304CEM

Web API Development

Assignment Brief

You need to develop a full stack, full restful web application. The exact topic you choose is entirely up to you, you can decide to create a news website, a Recipe website, a Cinema website, a portfolio website or any other ideas you are interested in.

You will need to do four tasks:

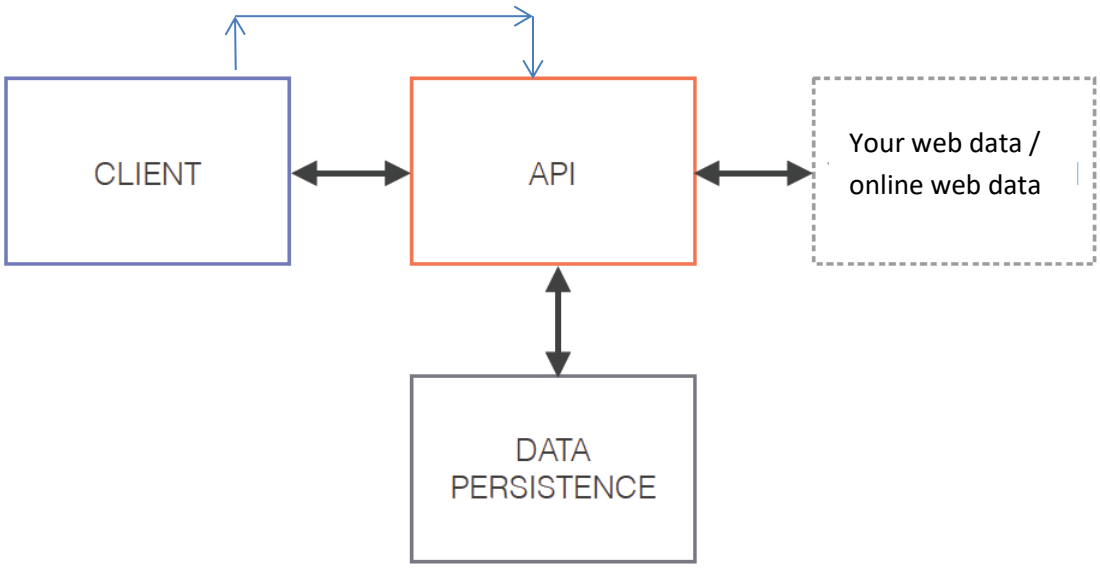
1- Create a persistent data storage (SQL, noSQL, JSON files, text files or any other technology you think suitable for your website) and write one page report about its design 2- A backend system in a form of a full Restful public web API using NodeJS, in case you are not comfortable with NodeJS you may only use Python, no other alternatives are permitted. For this part of the assignment the source code should be hosted on a GIT repository 3- A web application using combination of modern web UI technologies such as Bootstrap and JQuery. The front-end should communicate with your backend system, the source code should hosted on a GIT repository. 4- A link to your live application or a video recording of your application.

Task 1: Persistent Storage (10%)

You should create a persistent data storage, you can choose the technology you think will be most suitable for your application, it can be a SQL database (e.g. mySQL, SQL Server), a noSQL database (e.g. MongoDB, CouchDB) a files system, text files or any other technology. You should explain your database design and justify the design and the chosen technology in a one page report and submit it through moodle.

Architecture

The diagram below may help you understand the requirements. Notice that the client interacts with your API **OR** directly interact with the third-party API(s)s and with your data persistence solution.



Task 2: Backend System (Public Web API) (40%)

You should create a backend system for your application, this backend system should be in a form of a full Restful public web API, you will learn how to create such system during this module, your backend system should be written in NodeJS only, you can use any framework or library available, so for example you may use NodeJS + RESTIFY. Please note that you will get a zero grade if you choose other languages. Your backend system should serve at least the following functionality:

1- User Registrations
2- Login System and Authentication.
3- At least two CREATE methods using POST requests to insert data in the database (apart from user registration or login activities)
4- At least two UPDATE methods using PUT requests to update data in the database (apart from user registration or login activities)
5- At least two DELETE methods using DELETE requests to delete data from the database (apart from user registration or login activities)
6- At least two RETRIEVE methods using GET requests (apart from user registration or login activities)

For example, you may suggest developing a Recipe website, then your minimum requirements could be:

1- A public API for user's registrations
2- A public API for user's login
3- Public APIs to add a recipe, and add an existing recipe to a favourite list(2 CREATE methods)
4- Public APIs to update a recipe, a comment, and a specific user's profile information (2 UPDATE methods)
5- Public APIs to delete a recipe, and a specific recipe from a specific user's favourite list (2 DELETE methods).
6- Public APIs to retrieve a recipe, and a specific user's favourite list. (2 RETREIVE methods).

