RMIT UNIVERSITY

Further Web Programming
COSC2758 (Semester 2, 2021)
Assignment 2

Assessment Type	To be attempted individually. **No group work is allowed**. Submit online via Canvas → Assignments → Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications and updates may be made via announcements / relevant discussion forums.
Due Date	Week 12, Sunday 17 October 2021, 11:59 pm *Melbourne time*
Mandatory demo via Microsoft Teams	Week 13, 18-22 October, 2021 (*schedule and a booking system will be released closer to the deadline*). **No demo = No Marks**
Marks	45

1. Overview (you must read this first)

In this assignment, you will develop a full-stack web application to complete the front-end prototype built from assignment 1. You are to use the following stacks:

Front-end:	ReactJS
Middle-layer:	Node.js & Express.js with Sequelize ORM
Backend database:	Cloud MySQL

You are not allowed to change technology stacks to suit your convenience, experience and/or knowledge. Using stacks other than the ones mentioned above will FETCH A ZERO for the whole assignment.

The tasks are divided into *four* parts: PA (Pass), CR (Credit), DI (Distinction) & HD (High Distinction).

> The DI & HD section tasks will require self-research, you will not get straight answers in the course material. While we are happy to assist you on those tasks, most of the work and research must be done by you. This is done on purpose to prepare for you future work and rigours of the IT industry.

If you find a specification open to interpretation, post a query identifying the specification in the corresponding discussion board for assignment 2. Software development in real life does not come with a definitive roadmap and flowcharts complete with instructions. More often than so, it is the job of the developer to clarify requirements from the client. For the purpose of this assignment and course, the lecturer is considered as the client.

All of us have been affected by the unfortunate COVID-19 scenario and its aftermath. It is often hard to concentrate and study; but as a student enrolled in this course, it is your responsibility to regularly attend lectorial, lab and consultation session(s).

- Bring your questions to online discussion board, consultation sessions
- Watch the online recordings on a regular basis if you cannot attend the live sessions.
- Do NOT start the work on assignment at the last minute.
- Do NOT ask for last minute extensions, these are often rejected. Extensions can only be granted for personal and medical reasons, provided you can supply some evidence.

2. Learning Outcomes

This assessment relates to all of the learning outcomes of the course which are:

- Demonstrate proficiency with a web application development framework;
- Implement a range of techniques and procedures for developing a small to medium-scale web application;
- Demonstrate knowledge of and utilise software engineering patterns in development;
- Design and manage the development life-cycle of a complete application.

3. Assessment details (READ THIS FIRST)

The senior student club committee has accepted the prototype of VC (Vibe Check) website and now recommends that full stack version of website be developed. The committee recommends inclusion of extra features and constraints for the VC website- a summary is presented here, you will find more details in **Section 4: Tasks**.

The extra features and constraints include-

3.1
The committee wants the developer to use a Cloud MySQL database for the backend purposes. All the user and any other appropriate data for the website must be stored in the backend Cloud MySQL database (**rmit.australiaeast.cloudapp.azure.com**). You have been emailed your username, password and access link to **phpMyAdmin**- if you do not have these details, email Matthew Bolger at matthew.bolger2@rmit.edu.au

3.2
Two users cannot have the same username. Password must be stored in a hashed format in the database. **Please do not use MD5 as it is no longer considered secure by industry professionals**.

3.3
The full stack version of VC must be a multi-user website where users can follow and unfollow one or more other users. Posting form must have an option for uploading photos.

3.4
User can now submit interactions on posts such as: like and dislike.

3.5
All user inputs must be sanitised to avoid XSS attacks.

3.6
VC website will have a separate *admin portal* for moderating posts and generating user statistics. Admin does not need to login to access the portal ie no admin accounts exist. Admin dashboard is accessible via a separate Url to VC.

