RMIT UNIVERSITY

Further Web Programming
COSC2938 (Semester 2, 2023)
Assignment 2

Assessment Type	To be attempted individually OR in a group of 2. Submit online via Canvas→Assignments→Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.
Due Date	Week 12, Sunday 15 October 2023, 11:59 pm *Melbourne time*
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:

✓

Frontend:	ReactJS
Middle layer:	Node.js & Express.js with Sequelize ORM
Backend database:	MySQL

✗

- You are **not** allowed to change technology stacks to suit your convenience and/or knowledge.
- Using stacks other than the ones mentioned above will **FETCH A ZERO** for the whole assignment.
- Use of Object-Oriented React 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 this assignment and course, the lecturer is considered as the client.

Tips for assignment success:

- 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.

You are strongly recommended to work in a group as this is a longer assessment. If you want to join a group or change your assessment 1 group, please email lecturer at your earliest.

2. Learning Outcomes

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

CLO2: Demonstrate proficiency with a web application development framework

CLO3: Implement a range of techniques and procedures for developing a small to medium-scale web application

CLO4: Demonstrate knowledge of and utilise software engineering patterns in development

CLO5: Design and manage the development life cycle of a complete application.

3. Assessment details

The senior executive committee has accepted the prototype of the Loop Cinemas website and now recommends that full stack version of website be developed. The committee recommends inclusion of extra features and constraints for the Loop 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.

3.1. 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**. We recommend using bcrypt but any suitable hashing algorithm will do.

3.2. The full stack version of Loop Cinemas must be a multi-user website where users can login and post their reviews, with the reviews saved in the database.

3.3. Website visitors can now reserve a ticket to a particular movie session. More details are further along in this document.

3.4. Loop website will have a separate *admin portal* for moderating reviews and managing movies. Admin dashboard is accessible via a separate URL to Loop website.

3.5. Codebase must be well-commented.

3.6. GitHub should be used throughout the development lifecycle. The repository must be a part of **rmit-fwp-s2-23** 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 lecturer at matt.hayward@rmit.edu.au

3.7. The website must be fully styled and look professional. The content must make sense i.e., use of *lorem ipsum is not allowed.*

3.8. The digital assets (images, icons, audio & video) must be outsourced 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)

To proceed to higher parts, you must complete all the specifications in the lower part, you must not cherry pick specifications from various parts. As an example, complete all 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** Loop website with **–** **ReactJS**, **Node.js, Express.js (with Sequelize ORM)** & **Cloud MySQL database.** **Here is the architecture diagram for the website:**

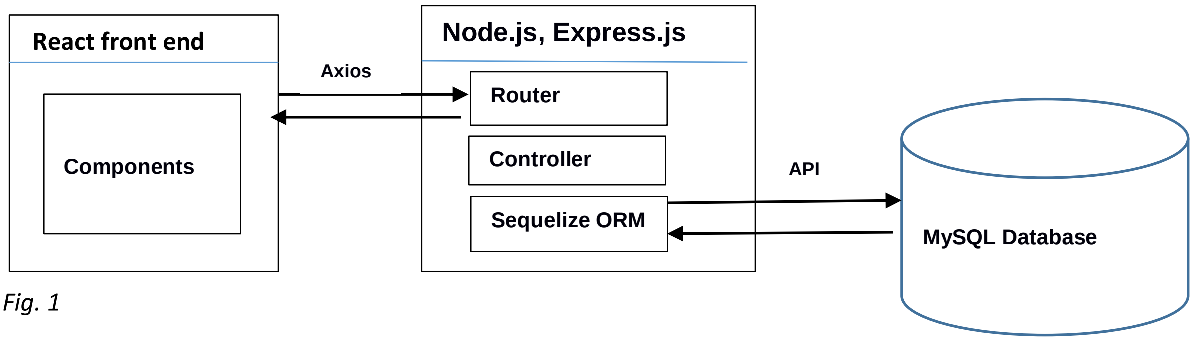

Fig. 1

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. (6 marks) **Database schema model files**

Create an ER (Entity-Relationship) diagram that will represent the database schema for the website. *This is for your draft work*, there is no need to submit the diagram. It 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 datatypes should these fields use? What kind of relationships exist among these tables? Is the database normalized? (i.e., avoid duplicated data, do not use too few or too many tables)*.

