

Western Engineering

Project Title: Western Online Timetable Application

A PROJECT REPORT

by:

Student Name	Student Number
Jiaxin Zhao	251105013

submitted for the requirements of the degree

of

MASTER OF ENGINEERING (M.Eng)

Project Supervisor: Dr.Jagath Samarabandu

Department of Electrical and Computer Engineering
ECE – 9000

University of Western Ontario
1151 Richmond Street, London, Ontario

August 2020

Table of Contents

Table of Contents.....	2
List of Figures.....	3
List of Tables.....	4
Abstract.....	5
Acknowledgement.....	6
Project Outline.....	7
1.Project Overview.....	7
2.Purpose.....	7
Development Environment & Technologies.....	9
1.Tools used.....	9
2.Techniques used.....	9
3.Node.js.....	9
4.Node.js running platform.....	10
5. API and RestFul API.....	10
6. Angular.....	11
7. Angular Running Environment.....	12
Project Organization.....	13
1.Server side implementation.....	13
2.Client side implementarion.....	22
Testing and Evaluation.....	29
1.User interface testing.....	29
2.Interface Mechanisms Testing.....	29
Installation guide.....	38
Scope of Future Work.....	38
Conclusion.....	40
References.....	41

List of Figures

Fig.1. Transmit process of Node and Express.....	14
Fig.2. Index page	29
Fig.3. Search Panel.....	30
Fig.4. Search panel based on selected fields(a).....	31
Fig.5. Result panel based on selected fields(a).....	31
Fig.6. Search panel based on selected fields(b).....	32
Fig.7. Result panel based on selected fields(b).....	32
Fig.8. Search panel based on registration	33
Fig.9. Result panel based on registration	33
Fig.10. Search panel based on course number	34
Fig.11. Result panel based on course number	34
Fig.12. Search panel based on all subjects	35
Fig.13. Result panel based on all subjects	35
Fig.14. Result panel and pagination.....	36
Fig.15. Filter by class name.....	36
Fig.16. Save by local storage.....	37

List of Tables

Table I HTTP Methods and description.....	11
Table II Explanation of fields in getTimeTableSchema API	17

Abstract

A University online timetable application is a temporal arrangement of a set of lectures, times and classrooms. Timetable can be defined as the optimization of giving activities, actions or events to staff, teachers, students in university. It is an important communication channel between school administrative department and other faculty members. However, with the advent of the intelligent age, displaying timetable information on the website has become a mainstream trend. All schools (elementary school, middle school, university) in Canada prefer to put their timetable information on the official website or app for students and teachers to check. Therefore, developing an online timetable application for university is essential and it is a great chance for our engineering graduate student to consolidate the knowledge we have learned in class and lay a solid foundation for our future career path.

The objective of this project is to find relevant timetable information by selecting keywords such as subject, component, campus, days, start time, end time, course suffix and or even search by entering course number. What we need to complete is the design and mock of the back-end REST APIs and the construction the web page using popular front-end framework. Main technologies we are using is Node.js for back-end and Angular/React for front-end. All the course information is stored in JSON format which is provided by WTS department. If this project finished successfully, each teacher and student in Western University can search timetable and relevant courses for them conveniently online. It is an good interface to let students compare courses with other schools as well.

Key words: online timetable application, Node.js, Angular, convenience

Acknowledgement

With a deep sense of gratitude, I would like to thank Dr. Jagath Samarabandu for his kind support and encouragement throughout the course of this project. There were times, that we were facing challenges due to the spread of covid-19 all around the world. During the process, I have tried to give up countless times, but the professor and teammates always encouraged me to proceed. Dr.Samarabandu offered me valuable guidance even if school was closed, he taught me how to overcome these challenges and also mentored me about the various software development principles and approaches that had to be undertaken to ensure that the software was engineered was robust and possessed a seamless interface. It was truly a very enriching experience to work under Dr.Samarabandu, especially in this challenging year.

Last but not the least, I would also like to thank my team, Bharti Upadhyay, Utkarsha Bakshi, Yiyun Yang who have always been my pillar of support and strength.

Project Outline

1. Project Overview

There is no doubt that online timetable application is very important to teachers and students in university nowadays. University administrative department provides course information, students and professors attend classes on time according to the timetable information including subject, class name, catalog number, class description, section, component, class number, days, start time, end time, location, instructor, status, campus etc.. At the same time, students are able to query relevant course information based on specific conditions such as subject, component, day of class, course suffix, starting time, ending time, campus, course number and whether to show courses for registration etc.. Online timetable application is aimed at bringing great study experience to students in Western University, which allows students to have a more comprehensive understanding of the relevant courses of their major and the professors who will teach them as well.

Although the website page and functions of the current western online timetable application are basically complete, it faces many problems, such as the user interface is too concise, the data storage function is incomplete, the front-end framework is out-dated, etc.. Therefore, as a team in Electronic Computer Engineering Department, we have the obligations to improve Western online timetable application. What we should do is to connect WTS to gain the course data we need, complete the back-end REST APIs design, mock server side of implementation, and use mainstream front-end frameworks to finish this online timetable application.

2. Purpose

Our task is to define back-end REST APIs in consultation with WTS staff in Western University who would provide us with all undergraduate course information. Then we can develop back-end functionality using node.js and a few front-end applications using Angular and React should be created. Finally, we can find the corresponding course information according to the input keywords.

• Search Information

(1) Subject

(2) Course Number

- (3) Course Suffix
- (4) Component
- (5) Campus
- (6) Starting Time
- (7) Ending Time
- (8) Day of Class
- (9) Whether the course can be registered

• **Course information**

- (1) Subject
- (2) Catalog Number
- (3) Class Name
- (4) Class Description
- (5) Section
- (6) Component
- (7) Class Number
- (8) Start Time
- (9) End Time
- (10) Day of Class
- (11) Location
- (12) Instructors
- (13) Enrollment Status
- (14) Requisites and Constraints
- (15) Campus

Development Environment & Technologies

In order to better improve the functions of online timetable application of Western University, we have adopted the technology of separating front and back ends to make the entire development process more standardized and efficient. The main technologies used in the back-end are Node.js, Restful API to accomplish inquiry and post of course selection while Angular is used to implement the front-end because it provides the capability to create single page application in a very clean and maintainable way.

1.Tools used:

- (1).VS Code
- (2).Github
- (3).Postman

2.Techniques used:

- (1).Front-end: Angular, Typescript
- (2).Back-end: Node.js, REST API
- (3).Data: JSON file

3.Node.js

Node.js uses JavaScript on the server and it is building on Chrome's V8 Javascript engine which is light and efficient. Node.js package ecosystem npm is the largest ecosystem of open source libraries in the world.

Node.js is a free, open source server environment which runs on various platforms (Windows, Linux, Unix, Mac OS X, etc..) . Compared with other back-end language like PHP or Java, it can eliminate the waiting time and simply continues with the next request. The most advantage of Node.js is that it is a single-threaded, non-blocking, asynchronously programming.

Overall, Node.js has a platform which allows developers to run Javascript on a computer or server, it can read, delete and update files. What's more, it can easily communicate with a database. Therefore, it is a nice choice for real-time services like chats, music app, or the timetable application. The inner working of Node.js includes V8 engine, modules, event emitter and the file system. And using node.js to create a web server is has become a trend.