You should submit a link to your GIT repository.

Task 3: The front-end System (45%)

You will need to prototype, design, and implement a single page application for your project using a mixture of HTML5, Vanilla Javascript, Bootstrap, JQuery. For more extra credits and challenge you may use Single Page Application frameworks such as AngularJS or Vue.JS. You should not use any other front-end system frameworks other than what is proposed above, the reason for that, is to make marking and assessment consistent. The web application should connect to the back-end server to read and write data. This application should at least allow the user to register, login, and perform the other required 12 operations (2 creates, 2 updates, 2 deletes, 2 retrieves).

You will be assessed on the design of your front-end and its, completeness, effectiveness and

Task 4: Live Application or Video Recording (5%)

You can evidence your work with either submitting a link to your live application **or** submitting a video recording of it.

1. Live demonstration link **OR**
2. A link to your reflective video (hosted on YouTube)

Live demonstration

If you choose to submit a link to your live application then you can host it anywhere on the internet, please make sure it is deployed and functional before the assignment deadline and that the deployed version matches the source code you submit also make sure the application stays live at least until the assessment is completed and final grades are released, you will be provided with a list of free hosts for each possible technology you may use.

Video

Once you have completed the API and the client, you need to explain how your API and client work. Rather than writing a report, you are required to record a short screencast of 8 min or less. This should cover all the points in the grading criteria and demonstrate your skills and knowledge of the subject. After uploading, you must change the video permissions from private to **unlisted**. This will allow the lecturers to access the video via the link but prevent it being publicly searchable.

Make sure you separately demonstrate the API (using cURL or Postman) and the client and ensure you justify your choices of language constructs and architecture.

1. Demonstrate your API showing the requests and responses (headers/body)
2. Demonstrate the back-end persistence showing how the data is stored
3. Run the unit and acceptance tests on the API explaining code coverage
4. Demonstrate the features of the web client
5. Run the unit and acceptance tests on the web client explaining code coverage
6. Video narration must be done by **yourself.** For any suspected voice noticed, marks will be deducted.

Final Submission

This assignment requires you to demonstrate the range of skills and knowledge required by industry and this is reflected in the coursework submission. You are required to submit the following. 1. Links to your Git remote repositories (hosted on University Github), you will have a repository for your backend API and a repository for your front-end. 2. A link to your live application where all components working together (persistent storage + API + Front-end) or a link to your reflective video (hosted on YouTube) 3

Source Code

Managing source code is a vital skill if you are to become a successful developer. For this assignment, you are required to track your API code and your client code in separate Git repositories and you will be marked on how efficiently you organise this.

You will be required to submit links to both your Git remotes hosted on Github. Both repositories should include full documentation available through the home page.

To ensure the code can be seen by your lecturers, make sure you give them **reporter** permission. Their usernames are:
alexng88

Grading

- 45% Front-end
- 40% API
- 10% Persistent Storage
- 5% Video Demo OR Live App & API

304CEM Grading Rubric for Front-end (45%)

	0	1	2	3	4	5
Version Control 5%	No access given to the remote repository on GitLab.	Code access provided but does not match the code demonstrated.	Access provided to GitLab however there is no evidence of regular commits.	Access has been given to the code on GitLab with evidence of regular commits.	Evidence of regular commits over an extended period of time.	Commits over an extended period of time, demonstrating the use of branching and merging.
Responsive 5%	No responsive design	An attempt for a responsive design but it is not working properly.	Around half of the pages are responsive, or there are all responsive with major issues in each page	Most of the pages are responsive, with very few major issue with design such as wrong placement of large components	Fully responsive design, with minor issues, such as wrong placement of some small elements or extra shifts in some components	Fully responsive design UI
CSS 10%	No CSS used at all	Most of the CSS are working from the template. No real modification or use of its customised CSS	Half of the website is based on the original and customised CSS which heavily dependent on the template CSS	Most of the pages used original CSS, with major components throughout the pages used the template CSS	Well designed and original CSS, with some elements used from the template	Well designed, original CSS creation, completely different from the standard template
Front-end code Architecture 5%	No architecture	An attempt for a front-end code architecture was made, but it is not working properly, or it is mostly incomplete	Code architecture is not complete, with major flaws in some pages	Generally good architecture that needs improving in some places.	Good architecture with some minor flow in the design of the code	Clear architecture, clear separation of JS, CSS and contents
Front-end Coding (HTML & JS) 10%	No code supplied through GitLab	An attempt has been made to write code to implement some of the basic functionality although this may not be successful. No attempt at documentation.	Working code base showing the application of basic programming principles. Code may contain linting errors and warnings.	Demonstration of the usage of modularity to organise the code. Code contains linting warnings but no errors.	The code is modular. Code contains no linting errors or warnings with minor errors in the code	Code contains no linting errors or warnings.
Completeness 5%	Zero achievement	At attempt has been made to create front-end for the requested API, however they still need major work and improvements	Have of the back-end services have been utilised in the front-end	Most of the requested back-end APIs were used in the front-end, with some having minor or major issue	Most of the requested back-end APIs have front-end pages ready to serve them	All requested backend APIs have been utilised in the front-end
Connecting to APIs 5%	No connecting to APIs from the front-end	An attempt for connecting and retrieving data from the APIs was done but it is mostly having issues in connecting	half of the pages connecting and retrieving data from the APIs with few having minor or major issues in connecting	Most of the pages connecting and retrieving data from the APIs with few having minor or major issues in connecting	All pages connecting and retrieving data from the API with few having minor or major issues in connecting	All pages connecting and retrieving data from the API

