

Name: K C A A Iroshan

Student Reference Number: 10638366

Module Code: PUSL3111	Module Name: API Software Development (19/SP/M)
Coursework Title: University to Industry (Web application for IPT program of NSBM)	
Deadline Date: 23/04/2020	Member of staff responsible for coursework: Dr. Rasika Ranaweera
Programme: BSc (Hons) Software Engineering	
Please note that University Academic Regulations are available under Rules and Regulations on the University website <a href="http://www.plymouth.ac.uk/studenthandbook">www.plymouth.ac.uk/studenthandbook</a> .	
Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.	
K. C. A. A. Iroshan 10638366 A. A. Ayesh Dulanja 10638431 M. D. S. Tharindu 10638387 T.J.M Siriwardhana 10638374 G H P Prabodhani 10638378 T. Dasanayake 10638504	
<b><i>We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.</i></b>	
Signed on behalf of the group: K C A A Iroshan	
Individual assignment: <b><i>I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.</i></b>	
Signed :	
Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.	
I *have <del>used</del> /not used translation software.	
If used, please state name of software.....	
Overall mark _____ %      Assessors Initials _____      Date _____	

## Acknowledgment

We wish to acknowledge with gratitude, the insightful guidance given by our module leader Dr. Rasika Ranaweera. You have been our guidance mentor. We have been extremely lucky to have a lecturer who cared so much about student work, and who responded to our questions so promptly regarding. Not only module contents whenever he taught many extra trends, but technologies regarding our module since the second year and it also increase our knowledge.

## Table of Contents

### Table of Contents

Acknowledgment.....	1
Table of Contents.....	1
Table of figures .....	3
1.0 Project identification .....	4
1.1 Introduction.....	4
1.2 Project Goals .....	4
1.3 Project Objectives.....	4
2.0 Planning .....	5
3.0 Analyze .....	5
3.1 Functionalities.....	5
3.2 Requirement Analysis.....	6
3.2.1 Functional Requirement .....	6
3.2.2 Non-Functional Requirements.....	6
4.0 Designing .....	7
4.1 Architecture Diagram.....	7
4.2 Use Case Diagram .....	8
4.3 Database Design .....	9
4.4.1. Extended Entity Relationship Diagram .....	9
4.4.2. Relational Mapping .....	10
4.4.3. Normalized Tables .....	11
4.4.4. Database Diagram .....	12

5.0 Implementation.....	13
5.1 Web App Implementation .....	13
5.1.1 Features of Web Application .....	13
5.1.2 Web Application Sample Screenshots .....	14
5.2 Mobile App Implementation .....	16
5.2.1 Features of Mobile Application .....	16
5.2.2 Mobile Application Sample Screenshots .....	17
5.3 API Implementation .....	18
5.3.1 Overview .....	18
5.3.2 API Project Dependencies .....	20
5.3.3 Model .....	20
5.3.4 Controller.....	22
5.3.5 Routes.....	23
5.3.6 Configuration.....	26
6.0 Tools and Technologies .....	26
5.1 Technology Used.....	26
5.2 Why Did We Use Them? .....	28
5.3 API Development Technologies.....	30
7.0 API Documentation .....	30
8.0 Instruction to Run .....	30
9.0 Contribution.....	31
10.1 Workload Matrix .....	31
10.2 LinkedIn Certificates .....	34
11.0 Turnitin Report.....	40
12.0 Google Driver Link of the Project.....	41
13.0 References.....	42

## Table of figures

Figure I -Architecture Diagram .....	7
Figure II- Use Case Diagram.....	8
Figure III- EER .....	9
Figure IV - Relational Mapping.....	10
Figure V - Normalized Tables.....	11
Figure VI - Database Diagram.....	12
Figure XI - Approved Students .....	14
Figure XII - Non-Approved students .....	15
Figure XIII - Rejected Students .....	15
Figure XVII - Student Page .....	16
Figure XXV - Login Screen.....	17
Figure XXVI - Signup Screen .....	17
Figure XXXIX- Project Structure.....	18
Figure XL- server.js .....	19
Figure XLI- package.json .....	20
Figure XLII- Models .....	20
Figure XLIII – db.js .....	21
Figure XLIV - A part of the user.model.js .....	22
Figure XLV – Controllers .....	23
Figure XLVI- a part of the user.controller.js.....	23
Figure XLVII-Routes.....	24
Figure XLVIII -admin.routes.js .....	24
Figure XLIX - expert.routes.js.....	24
Figure L - login.routes.js .....	25
Figure LI - resume.route.js .....	25
Figure LII - student.route.js .....	26
Figure LIII- user.routes.js .....	26
Figure LIV- Database Configuration .....	26

## 1.0 Project identification

### 1.1 Introduction

University to Industry is a platform that allows the industry personals to directly recruit university students. The platform allows a registered university student to maintain his profile and curriculum vitae which are made visible for the registered industry personals. The industry professionals can find out the students with qualifications and skills that they are looking for using the platform.

This project is mainly about the development of the API for this system. The REST API has been developed for the CRUD operations of the database that is being used for this system. The system is also consisted of a web application as the front end that uses the API to communicate with the database.

### 1.2 Project Goals

The main goal of this project is to develop a RESTful API web application and complete our project according to the given assignment criteria.

Learn something new and improve our technical skills and knowledge were our second goal as a team.

### 1.3 Project Objectives

- Carefully analyses the scenario and deliver a good effective solution.
- Figure out new developing tools and technologies which can fulfill the project requirements.
- Design and development of a functional web application and a mobile application.
- The source code should have been tested properly by using test data.
- The design clearly illustrated within the website documentation and clearly evident in the architecture.
- Proper project documentation.

## 2.0 Planning

Here, we identified and defined the purpose and project plan for the system development. There is the main task we have completed in this stage.

That is Requirement Gathering. Requirements gathering is an essential part of any project and project management. Understanding fully what a project will deliver is critical to its success.

Requirement gathering was mainly done by analyzing project criteria and observing the given scenario.

## 3.0 Analyze

### 3.1 Functionalities

From the IPT program manager(Admin) aspect,

- IPT program manager can access the websites admin panel by using username and password.
- IPT program manager can see details about any registered member.
- According to the given details IPT program manager can approve or reject any membership.
- IPT program manager can modify their account details and logout from the account.

From the student aspect,

- Any student can sign up for the website by giving required details.
- They can sign in using username and password.
- Student can edit their profile details and share their qualifications.

From the IT expert aspect,

- Any IT expert can sign up for the website by giving required details.
- They can sign in using username and password.
- Approved experts can filter by the category to find appropriate students and communicate with them.
- They can also can filter by the category and see the previous selected students and their contact details.

## 3.2 Requirement Analysis

### 3.2.1 Functional Requirement

1. Students can register themselves with details including a student ID, name, profession, email, affiliated university, password, etc.
2. Experts can register themselves with details include national ID, name, profession, email, affiliated company, password, etc.
3. Students or experts can update their profiles.
4. Anyone who logged in can access member details when the id is known.
5. An expert should be able to filter the students by category.
6. The IPT manager can remove invalid members, students or experts

### 3.2.2 Non-Functional Requirements

1. Availability- The system is available for accessing anytime with the maximum number of users
2. Reliability- The system is error-free and work according to the specifications mentioned.
3. Security- A third party cannot access the system without any authorization from the administration.
4. Accuracy- The system depends on real time information and the system provides real time updated data to the user. If any modification happened that will be updated in the system instantly.
5. Maintainability- The system can be easily maintained.