4.Node.js running platform

Download Node.js for Windows(x64) with node version v12.14.1 and choose VS Code as code editor which is a powerful editor. Then create western-timetable directory for western online timetable application and navigate into it. Afterwards use **npm init** command to create a package.json file for application which prompts developers to fill in a number of things such as name, version, github and the name of the initial entry point file (server.js). Finally install **express, body-parser, cors, fs, lodash** in western-timetable directory using **npm install** command which could be saved in the dependencies section of package.json file. The following shows the content of package.json file.

```
1. {
2.   "name": "web-final",
3.   "version": "1.0.0",
4.   "description": "",
5.   "main": "server.js",
6.   "scripts": {
7.     "test": "echo \"Error: no test specified\" && exit 1",
8.     "start": "node server.js"
9.   },
10.  "dependencies": {
11.    "body-parser": "^1.19.0",
12.    "cors": "^2.8.5",
13.    "dotenv": "^8.2.0",
14.    "express": "^4.17.1",
15.    "fs": "0.0.1-security",
16.    "lodash": "^4.17.15"
17.  },
18.  "author": "",
19.  "license": "ISC"
20. }
```

5.API and RestFul API

Most of applications which are popular today consist of client side and server side. Client side is the front end and interface of application while the server side is where user store and operate the data. HTTP protocol is used to communicate between client side and server side. So in the server side developers must expose a bunch of services that are accessible via HTTP protocol and the client can then directly call the services by sending HTTP request.

API implementation is widely used in server side. API is short for Application Program Interface and it is easy to see that APIs are everywhere in software development. It is a contract provided by one piece of software to another and the structured of APIs are request and response.

REST is indicating Representational State Transfer, it is an architecture style for designing network applications which is always relying on a stateless, client-server protocol, always HTTP. Besides, it treats server objects as resources that can be created or destroyed and it can be used by virtually any programming language. Currently, HTTP methods are the core of RESTful APIs and HTTP protocols can provide support to create, read, update and delete data.

Table I HTTP Methods and description

HTTP Method	Description	Endpoint Example
GET	Retrieve data from a specified resource	/api/customers /api/customers/1
POST	Submit data to be processed to a specified resource	/api/customers
PUT	Update a specified resource	/api/customers/1
DELETE	Delete a specified resource	/api/customers/1

6.Angular

Traditional web applications model are using AJAX to send request and response with HTML/JSON format which is time-consuming and disrupts user experience. But today single page application is introduced which is much more responsive, dynamic to the user and saving time. There are many frameworks allow us to create a single page application such as Angular, React and Vue but Angular is the most popular one in North America.

Angular was developed by Google and it was first released in 14 September 2016. The current release version is 10.0.5. Angular("Angular 2+") is an open-source web application framework led by Angular Team at Google. Originally, the rewrite of AngularJS was called "Angular 2" but this led to confusion among developers. Angular has MVC based architecture for application development and we can use it with npm package.

The core files in an Angular Application is src folder which contains important folders or files like environments, assets, index.html, main.js, app, modules and components. In environments developers can store their base url to connect with server side and assets are used to store public files such as pictures developers want to show on pages. Index.html file is entry point of our Angular application and `<app-root></app-root>` coming from app module is what to be shown on first page. Furthermore, the most significant module is app module which is also the root module. Every module should be included in app.module.ts file.

Component part includes app.component.html, app.component.css, app.component.ts, app.component.spec.ts. It is noted that .ts file can be viewed as a controller which can complete the logic function of Angular app.

7. Angular Running Environment

Prerequisites for building an Angular environment is that **node** and **npm** should be installed. After that opening command line for installing Angular environment. In order to create an Angular App, first is to install angular/cli using **npm install -g @angular/cli**, then create a new app by entering **ng new AngularApp** where user can choose **CSS/SCSS** as angular app style. After finishing installing, command **ng serve** can be typed to initiate a new angular app, if succeed, developers can open **localhost:4200** in the browser to see a new Angular app.

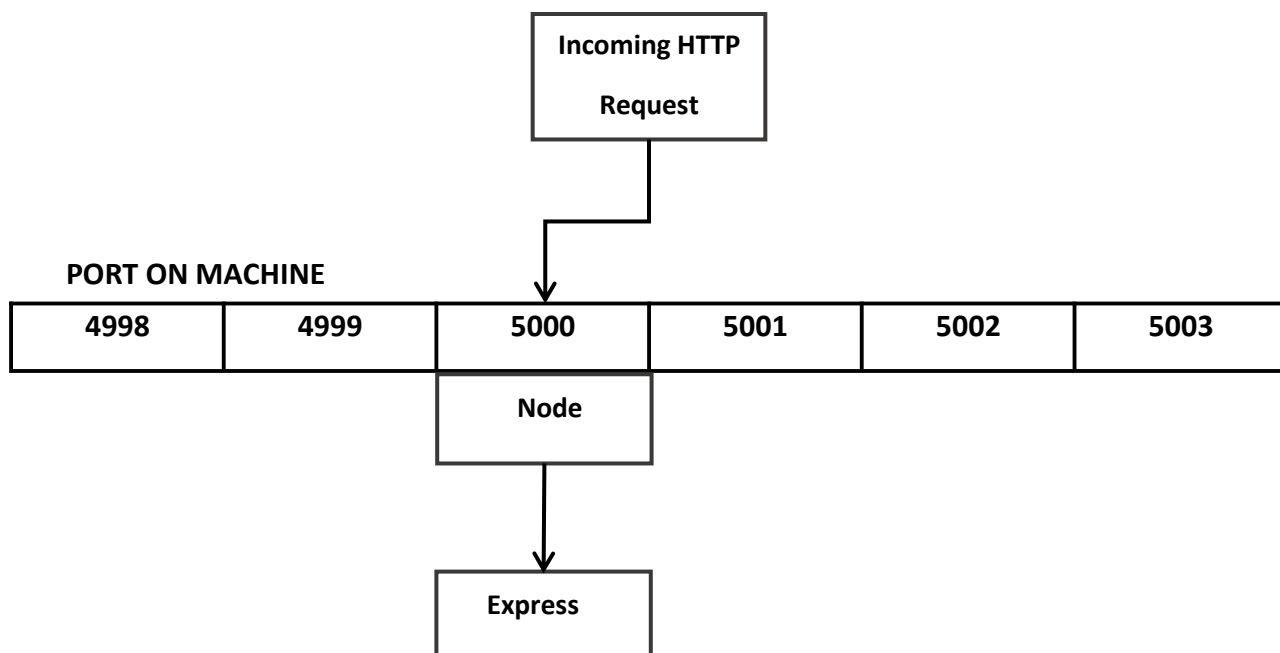
Project Organization

1. Server side Implementation

(1) Installing Express

Node is a Javascript runtime which used to execute code outside of the browser and express is a library that runs in the Node runtime. Express has helpers to make dealing with HTTP traffic easier, defines a routing table which is used to perform different actions based on HTTP Method and URL and allows to dynamic render HTML pages based on passing arguments to templates. To conclude, express is a minimal and flexible Node.js web application framework and aims to provide a robust set of features to develop web and mobile applications. We can install the express framework globally using **npm install express** which can be saved in **node modules** after installing successfully.

When running server side on our local machine, server is listening to HTTP traffic individual port that is like a door through which HTTP request can be allowed. Node.js is listening to some port and waiting for information to flow, and then throw it to express side of application. Node is important because it could handle all the http request, however, express is needed because it could make traffic a bit more easier. Express looks at the request and decides what chunk of code will handle or respond to the request.



