|  |  |  |
| --- | --- | --- |
| Module Code: **PUSL3111** | Module Name: **API Software Development** | |
| Coursework Title: **API development for IPT** | | |
| Deadline Date**: 7th of May 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 [www.plymouth.ac.uk/studenthandbook](http://www.plymouth.ac.uk/studenthandbook). | | |
| 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.  **D. P. I. Gayantha 10638279**  **K. H. M. Kavinda Maduranga** **10638236**  **H. P. W. N. Hasaranga** **10638200**  **P. C. N. Dissanayake** **10638271**  **S. S. Bandara** **10638294**  **G. H. Dhananjana 10638313**  ***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.***  Signed on behalf of the group: C:\Users\DigitalForce™\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Sign.jpg | | |
| Individual assignment: ***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.***  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 ~~used~~/not used translation software.  If used, please state name of software………………………………………………………………… | | |
| **Overall mark \_\_\_\_\_% Assessors Initials \_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_** | | |

|  |
| --- |
| Name: **Mr.** **Gayantha, D. P. I.** |
| Student Reference Number: **10638279** |



API SOFTWARE DEVELOPMENT| ASSIGNMENT

**API SOFTWARE DEVELOPMENT** DOCUMENTATION

GROUP No: 18

**ACKNOLEDGEMENT**

I would like to express my special thanks and gratitude to my Distributed Systems module lecturer Mr. Rasika Ranaweera who gave us the priceless opportunity to do this project on various topics, which also helped us in doing a lot of Research and we came to know and have to study so many new things I am really thankful to them and the Plymouth University.

Finally, I am grateful to all of those with whom I have had the pleasure to work during this project within the limited time frame. Each of the group members committed necessary tasks within this time period.

Best Regards,

D. P. I. Gayantha (Group Leader of Group 18 (DLE Group list))

TABLE OF CONTENTS

**Introduction5**

**System Architecture5**

**Documentation7**

**Technical Guide17**

**Group Members19**

**Appendix25**

INTRODUCTION

NSBM Green University Town is the first-ever green university in South Asia and more than 8000 thousand undergraduates are following their degree programs. But they haven’t proper understanding of experience in industrial life and how to thrive in the industry.

Then NSBM is introducing a program Dubbed "IPT" is a program that allows students to make a quick and simple transition to industrial life after NSBM studies. This is intended to have the first-hand experience of business life and understanding of how to succeed in the industry, with the expertise of seasoned these experts. NSBM is now working to set up an IPT website to show the experience. Students should show their credentials and professionals can see them and potentially hire them.

SYSTEM ARCHITECTURE

System has two Front-End clients. An Angular Web Application and an Android native application. Both front-end components utilize the full capability of the PHP RESTful API.

Following diagram shows the system Architecture of the ‘IPT Program’.

A screenshot of a cell phone

Description automatically generated

Figure 1: System Architecture Diagram

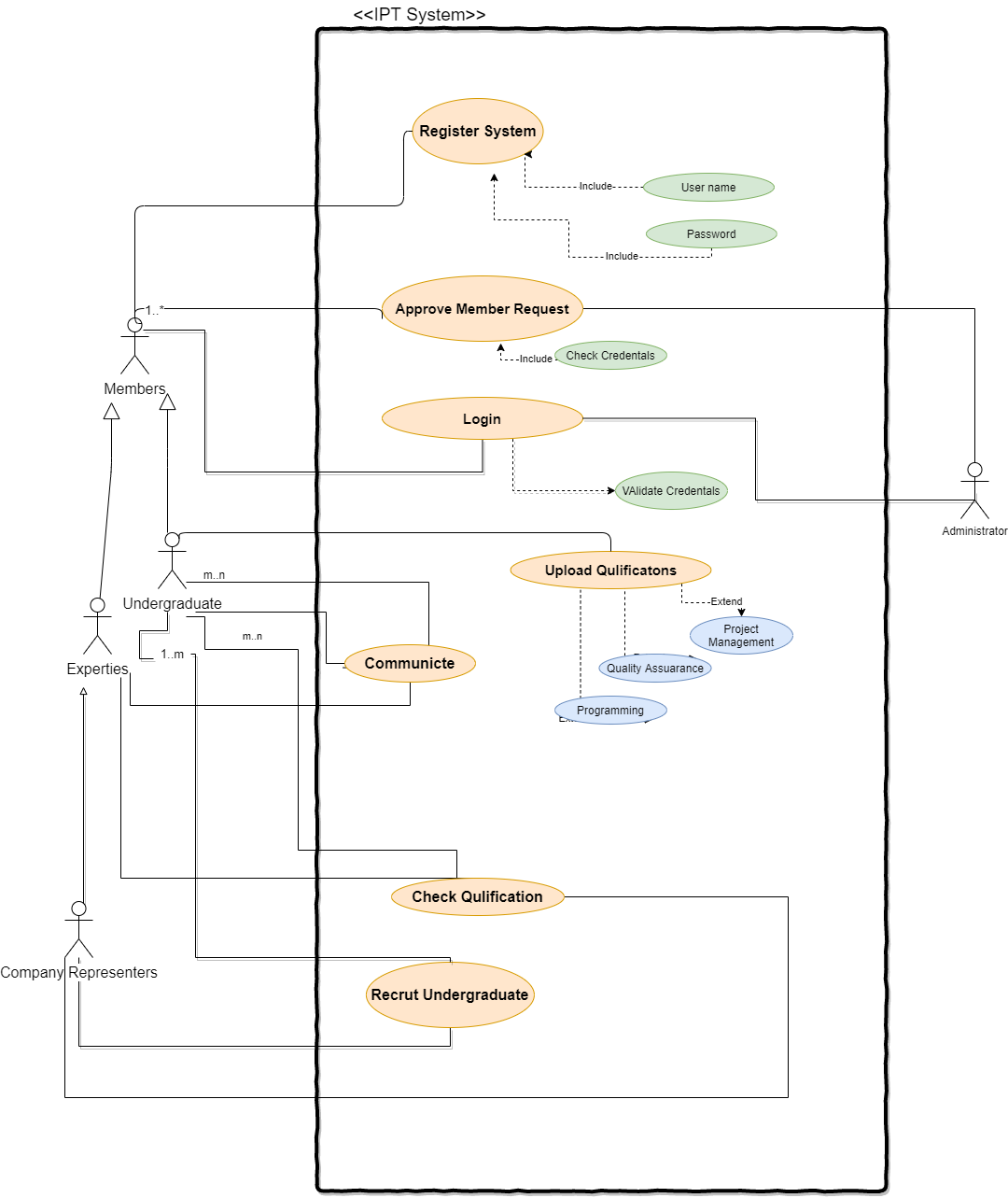


Figure 2: Use Case Diagram

DOCUMENTATION

IPT services are not limited to its own website. It provides several API based services to both undergraduates and experts. To use those services, users should have exceptional knowledge on certain areas. Therefor this documentation section will provide step by step instructions with necessary requirements to achieve service embedding process. Following instructions will serve promising amount of knowledge if wish to request our services.

# INSTALLATION AND REQUIREMENTS

Service installation process is a straightforward process and It’s an essential process to achieve afterward process. In order to implement any of our services, basically you should familiar with its supportive development technologies and its runnable environments. It can be run on any browser which supports JavaScript and data interchange formats such as Json and XML which are basically supported by every existing web browser. Our service supports following development technologies.

Web Development Languages

* + JavaScript and any JavaScript runtime environment such as Node JS, Vue JS, React JS
  + Java
  + ASP.NET

Mobile Application Development Technologies

* + Android
  + Flutter

As other API services, this service doesn’t require an API key or any user credential. It’s a free service indeed. In case of that it doesn’t require personal credential to access all the services that we provide.

# CLARIFICATION ON API DIALING

In order to make each data request you should call HTTP Request on your website by dialing a specified URL. Such URL has identical characteristics that you should know about. Following sections will clarify more about those characteristics.

Mainly given URL can divide into few sections as below.



Domain - A section that shows domain web site which is our IPT official web site.

Path - A section specify web path.

Parameters - Sections which allow to filter / query requested data.

# Content

Followings are few services we provide through this API.

GET all users information.

POST user’s information / Register.

GET specific user’s information.

UPDATE user’s information remotely.

DELETE user’s information remotely.

# GET all users information.

This specified URL will provide all users information remotely. Typically, members details page is looks like below snapshot and its URL as below.

http://localhost:5001/members/

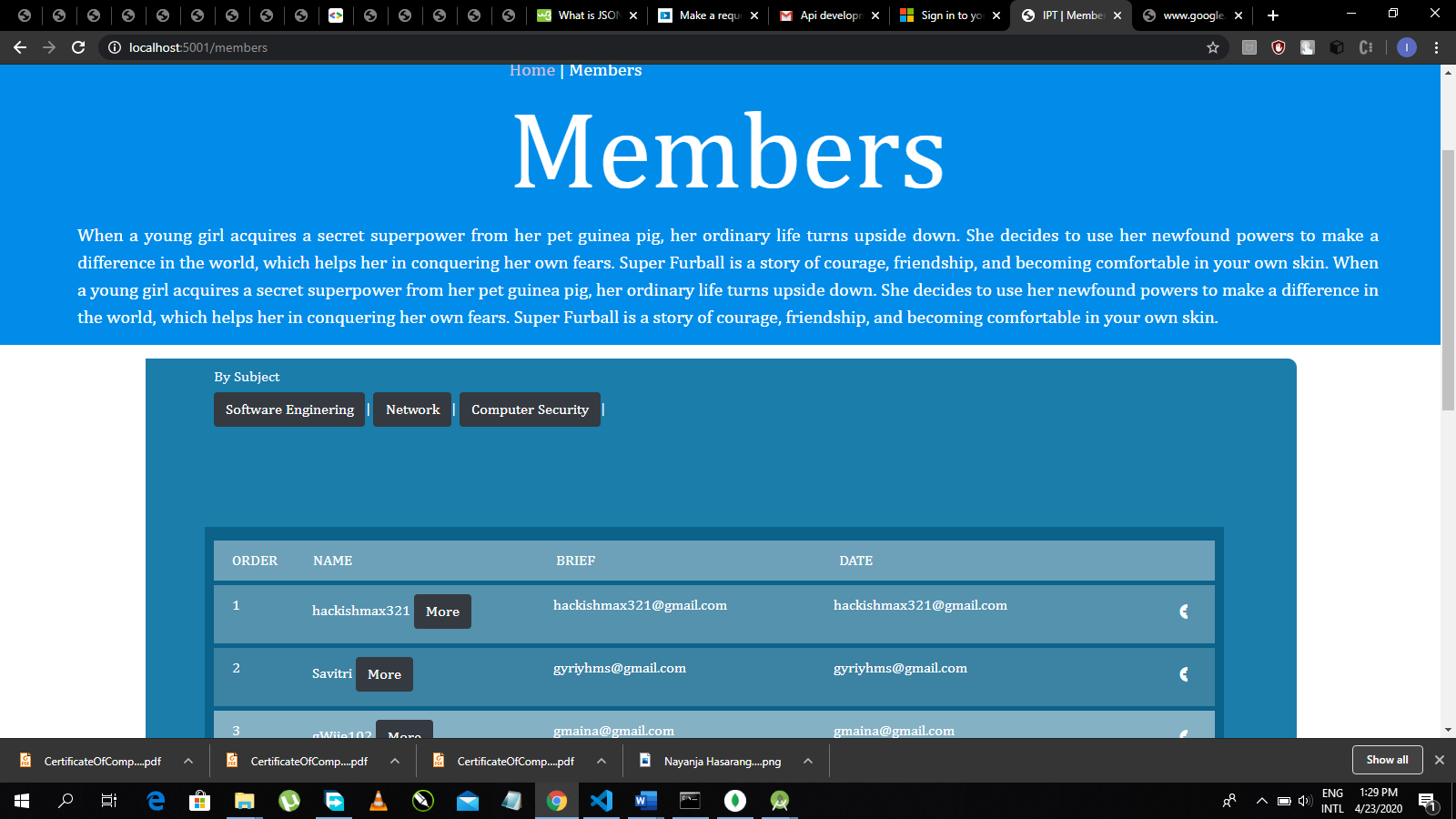


Figure 3: All registered Member's Details

Coping above URL and adding **data/json** you can retrieve those data. To get this service following guidelines should be proven.

http://localhost:5001/members/data/json

This URL will output following JSON string as the result.

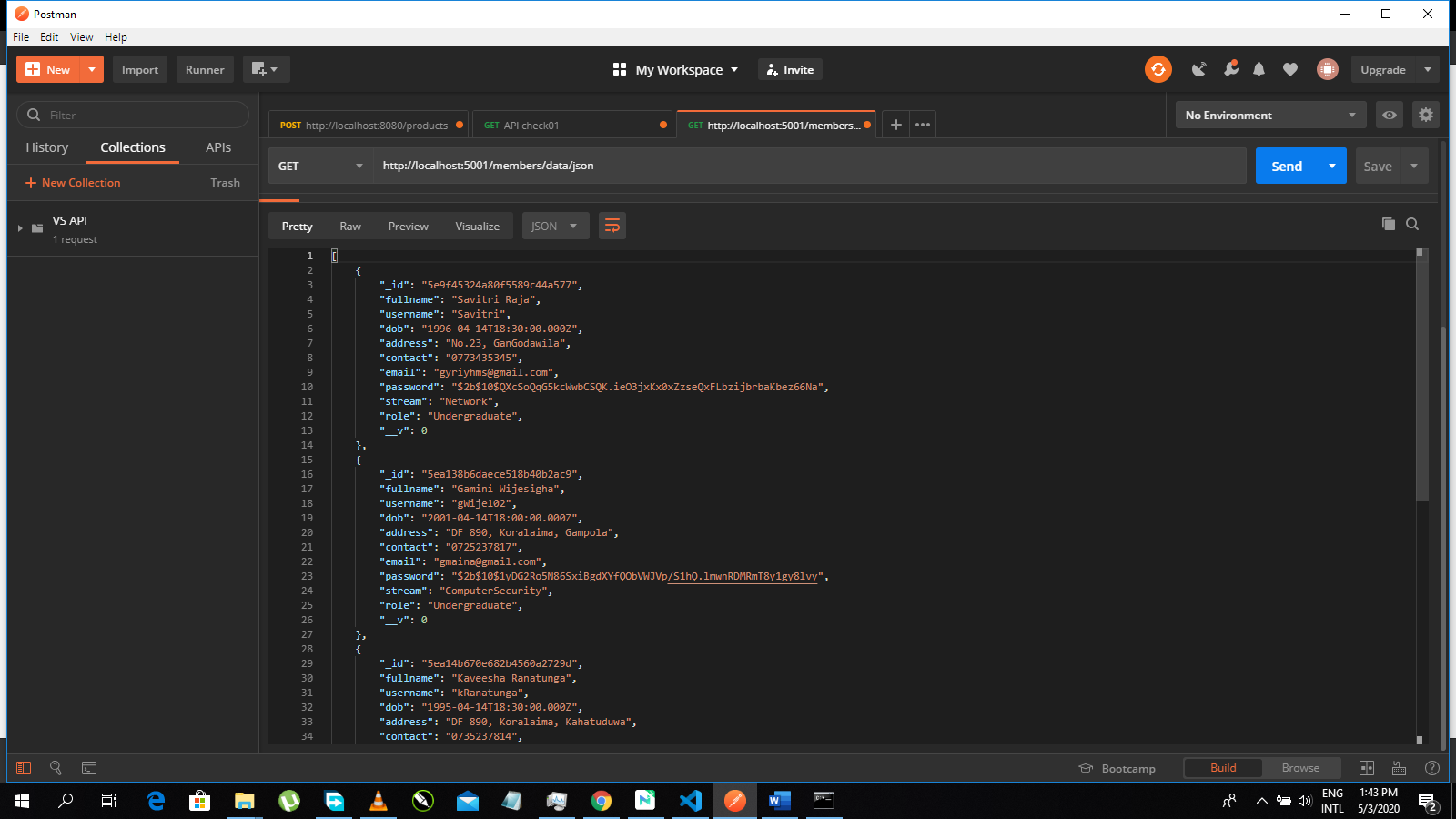


Figure 4: Json Output

http://localhost:5001/members/data/xml

This URL will output following XML block as the result.

