "Smith"`
	`}`

Constraints	• The ID must be valid and an item with this ID must exist. • Fields must follow the format and constraints specified in the create user service when specified. • The ID field must not be updated.
Success return	Status 200 A JSON object containing the updated details for the user.
Error return	• Status 422 if the ID is invalid or if any of the parameters have invalid format. • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

List users	60%+
Description	Lists all users.
URI	…/api/user
HTTP Method	GET
Parameters	• eventID (optional)
Body	-
Constraints	• If an eventID is provided, only users with reservations to that event should be returned.
Success return	Status 200 A JSON array containing the details for the listed users. The array may be empty if no users were found.
Error return	• Status 422 if any of the parameters are invalid. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

Organizer

Create organizer	40%+
Description	Creates a new event organizer.
URI	…/api/organizer/create
HTTP Method	POST
Parameters	-

Body	JSON object containing information used to create an event organizer.
	{
	"name": "Radisson Blu"
	}

Constraints	• All parameters must be provided. • The name must be between 2-255 characters long. • The name must be unique.
Success return	Status 200 A JSON object containing the details for the newly created item.
Error return	• Status 422 if any of the parameters are invalid or missing. • Status 409 if an organizer with the same name already exists. *All error returns must be accompanied by a relevant error message.*

Delete organizer	50%+
Description	Deletes an event organizer.
URI	…/api/organizer/delete
HTTP Method	DELETE
Parameters	ID (Integer)
Body	-
Constraints	• The ID must be provided, and must be a valid ID. • An organizer with this ID must exist. • The organizer must not be referenced in any other entities.
Success return	Status 200 The text "OK".
Error return	• Status 422 if the ID is invalid or the item could not be deleted. (e.g., when another item is referencing it). • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

List organizers	60%+
Description	Lists event organizers.
URI	…/api/organizer
HTTP Method	GET
Parameters	hasEvents (optional)
Body	-
Constraints	• When hasEvents is provided (Boolean), only organizers which have scheduled events should be returned.
Success return	Status 200 A JSON object containing room information `[ { "id": 1, "name": "Radisson Blu" }, { "id": 2, "name": "Logicom" } ]`
Error return	• Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

Event Types

Create event type	40%+
Description	Creates a new event type.
URI	…/api/event-type/create
HTTP Method	POST
Parameters	-

Body	JSON object containing information used to create an event type.
	{
	"name": "Conference"
	}

Constraints	• The name must be provided and must have a length of 2-255 characters.
Success return	Status 200 A JSON object containing the details for the newly created item.
Error return	• Status 422 if any of the parameters are invalid or missing. • Status 409 if a module with the specified code already exists. *All error returns must be accompanied by a relevant error message.*

Delete event type	50%+
Description	Deletes an event type.
URI	…/api/event-type/delete
HTTP Method	DELETE
Parameters	ID (Integer)
Body	-
Constraints	• The ID must be provided, and must be a valid ID. • An event type with this ID must exist. • The event type must not be referenced in any other entities.
Success return	Status 200 The text "OK".
Error return	• Status 422 if the ID is invalid or the item could not be deleted. (e.g., when another item is referencing it). • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

List event types	40%+
Description	Lists event types.
URI	…/api/event-types
HTTP Method	GET
Parameters	-
Body	-
Constraints	-
Success return	Status 200 A JSON array containing the details for the event types. The array may be empty if no event types were found. `[ { "name": "Conference" }, { "name": "Workshop" }, { "name": "Sports event" } ]`
Error return	• Status 422 if any of the parameters are invalid. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

Events

Create event	40%+
Description	Creates a new event.
URI	…/api/event/create
HTTP Method	POST
Parameters	-

Body	JSON object containing information used to create an event.
	`{`
	`"eventTypeID": 3,`
	`"organizerID": 3,`
	`"name": "Radisson Blu Larnaka International Marathon",`
	`"price": 30,`
	`"dateTime": 1731835800,`
	`"locationLatitude": 34.915147,`
	`"locationLongitude": 33.638146,`
	`"maxParticipants": 200`
	`}`

Constraints	• All parameters must be provided. • Note that numOfParticipants is excluded, and automatically initialized to 0. • The name must be non-empty, contain only alphanumeric characters, and have a length between 2-255 characters. • All IDs must be valid integers and exist in the database. • The price must be a non-zero positive real number. • The dateTime must be a valid timestamp, and be in the future. • locationLatitude must be a valid latitude, between the values -90<=X<=+90. • locationLongitude must be a valid longitude, between the values -180<=Y<=180. • maxParticipants must be a valid non-zero positive integer.
Success return	Status 200 A JSON object containing the details for the newly created item.
Error return	• Status 422 if any of the parameters are invalid or missing. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

Delete event	50%+
Description	Deletes an event.
URI	…/api/event/delete
HTTP Method	DELETE
Parameters	ID (Integer)
Body	-
Constraints	• The ID must be provided, and must be a valid ID. • An event with this ID must exist. • The event must not be referenced in any other entities.
Success return	Status 200 The text "OK".
Error return	• Status 422 if the ID is invalid or the item could not be deleted. (e.g., when another item is referencing it). • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