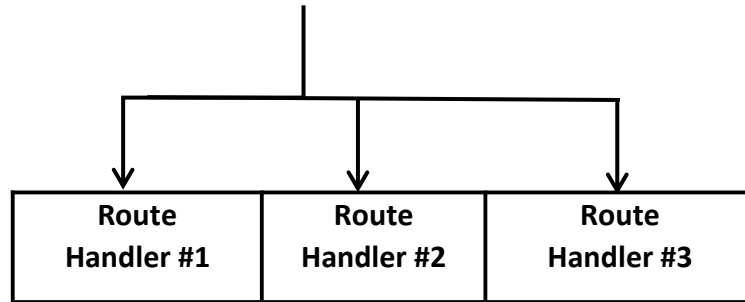


Fig.1. Transmit process of Node and Express

(2) Installing Body-parser

Node.js body-parser is a kind of middleware, aiming at parsing incoming request bodies in a middleware before handlers. It is available under the `req.body` property. It is extremely important when we are implementing POST method, but this cannot handle multipart bodies.

The main points of body-parser implementation are as follows:

- A.Processing different types of request bodies: such as text, json, urlencoded, etc., the format of the corresponding message body is different.
- B.Deal with different encodings: such as utf-8, gbk, etc..
- C.Handle different compression types: such as gzip, deflate, etc..
- D.Handling of other boundaries and exceptions

(3) Using CORS and Setting Headers

CORS is representing Cross-Origin Resource Sharing. The main reason to use CORS is that the front-end and back-end always have different origins. If we have created an API on sever side, the default would not be allowed to expose due to security reasons. So this problem can be solved by setting API as public for many people to use.

CORS can be installed by using command **npm install cors**, and then configure CORS in `app.js`.

```
1. res.setHeader(  
2.   'Access-Control-Allow-Origin', 'http://localhost:4200',  
3.   'Access-Control-Allow-Methods', "GET,POST,OPTIONS,DELETE,PUT",  
4.   'Content-Type', 'application/json',  
5.   'Access-Control-Allow-Headers', ' Origin, Content-Type, X-Auth-Token',  
6.   "HTTP/1.1 200 OK");
```

(4) Installing Fs

Given that the course information that our team obtained from the WTS department is **JSON** text mode, which is different from the previous common form of connecting MongoDB,

postgreSQL and other databases. Therefore, to better obtain data and perform addition, deletion, modification and query operations and complete the API document for all functions, we need to import the json file into app.js (Node.js platform).

Node.js file system module can allow us to work the file system on our local machine. File system module using **require('fs')** method and **fs.readFile()** method can be used to read files on our computer. It is common to read, create, update, delete and rename files in the file system module.

In our json directory there are two files. One is **timetableInfo.json** which contains all the key words(Component, Campus, CourseType, Designation, Subject, day, start_time, end_time, status) which are used to search for the course information. To make the format of the json file more standardized and easier to develop later, it has been assigned **id** and **description** to each obtained value, and let them exist in the form of key-value pairs.

```
1. {
2.   "Component": [
3.     {"id": "All", "desc": "All"},
4.     {"id": "LEC", "desc": "Lecture"},
5.     {"id": "TUT", "desc": "Tutorial"},
6.     {"id": "LAB", "desc": "Laboratory"} ],
7.   "Campus": [
8.     {"id": "Any", "desc": "Any"},
9.     {"id": "KC", "desc": "King's"},
10.    {"id": "HC", "desc": "Huron"},
11.    {"id": "UWO", "desc": "Main"},
12.    {"id": "BR", "desc": "Brescia"} ],
13.  "CourseType": [
14.    {"id": "All", "desc": "All"},
15.    {"id": "inPerson", "desc": "In Person"} ],
16.  "Designation": [
17.    {"id": "Any", "desc": "Any (any suffix)"},
18.    {"id": "Full", "desc": "Full Course ..."},
19.    ...
20.    {"id": "Z", "desc": "Essay Half-Course ..."} ],
21.  "Subject": [
22.    {"id": "ACTURSCI", "desc": "Actuarial Science"},
23.    {"id": "AMERICAN", "desc": "American Studies"},
24.    ...
25.    {"id": "WOMENST", "desc": "Women's Studies"},
26.    {"id": "WRITING", "desc": "Writing"}
27.  ]
28. }
```

The other one is **fullTimetable.json**, in this file, it contains all the course information of undergraduate in Western University. This is also the course result that should return at the end of the query.

```
1.  {
2.    "catalog_nbr": "1021B",
3.    "subject": "ACTURSCI",
4.    "className": "INTRO TO FINANCIAL SECURE SYS",
5.    "course_info": [
6.      {
7.        "class_nbr": 5538,
8.        "start_time": "8:30 AM",
9.        "descrlong": "",
10.       "end_time": "9:30 AM",
11.       "campus": "Main",
12.       "facility_ID": "PAB-106",
13.       "days": [
14.         "M",
15.         "W",
16.         "F"
17.       ],
18.       "instructors": [],
19.       "class_section": "001",
20.       "ssr_component": "LEC",
21.       "enrl_stat": "Not full",
22.       "descr": "RESTRICTED TO YR 1 STUDENTS."
23.     ]
24.   ],
25.   "catalog_description": "The nature and cause of financial security and insecurity; public, private and employer programs and products to reduce financial insecurity, including social security, individual insurance and annuities along with employee pensions and benefWTS.\n\nExtra Information: 3 lecture hours."
26. }
```

(5) Setting PORT

Traditionally, developers define listening port in app.js where to implement several APIs, but this time we choose a different way because most of the developers are not willing to let others know what port they are using, so they tend to use an environment variable ***process.env.PORT***. In this project, we have separated the port monitoring in another server.js file.

(6) Implementing API localhost:8080/timetable/getTimetableSchema

This API is used to get all fields generated in the front-end for users to choose such as subject, component, campus, designation, course type which returns an array of key-value pairs that provide descriptive labels for various ID values. Besides, there are 3 other fields (day/start_time/end_time) as a kind of list. What should be noted is that start_time (8:00 am - 7:00 pm), end_time (9:00 am - 10:00 pm) and Day of class (Monday - Friday) are fixed values which can be arranged as an array. Furthermore, the field status should be added and the default value is false. Some of the details of the fields related to this API are below:

Table II Explanation of fields in getTimeTableSchema API

Name	Description	Example
Component_id	Unique Id representing component	"LEC" for lecture
Component_value	Name of the component	Lecture, Tutorial, Lab
Campus_id	Unique Id representing campus	"HC" for Huron
Campus_value	Campus name where a lecture to be held	Main, Huron, King's
CourseType_id	Unique id for the types of courses offered	"inPerson" is for courseType "In Person"
CourseType_value	Name of the course types	In person, Distance studies
Designation_id	Unique Id representing each designation offered	"A" value maps to "First Term Half-Course (A)"
Designation_value	A string describing the designation type.	Second Term Half-Course (B), Essay Full Course (E)
Subject_id	Unique Id for the subjects offered	"ACTURSCI" for "Actural Science"
Subject_value	A string describing the subject	Theological Studies, Actural Science, Physiology
Start_time	When the course starts	8:00 am, 8:30 am, 9:00 am
End_time	When the course ends	9:00 pm, 9:30 pm, 10:00 pm
Day	The day of having classes	M, Tu, W, Th, F

Therefore, to complete the first `getTimetableSchema` API, we should read **timetableInfo.json** file by using **fs.readFileSync()** method and return the whole file as our result. When there is something wrong, return error.