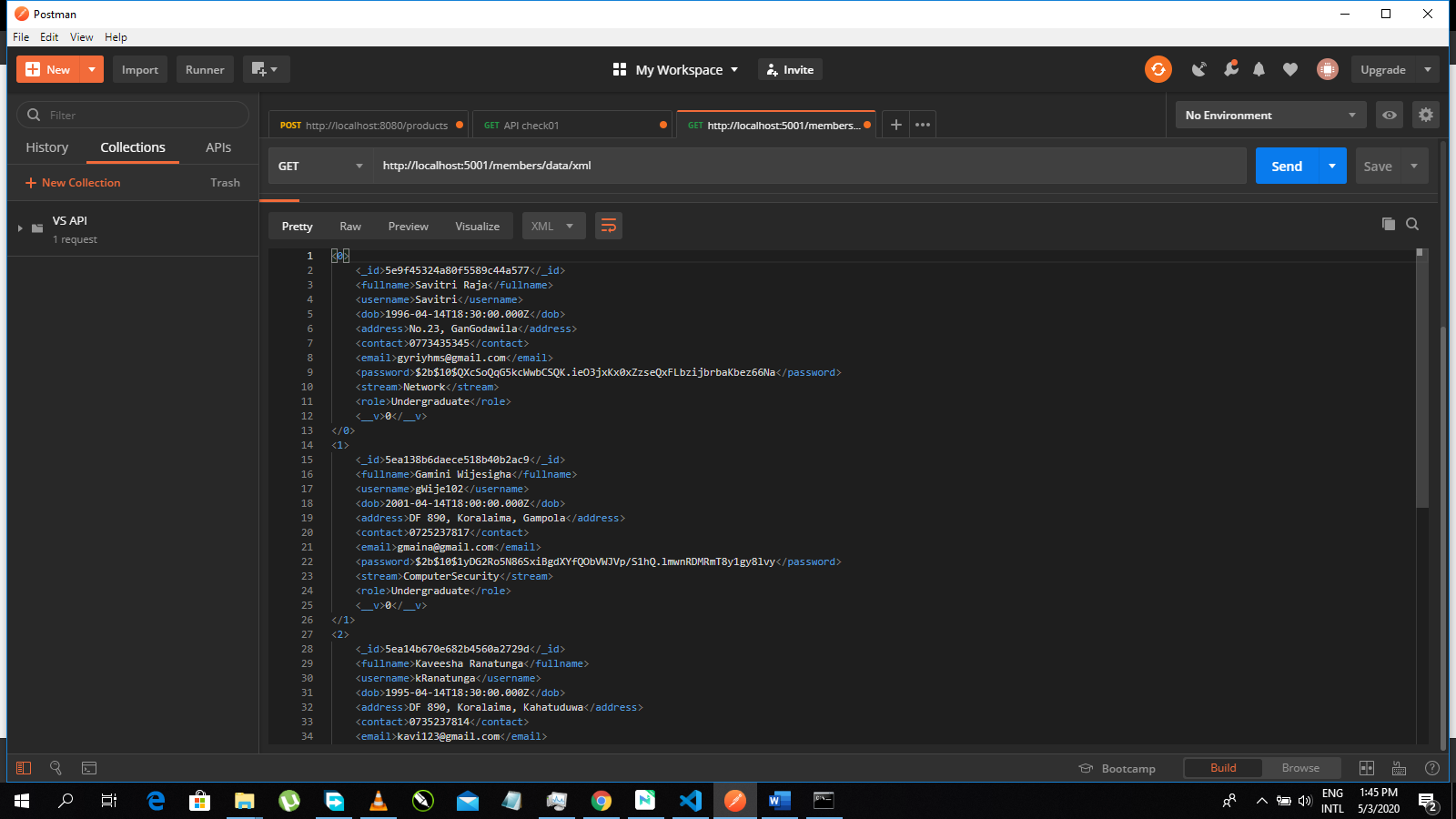


Figure 5: XML output

# GET users’ information based on Program category.

This specified URL will provide users information which is specified by Degree Program. To get this service following guidelines should be proven.

As the normal manner this URL will output json data of related category.

[http://localhost:5001/[CATEGORY]/member](http://localhost:5001/%5bCATEGORY%5d/member)s/data/json

e.g.: Category = Software Engineering

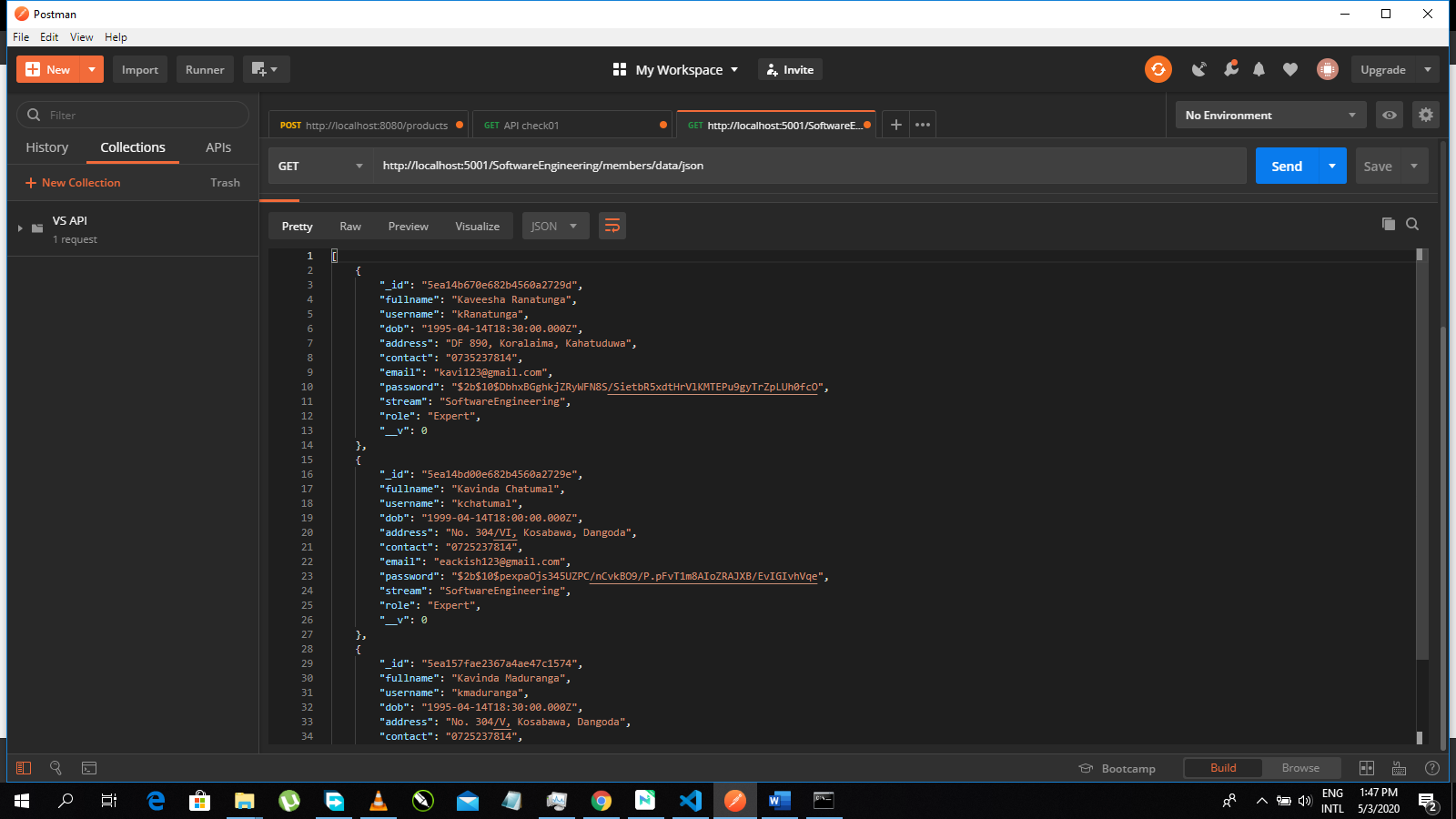


Figure 6: Get category data

As the normal manner this URL will output xml data of related category.

[http://localhost:5001/[CATEGORY]/member](http://localhost:5001/%5bCATEGORY%5d/member)s/data/xml

e.g.: Category = Network

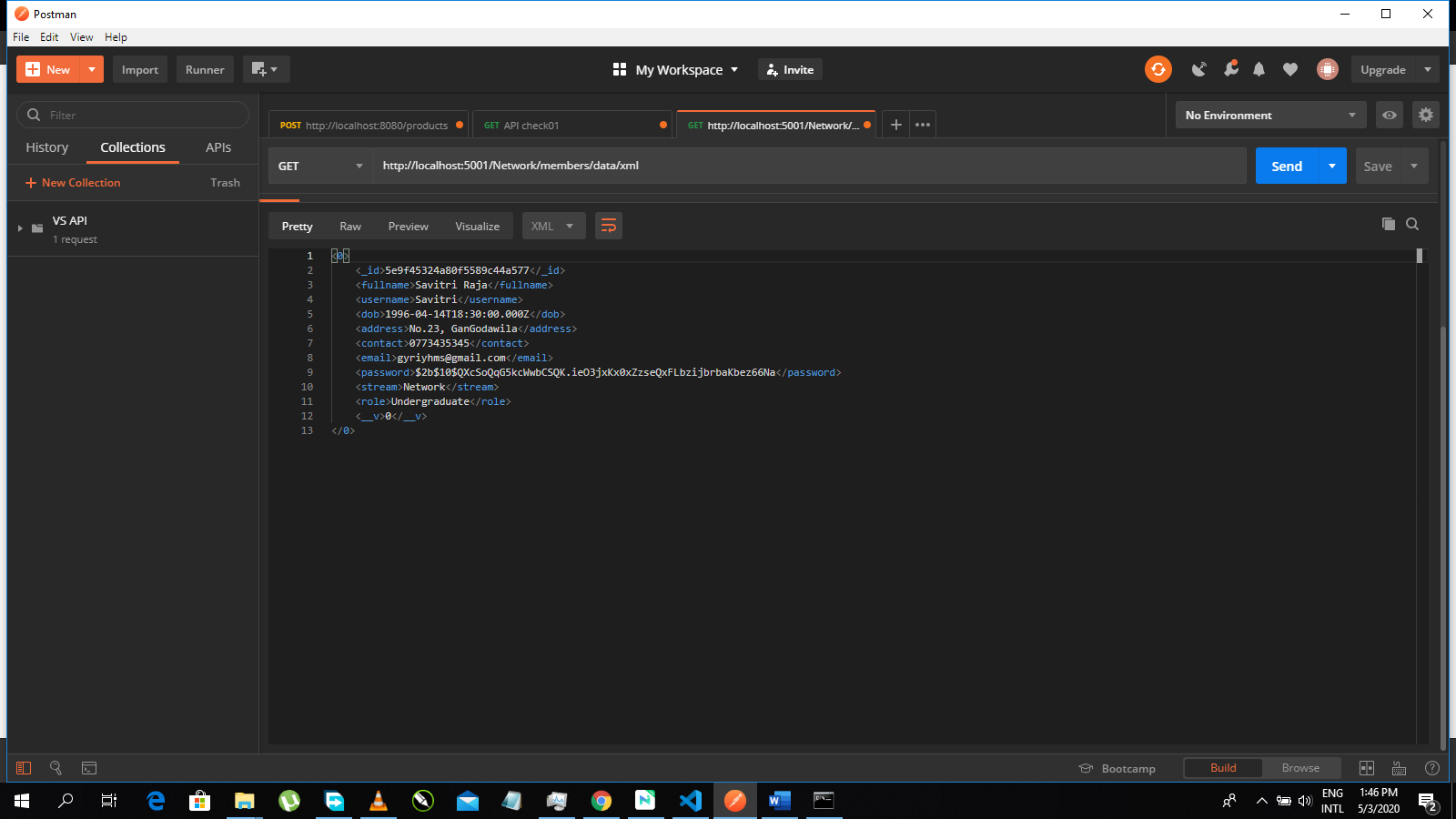


Figure 7: Get category data in XML

http://localhost:5001/[CATEGORY]/members

e.g.: Category = Software Engineering

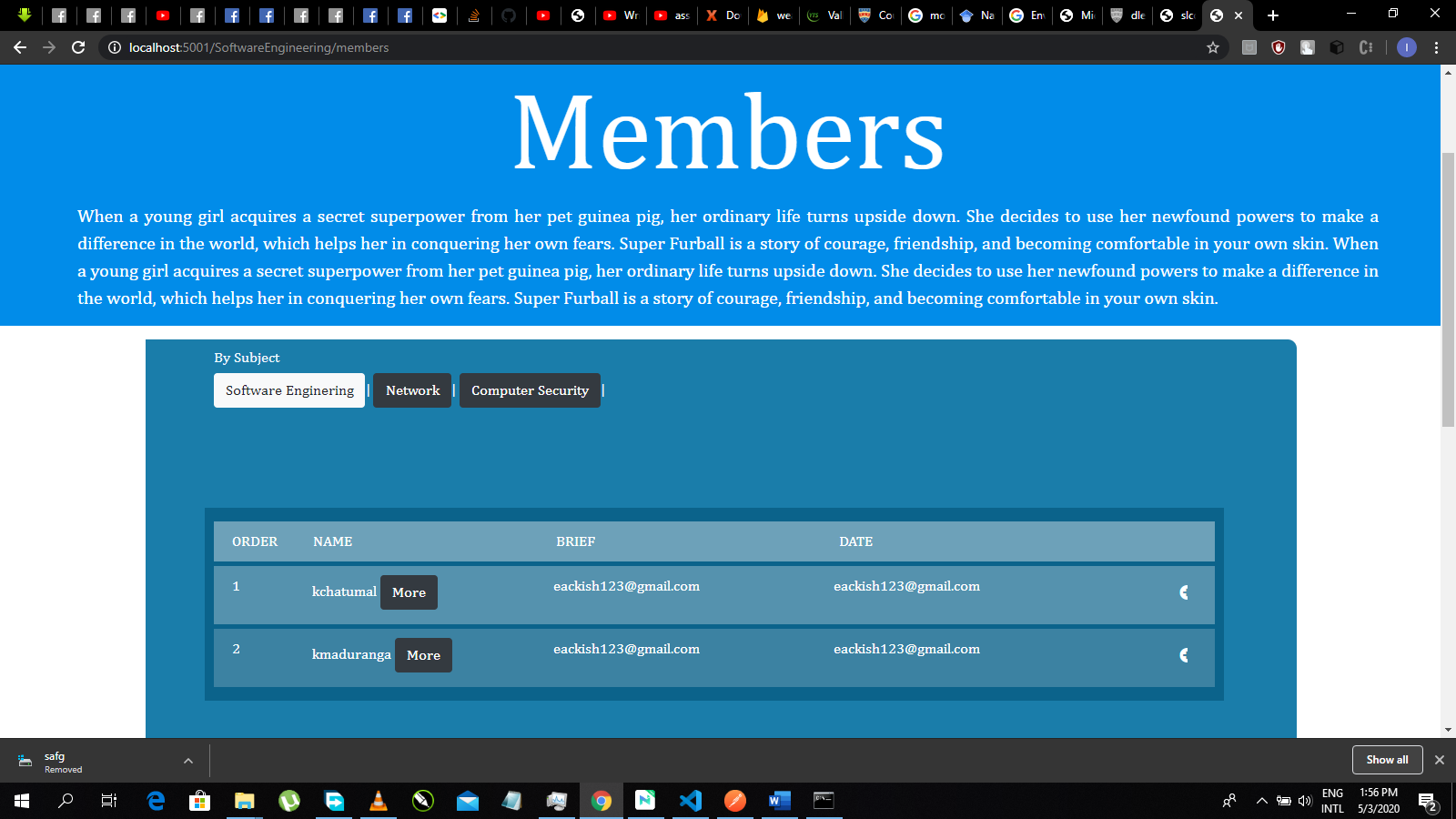


Figure 8: Get category data for GUI

e.g.: Category = Network

# 

Figure 9: Get category data for GUI

e.g.: Category = Computer Security

# 

Figure 10: Get category data for GUI

# GET selected user’s information.

This specified URL will provide single users information which is specified by Username. To get this service following guidelines should be proven.