## 4.0 Designing

### 4.1 Architecture Diagram

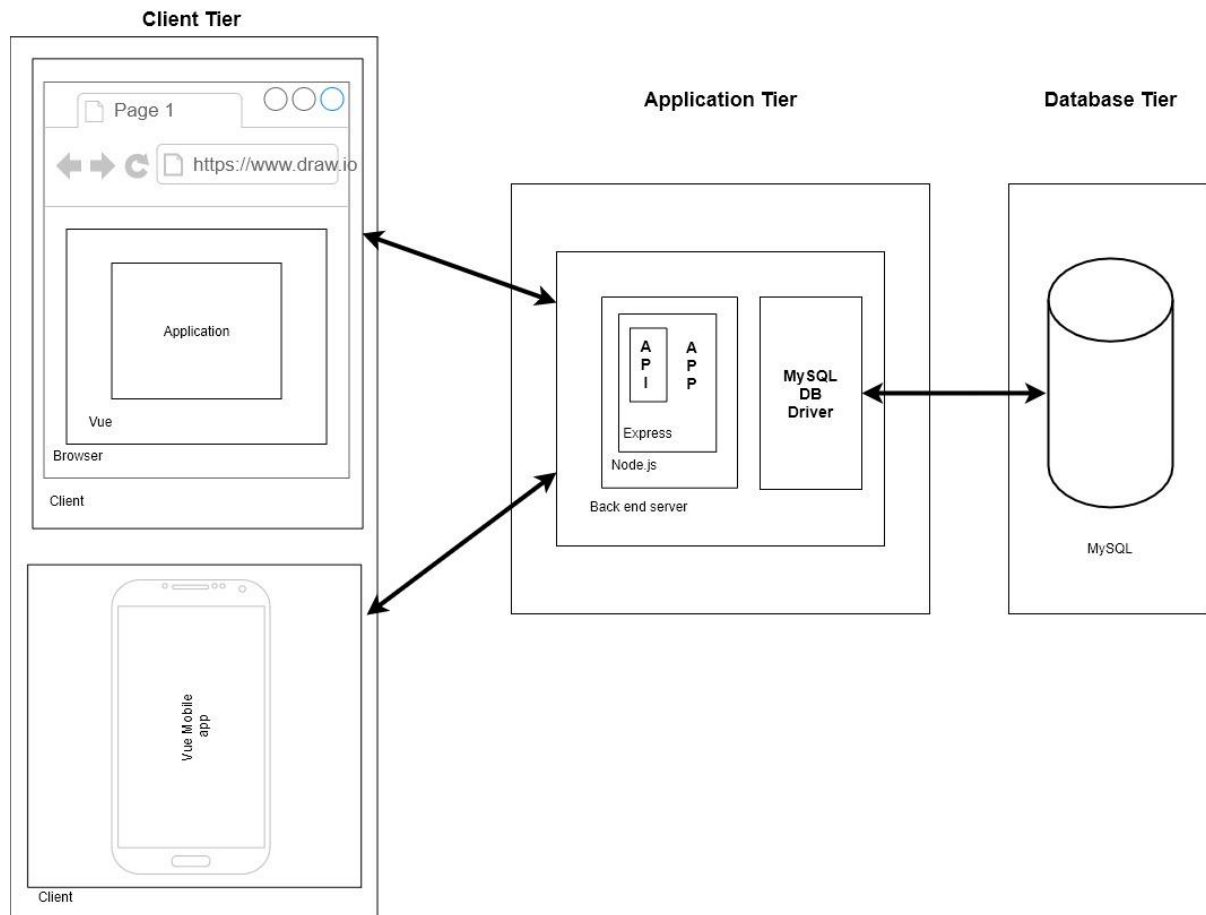


Figure 1 -Architecture Diagram



## 4.2 Use Case Diagram

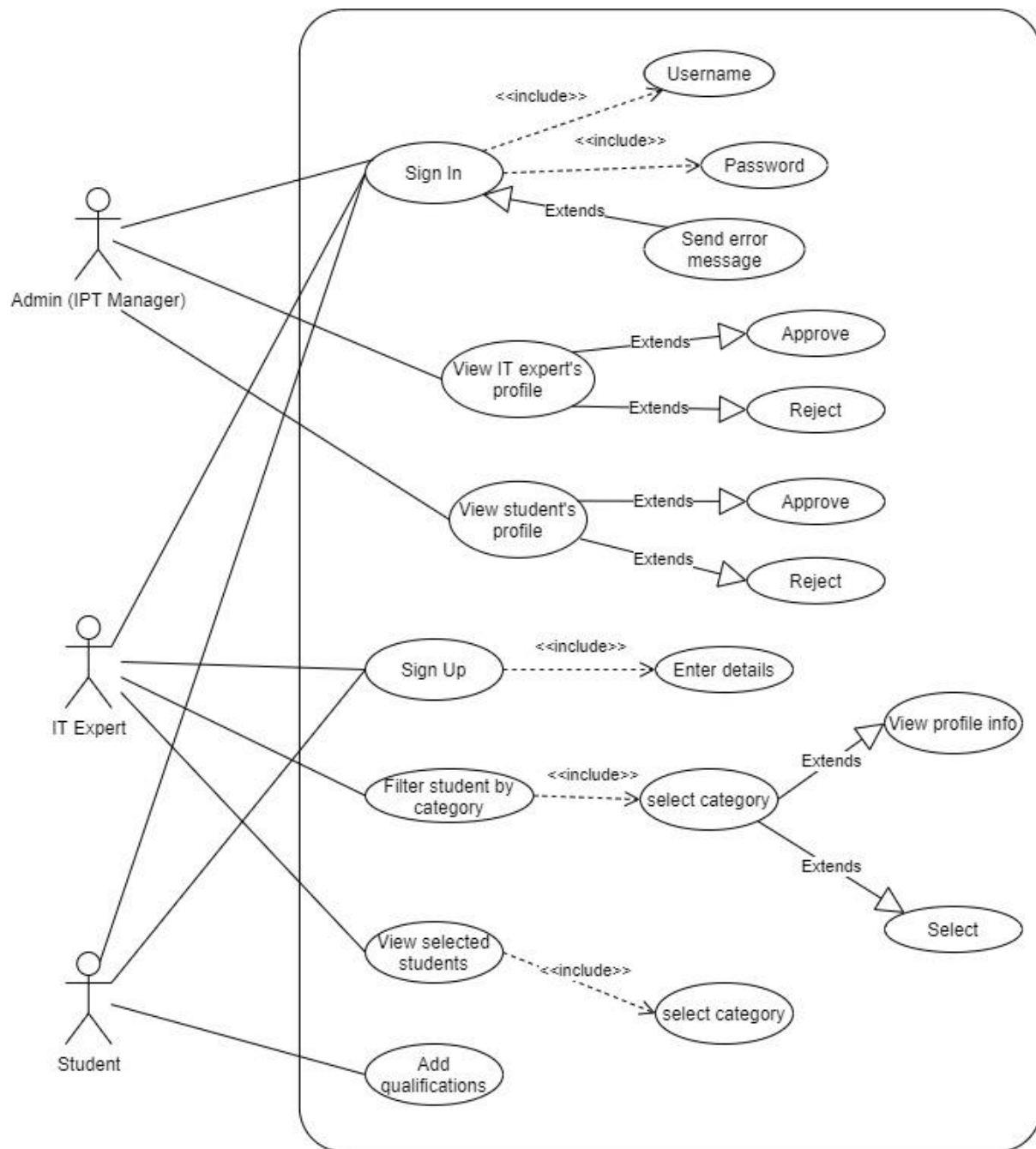


Figure II- Use Case Diagram

### 4.3 Database Design

#### 4.4.1. Extended Entity Relationship Diagram

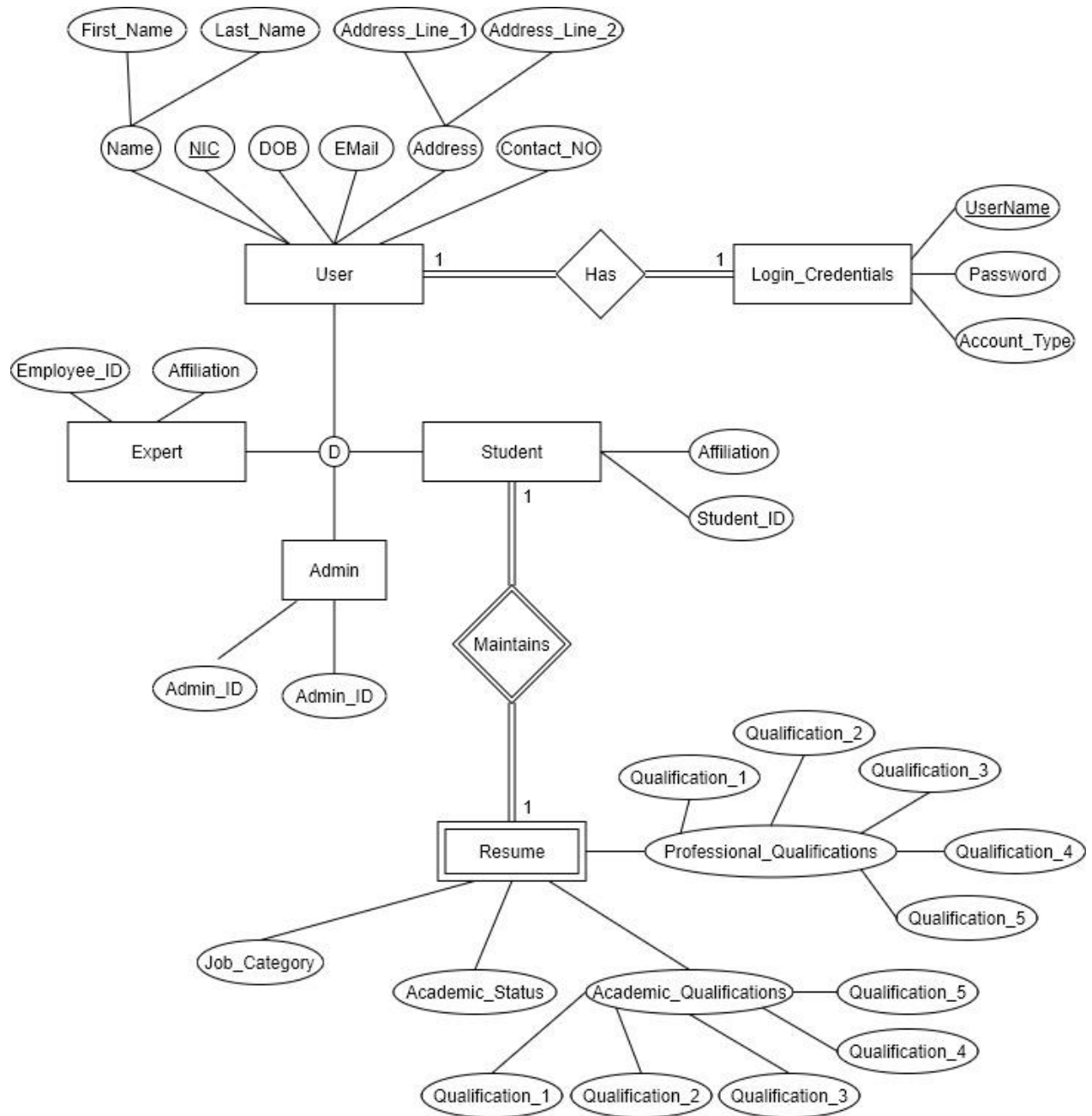


Figure III- EER

#### 4.4.2. Relational Mapping

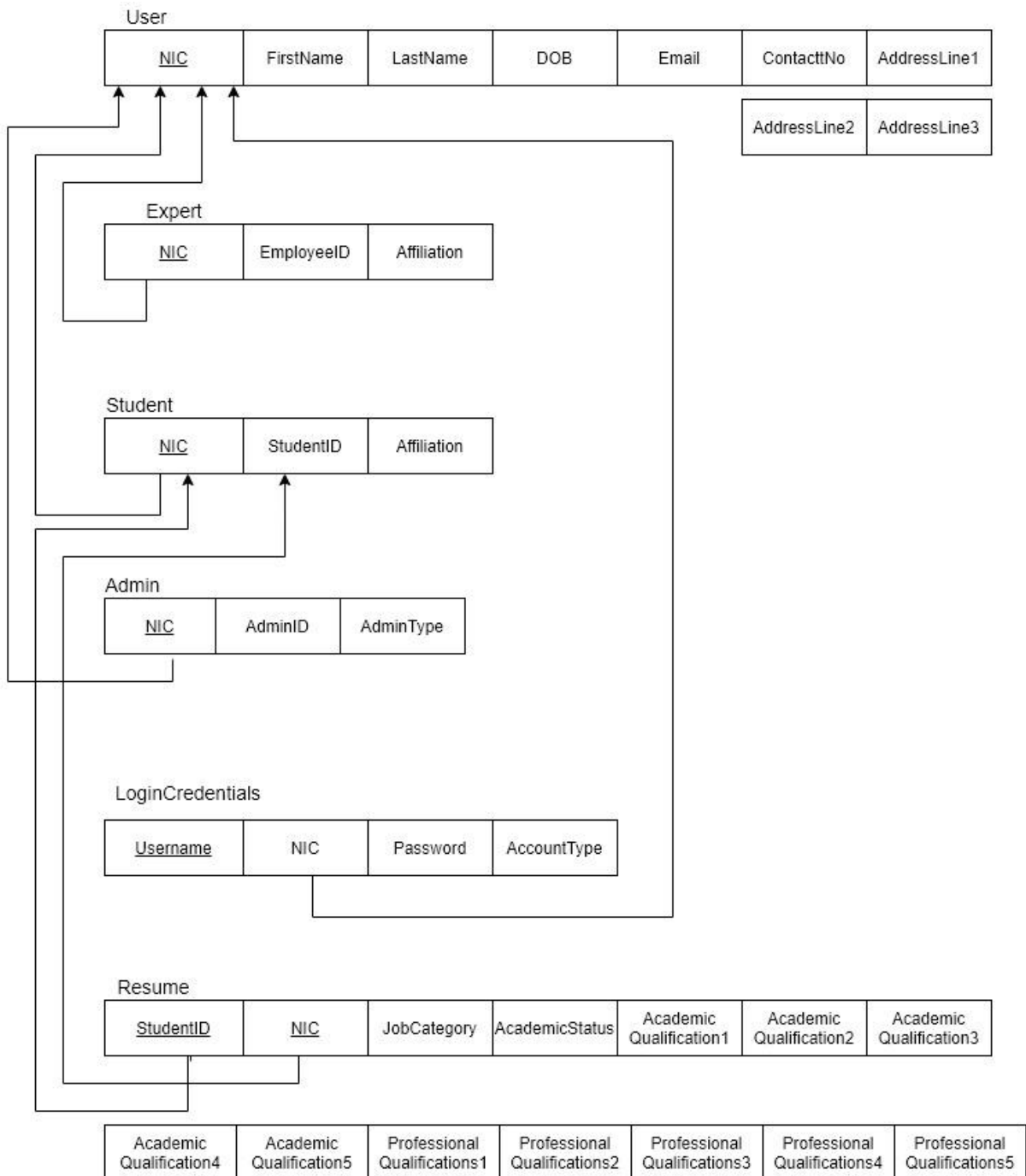


Figure IV - Relational Mapping

#### 4.4.3. Normalized Tables

User

<u>NIC</u>	FirstName	LastName	DOB	Email	ContactNo	AddressLine1
					AddressLine2	AddressLine3

Expert

<u>NIC</u>	EmployeeID	Affiliation
------------	------------	-------------

Student

<u>NIC</u>	StudentID	Affiliation
------------	-----------	-------------

Admin

<u>NIC</u>	AdminID	AdminType
------------	---------	-----------

LoginCredentials

<u>Username</u>	NIC	Password	AccountType
-----------------	-----	----------	-------------

Resume

<u>StudentID</u>	<u>NIC</u>	JobCategory	AcademicStatus	Academic Qualification1	Academic Qualification2	Academic Qualification3
Academic Qualification4	Academic Qualification5	Professional Qualifications1	Professional Qualifications2	Professional Qualifications3	Professional Qualifications4	Professional Qualifications5

Figure V - Normalized Tables

#### 4.4.4. Database Diagram



Figure VI - Database Diagram

## 5.0 Implementation

### 5.1 Web App Implementation

#### 5.1.1 Features of Web Application

- Common login Page

All the users can log in to the application using this common login. They can select their user type (Admin, Industry experts, Students) and enter their other credentials (Username, Relevant ID number, Password).

- Common Signup Page

Users can register to the system throughout this page.

- Admin Panel

Following tabs are available on Admin panel

- Students
  - Approved
  - Non- Approved
  - Rejected
- IT Experts
  - Approved
  - Non- Approved
  - Rejected

All the newly registered students and industry experts will display through this page under the Non Approved tab. Admin can approve or reject the registered users (Industry experts, Students).

All the approved users will display under the relevant "Approved" (both Experts and Students have separate tabs) tab and all the rejected users display under the relevant "Rejected" tab.

- Students Page

After successful log in, students will direct to their account. Using this account, once student come to this page it notifies whether that student is approved, pending or rejected student. Students can insert or update

their details and qualifications. Also rejected students can send request to admin for approval.

- IT Experts Page

After logged in, experts will direct to their account page. If they got the approved from the admin, they could view the approved student's details and select them. After selecting, they can view the selected students separately according to their qualification category.