(7) Implementing API `localhost:8080/timetable/getScheduleByAll`

The `getScheduleByAll` API is to get timetable details based on selected parameters. There are two kinds of selecting methods to implement. The first one is searching based on the following defaults parameters, the other way is searching by course number. At the same time, filtering courses with enrollment status of “Not full” if the user clicks the check box of showing only courses open for registration in the front-end.

```
1. defaults = {  
2.     subject: defSubjects,  
3.     start_time: timetableInfoJson.start_time,  
4.     end_time: timetableInfoJson.end_time,  
5.     campus: defCampus,  
6.     days: timetableInfoJson.day,  
7.     component: defComponent,  
8.     course_number:"",  
9.     status:"",  
10. }
```

- (a). Subject includes all the `subject_id` stored in the `timetableInfo.json`
- (b). `Start_time` is a list ["8:00 am", "8:30 am", "9:00 am"... "6:30 pm", "7:00 pm"]
- (c). `End_time` is a list ["9:00 am", "9:30 am", "10:00 am"... "9:00 pm", "9:30 pm", "10:00 pm"]
- (d). Campus includes all the campus value:["Any","King's","Huron","Main","Brescia"]
- (e). Days is a list ["M", "Tu", "W", "Th", "F"]
- (f). Component contains all the component id ["All","LEC","TUT","LAB"]
- (g). Course number is entered in front-end, and search course by course number
- (f). Status is checked in front-end to show whether to show courses only for registration

Firstly we need to justify the status of course the user choose, if the user click the checkbox of showing only courses open for registration in the front-end, filtering the course information with enrollment status “Not full” is important.

After determining the course information range, we use **filter()** method and **indexOf()** method to match between user select/input value and course information in previous selected course information range. The **indexOf()** method returns the index of the first occurrence of the

specified value in the string object that calls it, and searches from index. If the value is not found, -1 is returned.

Searching course information by course number can use different approaches and it should be noted that fuzzy interpolation is needed here, for instance, if we input 44 as the course number and the results should contain all course information with course number like 44, 444,4440, 4441 etc.. In our project, regular expressions and **match()** method are applied. And the **match()** method retrieves and returns the result of a string matching a regular expression.

In our project, if user selects all value in front-end, we treat it as null which is convenient for us to filter course information in the back-end.

Request:

```
1. {
2.   "subject": "ACTURSCI",
3.   "start_time": "12:30 pm",
4.   "end_time": "1:30 pm",
5.   "campus": "Main",
6.   "days": ["M", "W", "F"],
7.   "component": "LEC",
8.   "course_number": "",
9.   "status": "Not full"
10. }
```

Response:

```
1. {
2.   "length": 1,
3.   "result": [
4.     {
5.       "catalog_nbr": "2427B",
6.       "subject": "ACTURSCI",
7.       "className": "LONG TERM ACTUARIAL MATH I",
8.       "course_info": [
9.         {
10.          "class_nbr": 2663,
11.          "start_time": "12:30 PM",
12.          "desclong": "Prerequisite(s): A minimum mark of 60% in each of Act
uarial Science 2553A/B, either Calculus 2402A/B or Calculus 2502A/B, and Statistical Sc
iences 2857A/B. Restricted to students enrolled in any Actuarial Science module.",
13.          "end_time": "1:30 PM",
14.          "campus": "Main",
15.          "facility_ID": "MC-105B",
```

```

16.         "days": [
17.             "M",
18.             "W",
19.             "F"
20.         ],
21.         "instructors": [],
22.         "class_section": "001",
23.         "ssr_component": "LEC",
24.         "enrl_stat": "Not full",
25.         "descr": ""
26.     }
27. ],
28.     "catalog_description": "Models for the time until death, single life annuit
y and life insurance present values and their probability distributions; introduction t
o equivalence principle and premium calculations.\n\nExtra Information: 3 lecture hours,
1 tutorial hour."
29. }
30. ]
31. }

```

Request:

```

1.  {
2.    "subject": "",
3.    "start_time": "",
4.    "end_time": "",
5.    "days": ["M", "Tu", "W", "Th", "F"],
6.    "campus": "",
7.    "component": "",
8.    "course_number": "13",
9.    "status": ""
10. }

```

Response:

```

1.  {
2.    "length": 24,
3.    "result": [
4.      {
5.        "catalog_nbr": "1413",
6.        "subject": "APPLMATH",
7.        "className": "APP MATH FOR ENGRS I",
8.        "course_info": [
9.          {
10.           "class_nbr": 1422,
11.           "start_time": "1:30 PM",
12.           "desclong": "Prerequisite(s): One or more of Ontario Secondary Sch
ool MHF4U, MCV4U, or Mathematics 0110A/B."

```

```

13.         "end_time": "2:30 PM",
14.         "campus": "Main",
15.         "facility_ID": "TC-141",
16.         "days": [
17.             "M",
18.             "W",
19.             "F"
20.         ],
21.         "instructors": [],
22.         "class_section": "001",
23.         "ssr_component": "LEC",
24.         "enrl_stat": "Not full"
25.     }
26. ],
27.     "catalog_description": "Limits, continuity, differentiation of functions of
one variable with applications, extreme values, integration, the fundamental theorem o
f calculus, methods and applications of integration to areas, volumes and engineering a
pplications. Sequences and series, convergence, power series. Vector functions, partial
differential calculus, gradients, directional derivatives and applications.\n \nAntire
quisite(s): Calculus 1000A/B, Calculus 1301A/B, Calculus 1500A/B, Calculus 1501A/B, Mat
hematics 1225A/B, Mathematics 1230A/B.\n \nExtra Information: 3 lecture hours, 1 tutoria
l hour. Applied Mathematics 1413 is a suitable prerequisite for any course which lists
Calculus 1000A/B plus Calculus 1501A/B. Restricted to students in the Faculty of Engine
ering."
28. },
29. {
30.     "catalog_nbr": "1301A",
31.     "subject": "CALCULUS",
32.     "className": "CALCULUS II",
33.     "course_info": [
34.         {
35.             "class_nbr": 10122,
36.             "start_time": "1:30 PM",
37.             "desclong": "Prerequisite(s): A final mark of at least 55% in eith
er Calculus 1000A/B or Calculus 1500A/B.",
38.             "end_time": "3:30 PM",
39.             "campus": "Kings",
40.             "facility_ID": "KC-SA150",
41.             "days": [
42.                 "M",
43.                 "W"
44.             ],
45.             "instructors": [],
46.             "class_section": 570,
47.             "ssr_component": "LEC",
48.             "enrl_stat": "Not full",
49.             "descr": "PRIORITY TO STUDENTS REGISTERED AT AN AFFILIATED UNIVERSI
TY COLLEGE."
50.         }

```

```

51.         ],
52.         "catalog_description": "For students requiring the equivalent of a full course in calculus at a less rigorous level than Calculus 1501A/B. Integration by parts, partial fractions, integral tables, geometric series, harmonic series, Taylor series with applications, arc length of parametric and polar curves, first order linear and separable differential equations with applications.\r\n\r\nAntirequisite(s): Calculus 1501A/B, Applied Mathematics 1413. \r\n\r\nExtra Information: 4 lecture hours."
53.     }
54.     ...
55. ]

```

(8) Error Handler

The error-handling middleware is a good place to distinguish between error types and send them to the centralized error-handling component. When a module raises an error, it could then be caught by API router and be propagated to the middleware(e.g., Express.js). Afterwards, a centralized error handler is called to judge whether the error is operational. If yes, error could be handled by logging, mailing, sending events etc.. If no, the server has to restart gracefully.

Overall, error-handling is not an optional extra but rather an essential part of an application, both in the development stage and in production. The strategy of handling errors in a single component in Node.js will ensure developers save valuable time and write clean and maintainable code by avoiding code duplication and missing error context.

2.Client side Implementation

(1) Adding Angular HttpClient

First step is adding HttpClient to our AngularApp project because this services available as an injectable class, with methods to perform HTTP requests. Then opening src/app/app.module.ts file and adding **import { HttpClientModule } from '@angular/common/http'**. With HttpClient service we can consume REST API in back-end.