As the normal manner you will get specified user’s account with this url

http://localhost:5001/members/[USERNAME]/

e.g.: Username = Savitri

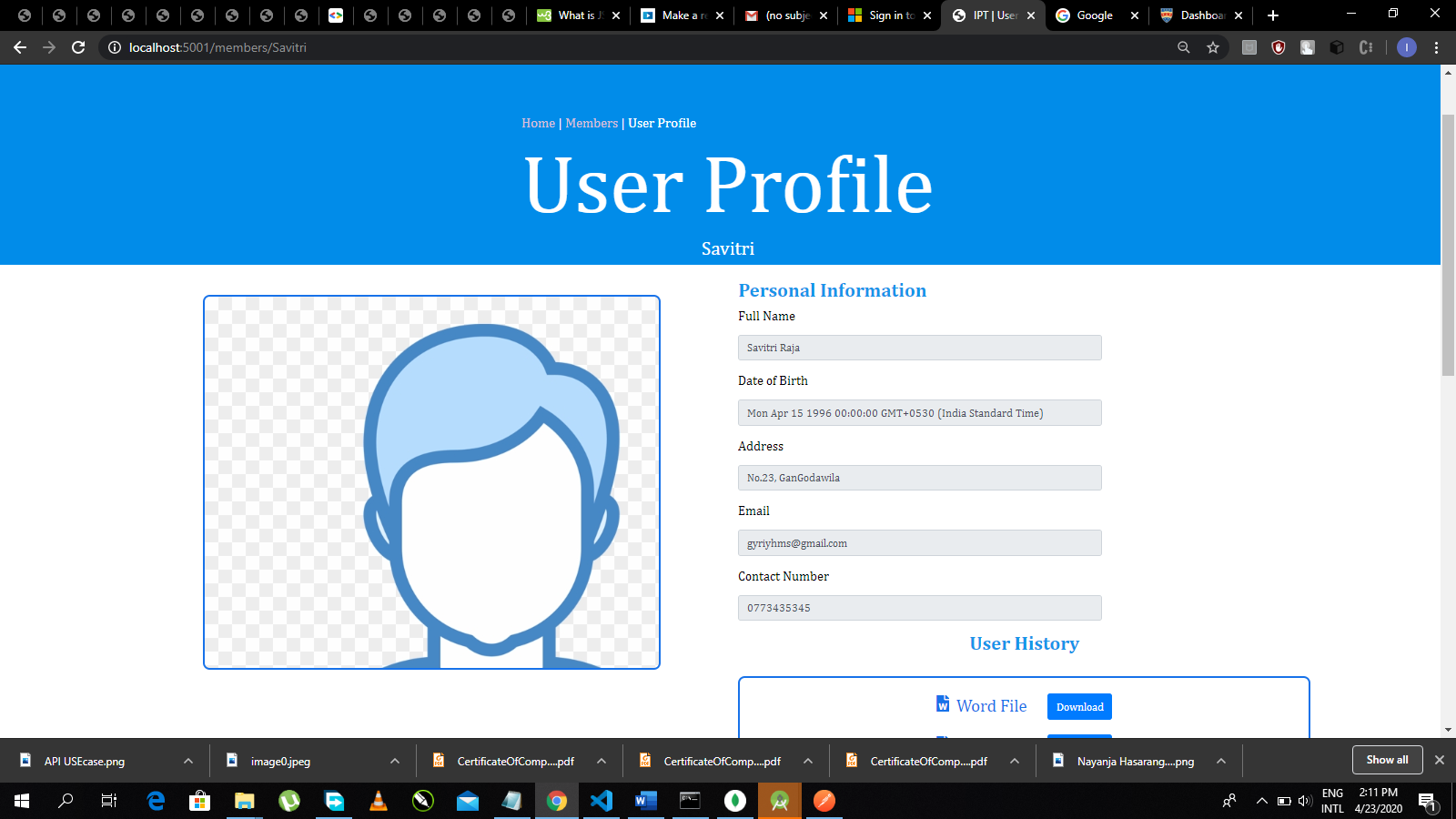


Figure 11: User profile of another User

http://localhost:5001/members/[USERNAME]/data/json

This URL will output following JSON string as the result.

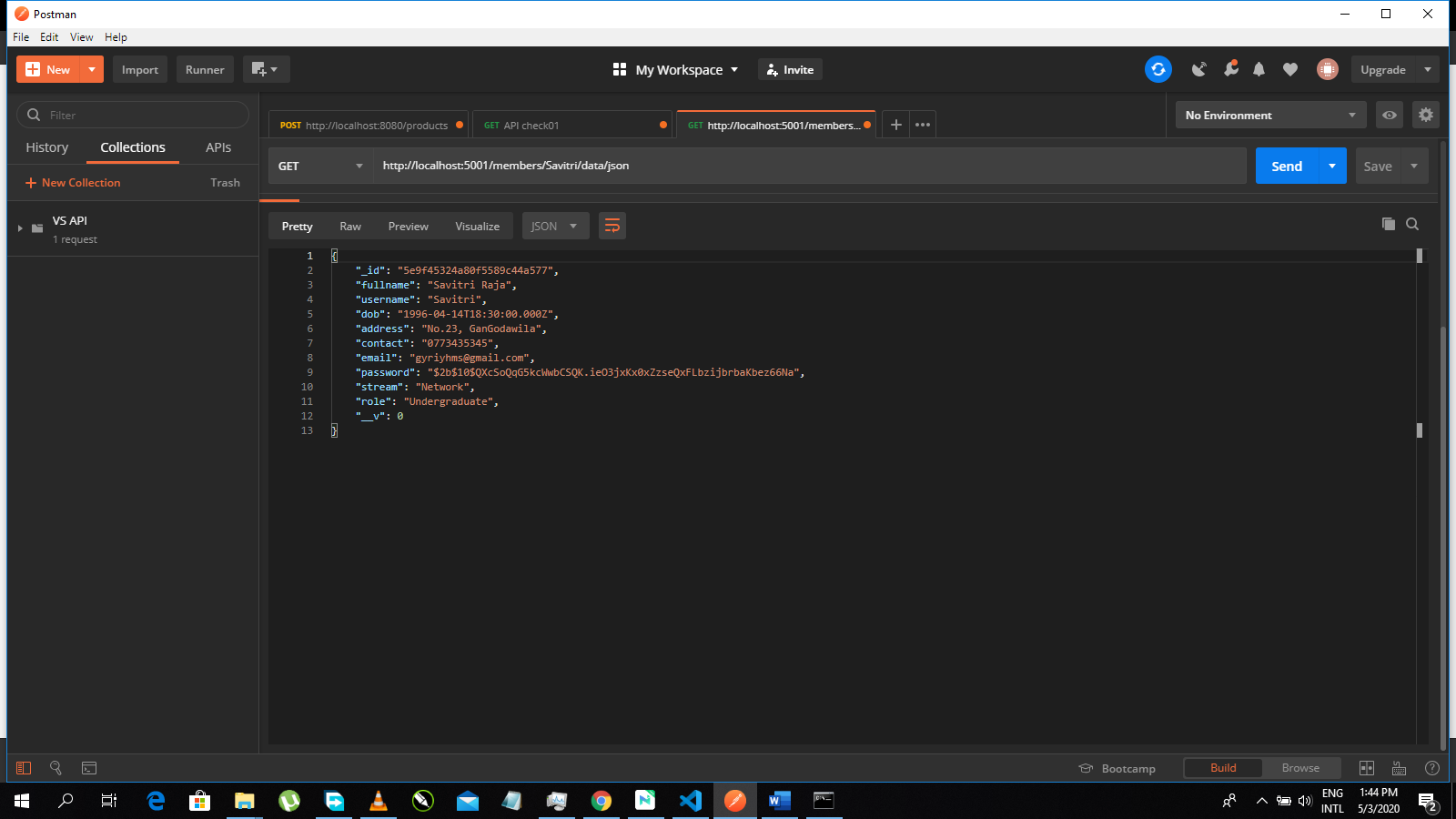


Figure 12: Json Output

http://localhost:5001/members/[USERNAME]/data/xml

This URL will output following XML as the result.

# 

Figure 13: XML Output

# POST user’s information.

This will allow external parties specially undergraduates and industries’ experts to register into IPT official web site and its thoroughly for personal use and it will not provide data for external use as above options.