3.7
Codebase must be well-commented. Absense of sensible code commenting will lead to **penalty of marks**.

3.8
GitHub should be used througout the development lifecyle. The repository must be a part of **rmit-fwp-s2-2021** organisation account. You have been emailed an invite to join this organisation during week 1 of the semester; if you did not accept the invite please email Matthew Bolger at matthew.bolger2@rmit.edu.au

3.9
The website must be fully styled and look professional. The content must make sense ie. use of *lorem ipsum is not allowed.*

3.10
The digital assets (images, icons, audio & video) must be outscourced from free websites. You should not steal someone else's assets to enhance the look and feel of your website. High-quality & free assets can be obtained from:

https://unsplash.com/	(Images)
https://uifaces.co/	(Avatars)
https://fonts.google.com/icons?selected=Material+Icons:home	(Icons)
https://www.flaticon.com/	(Icons)

In order to proceed to higher parts, you must complete all of the specifications in the lower part, you must not cherry pick specifications from various parts. As an example, complete all of the specifications in PA part before proceeding to CR part and so on. Please proceed to next page for the tasks. →

4. Tasks

Create a **full-stack** VC website with **– ReactJS, Node.js, Express.js (with Sequelize ORM)** & **Cloud MySQL database. Here is the architecture diagram for the website:**

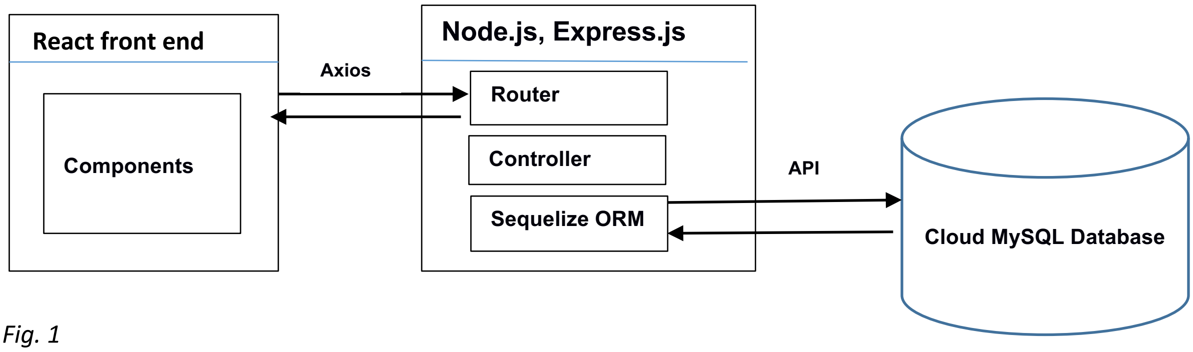

Fig. 1

React front-end app talks to API defined in Node + Express layer via Axios library. API created by you in Node + Express layer (*middle layer*) communicates with the backend database.

Create separate projects – *one for React frontend app* & *one for Node + Express + Sequelize middle layer*

The tasks are shown below:

PA part [23 marks]

a. (5 marks) **Database schema**

Create an ER (Entity-Relationship) diagram that will represent the database schema for the website. The diagram should display the tables with the fields, keys, constraints and relationship(s) between the tables. Think of these points- *How many tables do I need? Which fields do I need in these tables? What data-types should these fields use? What kind of relationships exist among these tables? Is the database normalized? (ie avoid duplicated data, do not use too few or too many tables).*

You can submit the diagram as an image or PDF file.

Create model files that represents the above tables, keys and constraints using Sequelize in the Node.js + Express.js (middle layer) project.

b. (10 marks) **Sign-up page**

Implement the sign-up page from assignment 1. This time the user details are stored in the Cloud MySQL database. The API in the middle layer should handle all the database operations. All the user (form) input must be sanitised and validated on the React end.

c. (3 marks) **Sign-in page**

Implement the sign-in page with API handling authentication to verify the username and password. All the user input must be santised and validated.

Introduce a **logout** link for the user.

Note: The API itself does not require authentication / authorisation; meaning all API endpoints can be accessed without access / bearer tokens.

d. (5 marks) **Profile and Profile management features**

Implement the profile and profile management feautures from assignment 1. The details of user profile must be fetched and modified via the API.