(2) Creating Interface

An interface a specification that identifies a related set of properties and methods to be implemented by a class. To be more specific, in Angular framework, interface is where we define the data type. In our online timetable application project, two interfaces need to be defined:

A.TimeTable Interface

```

1. export interface TimeTable{
2.     component: Array<string>;
3.     campus: Array<string>;

```

```

4.     courseType: Array<string>;
5.     designation: Array<string>;
6.     subject: Array<string>;
7.     start_time:[];
8.     end_time:[];
9.     day:[];
10. }

```

B.Result Interface

```

1.  export interface Result{
2.      length: Number,
3.      result: [
4.          {
5.              catalog_nbr: string,
6.              subject: string,
7.              className: string,
8.              course_info: [{
9.                  class_nbr: Number,
10.                 start_time: string,
11.                 descrlong: string,
12.                 end_time: string,
13.                 campus: string,
14.                 facility_ID: string,
15.                 days: [],
16.                 instructors: [],
17.                 class_section: string,
18.                 ssr_component: string,
19.                 enr1_stat: string,
20.                 descr: string
21.             }
22.         ],
23.         catalog_description: string
24.     ]
25. }
26. }

```

The reason for exporting TimeTable interface is to traverse all the information of the timetable schema(first API) and display the content of the corresponding field in the front-end framework in the form of a drop-down box for users to select.

(3) Creating Search Components

In an Angular App the Component is the main building block which contains the definition of the view and data. Data part shows how the view looks and behaves. Angular components are composed of plain javascript classes and using @component decorator.

The function of Component is to pass the data to the view by a process named Data Binding. Data Binding is binding the DOM elements to component properties which can also be used to display component class property values to the user, respond to the user event as well as change element style etc..

The Component consists of three main building blocks:

A.Template: The template defines the layout of the View and defines what is rendered on the page. Without the template, there is nothing for Angular to render to the DOM. Templates are always HTML page.

B.Class: Class is associated with Template (HTML Page). The Class is created with the Typescript, (javascript can also be used). Class Contains the Properties & Methods we should implement in our task.

C.MetaData: Metadata Provides additional information about the component to the Angular. Angular uses this information to process the class. The Metadata is defined with a decorator.

In our project, we need to add a new **search component** where to design user interface and add properties and functions. The following is the commands:

1. `cd AngularApp`
2. `ng generate component Search`

We'll get the following output in terminal:

1. `CREATE src/app/search/search.component.html (19 bytes)`
2. `CREATE src/app/search/search.component.spec.ts (614 bytes)`
3. `CREATE src/app/search/search.component.ts (261 bytes)`
4. `CREATE src/app/search/search.component.css (0 bytes)`
5. `UPDATE src/app/app.module.ts (467 bytes)`

(4) Adding Routing

This step is to configure the routing in Angular project. Going to the `src/app/app-routing.module.ts` file and adding the following routes:

A. Importing the search component.

B. This route is for redirecting the empty path to the search component, in other words, search component html file is the first page to be shown in the browser when users open our Angular app.

(5) Creating Service for Consuming REST API

According to the REST API designed before and the user interface of online timetable application, a search panel is essential. However, before creating a search panel, getting data to display is more significant and creating a service is significant.

To create a service for consuming `getTimetableSchema` API with Angular `HttpClient`. Terminal should be open with the following commands:

1. `cd AngularApp`
2. `ng generate service http`

An Angular service is a singleton that can be wired with components or other services via dependency Injection. They contain methods that maintain data throughout the life of an application. The main objective of a service is to organize and share business logic, models, or data and functions with different components of an Angular application. In our project, service is created to be accessible to our back-end REST API.

What's more, **Observable** can be used to transmit data in Angular service. **Observable** provides support for passing messages between publishers and subscribers inside an angular application. **Observable** offer significant benefit for event handling, asynchronous programming, and handling multiple values which is commonly used in returning back-end through API endpoint in Angular service. It is a significant bridge to connect front-end and back-end.

In the end, in `search.component.ts` file, we use **`subscribe()`** function to get all the returned data in **`getTimetable()`** in `http` service. To be more specific, **`subscribe()`** is a method on the **Observable** type.

(6) Handling HTTP Errors

Error handling is always an important part in Angular application. With `HttpClient`, network problems and front-end code errors return `ErrorEvent` instances. By verifying if an error is an instance of `ErrorEvent`, we can figure out which type of error we have and handle it accordingly.

(7) Building Searching Panel with Angular Component

In this search panel we should create 6 select box fields, 1 input box, 1 check box array, 1 check box and 1 submit button. Select box includes Subject, Course Suffix, Component, Starting Time, Ending Time. Check Box array is Day of Class. Input box is for user searching by course number and another 1 check box is to control whether to show courses only for registration.

There are two different approaches to forms in Angular. The first category is the template-driven forms which is creating html-input-elements and then use directives like ngModel to bind their value to a component's variable. However, this time we should design a search panel with 7 fields where we can select and input value interactively. Therefore reactive form is a better choice. Reactive forms use an explicit and immutable approach to managing the state of a form at a given point in time. Each change to the form state returns a new state, which maintains the integrity of the model between changes.

We use **FormBuilder**, **FormControl**, **FormGroup** to create a **Reactive Form** with select box, input and checkbox. **FormBuilder** provides syntactic sugar that shortens creating instances of a **FormControl**, **FormGroup**, or **FormArray**. It reduces the amount of boilerplate needed to build complex forms. **FormControl** is a class that is used to get and set values and validation of a form control such as <input> and <select> tag and **FormGroup** and has the role to track value and validity state of group of **FormControl**. To set a default value to our form control, we can pass a default value while instantiating a **FormControl** in our class and In HTML template, we will use these properties with form control. What's more, To validate a form control created by **FormControl**, **Validators** that belongs to @angular/forms library is applied. In the search form, subject, start_time, end_time, days, campus and component are set to be required and default value of course_number and status are null.

In HTML template, **FormControl** is used to create select boxes using tag <select></select> and <option></option> and input box as well. It is worth mentioning that **(change)** method combined with **\$event** is used to get selected value and ***ngFor** is to traverse all the subject value we get in **getTimetable()** function. We also create **getcouse()** method combined with **\$event** to get the value from the course number we have input.

As for the checkbox, **FormControl** is not used but rather use a common array with default value checked. However, a problem is how to delete days not checked. So we use the **(change)** function and complete a **getDays(\$event)** function. If the day is checked it can be pushed into daysArray otherwise **filter()** method is used to select checked array. Similarly, to obtain the value of the status, whether the user wants to search courses only for registration, we use **getStatus(\$event)** and **event.target.checked** to get the value. If it is checked, the values is "Not full".

(8) Building Result Panel with Angular Component

After finishing the design of search panel, next step is to design result panel where to show the course information based on searching conditions. A formal course information includes subject,

course number, class name, course description, section, component, class number, start time, end time, location, day of class, start time, end time, instructors, enrollment status, requisites and constraints and campus. Subject, class number, class name and course description should be highlighted and the remaining fields can be made into a table.

First step is to get all the suitable course information from the `getScheduleByAll` API through `http.service.ts` by adding results array and **search()** function.

Then coming back to `search.component.ts` to complete **onSubmit()** function using **subscribe()** function which means that the course information results should be displayed after clicking submit button. Given that the returning results are a bit of complex, we can convert them into JSON format which is more convenient to make result panel.

After getting the returning results from `httpClient`, the following task is to display all the obtained course information results in result panel. This time we choose a static table to show all the course information results. In order to create a table to display information, ***ngFor** can be used that is built in method in Angular to iterate through arrays, this time we put ***ngFor** in `<div>`. To better display the days information, a sub-table inside the original one is created. After displaying table with course information, CSS style is essential.

(9) Adding Pagination and Filtering

In order to make the result panel function more diversified, pagination and filtering by class name function are added.