http://localhost:5001/members/register

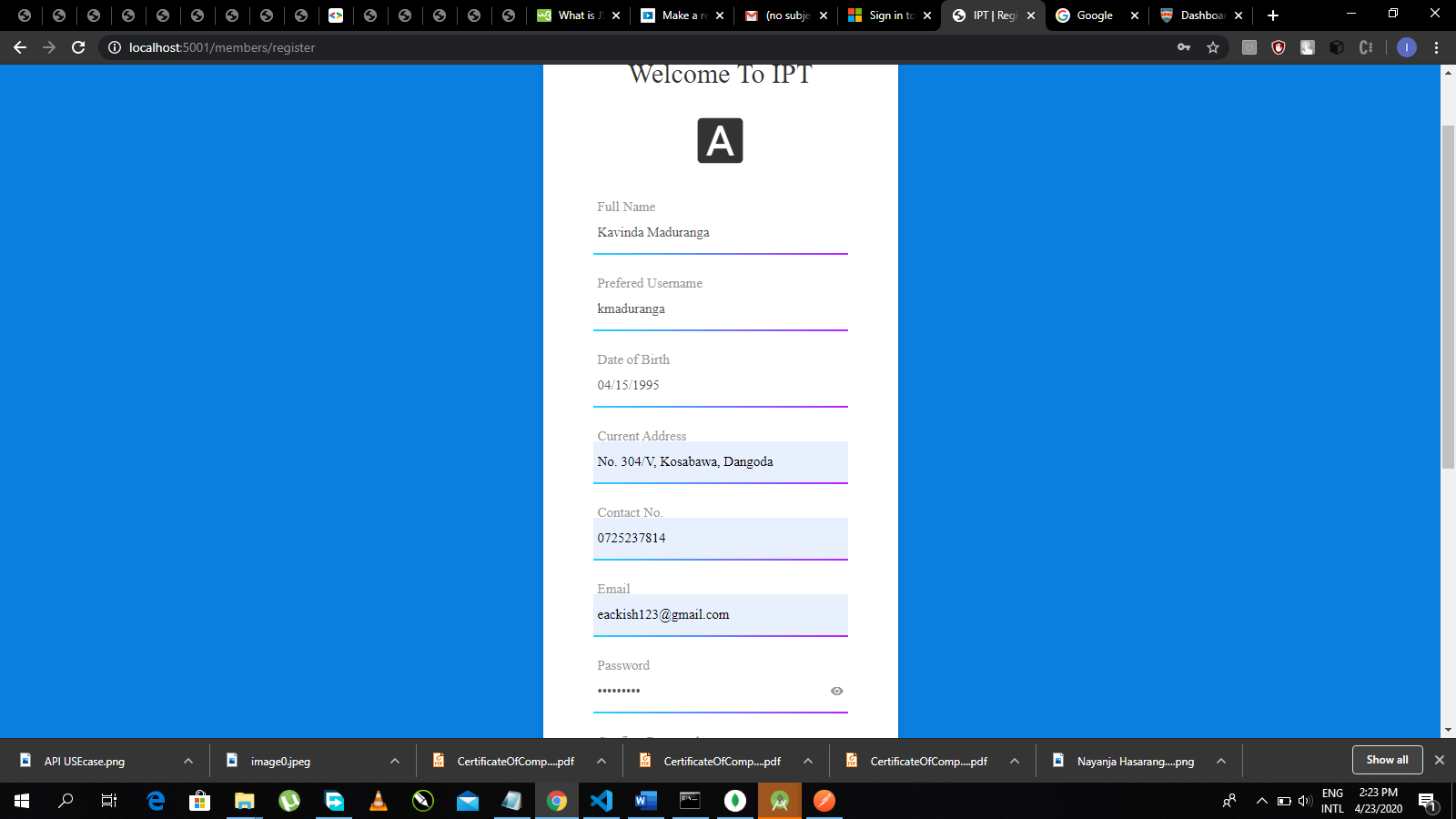


Figure 14: Registration Page part 01

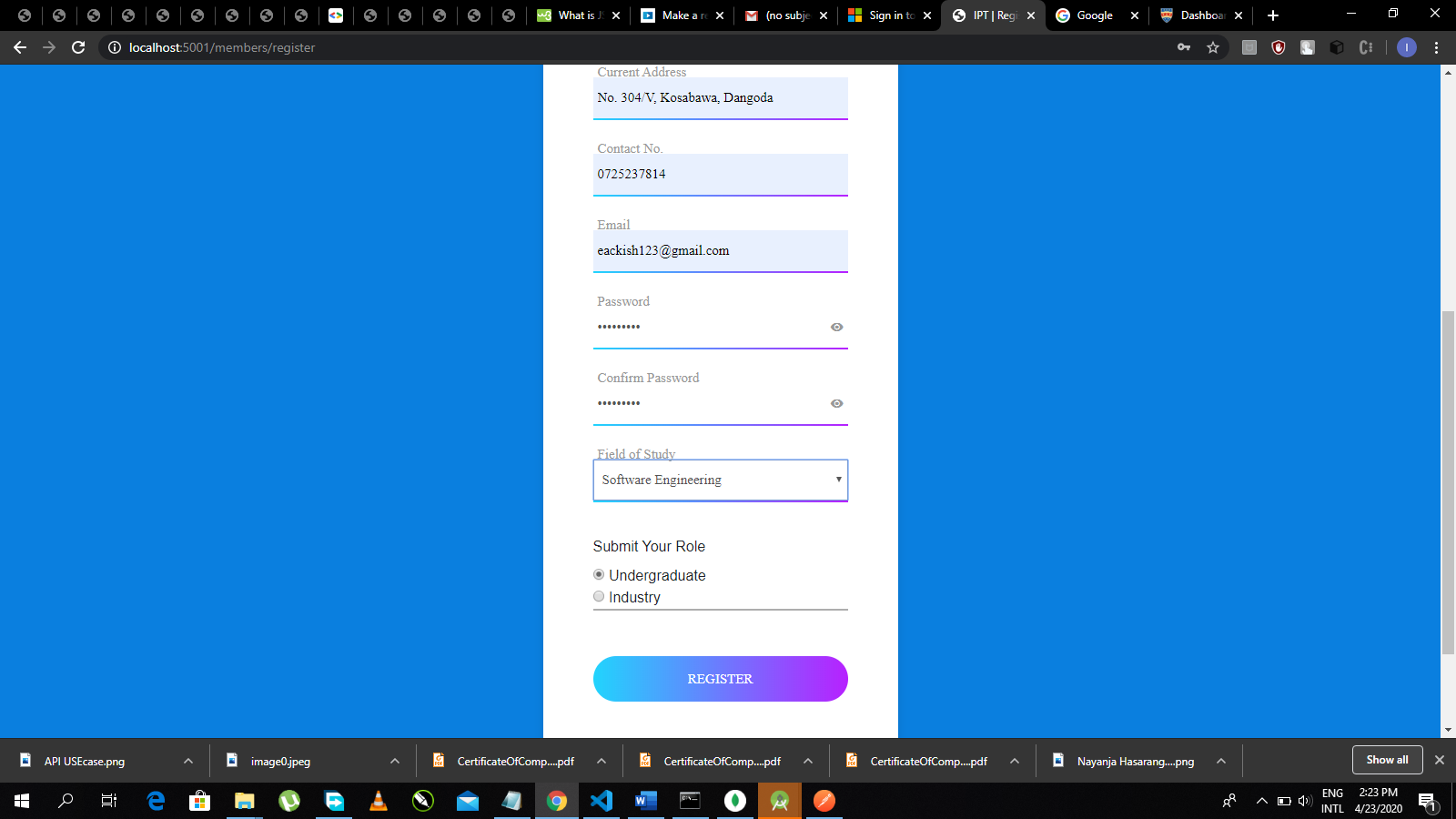


Figure 15: Registration Page part02

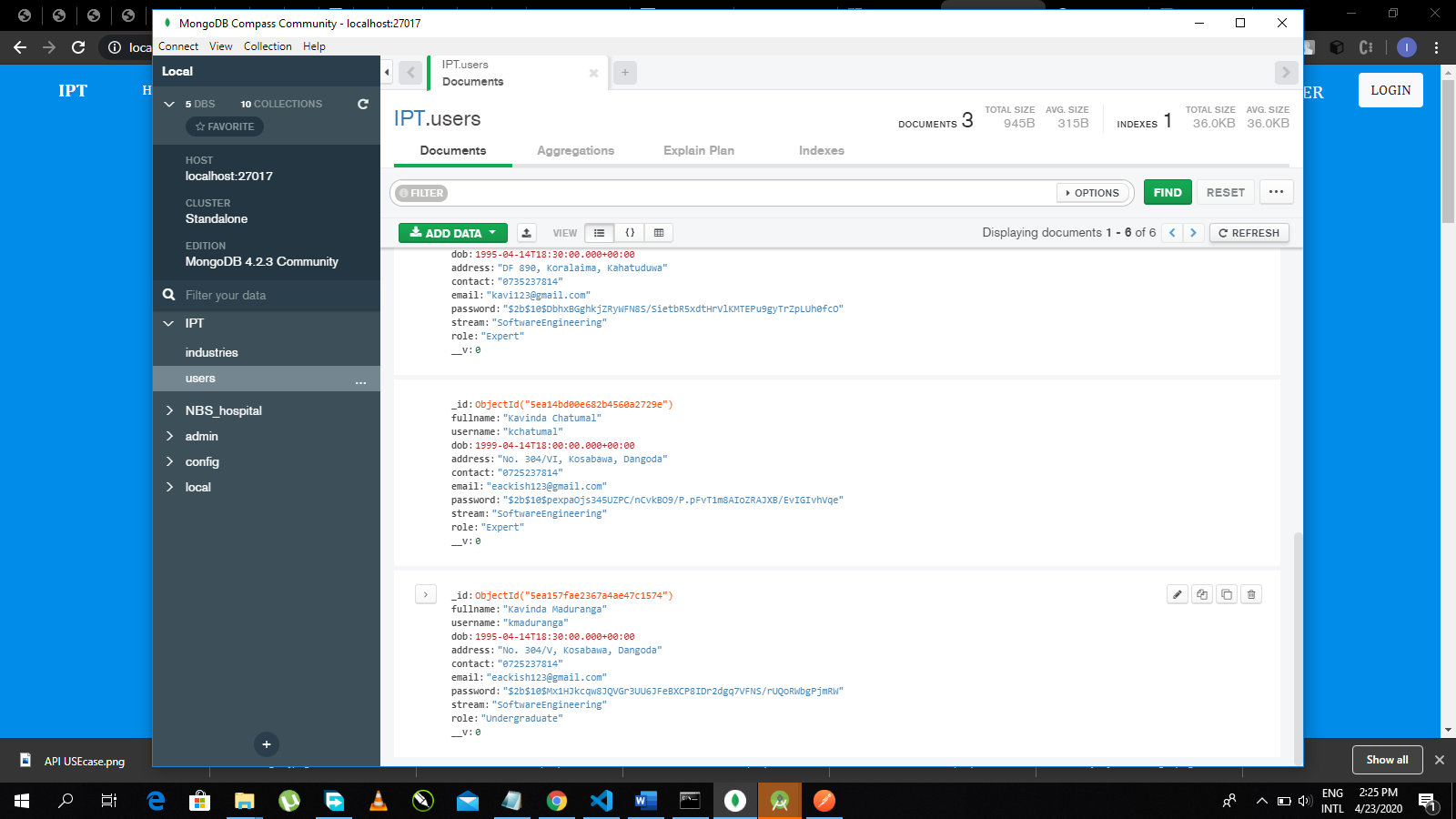


Figure 16: Changed Data in Mongo DB database

You can log into our official Web API using your personal credentials such as Username and Password. Following URL will give access to the Login page.

http://localhost:5001/members/login

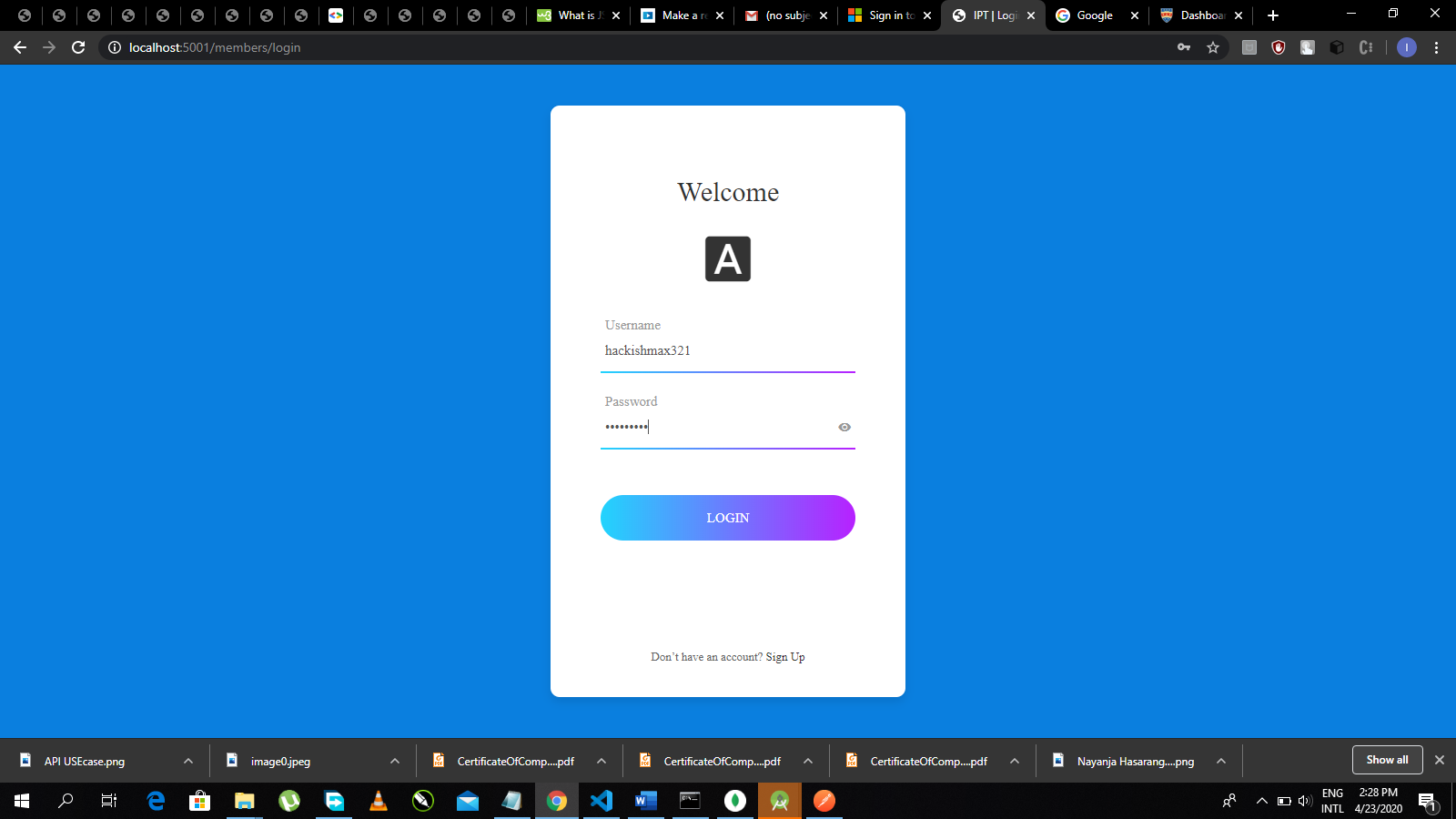


Figure 17: Login Page

# 

# UPDATE user’s information.

This will allow users to access his or her user profile settings and update particular personal data. Its URL can be recognized as below.

http://localhost:5001/ members/[USERNAME]/update/[COLUMN]



Figure 18: User Profile of Logged User

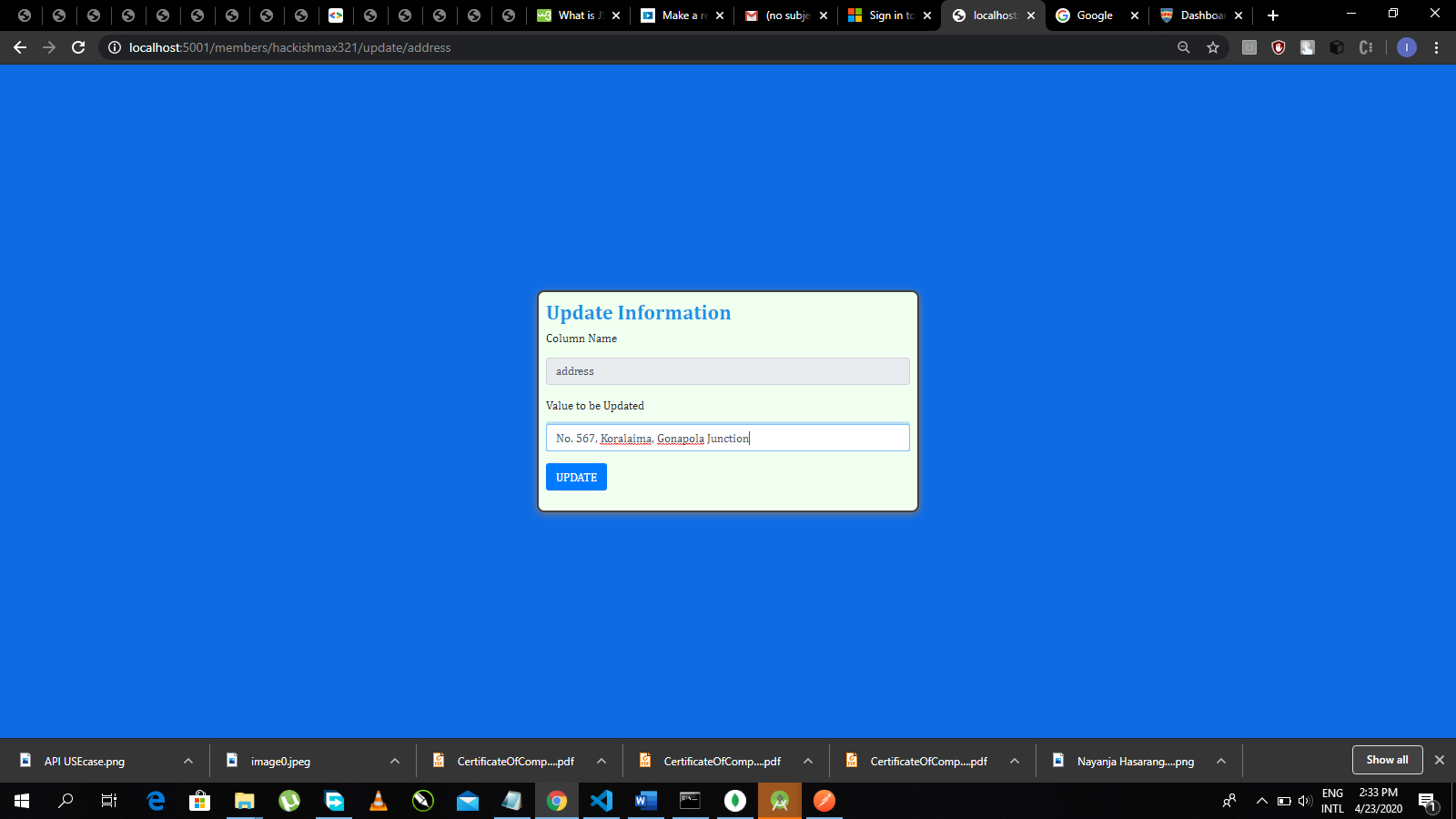


Figure 19: Editing Panel

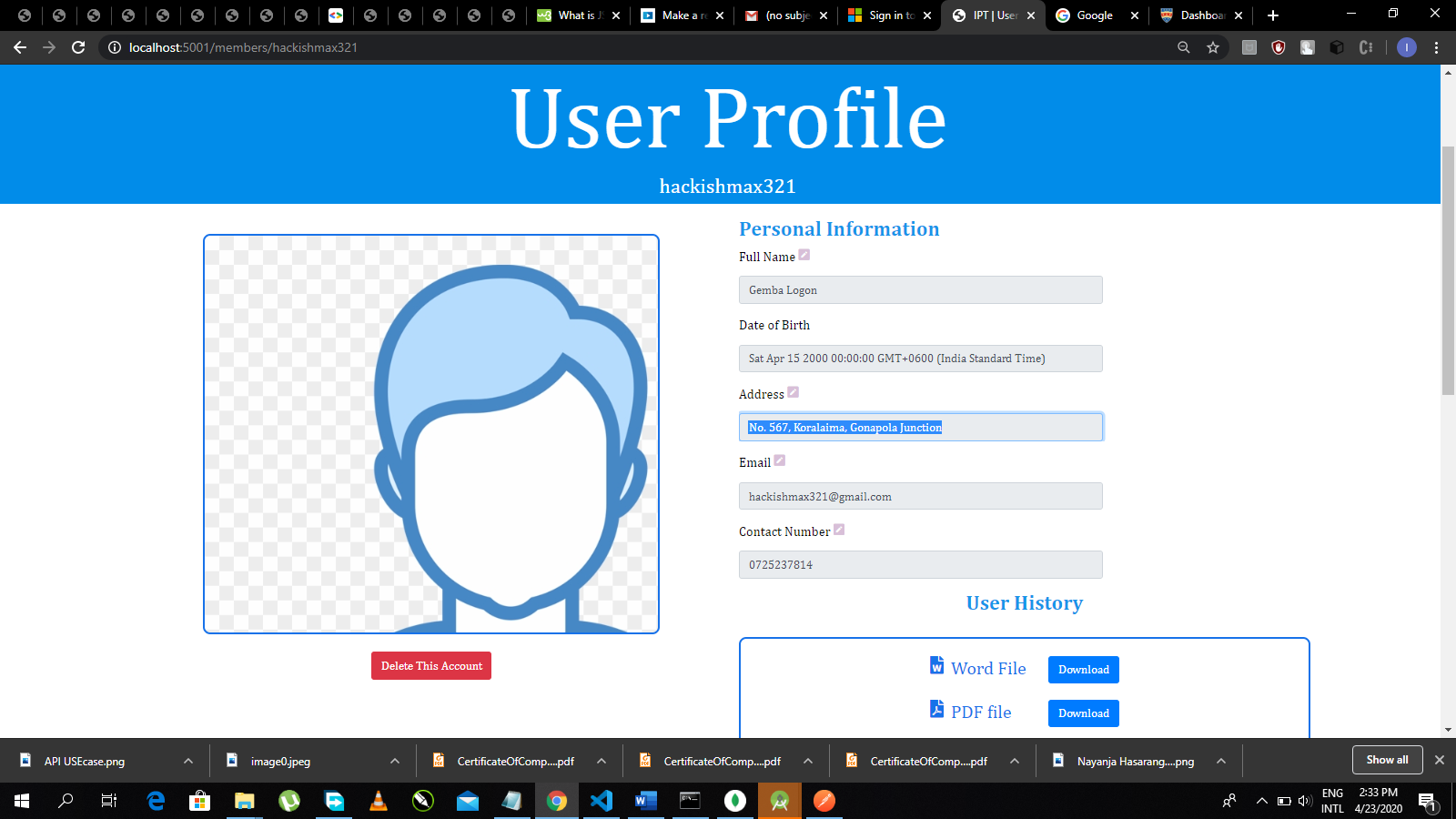


Figure 20: Updated Value

# DELETE user’s information.

Delete setting allow you to delete your account (not others). Also Administrator user will have authority to delete your account.