Get event	40%+
Description	Retrieves an event.
URI	…/api/event/**{ID}**
HTTP Method	GET
Parameters	-
Body	-
Constraints	• The ID must be provided, and must be a valid ID.
Success return	Status 200 The details of a reservation object.
Error return	• Status 422 if the ID is invalid. • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

Update event	50%+
Description	Updates an event's details.
URI	…/api/event/update
HTTP Method	PUT
Parameters	-

Body	A JSON object containing the data for the updated event.
	Example:
	`{`
	`"id": 4,`
	`"eventTypeID": 3,`
	`"organizerID": 3,`
	`"name": "Radisson Blu Larnaka International Marathon",`
	`"price": 30,`
	`"dateTime": 1731835800,`
	`"locationLatitude": 34.915147,`
	`"locationLongitude": 33.638146,`
	`"maxParticipants": 200`
	`}`

Constraints	• The ID must be valid and an item with this ID must exist. • Fields must follow the format and constraints specified in the create event service when specified. • The ID field must not be updated.
Success return	Status 200 A JSON object containing the updated details for the event.
Error return	• Status 422 if the ID is invalid or if any of the parameters have invalid format. • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

University of Central Lancashire
UCLan Cyprus

List events	60%+
Description	Retrieves a list of scheduled events based on various parameters.
URI	…/api/event
HTTP Method	GET
Parameters	• organizerID (optional) • eventTypeID (optional) • dateTime (optional) • userIDs (optional)
Body	-
Constraints	• If provided, the referenced organizer in organizerID must exist in the database. When this is provided, only events of this organizer should be listed. • If provided, the referenced eventTypeID must exist in the database. When this is provided, only events of this type should be listed. • If provided, the dateTime must be a valid UNIX timestamp. When provided, the dateTime must filter the returned events to only those which are after that time. • If provided, the userIDs must be a list of values separated by commas (CSV). When provided, only events which are attended by these users should be listed.

Success return	Status 200
	A JSON array containing objects for the scheduled sessions meeting the constraints.
	[
	{
	"id": 4,
	"eventTypeID": 3,
	"organizerID": 3,
	"name": "Radisson Blu Larnaka International Marathon",
	"price": 30,
	"dateTime": 1731835800,
	"locationLatitude": 34.915147,
	"locationLongitude": 33.638146,
	"maxParticipants": 200
	},
	{
	"id": 4,
	"eventTypeID": 3,
	"organizerID": 3,
	"name": "Radisson Blu Larnaka International Marathon - 5K Race",
	"price": 30,
	"dateTime": 1731835800,
	"locationLatitude": 34.915147,
	"locationLongitude": 33.638146,
	"maxParticipants": 300
	}
	]

Error return	• Status 422 if any of the parameters have an invalid format, or the constraints are not met correctly. • Status 404 if an item with a specified ID does not exist.

University of Central Lancashire
UCLan Cyprus

Reservations

Create reservation	40%+
Description	Creates a reservation.
URI	…/api/reservation/create
HTTP Method	POST
Parameters	-

Body	A JSON object containing the data for the reservation.
	Example:
	`{`
	`  "eventID": 2,`
	`  "userID": 4`
	`}`

Constraints	• Any referenced items must exist in the database. • Each user is only allowed to make one reservation per event. • The service must check whether there is a slot available for the reservation to be made (through maxParticipants in the Events table, and by first counting reservations previously made to this event). • If there isn't a slot available for this event, no reservation should be made, and the service should produce an error (see below for error returns). • When a reservation can be made, a new Reservation record should be created.
Success return	Status 200 A JSON object containing the details for the created reservation.
Error return	• Status 422 if any of the parameters have an invalid format, or the constraints are not met correctly, or if there aren't enough slots available to make the reservation. • Status 404 if an item with a specified ID does not exist. *All error returns must be accompanied by a relevant error message.*

Get reservation	40%+
Description	Retrieves a reservation.
URI	…/api/reservation/**{ID}**
HTTP Method	GET
Parameters	-
Body	-
Constraints	• The ID must be provided, and must be a valid ID.
Success return	Status 200 The details of a reservation object.
Error return	• Status 422 if the ID is invalid. • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

Delete reservation	50%+
Description	Deletes an event.
URI	…/api/reservation/delete
HTTP Method	DELETE
Parameters	ID (Integer)
Body	-
Constraints	• The ID must be provided, and must be a valid ID. • A reservation with this ID must exist. • When a reservation is deleted, the numOfParticipants counter in the corresponding event must be decreased by 1.
Success return	Status 200 The text "OK".
Error return	• Status 422 if the ID is invalid or the item could not be deleted. (e.g., when another item is referencing it). • Status 404 if an item with this ID does not exist. *All error returns must be accompanied by a relevant error message.*

List reservations	60%+
Description	Lists reservations
URI	…/api/reservation
HTTP Method	GET
Parameters	userIDs (optional) eventIDs (optional)
Body	-
Constraints	• IDs provided must be valid IDs. • If provided, userIDs must be a valid list of IDs separated by comma (CSV) and will cause the service to show reservations only for the users with these IDs. • If provided, eventIDs must be a valid list of IDs separated by comma (CSV) and will cause the service to show reservations only for the events with these IDs. • The userIDs and eventIDs parameters may **not** be used together. If they are used in combination, the service should produce an error.