A.Pagination

In `app.module.ts` and `search.component.ts` files add **import { NgxPaginationModule } from 'ngx-pagination'**, setting pagination parameters where items in per page is 5 and currentPage is variable p, p is set value by **(pageChange)"p=\$event"** method.

B.Filtering by class name

To realize filtering function, creating an input box and adding **search()** function in input box. In **search()** function, we use **match()** method and **toLowerCase()** function to filter appropriate class name.

(10) Adding Local Storage

Local storage is a new local storage API for HTML5. In fact, it is a small warehouse with a space of 5M. It is stored in the browser. We can use it through javascript in angular, it is actually used in typescript. In addition, it belongs to permanent storage.

Cookies can also be stored locally, but the storage space of cookies is small, and the storage space of each cookie is only 4K. Obviously, there is a problem of insufficient space for cookies. It is to solve the problem of insufficient storage space for cookies, so with local storage, the space size is generally 5M, but it may be different in different browsers.

Through Angular local storage, saving or deleting the information user submitted to the form. The common methods are **localStorage.setItem('key', value)** and **removeItem('key')**. Local storage is better to be as a service function.

(11)Other Functions

We have also optimized the page settings and user experience. For example, the page can adapt to the screen and automatically jump to the result panel when the user clicks the physical examination button. And in the future, there are more meaningful functions deserve to be explored.

Testing and Evaluation

1. User Interface Testing

Interface features are tested to ensure that design rules, aesthetics and related visual content is available for the user without error.

Individual interface mechanism are tested in a manner that is analogous to unit testing.


Interface mechanism is tested within the context of a use-case for a specific user category.

The complete interface is tested against selected use-cases to uncover errors in the semantics of the interface.

The interface is tested within a variety of environments(e.g., browsers) to ensure it will be compatible.

(1). Index page and search panel

COVID-19 Information website of Western University for information on our response to the pandemic.

Western  **Undergraduate**

Fall/Winter 2020-2021 Schedules

All fall courses will be delivered online until future notice.

Individual instructors will specify if and when a component of the course will take place synchronously per the published class schedule.

Search

Subject:	<input type="text" value="All Subjects"/>	Course Number:	<input type="text" value="eg. 1001"/>
Course Suffix:	<input type="text" value="Any (any suffix)"/>	Component:	<input type="text" value="All"/>
Starting Time:	<input type="text" value="All"/>	Ending Time:	<input type="text" value="All"/>

Fig.2. Index page

Search

Subject:	<input type="text" value="All Subjects"/>	Course Number:	<input type="text" value="eg. 1001"/>
Course Suffix:	<input type="text" value="Any (any suffix)"/>	Component:	<input type="text" value="All"/>
Starting Time:	<input type="text" value="All"/>	Ending Time:	<input type="text" value="All"/>
Campus:	<input type="text" value="Any"/>	Day of Class:	<input checked="" type="checkbox"/> Mon <input checked="" type="checkbox"/> Tue <input checked="" type="checkbox"/> Wed <input checked="" type="checkbox"/> Thu <input checked="" type="checkbox"/> Fri

☐ Show only courses open for registration.
*Note: may not be an accurate reflection during paper add/drop

Submit

Fig.3. Search Panel

2.Interface Mechanisms Testing

Forms: The select box, input and check box in the forms on web page of the Angular App will be tested by giving sample data and submitting the form.

Result Panel: After submitting data to the form, result panel will be tested whether the course information shown is relevant to form data.

Links: All the links in the Angular App will be tested that they are functional and accurate.

Pagination: Pagination controls in the Angular App will be tested whether it is 5 course in one page and whether it can move forward and backward for the page.

Filter: Searching by class name function in the result panel will be tested by giving sample data and when there is no keywords input, whether it could go back to original state.

Local Storage: Local Storage function will be tested by whether the submitted information could be saved in local storage after user clicks submit button.

(1). Result panel base on selected fields

Search

Subject:
Course Number:

Course Suffix:
Component:

Starting Time:
Ending Time:

Campus:
Day of Class: ☒ Mon ☒ Tue ☒ Wed ☐ Thu ☐ Fri

☐ Show only courses open for registration.
 *Note: may not be an accurate reflection during paper add/drop

Submit

Fig.4.Search panel based on selected_fields(a)

1 Results

CHEM 1302B - DISCOVERING CHEMICAL ENERGETIC

Course Description: An examination of how the fundamentals of energetics influence chemical processes. Topics include: gases, thermodynamics and thermochemistry, chemical equilibria, solubility, weak acids and bases, electrochemistry, and chemical kinetics. Antirequisite(s): The former Chemistry 1024A/B. Extra Information: 3 lecture hours, 1.5 laboratory hours (3 hours every other week).

Section	Component	Class Number	Day of Class	Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
002	LEC	3733	M W F	9:30 AM	10:30 AM	NCB-101		Full	STUDENTS CANNOT BE ADDED TO A LAB THAT IS ALREADY FULL. STUDENTS ENROLLING IN BOTH 1301A AND 1302B DO NOT NEED TO BE IN THE SAME SECTIONS FOR BOTH	Main

Fig.5.Result panel based on selected_fields(a)

Search

Subject:
Course Number:

Course Suffix:
Component:

Starting Time:
Ending Time:

Campus:
Day of Class: ☒ Mon ☐ Tue ☒ Wed ☐ Thu ☒ Fri

☐ Show only courses open for registration.
 *Note: may not be an accurate reflection during paper add/drop

Submit

Fig.6.Search panel based on selected_fields(b)

CHINESE 1650G - PERSPECTIVES ON CHINA

Course Description: An examination of China as it emerges in the era of globalization. Contents include territory, people, society, language, science and technology, development and sustainability. Analysis of dominant and diverse realities will provide an essential basis for an appreciation of continuity and change in China. Students will learn how to access major sources of information and critically to evaluate perspectives and debates. Taught in English. Extra Information: 3 hours.

Section	Component	Class Number	Day of Class	Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
550	LEC	8344	M	2:30 PM	5:30 PM	HC-W101		Full		Huron

CHINESE 1651F - CHINESE SYMBOLS & ICONS

Course Description: This course surveys traditional symbols and icons still prevalent in China's everyday life, ranging from "yin-yang", "dragon", "mandarin ducks" and "the double-happiness", to "the three stars", "Lord Guan" and "Avalokitesvara" (Guanyin). Treating these symbols and icons as image-signifiers, the course illustrates the socio-historical contexts that have shaped major symbolism in China. Students will gain a basic understanding of Chinese culture and develop skills in critical examination of cultural phenomena. Taught in English. Extra Information: 3 hours.

Section	Component	Class Number	Day of Class	Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
550	LEC	8251	M	2:30 PM	5:30 PM	HC-W101		Full		Huron

Fig.7.Result panel based on selected_fields(b)

(2). Result panel based on registration

Search

Subject:	Calculus	Course Number:	eg. 1001
Course Suffix:	Any (any suffix)	Component:	Lecture
Starting Time:	All	Ending Time:	All
Campus:	Main	Day of Class:	<input checked="" type="checkbox"/> Mon <input type="checkbox"/> Tue <input checked="" type="checkbox"/> Wed <input type="checkbox"/> Thu <input checked="" type="checkbox"/> Fri

☒ Show only courses open for registration.
*Note: may not be an accurate reflection during paper add/drop

Submit

Fig.8.Search panel based on registration

CALCULUS 1000A - CALCULUS I

Course Description: Review of limits and derivatives of exponential, logarithmic and rational functions. Trigonometric functions and their inverses. The derivatives of the trig functions and their inverses. L'Hospital's rules. The definite integral. Fundamental theorem of Calculus. Simple substitution. Applications including areas of regions and volumes of solids of revolution. Antirequisite(s): Calculus 1500A/B, the former Calculus 1100A/B, Applied Mathematics 1413. Extra Information: 4 lecture hours.