http://localhost:5001/ members/[USERNAME]/delete



Figure 21: Deleted button in authentic user's profile

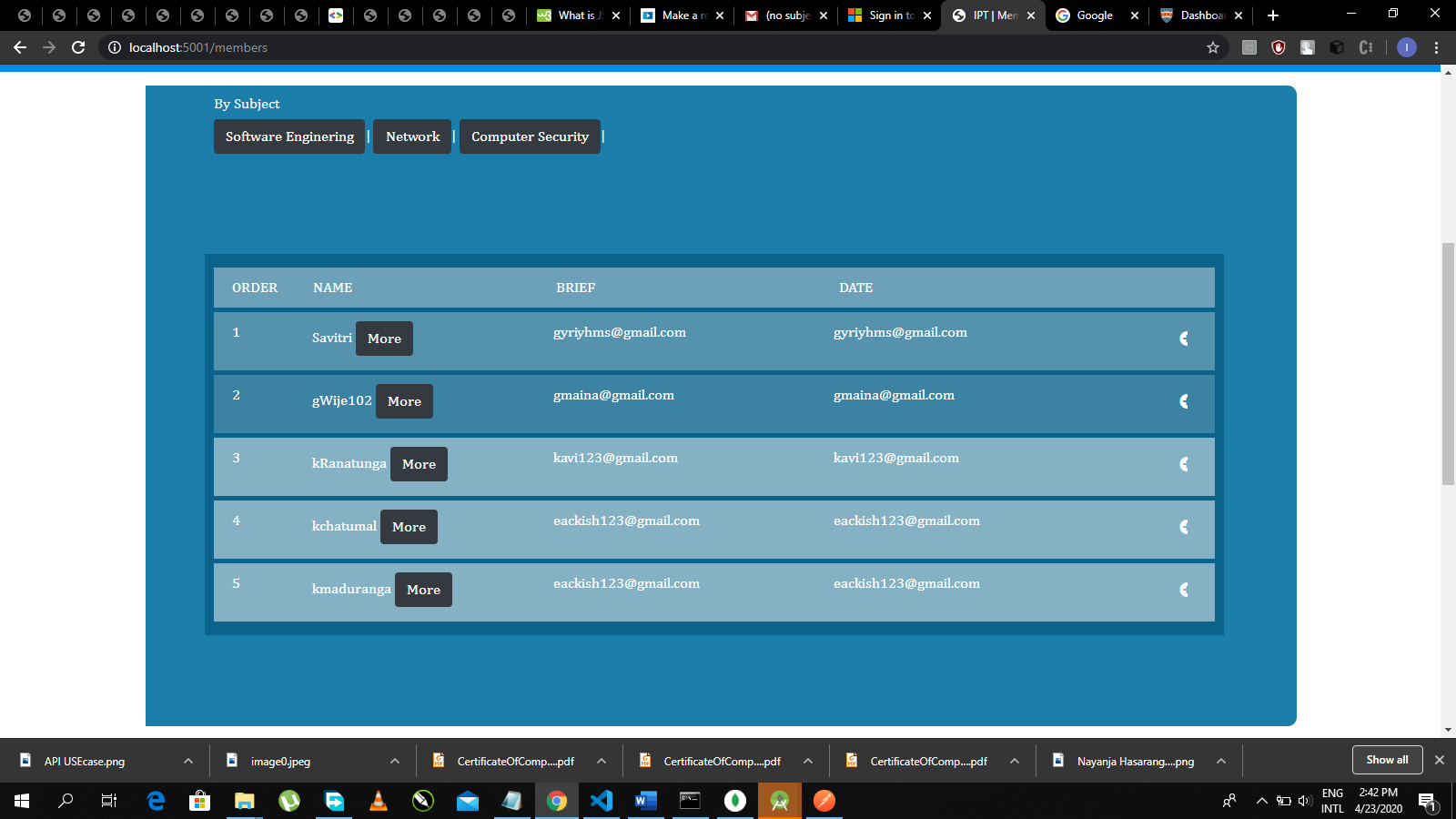


Figure 22: Updated User List after one User (hackishmax321) has deleted

# GET Industrial Data

This setting allows you to access industry page. you can also add industries as a verified user (expert).

http://localhost:5001/ industry

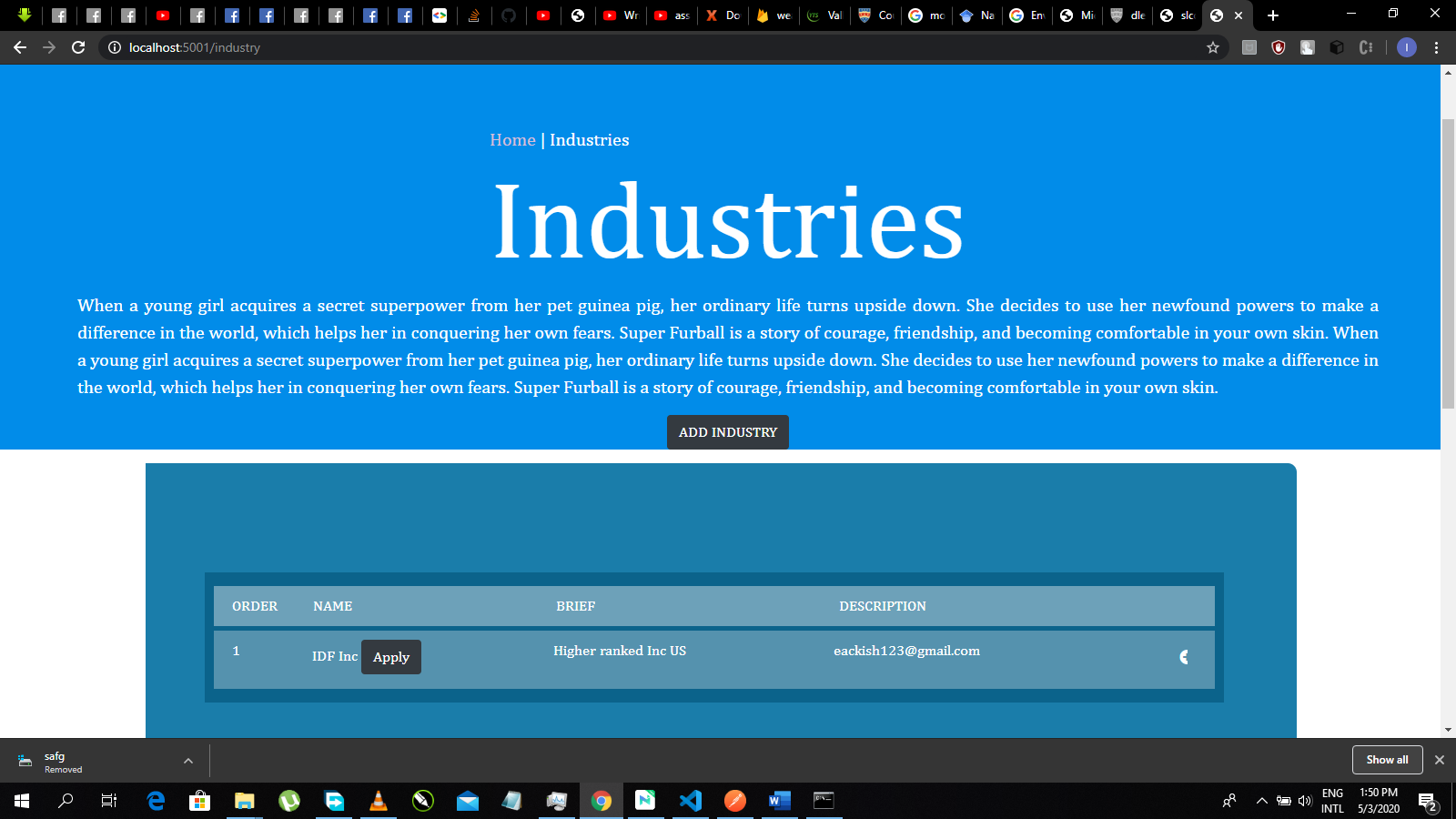


Figure 23: industries page

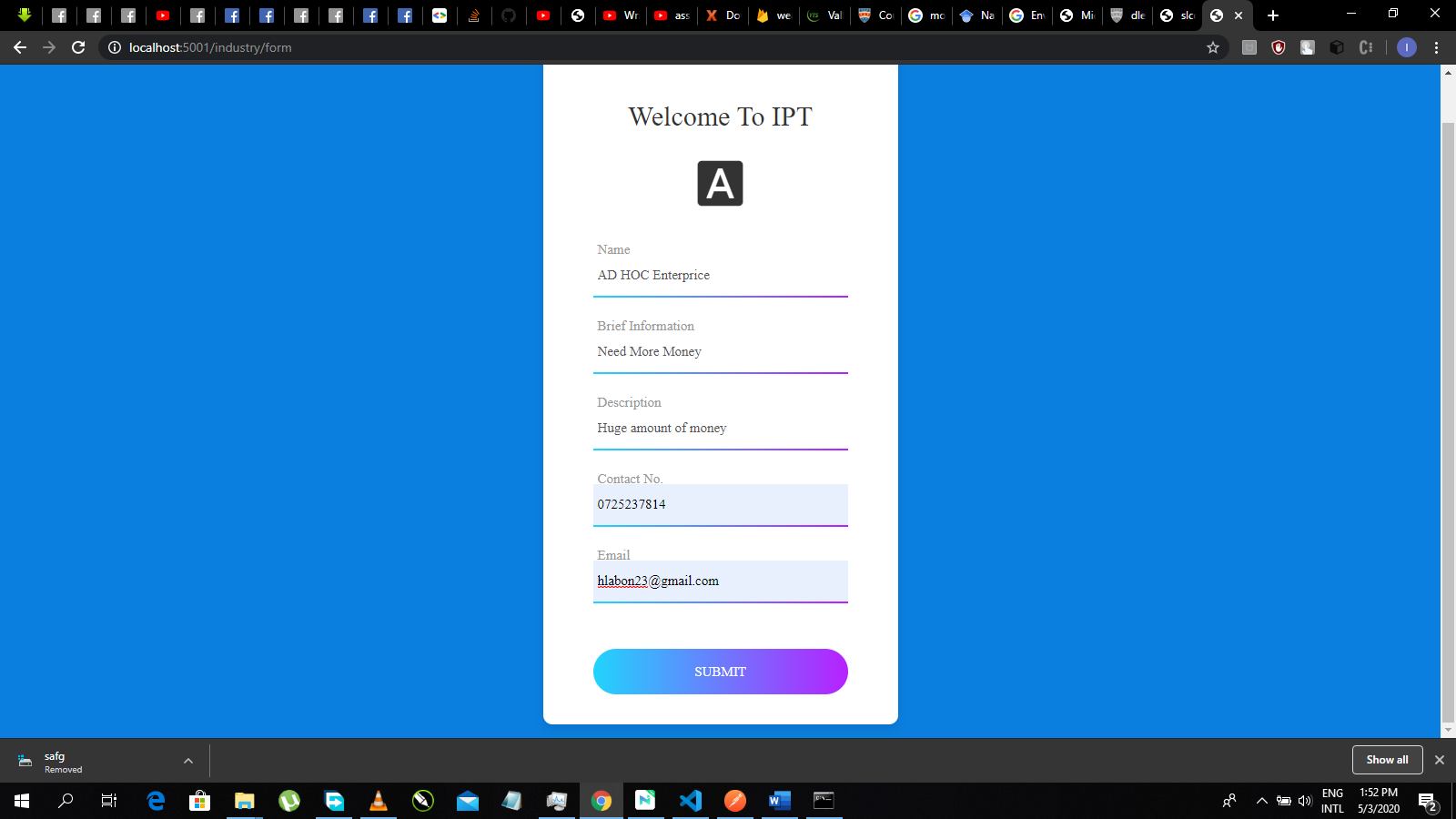


Figure 24: Industries form to add ne Industries

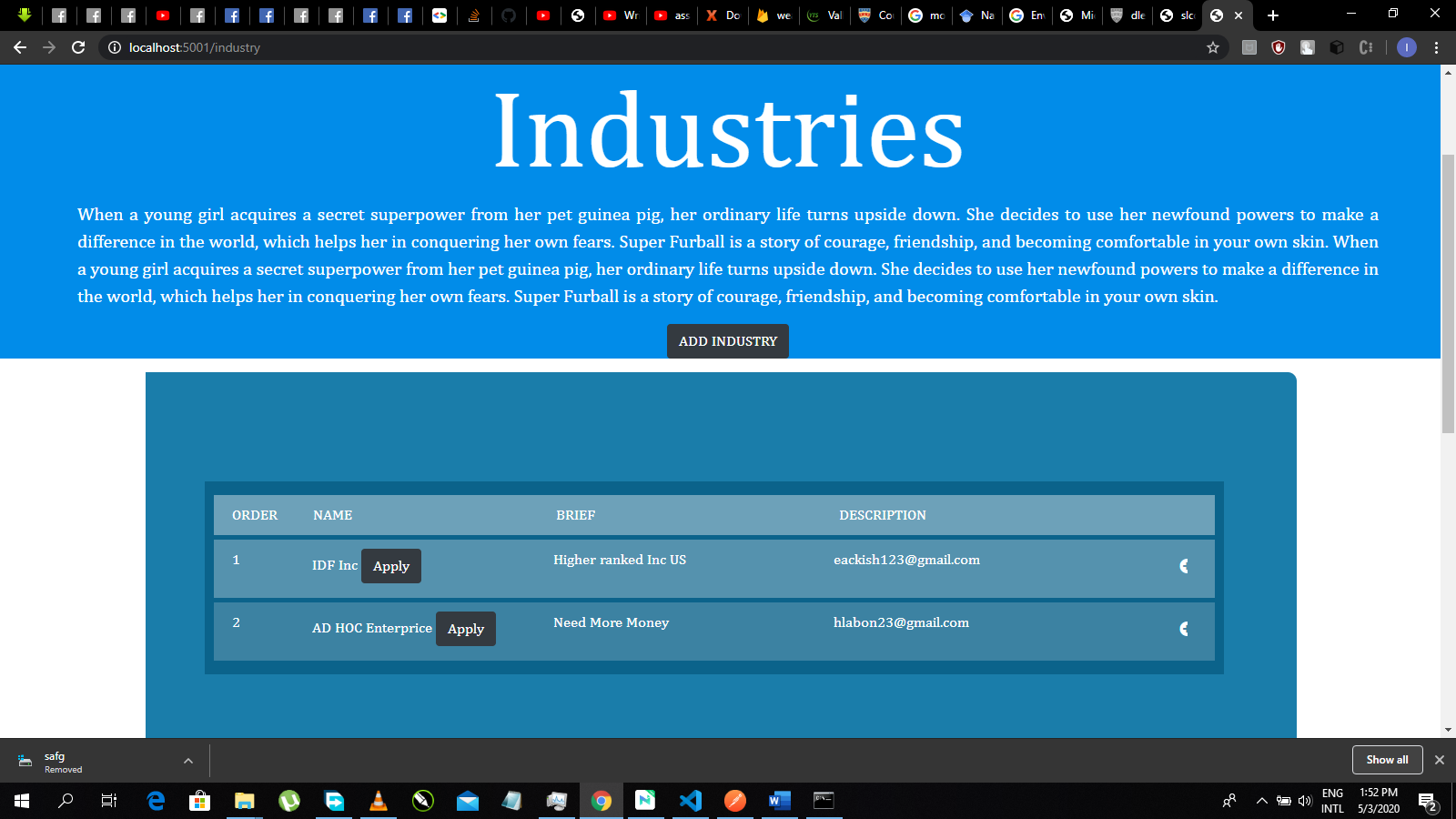


Figure 25: Newly added Industry

# RETRIVING user data in JavaScript

In any above way, you can access our services as below.



