ISYS1124/1126 ASSIGNMENT # 2

LARAVEL & NODE.JS

Marks allocated:	**20 (*to be marked out of 60 during a face2face demo session*)**
Deadline:	**Wednesday 12.10.2016 (11:59 PM)**
Demo:	**Thursday/13.10.2016 and Friday /14.10.2016 (NO DEMO == NO MARKS)**
Demo via:	**Your laptop**
Submit via:	**Blackboard *as a single zipped archive of your GitHub repository*.**
Work mode:	In a group of **2-5** (no individual submissions will be accepted)

2.1 Overview

In this assignment, you will develop web database applications using **Laravel (***Part A*) and **Node.js (***Part B*). You may use **Sqlite3** or **MySQL** as the back-end database for Laravel part. You are to use **MongoDB** as the back-end database for the Node.js part.

- **Use of any other frameworks other than Laravel in part A and Node.js in part B of the assignment will fetch you ZERO.**
- ***Before starting the assignment,*** *make sure that you have read the section on '**Laravel project issues**' on **Software** page of Blackboard**. Also make sure that you have attempted lab sheets 5, 6** and **7.***
- **This is a longer assignment, start ahead of time.**

2.2 Scenario

A senior member of the ITS was very impressed by your design and implementation of their ticket submission system. They recommended you to the chairman of a successful cinema group known as Mavericks Inc. (MI) MI's current website is old and was designed in 2005. It is in a desperate need of a facelift and a complete refurbishment. The current website uses PHP and badly written PHP-PDO code. The website apart from its obvious design issues has various architectural flaws. After a meeting with the senior members of MI, it was decided that a completely new website must replace the antiquated website.

MI's new website will have two separate parts- ticket booking (TB)and the movie blogs (MB). It was agreed upon to keep the implementation of these separate due to- technical, financial and other strategic reasons. It was finally decided that- TB would be implemented using **Laravel 5.2.*** framework and MB part would be implemented using **Node.js**.

2.3 Tasks:

Part A (30 marks): *Using Laravel 5.2.* and Sqlite3 or MySQL*

All of the relevant data must be stored in the backend database using **Eloquent ORM.** Your database should be normalised and reflective of your model and migration classes**. Use of embedded SQL is strictly prohibited**. MI's website features-

1. (2 marks) A well-defined layout page for all of the blade view pages. Layout should define the basic sections of view pages- header, footer, navigation bar and content areas.

2. (2 marks) Home page should show a list of featured movies in a carousel or an image gallery.

3. (2 marks) Movies category page should divide the movies into two categories – Now Showing and Coming Soon.

4. (4 marks) Search page should offer both – search by cinema and search by movie options. You can assume that MI has cinemas in various locations. You can assume the number of sessions.

5. (3 marks) Login page must have a link to separate Registration page (a user registration form where a user can register before purchasing the tickets).
    a. Add appropriate **Laravel server-side validation** to Login and Registration forms.
    b. You must also provide logout feature for a logged in user.
    c. User details must be stored in the database.

6. (2 marks) Customers can click on any movie and then enter the amount of tickets they want to buy as well as select the session for the movie before they can click on Add to cart button. Provide options for different types of tickets such as Adult / Children / Concession etc. On the cart page they should be able to change the amount of tickets they wish to purchase or remove any movie ticket. They should be able to purchase tickets for multiple movies. Seating chart is **not** required.

7. (4 marks) Cart page would lead the users to the payment page, a simple form asking for some details like name, address, suburb, postcode, mobile number and the credit card details
    a. Upon the form submission, there needs to be a **Laravel server side validation** for all the details filled by the customer.
    b. You may use **jQuery** for credit-card field validation and expiry date check
    c. Booking details must be stored in the database.
    d. Summary page to display a booking success message along with booking details

8. (4 marks) A logged in user should be able to see an extra link called- **My Account** section/page where
    a. they should be able to see the details of their upcoming booked sessions.
    b. A CRUD feature where they should be able to add, edit (*reason to add a 'Coming Soon' movie to a* wish list) and delete the wish list items

9. (3 marks) Create **three** unit tests. Choose the context of your tests. Think where can these tests be applied? Do not write scanty tests that are not useful and have no significance.

10. (4 marks) Add **two** security features to the user website. Examples: sanitizing form inputs is one, think of another one.

Part B (20 marks): *Using Node.js and MongoDB*

Note: MI wants this part for testing purposes-only. This application is for future release only. As a result, they want the database separate from Part A. Normal users from Part A will not be testing this part. This part will be tested by MI's admin staff members.

11. Create a blog website application using Node.js that has the following features-

    a. (4 marks) Home page displays movie information (name/title, rating, description, actors, any other relevant information) fetched via a remote web service REST API.

    b. (4 marks) A provision to register users that will allow them to add comments on the movies or write blogs

    c. (4 marks) Logged in users may comment to a blog of another person. **Moderation by admin is not required** for this part.

    d. (4 marks) All the relevant data such as user details and comments must be stored in a **MongoDB database**.

    e. (4 marks) Use of Bootstrap or any 3rd party plugin/library to create a professional user interface for this website.

Part C (10 marks): *General housekeeping (Code elegance and documentation)*

12. (6 marks = 1 + 1 + 2+2) General code elegance, directory structure of projects, elegant use of MVC and well-documented list of external resources used in Parts A and B.

13. (4 marks) GitHub repository history**, with each member having their own account and actively contributing**.

BONUS TASKS

This is extra and not a part of the assignment tasks. Attempt this part only if you have the time and inclination.

No help will be provided for these tasks. You need to do your own research and complete these parts.

You can get extra 5 marks added to your examination score (out of 100) on completion of the following tasks-

14 (2 marks) Add an admin feature to Part A. Admin will add the movie and tickets information. This will require a separate Admin page/dashboard and some more tables in the database.

15 (3 marks) Add another functional login option to Part A namely a social media login (Facebook or Twitter or Facebook).

2.4 What and how to submit?

a) Zip all of the files **from your github repository (*this avoids the bulky plugin/vendor folders in your project directory)*** and one of the group members to submit a single zipped archive (**assignment1.zip**) via your group link in Blackboard

b) Make sure you attend the demo on either of 13.10.2016 (Thursday) OR 14.10.2016 (Friday).

Late submission and Penalty

a) A penalty of 10% per day of the total marks will apply for each day late, including both weekend and weekdays.

b) After five days, you will receive a zero for the whole assignment.

c) Extension requests should only be emailed to the lecturer (**shekhar.kalra@rmit.edu.au**)

d) No demo = No marks

Plagiarism

Blackboard contains an in-built tool known as **turnitin**. It compares all of the submitted assignments for the whole batch and then compares the text against the resources in the internet. It flags a similarity report when it finds huge chunks of copied text.

All assignments will be checked with plagiarism-detection software; any student found to have plagiarised would be subject to disciplinary action. Plagiarism includes

- submitting code that is not your own or submitting text that is not your own
- allowing others to copy your work via email, printouts, social media etc.
- posting assignment questions (in full or partial) on external technical forums
- copying work from/of previous semester students

A disciplinary action can lead to

- a meeting with the Head of School
- a score of zero for the assignment
- a permanent record of copying in your personal university records and/or
- expulsion from the university, in some severe cases

All plagiarism will be penalised. There are no exceptions and no excuses. You have been warned.