### 5.1.2 Web Application Sample Screenshots

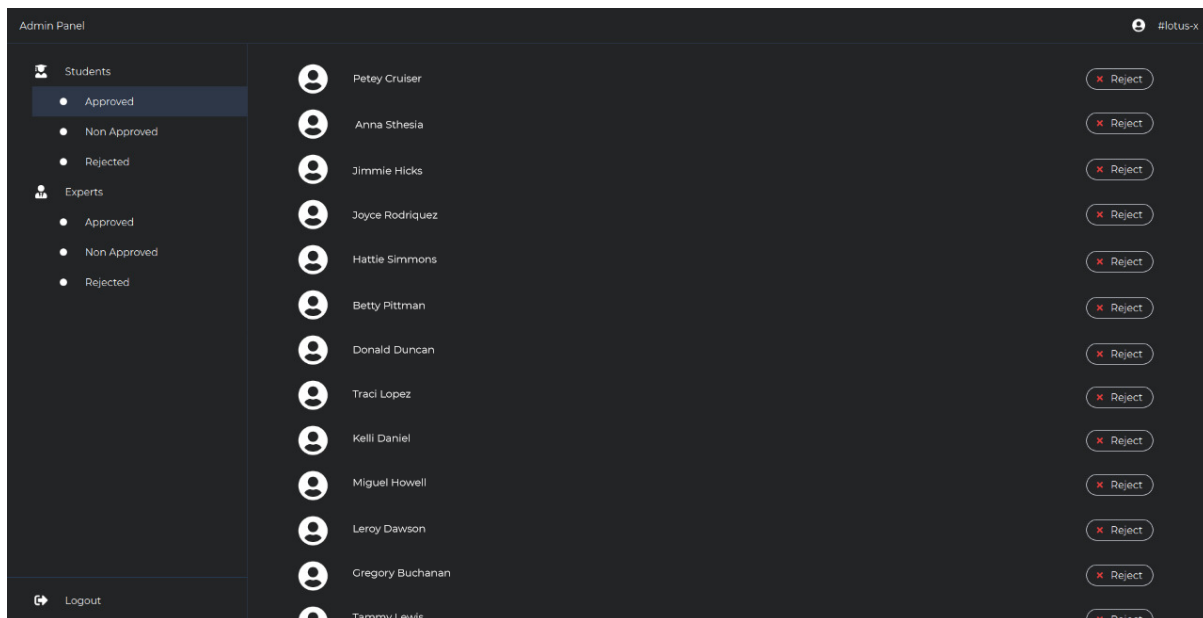


Figure VII - Approved Students

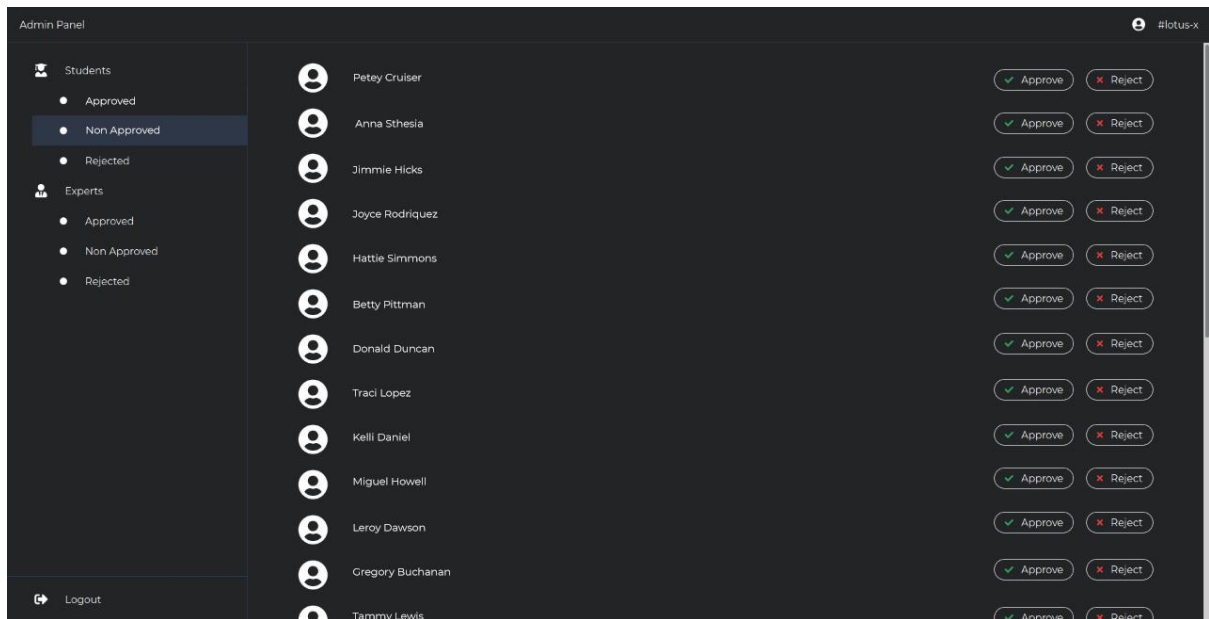


Figure VIII - Non-Approved students

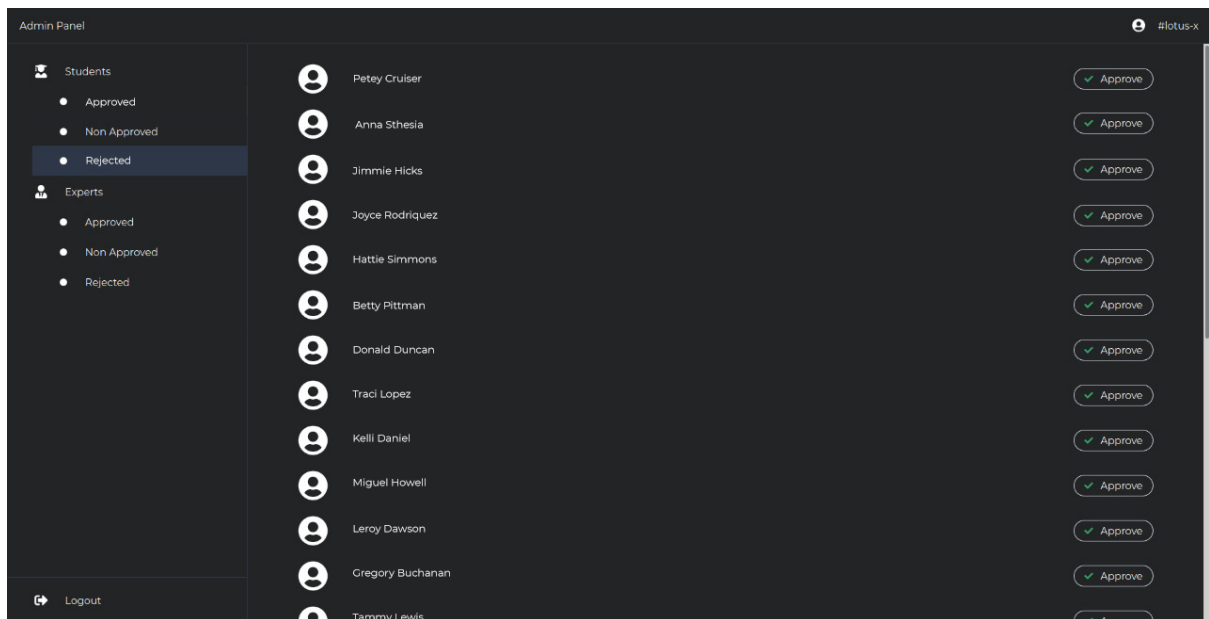


Figure IX - Rejected Students



Student Portal

logout

Primary Details

University Name

University ID

Study Year

LinkedIn Profile URL

Technical Details

Interested In -

Software Designing

Frontend Developer

Backend Developer

Database Administration

Quality Assistance

Programming Languages -

☒ JavaScript

☐ C#

☐ Java

☐ Dart

☐ Ruby

☐ C

☐ C++

☐ Python

Industrial Technologies -

☐ React

☐ React Native

☐ Angular

☐ VueJS

☐ Flutter

☐ .NET

☐ NodeJS

☐ SpringBoot

Additional Notes -

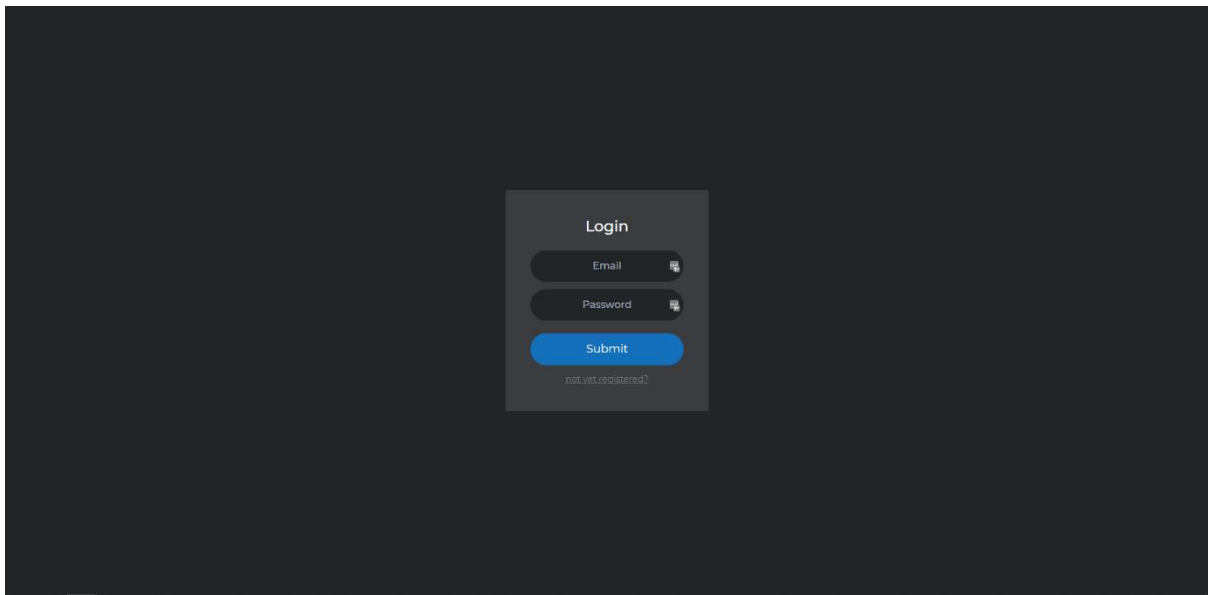
Figure X - Student Page

## 5.2 Mobile App Implementation

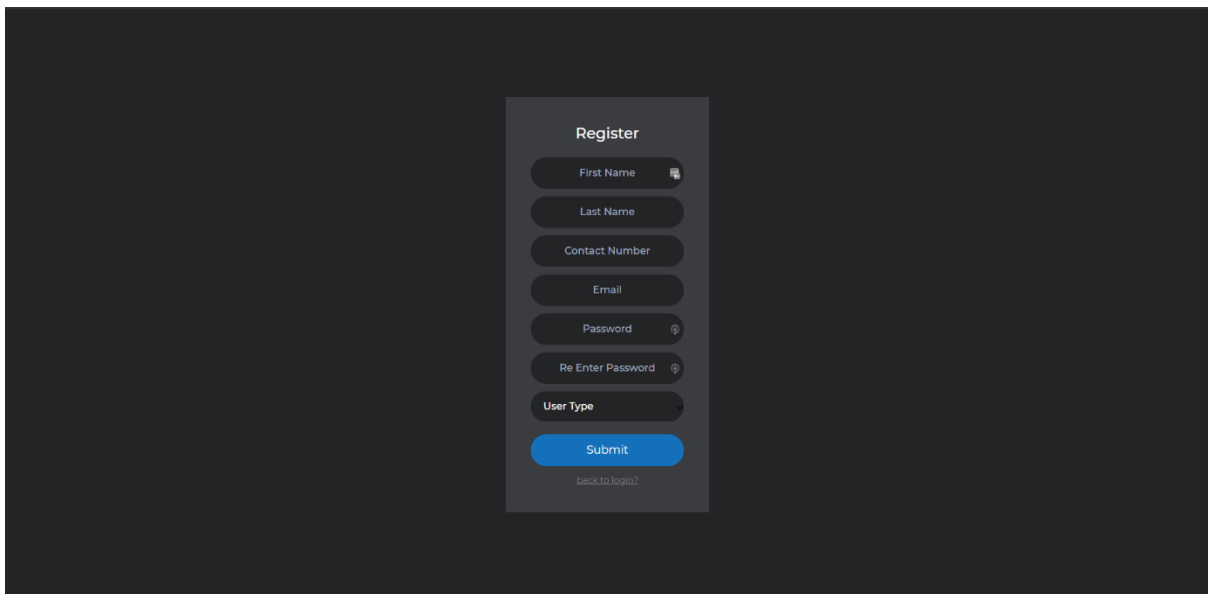
### 5.2.1 Features of Mobile Application

All the features in web application can be seen in mobile application except Admin panel.

### 5.2.2 Mobile Application Sample Screenshots



*Figure XI - Login Screen*



*Figure XII - Signup Screen*

## 5.3 API Implementation

### 5.3.1 Overview

Apart from node modules in the backend **API** folder is heart of this application. Because it contains Restful Web API.

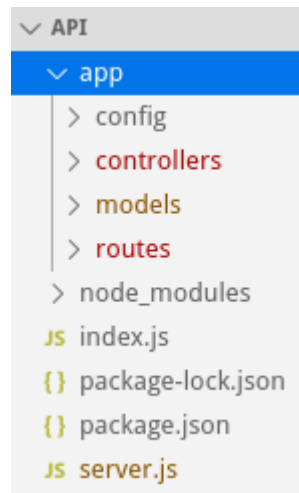


Figure XIII- Project Structure

In server.js file contains configurations of server. It has whole responsible of routing and same time working as server.

```
JS server.js > ...
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const PORT = 5001;
4
5  const app = express();
6
7  app.use(bodyParser.json());
8
9  app.use(bodyParser.urlencoded({extended:true}));
10
11 app.get("/", (req,res)=>{
12   res.json({message:"Welcome to University_To_Industry"});
13 });
14
15 require("./app/routes/user.routes.js")(app);
16 require("./app/routes/login.routes.js")(app);
17 require("./app/routes/expert.route.js")(app);
18 require("./app/routes/admin.route.js")(app);
19 require("./app/routes/student.route.js")(app);
20 require("./app/routes/resume.route.js")(app);
21
22 app.listen(PORT,()=>{
23   console.log("Server is running on PORT : "+PORT);
24 });
25 |
```

Figure XIV- server.js

### 5.3.2 API Project Dependencies


```
{ } package.json >  author
1  {
2    "name": "universitytoindustry",
3    "version": "1.0.0",
4    "description": "This system is for a platform that allows t
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "GPL-3.0-or-later",
11   "dependencies": {
12     "bcrypt": "^4.0.1",
13     "body-parser": "^1.19.0",
14     "express": "^4.17.1",
15     "mysql": "^2.18.1"
16   }
17 }
18
```

Figure XV- package.json

### 5.3.3 Model

First, we designed EER and summarize table meta details as in normalized table section.

▼ models	
JS	admin.model.js
JS	db.js