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

b. (3 marks) **Sign-up page**

Implement the sign-up page from assignment 1. This time the user details are stored in the MySQL database. The API in the middle layer should handle all the database operations. All the form input validations must be handled 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 form input validations must be handled on the React end. Upon logging, the user must see a welcome message in the format of *Welcome* username. Introduce a **logout** link for a logged in user.

d. (6 marks) **Profile** & **Profile management features**

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

e. (5 marks) **Unit tests**

Add five meaningful (*significant and contextual to the tasks b, c & d*) unit tests. You must explain the purpose of each of the test as comment entries in the test file.

CR part [8 marks]

f. (6 marks) **Reviews feature**

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

1) User post must be saved in the MySQL database via the API.
2) The maximum length of a review increased to 600 characters and it can now include formatted text.

(2 marks) Add a contextual unit test for this specification.

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

DI part [4 marks]

g. (2) **Ticket reservation**

A key feature of a cinema website is the ability to reserve a ticket for a particular session of a movie. The general user flow is that a logged in user can select a movie, select the session time and then select the number of seats they wish to reserve.

The details of their reservation should be stored in your MySQL database.

The user should then be able to see a simple list of reservations in their account.

For the purpose of this assignment payment for the reservation is not required.

The design of the user interface is left for you.

h. (2) **Ticket limits**

Cinema sessions normally have a limited number of seats available. Loop Cinemas would like to enforce limits around the total number of seats able to be sold. For the purposes of this assignment we will assume that Loop Cinemas has a single screen with 10 seats. So a maximum of 10 tickets should be able to be reserved for any session.

Display the number of available tickets next to each session, and disable the link to reserve a ticket if the session is sold out.

POSTGRADUATE STUDENTS: In your README, you need to explain the following:

- If a user wishes to cancel their reservation, how will your solution change to include such a scenario for the DI part? How will your futureproof your solution to include such a requirement?

Please proceed to next page for the HD tasks →

HD part [10 marks]

Please note the requirements for this part:

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

Requirement # 1: Create separate project(s) for this section.

Requirement # 2: The look and appeal of the admin dashboard must be different to the Loop Cinemas website. Spend some time thinking of an appropriate user interface.

Requirement # 3: 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 > Component 1 > Component 2). This is to make sure that you understand and correctly implement hooks for the state management.

Requirement # 4: You must utilise **useReducer** and **useContext** hooks for the state management for this part.

Requirement # 5: **For the data fetching part, you must use GraphQL- i.e.,** use of **REST API** is **not allowed**.

i. (2 marks) Admin should be able to perform the following (**database fetch via GraphQL**)-

   1) Delete reviews(s) if deemed inappropriate (you will need to explain during demo why a review is considered inappropriate?)

      Deleting a review must show some corresponding message on the reviews page on the Loop Website such as- *[**** This review has been deleted by the admin ***]*

   2) Block and unblock a user: *blocking a user will not allow a user to leave a review until the admin unblocks the account*

j. (3 marks) Admin should also be able to generate the following data analytics visually (**database fetch via GraphQL**):

   1) Number of ticket reservations per day.
   2) Average number of reviews per movie
   3) Number of views of each movie page (if your movies are displayed in a modal when you click on them, that is okay, count this as a page view).

   Each of the metric must be graphically depicted. No marks will be offered for a pure tabular representation.

   Note: do you understand the exact nature of these metrics? You need to do some reading first and find out how social media websites/apps use these metrics. Then define your understanding of the metric in the corresponding code file(s).

k. (5 marks) Add a feature to the admin panel that lets admins create, edit and update movie information. This information should be stored in your database and retrieved from the database when displayed on the main Loop Cinemas website.

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.

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**.