Figure 26: Extracted data from API displayed in another web site

Code should be as below.

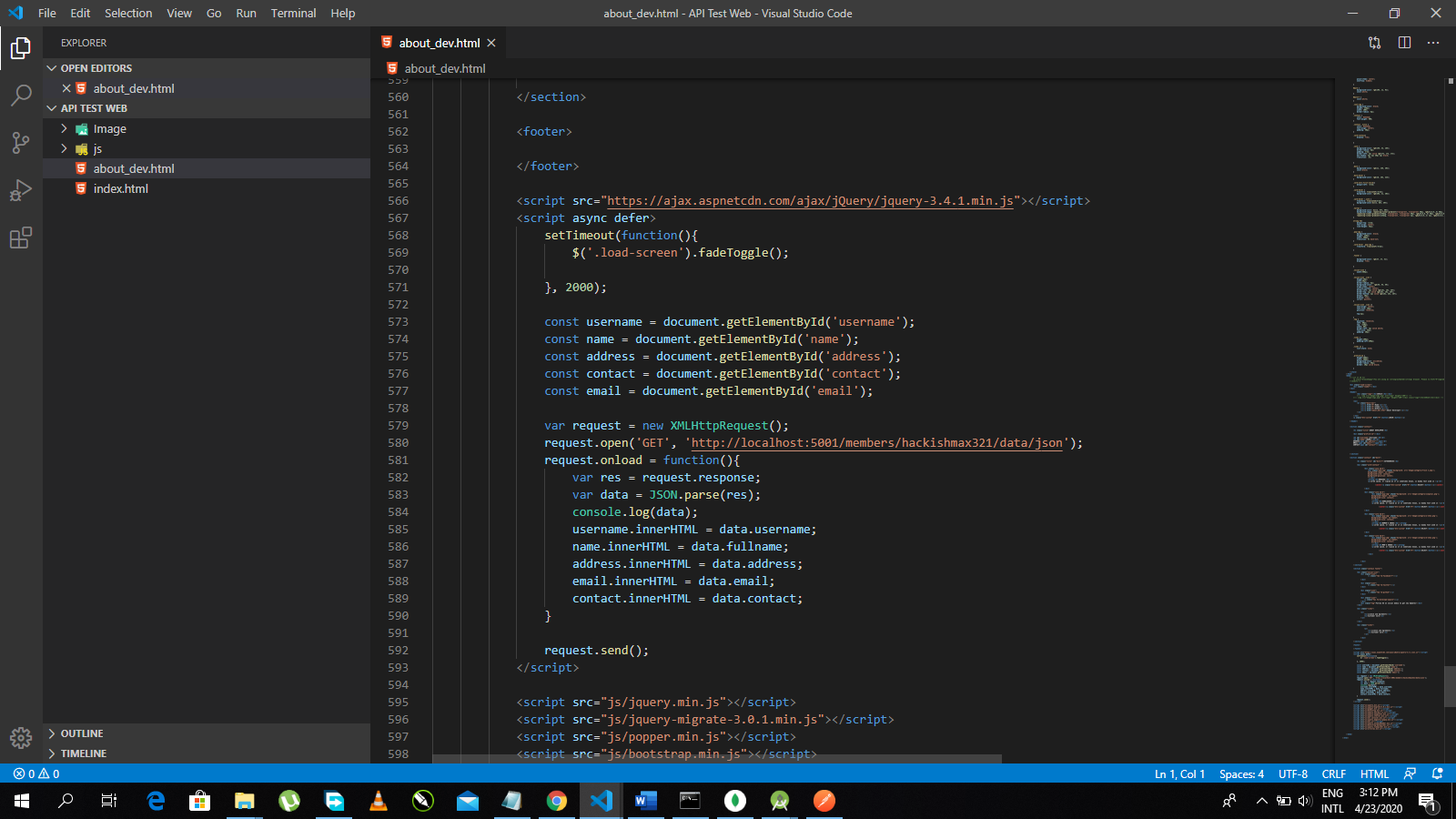


Figure 27: Making a API request

# RETRIVING user data for Android mobile Application

In any above way, you can access our services as below.

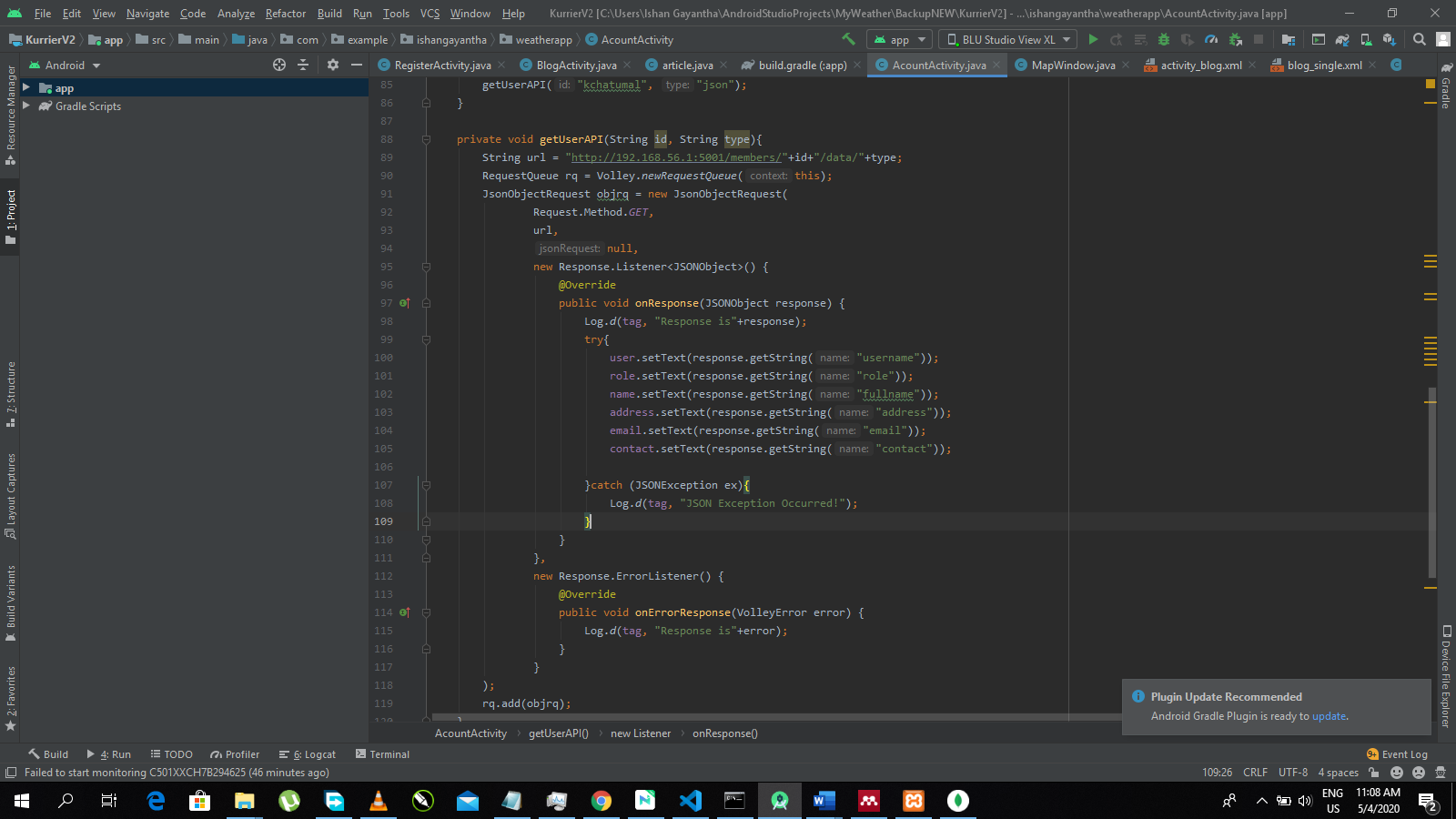


Figure 28: Http client-side request in Android App

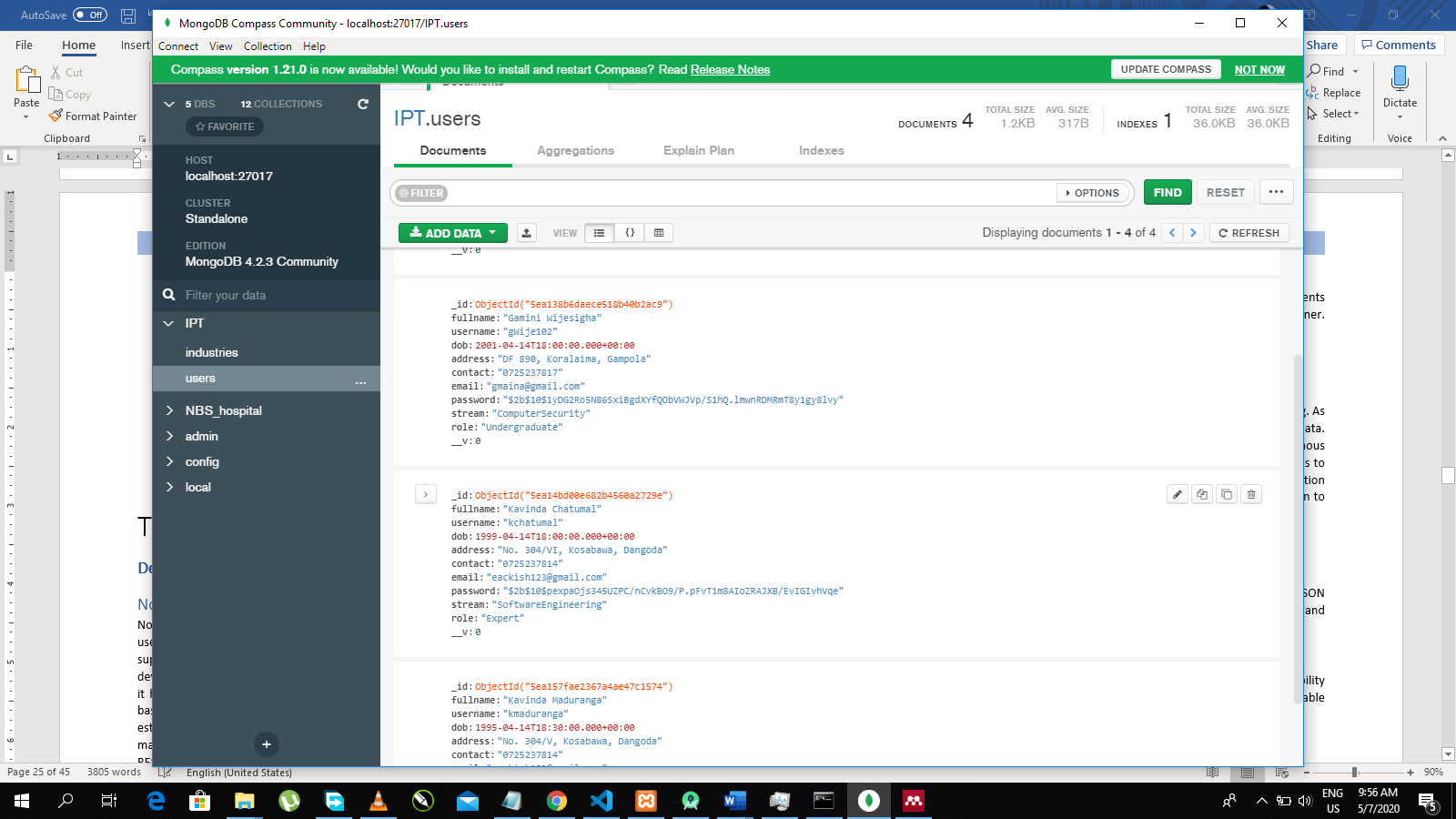
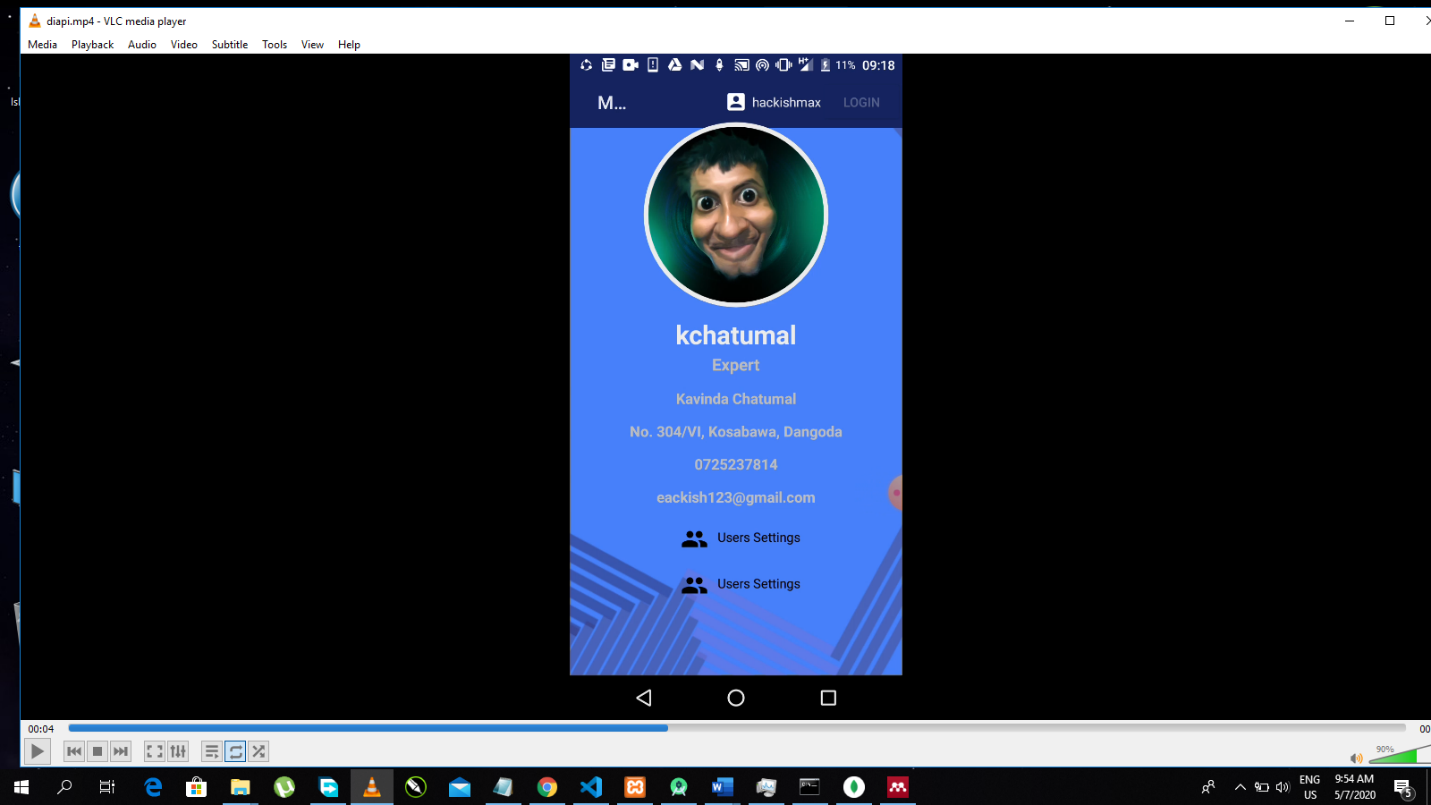
 

Figure : Extract User Data to Mobile Application

TECHNICAL GUIDE

# **Development Technologies**

# Node JS

Node JS is a popular JavaScript runtime environment such as Vue JS and REACT JS and it often use for industrial and commercial application developments. Because of its higher supportiveness for scalable and robust developments in a reliable manner. As well it’s a good development technology for API development because its capability to design a Restful API. Also, it has numerous supportive packages which are acted as Middleware to carry out processes based on Request Response cycle. Express is one of a famous middleware which often uses to establish a server. Other middleware packages are such as body-parser, app.use, app.routeand many other. During Request Response cycle these layers will communicate each other to handle RESTful services. Also, there were few reasons to choose Node JS to implement REST API.