JS	expert.model.js
JS	login.model.js
JS	resume.model.js
JS	student.model.js
JS	user.model.js

Figure XVI- Models

Also, in the backend folder it has db.js file which is store connection string to the database.

```
app > models > JS db.js > ...
1  const mysql = require('mysql');
2  const dbConfig = require('../config/db.config.js');
3
4  const connection = mysql.createConnection({
5    host : dbConfig.HOST,
6    user : dbConfig.USER,
7    password : dbConfig.PASSWORD,
8    database : dbConfig.DATABASE
9  });
10
11 connection.connect(error=>{
12   if(error) throw error;
13   console.log("Connection successful");
14 });
15
16 module.exports = connection;
```

*Figure XVII – db.js*

For an example we added a part of the User model.

```

app > models > JS user.model.js > User.delete
1  const sql = require('./db.js');
2
3  const User = function(user){
4      this.NIC = user.NIC;
5      this.First_Name = user.First_Name;
6      this.Last_Name = user.Last_Name;
7      this.DOB = user.DOB;
8      this.Address_Line_1 = user.Address_Line_1;
9      this.Address_Line_2 = user.Address_Line_2;
10     this.Email = user.Email;
11     this.Contact_No = user.Contact_No;
12 };
13
14 User.create = (newUser, result)=>{
15
16     sql.query("INSERT INTO User SET ?",newUser,(err,res)=>{
17         if(err){
18             console.log("Error: "+err);
19             result(err,null);
20             return;
21         }else{
22             console.log("Created record: ", {...newUser});
23             result(null,{...newUser});
24             return;
25         }
26     });
27 };
28

```

Figure XVIII - A part of the user.model.js

#### 5.3.4 Controller

After implementing models then we implemented controllers for models. In controllers we import models and ad methods which are GET, POST, PUT, DELETE HTTP methods.

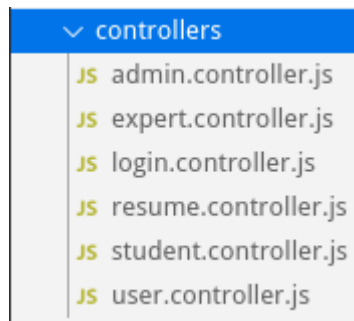


Figure XIX – Controllers

For example we added a part of the user controller that contains the controller of just record creation.

```
app > controllers > JS user.controller.js > deleteAll > exports.deleteAll > User.deleteAll() callback
1  const User = require('../models/user.model.js');
2
3  exports.create = (req,res)=>{
4      if(!req.body){
5          res.status(400).send({
6              message : "Content cannot be empty"
7          });
8      }else{
9          User.create(new User(req.body), (err,data)=>{
10             if(err){
11                 res.status(500).send({
12                     message : err.message || "An error has been occurred"
13                 });
14             }
15             else{
16                 res.send(data);
17             }
18         });
19     }
20 };
21
```

Figure XX- a part of the user.controller.js

### 5.3.5 Routes

Routes are defining the URI of particular resources along with certain HTTP methods.