Success return	Status 200 + a list of reservation data:
	`[`
	`  {`
	`  "eventID": 2,`
	`  "userID": 4`
	`  },`
	`  {`
	`  "eventID": 2,`
	`  "userID": 5`
	`  }`
	`]`

Error return	• Status 422 if any ID is invalid, or a combination of the userIDs and eventIDs parameters are used together. • Status 404 if any item with a specified ID does not exist. *All error returns must be accompanied by a relevant error message.*

General guidelines and constraints

- When a requested resource is not found, return an error message with code (404).
- When a request has missing or invalid parameters, return an error message with code (422).
- When a request to create/edit a resource has a conflict, return an error message with code (409).
- For any other client-side errors (besides 404, 409, 422), return an error message with code (400).
- For any unexpected, server-side errors, return an error message with code (500).

You can (and must) return relevant status error codes and error messages in cases where you may see fit.

Testing

You are required to provide tests for showing that the RESTful web services you have developed satisfy the requirements. You should provide sufficient evidence which may include the **Postman script** and a **1-page verbal description** of how it verifies:

- CRUD operations
- Error handling
- Persistence
- Concurrency

Your testing script must attempt to cover a wide range of possible inputs and outputs for your web services.

University of Central Lancashire
UCLan Cyprus

Marking scheme

This assignment counts towards 50% of the module grade. A summary of the marking criteria for each component is defined below. Full details surrounding the grade boundaries can be found in the University's Assessment Handbook located here.

Part 1 (Implementation + Testing)

Each level assumes sufficient completion of all previous levels.

If you do not achieve the minimum requirements for 40% (Pass/Third class) then your grade will be 0-35% (Fail).

40%+ (Pass/Third class)
• Submit a Node.js project which works without major launch/runtime errors. • Implement most of the C (Create) and R (Retrieve) calls, possibly without full conformance to the required constraints. • Include evidence of testing of the corresponding calls, in the form of a Postman script or a command line script (e.g., using curl).
50%+ (Lower second class)
• Implement all the C (Create), R (Retrieve), U (Update) and D (Delete) calls, possibly without full conformance to the required constraints. • Include evidence of testing of the corresponding calls, in the form of a Postman script. • Produce sensible error codes and messages. • Provided API is properly tested with a Postman script.
60%+ (Upper second class)
• Implement all calls outlined in the specified API, with nearly full conformance to the required constraints. • Include evidence of testing of the corresponding calls, in the form of a Postman script. • Provide evidence that your API call to create a reservation (/api/reservation/create) behaves as expected when used concurrently by many users (i.e., sessions cannot be scheduled at the same time). • Use transactions as needed.
70%+ (First class)
• Provide evidence that your full API behaves as expected when used concurrently by many users. Where needed, use transactions. • Well documented code that is clear to follow and understand. • Elaborate Postman test script with validation of return error codes and returned values where appropriate, for most of the calls.
85%+ (High first class)
• Full implementation of all calls, and full realization of all identified constraints. • Implementation of a management GUI as a simple front-end application that allows users to manage data by utilizing the system's API. For this you can use **EJS.** • Very well documented code that is of comparable quality to the documentation found in widely used software libraries, with considerations for efficiency and clarity. • Extensive testing script covering all calls and all aspects of the API.

University of Central Lancashire
UCLan Cyprus

Part 2 (Questions)

The same marking scheme applies for either of the two questions.

If you do not achieve the minimum requirements for 40% (Pass/Third class) then your grade will be 0-35% (Fail).

40%+ (Pass/Third class)
• Submitted an answer that covers the questions to some extent but there are several vague or unclear points. • Includes some evidence of literature research and analysis but remains largely rudimentary.
50%+ (Lower second class)
• Submitted an answer which answers the question however it suffers from flaws in content and in structure. • Includes evidence of literature research and analysis.
60%+ (Upper second class)
• A decent effort that answers the question and is well structured and well expressed. • Includes clear evidence of literature research and analysis, using high quality references, and applying best practice in citing.
70%+ (First class)
• An excellent effort that fully answers the questions and is very well structured and expressed. • Includes excellent evidence of literature research and analysis, using high quality references, and applying best practice in citing. • There is evidence of synthesis of knowledge from combining the data in the studied material, and from the student experience with Part 1.

University of Central Lancashire
UCLan Cyprus

Reflecting on Feedback: how to improve.

From the feedback you receive, you should understand:

- The grade you achieved
- The best features of your work
- Areas you may not have fully understood
- Areas you are doing well but could develop your understanding.
- What you can do to improve in the future - feedforward

Next Steps:

- List the steps have you taken to respond to previous feedback.
- Summarize your achievements
- Evaluate where you need to improve here (keep handy for future work):