## Asynchronous Request Handling

JavaScript single thread running programming language which support Concurrent Processing. Therefor Node JS which based on JavaScript V8 engine, also supports same method of processing and that methods are named as Event Loops. However, Node JS has higher performance capability comparing to JavaScript.

## Availability of Free tools and Packages

Unlike other developing technologies Node JS has NPM to manage its repositories. This package manager provides easy options to addon different packages allowing us improve capabilities of Node JS. As well it supports more 350,000 packages which is second to Yarn repository.

## Easiness

Node JS always supports effective developing by allowing us to develop different components using less amount of code lines. Also, it allows us to maintain and page code in effective manner. As well it supports sharing and code reuse comparing to many other languages.

# Mongo DB

Mongo DB is a Database Management Software which supports JSON formatted data storing. As the most used database technology it follows clear, light weight structure to maintain its data. Also, its Database engine support faster querying which is perfectly suit for Node JS Asynchronous callbacks. Even provide reliable setting to maintain tables / collection structures allowing us to update their structure as we want. Selecting Mongo DB for our API project was a fine selection in order to make expandable table / collection structures. At the end it was a good decision to choose Mongo DB alongside Node JS, CRUD operations.

# JSON (JavaScript Object Notation)

JSON is a syntax with pre specified notations and rules to store data in order to exchange. JSON uses for many data transactions including Cross – Platform transactions, CRUD responses and many other transactions.

# XML (Extensible Markup Language)

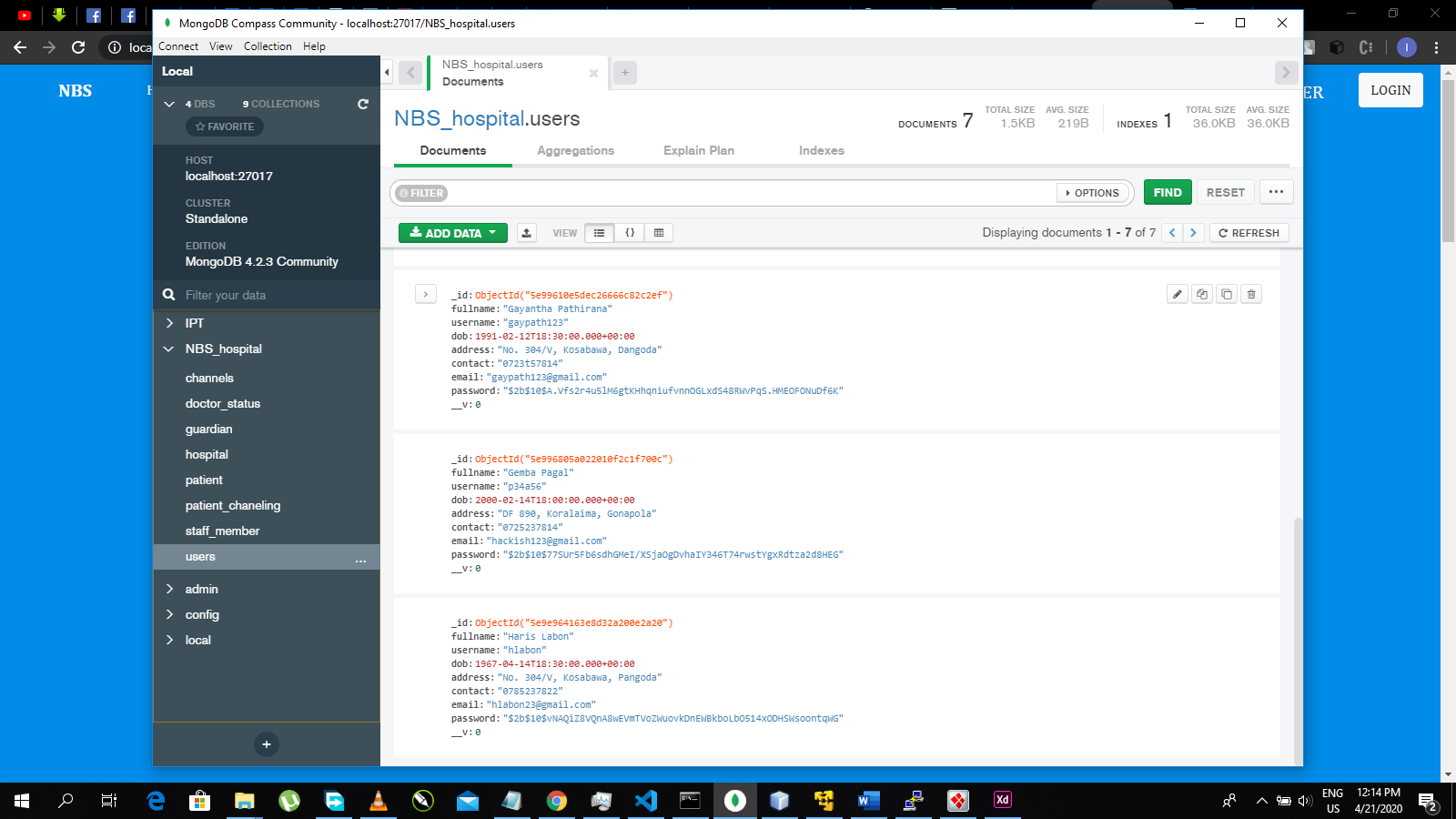
XML was widely used data storing and transaction method before JSON. Unlike JSON it has ability to store styles, template data, data related to electronics. For this project, XML became a reliable data providing method and used to send data based on different Server request types.

# Postman

Postman is an automated testing tool that has capability make HTTP requests in different request methods and observe response. It became useful to uncover anomalies in between communication. It became useful to exercise different Request headers and its validity.

# Bcrypt

Bcrypt is a node repository which facilitate standard level encryption for texts. Encrypted tests or credentials don’t need to decrypt or extract in order to validate because Bcrypt has ability to compare such encrypted text.



# Helmet

Helmet is also a node repository that provide set security standards and methods to secure web applications. In default mode, it will prevent clickjack attacks, sniffing attacks from clients, cross site scripting (XSS) and include important security policies like HTTP Strict Transport Security with many other facilities.

GROUP MEMBERS

1. Ishan Gayantha – (ID 10638279)

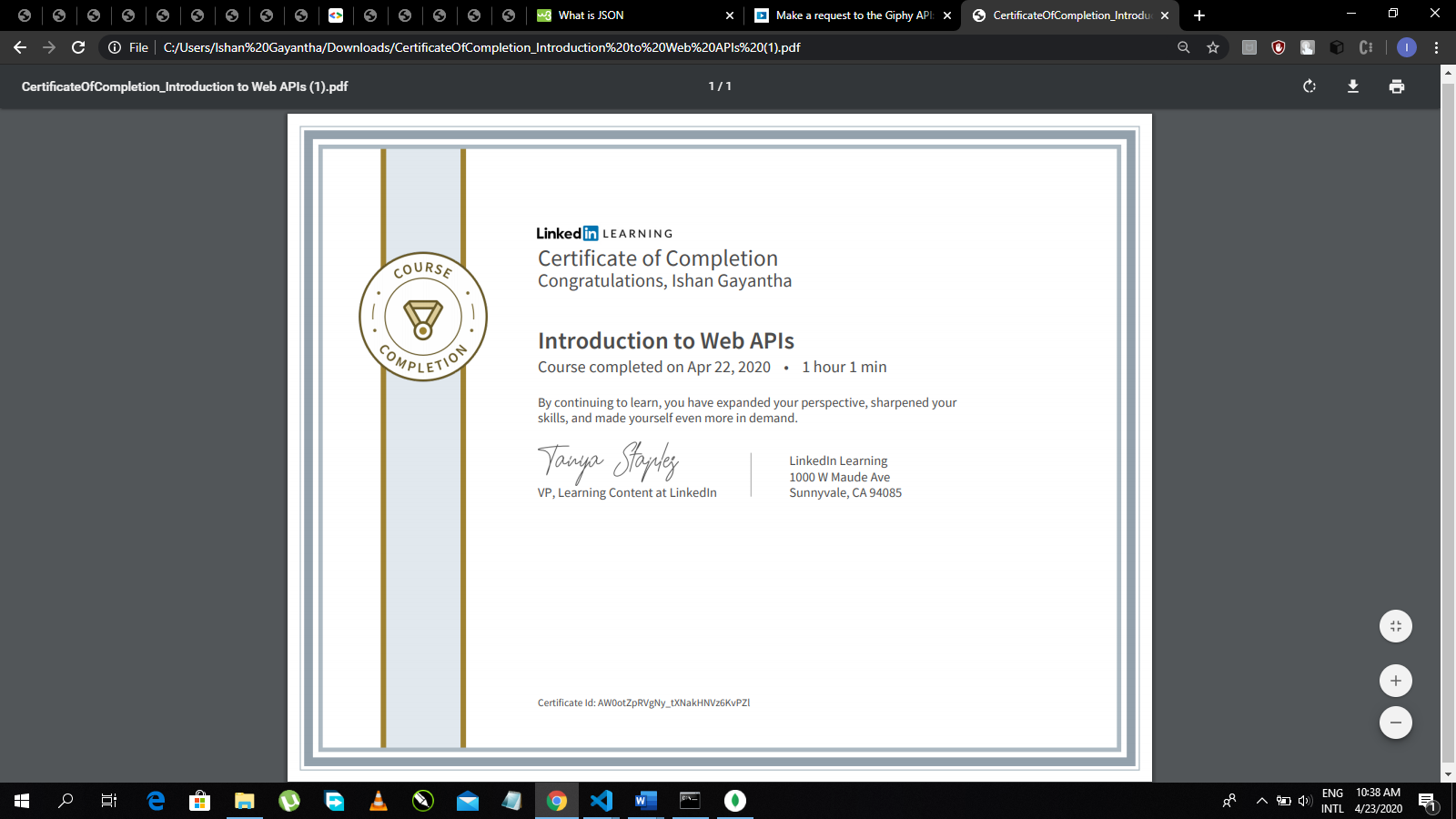
As the leader of this project, I had authority to exercise project management activities including gathering team members, held discussions and guide others to do their tasks. Under Project management several researches handled by me to understand the background of the project. Several courses were followed including Introduction to the Web API course in LinkedIn Learning environment (related certificate has given below).

During the development my responsibilities were as below;

Create Template Design for core API development

* Necessary HTML, CSS, and other necessary packages were added to create better environment to start developments

Members Controller with GET, POST, PUT and DELETE http request methods.

* After implementing core Node JS settings which are Express Server, core middleware, essential packages, it was ready to establish REST API
* Essential HTTP request method GET used for essential routings and data accesses.
* POST request method was used to Login and Register forms.
* PUT request method used to update user profile settings.
* DELETE request method used to remove existing user.
* Handle necessary CRUD options to perform data transactions.

1. Nayanja Hasarnga – (ID - 10638200)

As a developer of this group I had to follow leader’s instructions and did what he asked. I also had to get necessary knowledge by following necessary courses. Web API course in LinkedIn Learning environment is one of them.

Database Design.

* Design and Develop Mongo DB database to store API related data.

Design a sample template web site to extract API information.

* Program HTML and CSS design for sample web template.
* Use cross platform settings and DNS settings to available data communication between Web API and sample template.
* Program JavaScript to make XMLHTTP request.
* Embed Data into the sample site.





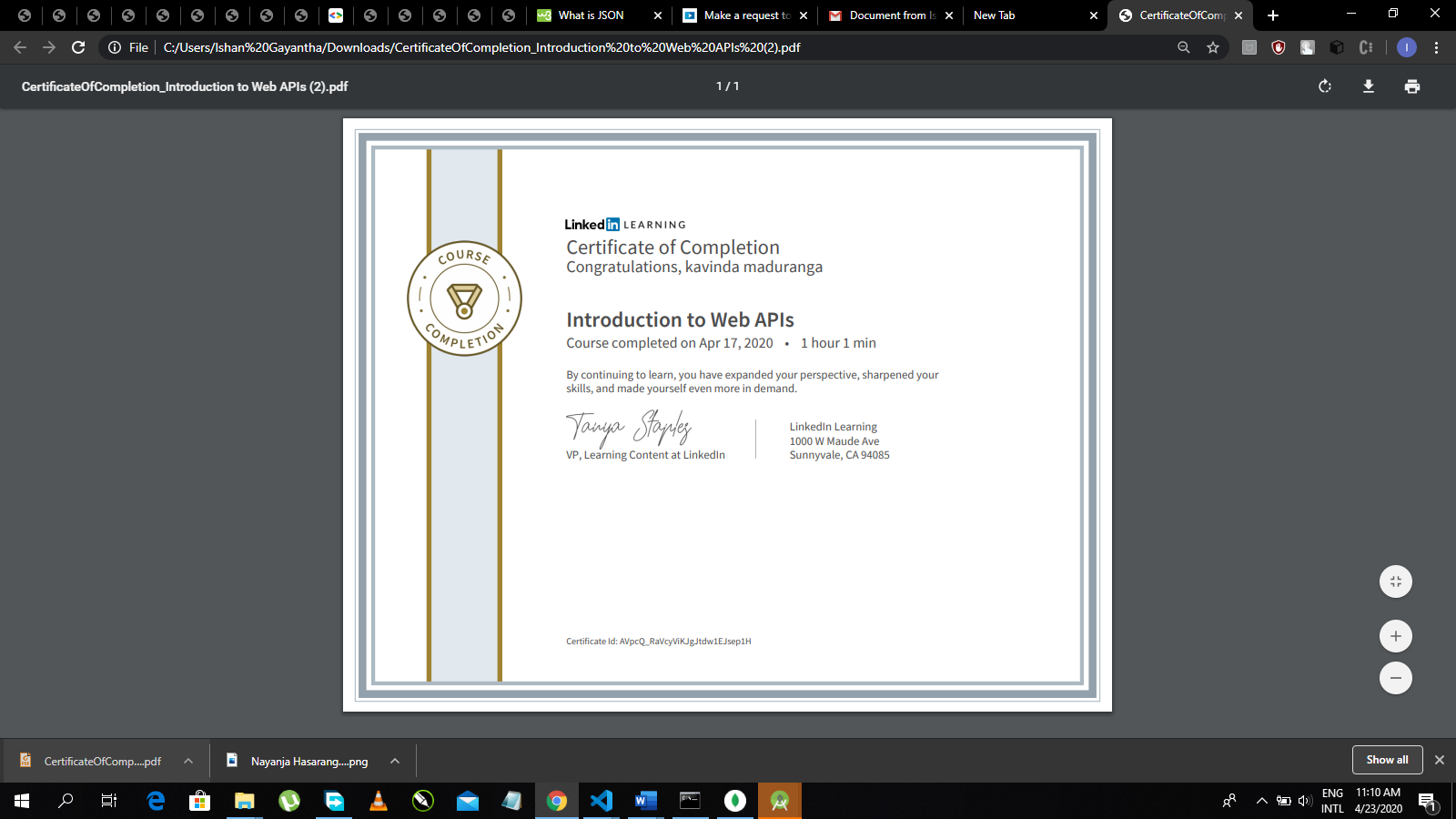
1. Kavinda Maduranga (ID - 10638236)

As a developer of our team my task is to develop essential controllers out of given Minimum requirements. Undergoing this process as our group leader guides, I could achieve following development factors.