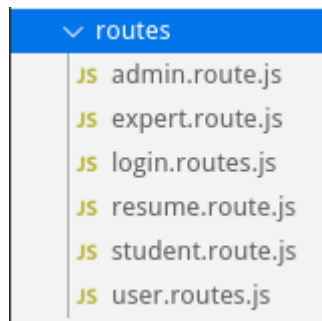


Figure XXI-Routes

```
app > routes > JS admin.route.js > <unknown> > module.exports
1  module.exports = app => {
2    const admins = require("../controllers/admin.controller.js");
3    app.post("/admins",admins.create);
4    app.get("/admins",admins.findAll);
5    app.get("/admins/:NIC",admins.findByNIC);
6    app.get("/admins/adminid/:adminid",admins.findById);
7    app.put("/admins/:NIC",admins.update);
8    app.put("/admins/adminid/:adminid",admins.updateById);
9    app.delete("/admins/:NIC",admins.delete);
10   app.delete("/admins/adminid/:adminid",admins.deleteById);
11   app.delete("/admins",admins.deleteAll);
12 }
```

Figure XXII -admin.routes.js

```
app > routes > JS expert.route.js > <unknown> > module.exports
1  module.exports = app =>{
2    const expert = require('../controllers/expert.controller.js');
3
4    app.post("/experts",expert.create);
5    app.get("/experts",expert.findAll);
6    app.get("/experts/:NIC",expert.findByNIC);
7    app.get("/experts/employeeid/:empid",expert.findById);
8    app.put("/experts/:NIC",expert.update);
9    app.put("/experts/employeeid/:empid",expert.updateByEmpId);
10   app.delete("/experts/:NIC",expert.delete);
11   app.delete("/experts/employeeid/:empid",expert.deleteByEmpId);
12   app.delete("/experts",expert.deleteAll);
13 }
```

Figure XXIII - expert.routes.js

```

app > routes > js login.routes.js > <unknown> > module.exports
1  module.exports = app => {
2
3      const login = require('../controllers/login.controller.js');
4
5      app.post("/logins", login.create);
6      app.get("/logins", login.findAll);
7      app.get("/logins/:NIC", login.findByNIC);
8      app.get("/logins/username/:username", login.findByUsername);
9      app.get("/logins/password/:username", login.getPasswordByUsername);
10     app.get("/logins/verifypassword/:username/:password", login.verifyPassword);
11     app.put("/logins/:NIC", login.update);
12     app.delete("/logins/:NIC", login.delete);
13     app.delete("/logins", login.deleteAll);
14
15 };

```

Figure XXIV - login.routes.js

```

app > routes > js resume.route.js > <unknown> > module.exports
1  module.exports = app => {
2      const resumes = require("../controllers/resume.controller.js");
3      app.post("/resumes", resumes.create);
4      app.get("/resumes", resumes.findAll);
5      app.get("/resumes/:NIC", resumes.findByNIC);
6      app.get("/resumes/studentid/:studentid", resumes.findByStudentId);
7      app.get("/resumes/category/:category", resumes.findAllByCategory);
8      app.get("/resumes/academicstatus/:academicStatus", resumes.findAllByAcademicStatus);
9      app.put("/resumes/:NIC", resumes.update);
10     app.put("/resumes/studentid/:studentid", resumes.updateByStudentId);
11     app.delete("/resumes/:NIC", resumes.delete);
12     app.delete("/resumes/studentid/:studentid", resumes.deleteByStudentId);
13     app.delete("/resumes", resumes.deleteAll);
14 };

```

Figure XXV - resume.route.js

```

app > routes > js student.route.js > <unknown> > module.exports
1  module.exports = app => {
2      const students = require("../controllers/student.controller.js");
3      app.post("/students", students.create);
4      app.get("/students", students.findAll);
5      app.get("/students/:NIC", students.findByNIC);
6      app.get("/students/studentid/:studentId", students.findByStudentId);
7      app.put("/students/:NIC", students.update);
8      app.put("/students/studentid/:studentid", students.updateByStudentId);
9      app.delete("/students/:NIC", students.delete);
10     app.delete("/students", students.deleteAll);
11 };

```

Figure XXVI - student.route.js

```
app > routes > JS user.routes.js > <unknown> > module.exports
1  module.exports = app => {
2
3      const users = require("../controllers/user.controller.js");
4
5      app.post("/users", users.create);
6      app.get("/users", users.findAll);
7      app.get("/users/:NIC", users.findByNIC);
8      app.get("/users/firstname/:Name", users.findByFirstName);
9      app.get("/users/lastname/:Name", users.findByLastName);
10     app.put("/users/:NIC", users.update);
11     app.delete("/users/:NIC", users.delete);
12     app.delete("/users", users.deleteAll);
13 }
```

Figure XXVII- user.routes.js

### 5.3.6 Configuration

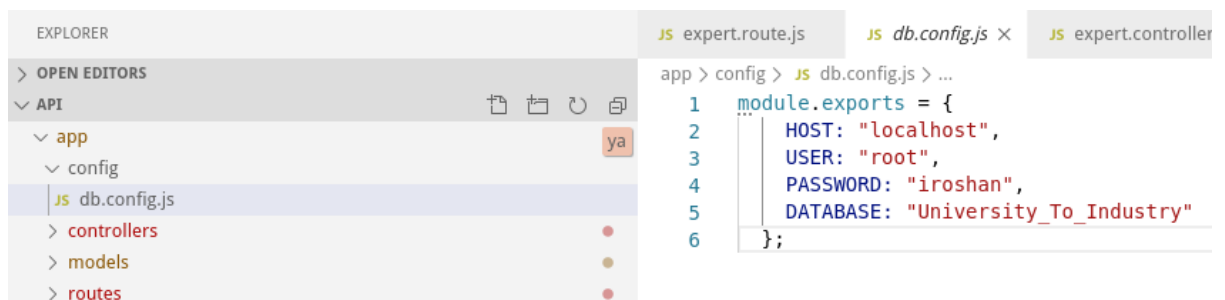


Figure XXVIII- Database Configuration

## 6.0 Tools and Technologies

### 5.1 Technology Used

To develop this project basically, we used the following technologies.

For the backend implementation

- NodeJS

“Node.js is an open-source, cross-platform, javascript runtime environment that executes Javascript code outside of a web browser”<sup>1</sup>.

- ExpressJS

“Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js”<sup>2</sup>.

It is Back-end web application framework running on top of Node.js

- MySQL.

“MySQL is an open-source relational database management system (RDBMS). MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos, and Tru64. A port of MySQL to OpenVMS also exists.”<sup>3</sup>

For the frontend implementation

- Vue Framework

“Vue is an open-source Model–view–ViewModel JavaScript framework for building user interfaces and single-page applications”<sup>4</sup>.

Apart from above we used,

---

<sup>1</sup> (Anon., n.d.)

<sup>2</sup> (Anon., n.d.)

<sup>3</sup> (Anon., n.d.)

<sup>4</sup> (Anon., n.d.)

Bcrypt - which is a node.js password hashing method to hashing passwords.

Body-Parser - It simply let handle JSON objects in backend level.

JWT – generate token and authorize to only authenticated users.

## 5.2 Why Did We Use Them?

### ✓ Node.js

Node.js is built on the JavaScript runtime of Chrome. The purpose is to develop scalable and fast network application.

The following are some of the key features that make Node.js the first choice of software architects.

#### - **Asynchronous and Event-Driven**

“All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the following API subsequent to calling it and a notification mechanism of Events of Node.js encourages the server to get a response from the past API call”<sup>5</sup>.

#### - **Fast**

“being based on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution”<sup>6</sup>.

#### - **Single-Threaded but Highly Scalable**

“Node.js uses a single-threaded model with event looping. The event mechanism causes the server to respond in a non-blocking way and makes the server profoundly versatile as instead to conventional servers which make limited threads to handle requests. Node.js uses a single-threaded

---

<sup>5</sup> (Anon., n.d.)

<sup>6</sup> (Anon., n.d.)

program and the similar program can offer support to a much larger amount of requests than conventional servers like Apache HTTP Server”<sup>7</sup>.

#### - **No Buffering**

Node.js does not buffer data. This is one of the key features of this framework. It does not buffer in order to maintain higher level or performance. Node applications simply output lumps of data without buffering and this reduces any performance overhead that could cause due to buffering.

Express and Node Js both go combine manner in the development of backend. Those two interact with each other and shared their components to make interactions with frontends.

- **Express.js**

ExpressJS is a Node JS framework. It is a prebuilt framework that can be helpful in development of server side web applications. It can be used for the development of fast web applications. Express JS packs the properties of the Node JS and as a result it has inherently got great performance.

Other great characteristics are "simplicity, minimalism, flexibility and scalability"

Express JS to Node JS is basically what Bootstrap to HTML/CSS.

- **MySQL**

While developing the database both MongoDB and MySQL were considered. But MySQL was used as it was asked specifically in the coursework documentation and due to the relational nature of the data that is used in the system, a relational database was the ideal database system and therefore MySQL was used.

---

<sup>7</sup> (Anon., n.d.)

- Vue

Vue.js is a modern JavaScript framework for the development of single page web applications. It is an open source framework with MIT open source license.

Vue which permits you to put the HTML, CSS, and JavaScript inside to scope your component accurately. It is likewise valuable since you get fine aides like babel to deal with new syntax in JS like async/await.

### 5.3 API Development Technologies

The API and the backend of the system is developed with NodeJS which is a JavaScript framework that can be used for the development of the back end of the system.

The system has a REST API developed using NodeJS for the CRUD operations with the database.

HTTP GET, HTTP POST, HTTP PUT, and HTTP DELETE methods are used in the REST API of this project and HTTP protocol is used for the data transmission between the system components.

## 7.0 API Documentation

<https://documenter.getpostman.com/view/8511782/Szf9W7Hs>

A pdf of API documentation is attached with this project folder for more convenience.