Section	Component	Class Number	Day of Class				Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
001	LEC	3370	M	Tu	W	Th	8:30 AM	9:30 AM	UCC-146		Not full		Main

CALCULUS 1000B - CALCULUS I

Course Description: Review of limits and derivatives of exponential, logarithmic and rational functions. Trigonometric functions and their inverses. The derivatives of the trig functions and their inverses. L'Hospital's rules. The definite integral. Fundamental theorem of Calculus. Simple substitution. Applications including areas of regions and volumes of solids of revolution. Antirequisite(s): Calculus 1500A/B, the former Calculus 1100A/B, Applied Mathematics 1413. Extra Information: 4 lecture hours.

Section	Component	Class Number	Day of Class		Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
001	LEC	1071	M	W	7:00 PM	9:00 PM	SEB-1059		Not full		Main

Fig.9.Result panel based on registration

(3). Result panel based on course number

Search

Subject:	All Subjects	Course Number:	32
Course Suffix:	Any (any suffix)	Component:	Lecture
Starting Time:	All	Ending Time:	All
Campus:	Any	Day of Class:	<input checked="" type="checkbox"/> Mon <input checked="" type="checkbox"/> Tue <input checked="" type="checkbox"/> Wed <input checked="" type="checkbox"/> Thu <input checked="" type="checkbox"/> Fri

☒ Show only courses open for registration.
*Note: may not be an accurate reflection during paper add/drop

Submit

Fig.10.Search panel based on course number

AH 2632G - CANADIAN ART

Course Description: An introduction to the visual arts of Canada in the 20th century, including First Nations and Inuit art, cultural policy, and collecting and curatorial practices in Canada. Key movements in Canadian art are discussed in relation to the social and political context. Antirequisite(s): the former VAH 2272F/G, the former VAH 2276E. Extra Information: 3 hours: lecture, blended or online format.

Section	Component	Class Number	Day of Class	Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
200	LEC	9903	Th	11:30 AM	1:30 PM	NCB-117		Not full		Main

CGS 3202G - SEMINAR IN GLOBAL STUDIES

Course Description: Organized around participation in colloquia, workshops, and presentations with community-based organizations, movements, and professionals, this course gives students opportunities to deepen understandings and insights into community-driven learning and scholarship around problems at stake in Centre for Global Studies academic programming. Consult with the Centre for this year's topic.

Section	Component	Class Number	Day of Class	Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
550	LEC	9295	W	2:30 PM	5:30 PM	HC-V208		Not full	REGISTRATION BY PERMISSION OF INSTRUCTOR. TOPIC: FEMINIST AND ANTI-RACIST WORK FROM ACADEMY	Huron

Fig.11.Result panel based on course number

(4). Result panel based on all subjects

Search

Subject:	All Subjects	Course Number:	eg. 1001
Course Suffix:	Any (any suffix)	Component:	All
Starting Time:	5:30 pm	Ending Time:	All
Campus:	Main	Day of Class:	<input checked="" type="checkbox"/> Mon <input type="checkbox"/> Tue <input checked="" type="checkbox"/> Wed <input type="checkbox"/> Thu <input checked="" type="checkbox"/> Fri

☐ Show only courses open for registration.
*Note: may not be an accurate reflection during paper add/drop

Submit

Fig.12.Search panel based on all subjects

2 Results

Search by Class Name

ECE 2238B - INTRO TO ELECTRICAL ENGINEERING

Course Description: DC circuit analysis, fundamentals of DC circuit analysis, Ohm's Law, KCL, KVL, Thévenin and Norton Equivalent circuits, maximum power transfer; linear analog circuits, diodes, transistors, operational amplifiers, biasing, gain, frequency response. Antirequisite(s): ECE 2205A/B, ECE 2231A/B. Extra Information: 3 lecture hours, 1 tutorial hour, 1 laboratory hour.

Section	Component	Class Number	Day of Class	Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
001	LEC	2590	M	5:30 PM	6:30 PM	SEB-2200		Not full	RESTRICTED TO SOFTWARE, GREEN PROCESS AND INTEGRATED ENGINEERING STUDENTS.	Main

MEDIEVAL 1022 - INTRO TO MEDIEVAL STUDIES

Course Description: This course will introduce civilization and thought in Europe and the Mediterranean between 400 and 1500, with emphasis on the medieval roots of many modern institutions and attitudes, including philosophy, technology, law, governance, courtly love and attitudes to women, warfare, art and archaeology, Christianity and Islam, literature, music and coinage. Antirequisite(s): Medieval Studies 1025A/B, Medieval Studies 1026A/B, and the former Medieval Studies 1020E. Extra Information: 3 hours.

Fig.13.Result panel based on all subjects

(5). Result panel and pagination

ACTURSCI 3424B - SHORT TERM ACTUARIAL MATH I

Course Description: Insurance loss frequency and severity models; aggregate loss models; risk measures; ruin theory; coverage modifications. Extra Information: 3 lecture hours.

Section	Component	Class Number	Day of Class			Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
001	LEC	2811	M	W	F	9:30 AM	10:30 AM	AHB-1B02		Not full		Main

« 1 2 3 4 5 ... 78 »



1878 - 2020 Western University

Western University
1151 Richmond Street
London, Ontario, Canada, N6A 3K7
Tel: 519-661-2111
Contact Us
Privacy | Terms of Use | Accessibility

Key Topics:

News & Publications
Media Relations
Teams & Expertise
Digital
Western Brand

Resources for:

Future Students
Current Students
Faculty/Staff
International
Community
Alumni

Fig.14.Result panel and pagination

(6). Filter by class name

ELE

ECE 2238B - INTRO TO ELECTRICAL ENGINEERING

Course Description: DC circuit analysis, fundamentals of DC circuit analysis, Ohm's Law, KCL, KVL, Thévenin and Norton Equivalent circuits, maximum power transfer; linear analog circuits, diodes, transistors, operational amplifiers, biasing, gain, frequency response. Antirequisite(s): ECE 2205A/B, ECE 2231A/B. Extra Information: 3 lecture hours, 1 tutorial hour, 1 laboratory hour.

Section	Component	Class Number	Day of Class	Start Time	End Time	Location	Instructors	Enrollment Status	Requisites and Constrains	Campus
001	LEC	2590	M	5:30 PM	6:30 PM	SEB-2200		Not full	RESTRICTED TO SOFTWARE, GREEN PROCESS AND INTEGRATED ENGINEERING STUDENTS.	Main

« 1 »

Fig.15.Filter by class name

(7). Save by local storage

Key	Value
Courses	[{"subject": "", "start_time": "", "end_time": "", "days": ["M", "W", "F", "Tu", "Th"], ...}


```
▼ [{"subject": "", "start_time": "", "end_time": "", "days": ["M", "W", "F", "Tu", "Th"], "campus": "",...},...]
▶0: {subject: "", start_time: "", end_time: "", days: ["M", "W", "F", "Tu", "Th"], campus: "",...}
▶1: {subject: "CALCULUS", start_time: "", end_time: "", days: ["M", "W", "F"], campus: "Main",...}
▶2: {subject: "CALCULUS", start_time: "", end_time: "", days: ["M", "W", "F"], campus: "Main",...}
▶3: {subject: "CALCULUS", start_time: "", end_time: "", days: ["M", "W", "F"], campus: "Main",...}
▶4: {subject: "CALCULUS", start_time: "2:30 pm", end_time: "", days: ["M", "W", "F"], campus: "Main",...}
▶5: {subject: "CHINESE", start_time: "2:30 pm", end_time: "", days: ["M", "W", "F"], campus: "Huron",...}
▶6: {subject: "CHINESE", start_time: "2:30 pm", end_time: "", days: ["M", "W", "F"], campus: "Huron",...}
▶7: {subject: "CHINESE", start_time: "", end_time: "", days: ["M", "W", "Tu", "Th", "F"], campus: "Huron",...}
▶8: {subject: "BUSINESS", start_time: "", end_time: "", days: ["M", "W", "Tu"], campus: "Main",...}
▶9: {subject: "CHEM", start_time: "9:30 am", end_time: "", days: ["M", "W", "Tu"], campus: "Main",...}
▶10: {subject: "CHEM", start_time: "", end_time: "", days: ["M", "W", "Tu"], campus: "Main", component: "",...}
```