CR part [8 marks]

e. (8 marks) **Posting feature**

Implement the posting feature from assignment 1 with the following additional requirements-

i. User post must be saved in the database.
ii. User may reply to their own post or to another user's post.
iii. The maximum length of a post is 600 characters.
iv. User should be able to upload an image alongside a post.

Note: For the following (DI and HD sections), you will need to do some independent research.

DI part [4 marks]

f. (2 marks) **Follow** & **Unfollow**

Add a new feature to VC- a user should be able to follow and unfollow another user. The follow and unfollow logic must be written in the API and appropriate details stored in the database.

For this part, a user should able to see a list of users they can follow. For each user you will need to keep track of whom they are following.

The user should be able to view a list of the users they are following in the React app with an option of unfollowing them.

The design of the user interface is left for you.

g. (2 marks) **Like** & **Dislike**

A core feature of any social media platform is the ability for users to interact with shared content. For the posts that are created on the VC website, add the options to like and dislike individual posts. The logic of likes and dislikes may be written in the API and the appropriate details stored in the database.

Each post or comment must display the total number of likes and dislikes.

The design of the user interface is left for you.

Please proceed to next page for the HD tasks->

HD part [10 marks]

Create separate project(s) for this section. Please note the requirements for this part-

You must use functional components only.

Use of Redux, MobX or any similary library is not allowed for state management. You must use React Hooks only.

For the data fetching part, you must use GraphQL- ie. use of **REST API** is **not allowed.**

Your task is to create a React Admin Dashboard with the following features-

h. (2 marks) The look and appeal of the admin dashboard must be different to the VC website. Spend some time thinking of an appropriate user interface.

   The admin dashboard must consist of multiple components with a nested hierarchy. The hierarchy must be at least **3-4** levels deep (as an example App > Dashboard > Table > Table User). This is to make sure that you understand and correctly implement hooks for the state management.

   You may use a third party library to enhance the user interface of the dashboard. You will be asked to justify your choice of third party library during the demo.

i. (4 marks) Admin should be able to perform the following (**database fetch via GraphQL**)-

   i. Modify user details: *admin should be able to modify their basic profile information*

   ii. Delete posts(s) if deemed inappropriate: *deleting a post must show some corresponding message on the user page in VC website such as- [**** This post has been deleted by the admin ***]*

   iii. Block and unblock a user account: *blocking a user will not allow a user to login until the admin unblocks the account*

   iv. Delete a user completely: *this will purge the user data from the database*

j. (4 marks) Admin should also be able to generate pictorial representations of user statistics such as ((**database fetch via GraphQL**):

   i. Number of users using VC per day.
   ii. Following user statistics for each user.
   iii. Most popular post of all the posts made on the VC website.
   iv. Time spent on VC on a dialy basis by each user.

Submission instructions →

5. Submission & Mandatory Demo

- Zip all website files **EXCEPT NODE MODULES FILES** and submit single zipped archive with .zip extension via Canvas submission link for this assignment.

- You must demo your assignment in week 13 via Microsoft Teams- a schedule and a booking link will be published closer to the deadline. **No demo will lead to no marks**. **You must submit the assignment prior to the demo**.

After the due date, you will have 5 business days to submit your assignment as a late submission. Late submissions will incur a penalty of 10% per day. After these five days, Canvas will be closed, and you will lose ALL the assignment marks.

Assessment declaration:

When you submit work electronically, you agree to the assessment declaration:

https://www.rmit.edu.au/students/student-essentials/assessment-and-results/how-to-submit-your-assessments

6. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Contract cheating- paying someone to do your work
- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students
- Posting assignment tasks on technical forums (*reddit, stack exchange, etc.*) and asking for solution(s)

For further information on our policies and procedures, please refer to:

https://www.rmit.edu.au/students/student-essentials/assessment-and-results/academic-integrity

7. Marking Guidelines

The marks allocated have been added to each of the tasks. **Please read rubrics for details**.