## 8.0 Instruction to Run

1. Start MySQL database service on your computer.
2. If it is a windows system, install XAMPP and click on the start button next to MySQL.

3. Then a database client program such as MySQL workbench or PhpMyAdmin can be used to import the database.
4. If it's a Linux system, enable MariaDB server by running the following command in the terminal.
5. `systemctl start MariaDB`
6. Then importing the database can be done by using a client program such as Deaver, MySQL workbench or phpMyAdmin.
7. After importing the database, navigate to the "API" folder of our submitted project folder.
8. Open a terminal inside and run the command
9. `"npm init"` to install all the dependencies.
10. Then `"node server.js"` to run the API.

## 9.0 Contribution

### 10.1 Workload Matrix

Member	10638366 K C A A Iroshan	10638387 M D S Tharindu	10638374 T J M Siriwardhana	10638504 L T S Dassanaya ke	10638378 G H P Prabodha ni	10638431 A A A Dulanja
API : <ul style="list-style-type: none"> <li>• Project Set up</li> <li>• Database Configuration (db.config.js) and Database Connection(db.js)</li> <li>• Server set up</li> <li>• Model, Controller and Routes programming for the database relation 'Resume'.               <ul style="list-style-type: none"> <li>◦ resume.model.js</li> <li>◦ resume.controller.js</li> </ul> </li> </ul>	✓					



◦ resume.route.js						
API : <ul style="list-style-type: none"> <li>Model, Controller and Routes programming for the database relation 'User' <ul style="list-style-type: none"> <li>user.model.js</li> <li>user.controller.js</li> <li>user.routes.js</li> </ul> </li> </ul>				✓		
API : <ul style="list-style-type: none"> <li>Model, Controller and Routes programming for the database relation 'login' <ul style="list-style-type: none"> <li>login.model.js</li> <li>login.controller.js</li> <li>login.routes.js</li> </ul> </li> <li>Password hashing using bcrypt.</li> </ul>			✓			
API : <ul style="list-style-type: none"> <li>Model, Controller and Routes programming for the database relation 'Expert' <ul style="list-style-type: none"> <li>expert.model.js</li> <li>expert.controller.js</li> <li>expert.routes.js</li> </ul> </li> </ul>		✓				
API : <ul style="list-style-type: none"> <li>Model, Controller and Routes programming for the database relation 'Admin' <ul style="list-style-type: none"> <li>admin.model.js</li> <li>admin.controller.js</li> <li>admin.routes.js</li> </ul> </li> </ul>					✓	
API : <ul style="list-style-type: none"> <li>Model, Controller and Routes programming</li> </ul>						✓

for the database relation 'Student' <ul style="list-style-type: none"> <li>◦ student.model.js</li> <li>◦ student.controller.js</li> <li>◦ student.routes.js</li> </ul>						
System Analysis and Database Design		✓			✓	
Database Development <ul style="list-style-type: none"> <li>• Table creation</li> <li>• Constraints</li> <li>• Foreign Keys</li> </ul>		✓				
Web Application Design				✓		✓
Web Application Development				✓		
Web Application Session handling	✓			✓		
Web Application authentication		✓		✓		
Mobile Application Design					✓	✓
Mobile App development				✓		✓
API Documentation	✓					
API Testing <ul style="list-style-type: none"> <li>✓ CRUD operations testing.</li> <li>✓ Error handling and console logs review</li> </ul>	✓					
API Testing <ul style="list-style-type: none"> <li>• Passwords Hashing and Comparison using bcrypt: testing and debugging</li> </ul>	✓		✓			
Database Testing <ul style="list-style-type: none"> <li>• Constraint evaluation</li> </ul>			✓			

<ul style="list-style-type: none"> <li>• CRUD operations testing</li> <li>• Field formats and size limits evaluation and testing.</li> </ul>						
Web Application Testing			✓	✓		✓
Mobile Application Testing				✓		✓
Report Creation					✓	
LinkedIn Learning Certificate	✓	✓	✓	✓	✓	✓

## 10.2 LinkedIn Certificates

1. K C A A Iroshan 10638366



2. M D S Tharindu 10638387



LinkedIn LEARNING

Certificate of Completion  
Congratulations, Sewmal Tharindu

## Introduction to Web APIs

Course completed on Feb 12, 2020 • 1 hour 1 min

By continuing to learn, you have expanded your perspective, sharpened your skills, and made yourself even more in demand.

