Interactive Web Applications

cct College Dublin
Computing • IT • Business

Module Title:	Interactive Web Applications
Assignment Type:	Continuous Assessment 2
Project Start Date:	14/04/2020
Assignment Compiler:	Mikhail Timofeev (mikhail@cct.ie)
Weighting:	35%
Due Date:	**23:59, 24/04/2020**
Method of Submission:	**GitHub** for the code, **Moodle** for the report
Late submission:	According to the accepted **CCT** guidelines
Last document update:	14/04/2020

Module Learning Outcomes Assessed

MLO1 Assess the limitations of static Web solutions and design dynamic interactive web application solutions using a variety of industry standard Web application APIs and technologies

MLO4 Analyse existing web applications needs to further integrate new cross-site communications services

MLO5 Review and critique currently available development frameworks to aid user experience and recently developed technologies for future implementation

MLO7 Design embeddable services that can integrate simple web applications into hosted web services

Assignment Description

Choose an area of your interest to design, create and deploy an interactive web app that incorporates:

[30%] Create an **API** service (you **must** use **Node.js** and **MongoDB**) that supports **full CRUD** for your existing **CA1 app**

[30%] Update your existing **CA1 app** to allow populating the **HTML** table with **JSON** data provided by the **API**, implement the ability to **Add** and **Delete** items from that table **using your new API only** from your front-end. You are free to use any available front-end frameworks/plugins to implement that functionality (Vanilla JavaScript, jQuery, Angular, etc.)

[15%] Deploy your application to **Heroku** and your database to **MongoDB Atlas**.

[15%] Possess a coherent commit history on **GitHub** (at least 8 valid and logical commits) and have your final submission code present in your **Git** repository

[10%] The finished application should be accompanied by a **report** (5 pages max) that outlines the difference between the existing **CA1** and the resulting **CA2** apps, reviews which approach was better in your opinion (**CA1** vs. **CA2**) and explains the reasons of choosing a particular framework/plugin for your app at Step 2. The cover sheet for the report should include a **full student name**, **student number**, **project title**, the project's **GitHub repository link** and **the link to the deployed application on Heroku**.

Interactive Web Applications

cct College Dublin
Computing • IT • Business

Additional Information

- This project is an **individual project** and you have all the freedom to work on a topic of your interest.

- The application must be developed using an online IDE – **Gitpod.io** (the free tier includes 100 hours a month for developing/debugging/running you app), and all of your code from day one should be stored in your personal **GitHub** repository.

- Do not store your **MongoDB Atlas** credentials in your code committed to **GitHub** – you will get **0 marks** for your deployment, if this is done. You can also get 0 marks for your deployment, if your app **is not running on Heroku** or if your app **does not connect to your cloud-provisioned MongoDB Atlas database**.

- You will get **0 marks** for the report, if some of the required items (such as full student name, student number, project title, etc.) are not present on the cover sheet. **Also note, that GitPod.io (Snapshot or a Running Workspace) link does not equal a GitHub link.**

- You **must answer all of these 3 questions** in your report or it will get **0 marks** for it.

- All of the code and the report will be checked for **plagiarism**, so make sure **everything is referenced** through the use of comments. **The code that is not referenced properly will not be marked and will be reported for plagiarism**.

- **Outside of special circumstances, any code submitted to GitHub after the deadline – will not be marked**.

- Any other **technical details** that might come up during the course of this project should be clarified directly with your lecturer.

- A **grading rubric** for each of the grade items is outlined below. Please use this as a guide for the development of this project.

	API (30 marks)	**JSON consumption, Add and Delete (30 marks)**	**Deployment (15 marks)**	**GitHub history (15 marks)**	**Report (10 Marks)**
<40	API is poorly structured with insufficient levels of complexity and does not provide CRUD	Data fails to pass from server to client. No Add or Delete functionality.	The app/database is not deployed properly and throws errors.	No sufficient GitHub commit history	Badly formatted and referenced report addressing only a few of the elements required, demonstrating no analyses and consideration of technologies
40 – 49	API returns data with some errors, data is somewhat structured with minimal complexity, some of CRUD is present	Data just passes from server to client-side, but not adequately displayed. Some form of Add or Delete functionality is present.	The app/database is somewhat deployed, but random errors occur	Some evidence of commits to GitHub	Poorly formatted and referenced report addressing only a few of the elements required, demonstrating little analyses and consideration of technologies
50 – 59	API returns data with no errors, which is adequately structured with minimal complexity, most of CRUD is present	Data passes from server to client-side and is displayed on the front-end. Add or Delete functionality is present, but works with errors	Both app and database are deployed and can talk to each other	Limited amount of commits to GitHub, the commits are not logical or reflect only minor changes	Well formatted and referenced report addressing most elements required, demonstrating adequate analyses and consideration of technologies
60 – 69	API returns data with no errors, data is well structured with several levels of complexity, full CRUD is present	Data passes from server to client-side and well-displayed on the front end. Both Add and Delete functionality is present.	Both app and database are deployed with the correct settings	An appropriate level of GitHub commits	Well formatted and referenced report addressing all elements required, demonstrating good analyses and consideration of technologies
70 +	API returns data with no errors, is excellently structured with many levels of complexity, full CRUD is functionality is well-designed and fully works.	Data just passes from server to client-side with an excellent way of rendering information on the front-end. Add and Delete functionality is well-implemented.	Both app and database are deployed with the correct settings and environment variables	A full and logical history of GitHub commits is present showcasing work from day 1	Excellently formatted and referenced report addressing all elements required, demonstrating excellent analyses and consideration of technologies