304CEM Grading Rubric for Backend API (40%)

	0	1	2	3	4	5
Version Control 5%	No access given to the remote repository on GitLab.	Code access provided but does not match the code demonstrated.	Access provided to GitLab however there is no evidence of regular commits.	Access has been given to the code on GitLab with evidence of regular commits.	Evidence of regular commits over an extended period of time.	Commits over an extended period of time, demonstrating the use of branching and merging.
Automated Testing 5%	No tests have been written or run.	Limited attempt at flawed tests.	An attempt has been made to write simple tests	Evidence of a limited number of tests written and run.	A range of tests showing how they contribute to code quality.	A full suite of automated tests ensuring full code coverage
API Design 10%	No API demonstrated in the screencast	An attempt has been made to implement a basic API however this does not work as expected	Simple functional API demonstrating a basic understanding of REST principles (resources, collections, methods and headers)	The API is fully functional and includes an authentication mechanism. The API demonstrates a good understanding of REST principles.	The API demonstrates user registration and authentication. It provides feedback for invalid requests through appropriate response codes and messages. Limited HATEOAS support.	Fully REST-compliant API that includes filtering and sorting and conditional GET requests. It makes use of the full range of request and response headers. Full HATEOAS implementation.
Architecture 5%	No code supplied through GitLab	An attempt to write the API however it fails to work correctly.	All code for routing and business logic maintained in a single file	Code split into several files but overlap between routing and business logic.	Clear separation between routing and business logic code.	Clear separation between routing and business logic code with no code duplication.
Coding 10%	No code supplied through GitLab	An attempt has been made to write code to implement some of the basic functionality although this may not be successful. No attempt at documentation.	Working code base showing the application of basic programming principles. Code may contain linting errors and warnings. No attempt at code documentation	Demonstration of the usage of modularity to organise the code. Code documentation is incomplete. Code contains linting warnings but no errors.	The code is modular and includes full exception-handling. Code is fully annotated and explained. Code contains no linting errors or warnings and is fully documented.	The API demonstrates a wide range of appropriate language constructs including clear modular structure. Code contains no linting errors or warnings and is fully documented.
Completeness 5%	No APIs were developed at all, or all of the created APIs are not working	Only few of the requested APIs were created, the exiting APIs may lack functionality	Half of the requested APIs were created and working properly, the existing APIs may not work or lack functionality	Most of the requested APIs were created, the existing APIs may not work or lack functionality	All requested API were created, with some of them may not working or lack functionality	All requested APIs were created and all working correctly

304CEM Grading Rubric for Persistent Storage (10%)

	0	1	2	3	4	5
Persistent Storage Design 10%	No Persistent storage at all	An attempt has been made a persistent storage however it is not working or lacking most of the functionality	Persistent storage working, however there is major work needed to improve the design.	A working persistent storage design with some major issues	Well-designed persistent storage with few minor improvements needed to improved efficiency	Well design and fully functional persistent storage

304CEM Grading Rubric for Video or Live Application (5%)

	0	1
Video 10%	No video submitted	A video was submitted
Live Application 10%	No live Application	A live application was submitted

Marks will be deducted on late submission.

1 week	Your marks x 90%
2 weeks	Your marks x 80%
More than 2 weeks	Your marks x 0%

Deadline

31 July 2019