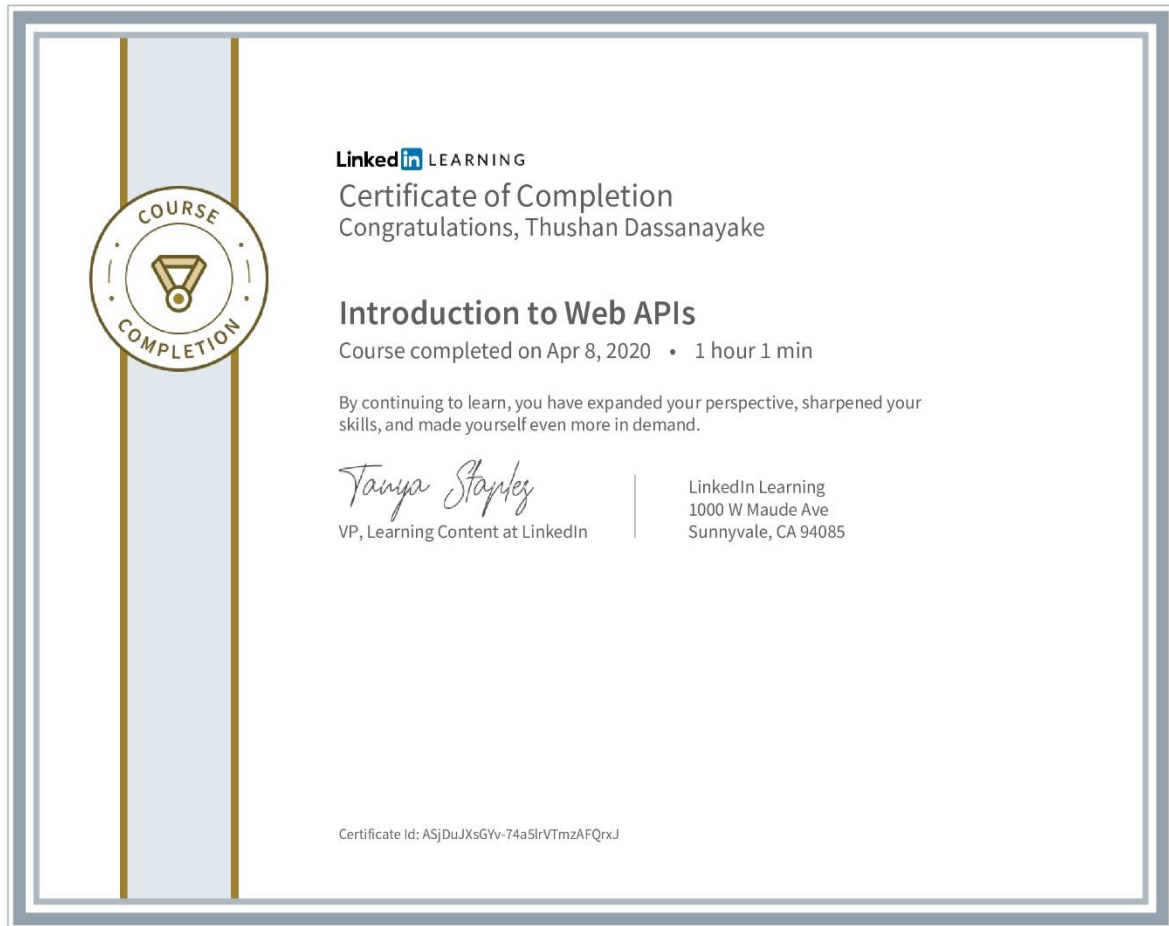
*Tanya Staples*

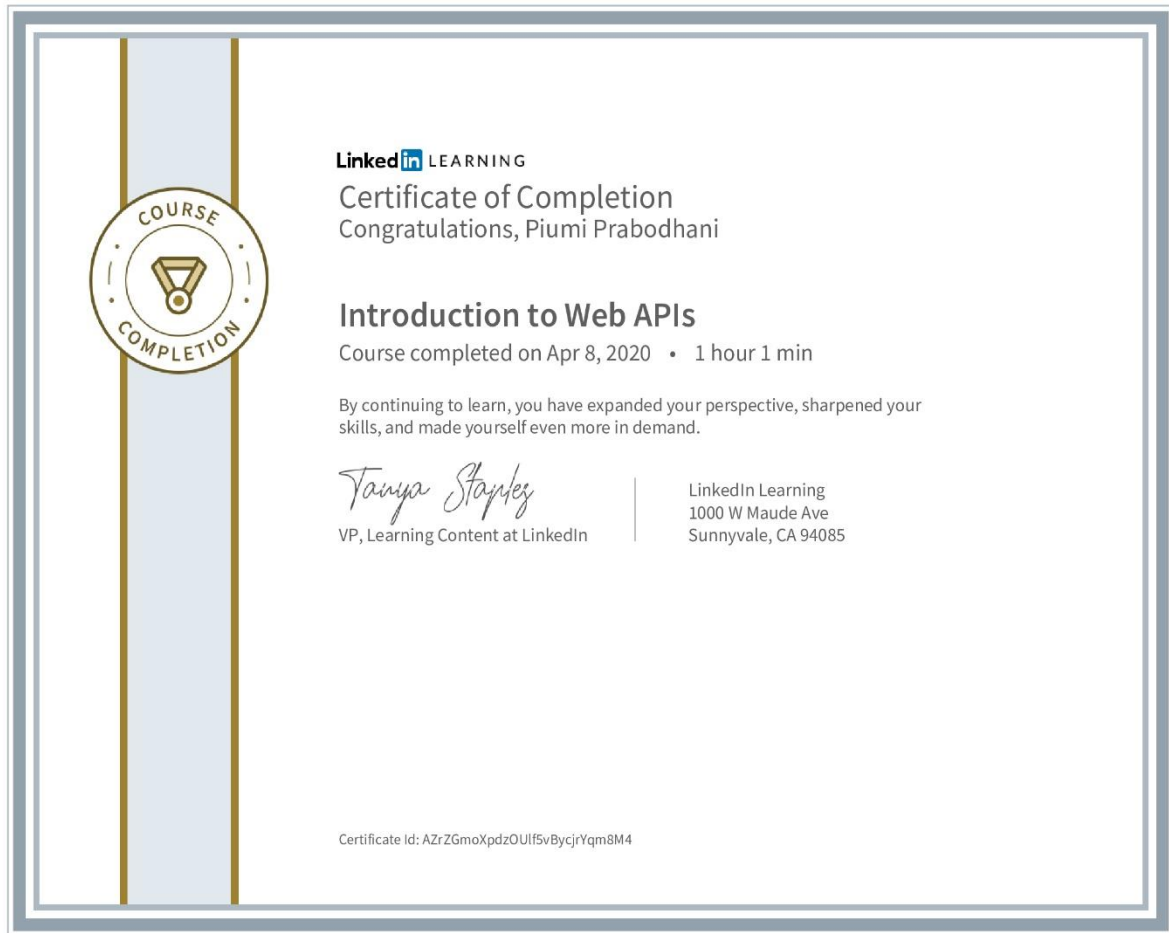
VP, Learning Content at LinkedIn

LinkedIn Learning  
1000 W Maude Ave  
Sunnyvale, CA 94085

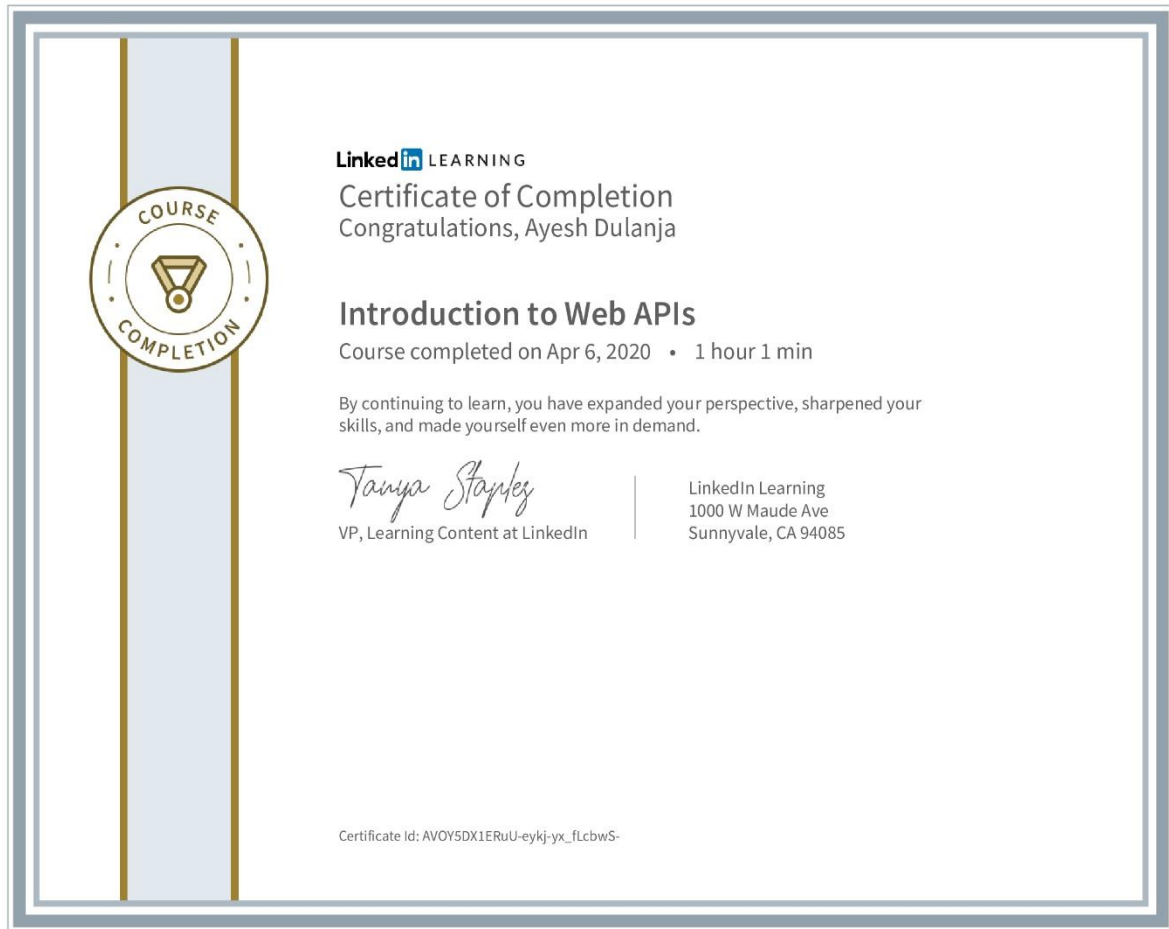
Certificate Id: ATbwoGu3TUSMwEd9ODCEjaunu4MV











## 11.0 Turnitin Report

## PUSL3111\_Group\_24\_API\_Report.pdf

### ORIGINALITY REPORT

9%

SIMILARITY INDEX

5%

INTERNET SOURCES

0%

PUBLICATIONS

7%

STUDENT PAPERS

### PRIMARY SOURCES

1

Submitted to Jyväskylä University

Student Paper

1%

2

www.manomayasoft.com

Internet Source

1%

3

Submitted to University of Bedfordshire

Student Paper

1%

4

Submitted to University of Brighton

Student Paper

1%

5

Submitted to London School of Commerce

Student Paper

1%

6

Submitted to University of Greenwich

Student Paper

<1%

7

hdl.handle.net

Internet Source

<1%

8

kodytechnolab.com

Internet Source

<1%

9

Submitted to Griffith College Dublin

Student Paper

<1%

## 12.0 One Driver Link of the Project

[https://liveplymouthac-my.sharepoint.com/:f:/r/personal/karunarathna\\_iroshan\\_students\\_plymouth\\_ac\\_uk/Documents/API%20Coursework?csf=1&web=1&e=XSMuG8](https://liveplymouthac-my.sharepoint.com/:f:/r/personal/karunarathna_iroshan_students_plymouth_ac_uk/Documents/API%20Coursework?csf=1&web=1&e=XSMuG8)

## 13.0 References

- Anon., n.d. *Algoworks*. [Online]  
Available at: <https://www.algoworks.com/blog/why-use-expressjs-over-nodejs-for-server-side-coding/>  
[Accessed 20 04 2020].
- Anon., n.d. *tutorialspoint*. [Online]  
Available at: [https://www.tutorialspoint.com/nodejs/nodejs\\_introduction.htm](https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm)  
[Accessed 20 04 2020].
- Anon., n.d. *Vue.js*. [Online]  
Available at: <https://vuejs.org/v2/guide/>  
[Accessed 20 04 2020].
- Anon., n.d. *wikipedia*. [Online]  
Available at: <https://en.wikipedia.org/wiki/Node.js>  
[Accessed 19 04 2020].
- Anon., n.d. *wikipedia*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Express.js#cite\\_note-4](https://en.wikipedia.org/wiki/Express.js#cite_note-4)  
[Accessed 19 04 2020].
- Anon., n.d. *wikipedia-MySQL*. [Online]  
Available at: <https://en.wikipedia.org/wiki/MySQL#Overview>  
[Accessed 18 04 2020].
- Anon., n.d. *wikipedia-Vue*. [Online]  
Available at: <https://en.wikipedia.org/wiki/Vue.js>  
[Accessed 18 04 2020].