Fig.16.Save by local storage

Installation Guide

The back-end code for running this project can be installed from my github repository by using the following command in your local github repository:

git clone <https://github.com/ubakshi2/western-timetable.git> **Jiaxin Branch**, not master branch because we have changed the previous API designed before.

This repository consists of the back-end code and JSON file. The user can follow the instructions in this report to setup the back-end and server on their own system. The user should make sure to install all the dependencies before running the server as discussed in this report before.

The front-end code for running this project on your local computer can also be installed from my github repository by using the following command in your local github repository:

git clone <https://github.com/ubakshi2/western-timetable.git> **Jiaxin Branch**, not master branch because Yiyun and I use Angular to implement the front-end.

In order to run this code make sure you have installed node.js and Angular CLI on your system. Also, when running this application, please make sure back-end port is listening by entering **nodemon/node server.js** in terminal and front-end is running by entering **ng serve** in terminal. Finally, open your browser and enter **localhost:4200**. Preferably use Google Chrome. What should be noted is that every time you search for a new course, the page should be turned into the first page.

When you download this repository from github there may be something wrong when you are running it. There are three things needed to be installed in terminal.

1.Server side: **npm i express**

2.Client side: **npm i -g @angular/cli**

npm i --save-dev @angular-devkit/core

Scope of Future Work

After nearly 5 years of rapid development, the current front-end development technology stack has entered a mature stage. After the emergence of frameworks such as React and Angular, the complexity of the front-end code development has been basically solved. Coupled with Node's solution to the separation of front-end and back-end, the front-end technology stack itself is actually very mature.

Node.js has already given a good start to front-end development. This head is to let front-end developers understand the details of the HTTP protocol and understand the conventional API development. I believe that many web developers have already understood why we need to separate the front and back ends. The main reason for this is that apart from the separation of code development and deployment, there is also a part to prevent people who do not understand the HTTP protocol from destroying the interface layer. Therefore, understanding the front-end of the HTTP protocol will gradually cover this part of the task of back-end development, and understanding the back-end of the HTTP protocol will also learn front-end development because of the maturity of the three major framework development models. Furthermore, the trend is evolved into full-stack development.

Conclusion

In August, our online timetable application for Western project is gradually coming to an end. Recalling the tension and anxiety of joining the project team in the second semester, the project is about to end successfully. In the past few months, we have encountered too many challenges. From university closure in March, it was hard for us to get course data from WTS, to the unfamiliarity of javascript method and syntax when completing back-end REST API, and to building the front-end Angular framework with various bugs encountered. I have overcome so many problems during the project process, for instance I don't know how to filter information in back-end and I do not fully understand the meaning of first API in the beginning. When it comes to the front-end framework, I do not know whether material table to display timetable information is a better choice, but we choose static table as last because material table is suitable for column reference, which may destroy our designed table structure.

I am so appreciated that Dr Jagath Samarabandu gives me this chance to collaborate with three other excellent students to complete this online timetable application project. It is an unforgettable experience because it not only give me an opportunity to learn and review knowledge in web development course but also I have gained valuable friendship and understand the importance of mutual help and cooperation in a team, which is so meaningful for my future career path. As for the professional knowledge. I also deeply understand that I still have many shortcomings in javascript and front-end frameworks through this project and I know I should learn more from others and search for questions actively if I want to make great progress.

More importantly, through this team development, I am also determined to become a professional developer after graduation. No matter how tortuous the road ahead, I believe that as long as I work hard, I will surely break into the sky.

References

1. https://www.tutorialspoint.com/angularjs/angularjs_overview.htm
2. https://www.w3schools.com/nodejs/nodejs_intro.asp
3. https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm
4. <https://www.youtube.com/watch?v=w-7RQ46RgxU&list=PL4cUxeGkcC9gcy91rvMJ75z9maRw4byYp&index=1>
5. <https://www.youtube.com/watch?v=1US-Pl3yKV&list=PL4cUxeGkcC9gcy91rvMJ75z9maRw4byYp&index=2>
6. https://www.youtube.com/watch?v=TlB_eWDSMt4
7. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/development_environment
8. <https://www.youtube.com/watch?v=Q-BpqyOT3a8>
9. <https://searchapparchitecture.techtarget.com/definition/RESTful-API>
10. <https://www.youtube.com/watch?v=SLwpqD8n3d0>
11. <https://www.youtube.com/watch?v=pTecle8oyc8&t=6787s>
12. https://www.youtube.com/watch?v=qkspwgbVocA&list=PL8p2I9GklV45GP23mlinXabp4d4ASdtz_K&index=3
13. <https://www.udemy.com/course/node-with-react-fullstack-web-development/learn/lecture/7593680#overview>
14. https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm
15. <https://www.cnblogs.com/chyingp/p/nodejs-learning-express-body-parser.html>
16. <http://expressjs.com/en/resources/middleware/body-parser.html>
17. https://www.youtube.com/watch?v=x_Z6iYY5ibc
18. https://www.w3schools.com/nodejs/nodejs_filesystem.asp
19. F. Leymann — Web services: Distributed applications without limWTS ||, in Proc. BTW'03 (Leipzig, Germany, February 26 - 28, 2003), Lecture Notes in Informatics, vol. P-26, Gesellschaft fuer Informatik (GI), Bonn, Germany.
20. M. P. Papazoglou, and D. Georgakopoulos, — Serviced-oriented computing ||, Communications of ACM, 46 (2003), 10, 25 - 28
21. https://www.w3schools.com/jsref/jsref_filter.asp
22. https://www.w3schools.com/jsref/jsref_map.asp
23. https://www.w3schools.com/jsref/jsref_includes.asp
24. <https://www.toptal.com/nodejs/node-js-error-handling>
25. <https://www.freecodecamp.org/news/angular-8-tutorial-in-easy-steps/>
26. <https://www.tektutorialshub.com/angular-tutorial/>
27. <https://vegibit.com/how-to-use-an-interface-in-angular/>
28. <https://angular.io/api/core/Component>
29. <https://www.tektutorialshub.com/angular/angular-components/>
30. https://www.w3schools.com/angular/angular_services.asp
31. <https://medium.com/techiediaries-com/what-is-an-angular-service-angular-9-service-by-example-466aa10fc6e6>
32. <https://dzone.com/articles/what-is-a-service-in-angular-js-why-to-use-it>
33. <https://angular.io/guide/reactive-forms>
34. <https://www.concretepage.com/angular-2/angular-2-formcontrol-example>
35. <https://medium.com/@mjthakur413/how-to-create-table-in-angular-7-using-ngfor-3c2c0875b955>
36. <https://www.slideshare.net/AdityaJain335/time-table-management-system-software-report>
37. https://blog.csdn.net/whm18322394724/article/details/80302504?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.channel_param&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.channel_param