Design Industry Controller with Respective GET methods

* Several GET methods were created to extract content related to industries.
* Data Models were created to preform necessary CRUD operations.

Design Category Controller with GET methods

* GET methods were created to category displaying settings.

1. P.C.N Disanayake – (ID – 10638271)

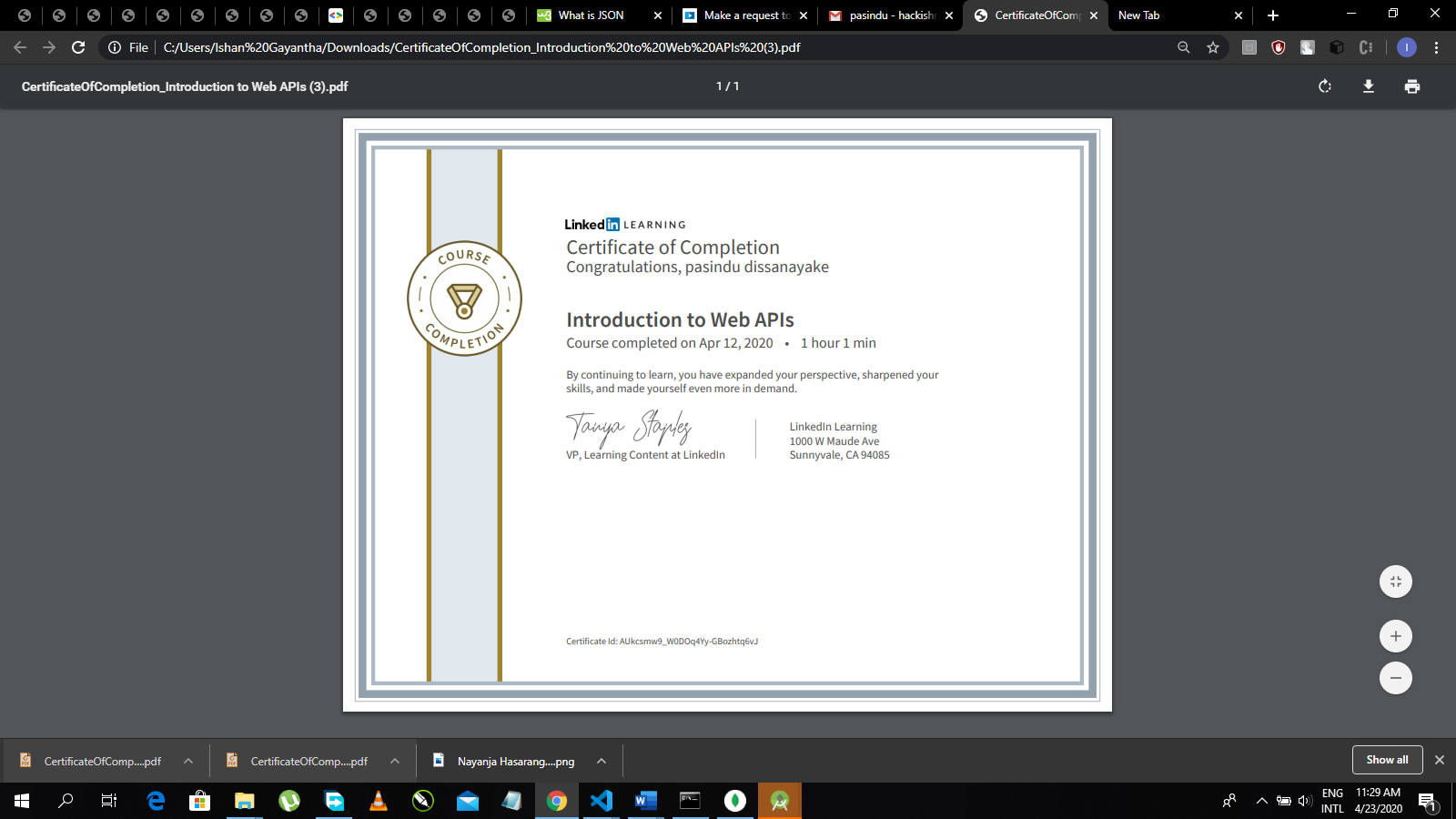
As a Group member of Group, no 18, I did what our leader asked me to do. Therefor I was allocated to implement security at the end of the project. Followings are the standards that I implemented.

Implement encryption methods such as Bcrypt

* Implementation of Bcrypt for login and registration with necessary options.
* Implement mechanism to store Encrypted passwords to Mongo DB.

Implement Helmet package

* Implement XSS related security methods.
* Implement Header hiding mechanisms.



1. S. S. Bandara – (ID – 10638294)

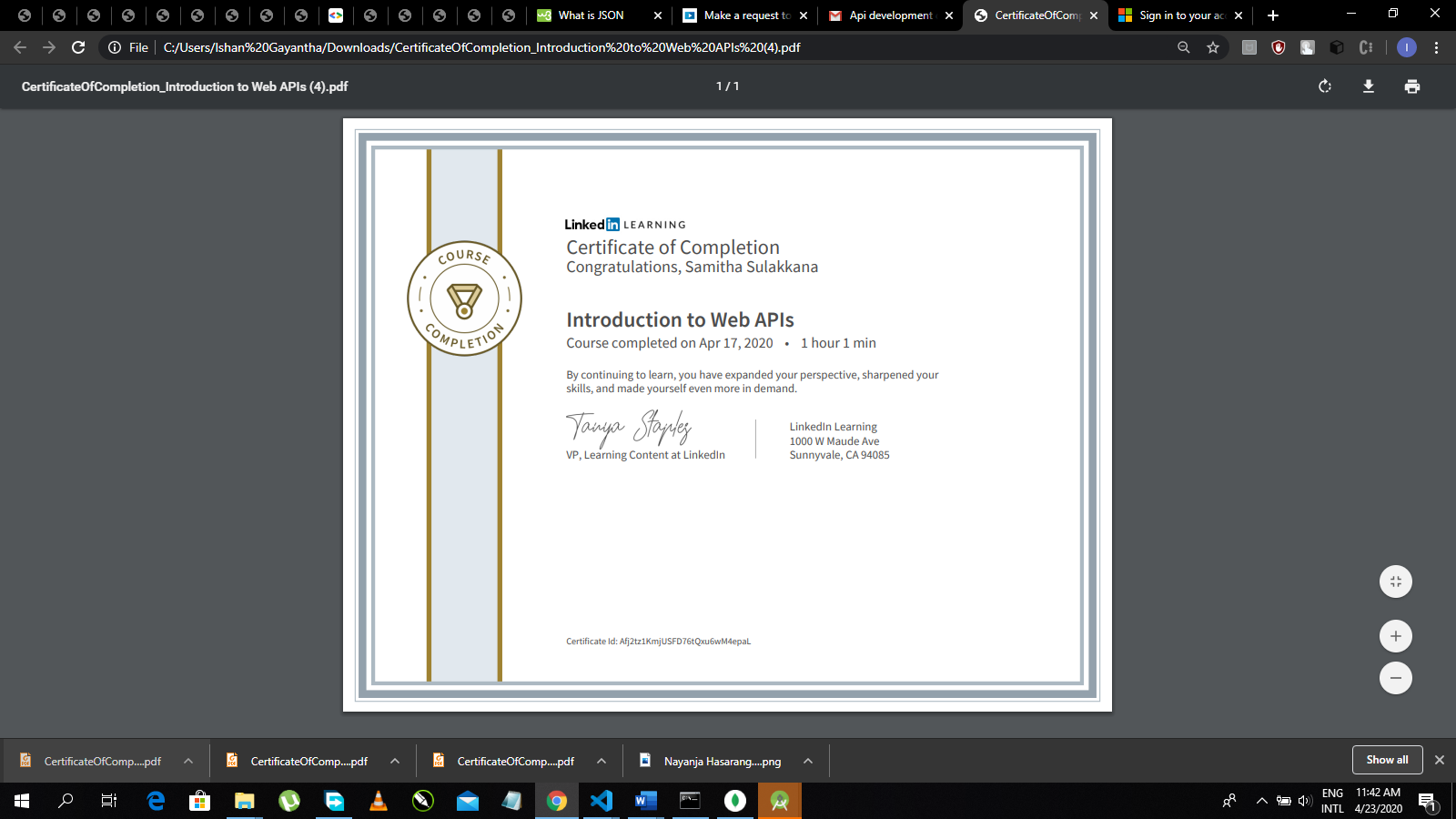
As our group member pointed my objective was to design mobile application and extract necessary information from the core API. Following a course on API design helped me to achieve what I needed to achieve.

Creating a mobile Application template

* Creating Mobile Application template with XML and Java
* Include 3 activity widows with necessary content.

Extracting API information

* Needed to configure localhost settings to access and between communication
* Implement necessary Fetching set up to extract data
* View extracted data



1. G. H. Dhananjana (ID – 10638313)

As our group leader my objective was to handle status related errors handling. Given LinkedIn tutorial also helped me to understand what I wanted to develop.

Status based error handling

* redirect error with status codes such as 404 – Page Not Found, 503 – Server Error / Entrance Restricted
* redirect Json data not found errors
* redirect Text based errors



APPENDIX

1. Typical Routes for web pages

// const express = require('express');

const path = require('path');

const validate = require('express-validator');

const session = require('express-session');

const bcrypt = require('bcrypt');

const saltRounds = 10;

// Data Models

const User = require('../models/user');

module.exports = function(app){

    // Expess Validator | Session

    app.use(validate());

    app.use(session({secret: 'max', saveUninitialized: false, resave: false}));

    // [Typical Routes] =====================================================

    app.get('/', (req, res, next)=>{

        if(req.session.username){

            res.render('home', {username: req.session.username, role: req.session.role});

        } else {

            res.render('home');

        }

    });

    app.get('/aboutus', (req, res, next)=>{

        if(req.session.username){

            return res.render('about us', {username: req.session.username, role: req.session.role});

        }

        res.render('about us');

    });

    app.get('/contactus', (req, res, next)=>{

        if(req.session.username){

            return res.render('contact us', {username: req.session.username, role: req.session.role});

        }

        res.render('contact us');

    });

    app.get('/service', (req, res, next)=>{

        if(req.session.username){

            return res.render('information', {username: req.session.username, role: req.session.role});

        }

        res.render('information');

    });

    // ======================================================================

    // [Controllers] ========================================================

// Members Controller

    require('../controllers/membersControl')(app);

// Industry Controller

    require('../controllers/industryControl')(app);

    // ======================================================================

    // [Error Handling] =====================================================

// Status Code / JSON / Text invalidates Redirect

    app.use(function(req, res, next){

        res.status(404);

        // HTML not Found

        if(req.accepts('html')){

            res.render('err404', {url: req.url});

            return;

        }

        // Json not Found

        if(req.accepts('json')){

            res.send({error:'Not Found'});

            return;

        }

        // Plain Text not Found

        res.type('txt').send('Not found');

    });

    // ========================================================================

}

        }

        // Plain Text not Found

        res.type('txt').send('Not found');

    });

    // ========================================================================

}

1. Members Controller

const path = require('path');

const xml = require('xml-js');

const xml2 = require('jsontoxml');

const validate = require('express-validator');

const session = require('express-session');

const bcrypt = require('bcrypt');

const saltRounds = 10;

// Data Models

const User = require('../models/user');

const Channel = require('../models/industry');

module.exports = function(app){

    // Expess Validator | Session

    app.use(validate());

    app.use(session({secret: 'max', saveUninitialized: false, resave: false}));

// Member Login

    app.route('/members/login')

    .get((req, res, next)=>{

        res.render('login');

    })

    .post((req, res, next)=>{

        //DB

        User.find({username: req.body.regUsername}, (error, resp)=>{

            if(error) throw error;

            if(resp){

                var message = '';

// Password Encryption using BCrypt

                bcrypt.compare(req.body.regPassword, resp[0].password, function(error, response){

                    if(error) throw error;

                    if(response==true){

                        req.session.username = req.body.regUsername;

                        req.session.role = "Member";

                        req.session.email = resp[0].email;

                        res.redirect('/');

                        return;

                    } else if(resp[0].password==req.body.regPassword){

                        //Sessions

                        req.session.username = req.body.regUsername;

                        req.session.role = "Member";

                        req.session.email = resp[0].email;

                        res.redirect('/');

                    } else {

                        console.log(message);

                        res.redirect('/login');

                    }

                });

            }

        });

    });

// Logout

    app.route('/members/logout').get((req, res, next)=>{

        // if (config.debug)

        //     console.log("user logout");

        var backURL=req.header('Referer') || '/';

        req.session.destroy();

        res.redirect(backURL);

    });

   // Member Register

  app.route('/members/register')

    .get((req, res, next)=>{

        res.render('register');

    })

    .post((req, res, next)=>{

        if(req.body.pass==req.body.cpass){

            bcrypt.hash(req.body.pass, saltRounds, function(error, hashc){

                if(error) throw error;

                var user = new User();

                user.fullname = req.body.fullname;

                user.username = req.body.username;

                user.dob = req.body.dob;

                user.address = req.body.address;

                user.contact = req.body.contact;

                user.email = req.body.email;

                user.password = hashc;

                user.stream = req.body.straem;

                user.role = req.body.role;

                user.save(function(error){

                    if(error) throw error;

                    res.redirect('/');

                });

            });

        }

    });

// GET Members details all in list

    app.route('/members')

    .get((req, res, next)=>{

//Mongo DB select Query with Find key word (find ({}) == Select \*)

        User.find({}, function(error, members){

            if(req.session.username){

                return res.render('members', {username: req.session.username, role: req.session.role, users: members});

            }

            res.render('members', {users: members});

        });

    });

  //GET Members Detail as rough data in JSON or XML

   app.route('/members/data/:type')

    .get((req, res, next)=>{

        User.find({}, function(error, members){

            if(error) throw error;

                if(req.params.type=='json'){

                res.set('content-type','application/json');

                res.end(JSON.stringify(members));

            } else if(req.params.type=='xml'){

                res.set('content-type','application/json');

                var xmldata = xml.json2xml(JSON.stringify(members), {compact: true, spaces: 4});

                res.send(xmldata);

            }

        });

    });

// GET members according to their category

    app.route('/:category/members')

    .get((req, res, next)=>{

        User.find({stream: req.params.category}).exec(function(error, members){

            if(error) throw error;

            if(req.session.username){

                return res.render('members', {username: req.session.username, role: req.session.role, users: members});

            }

            res.render('members', {users: members});

        });

    });

// GET members according to their category as rough data in JSON and XML

    app.route('/:category/members/data/:type')

    .get((req, res, next)=>{

        User.find({stream: req.params.category}).exec(function(error, members){

            if(error) throw error;

            if(req.params.type=='json'){

                res.set('content-type','application/json');

                res.end(JSON.stringify(members));

            } else if(req.params.type=='xml'){

                res.set('content-type','application/json');

                var xmldata = xml.json2xml(JSON.stringify(members), {compact: true, spaces: 4});

                res.send(xmldata);

            }

        });

    });

// GET Member’s user Profile details based on his or her username (id)

    app.route('/members/other/:id')

    .get((req, res, next)=>{

        User.findOne({username:  req.params.id}).exec(function(error, member){

            if(error) throw error;

            if(req.session.username){

                return res.render('user\_profile\_other', {username: req.session.username, role: req.session.role, user: member});

            }

            res.render('user\_profile\_other', {user: member});

        });

    });

//GET access logged user’s Profile

    app.route('/members/:id')

    .get((req, res, next)=>{

        User.findOne({username:  req.params.id}).exec(function(error, member){

            if(error) throw error;

            if(req.session.username){

                return res.render('user\_profile', {username: req.session.username, role: req.session.role, user: member});

            }

            res.render('user\_profile', {user: member});

        });

    });

//GET selected user’s details as rough data in JSON or XML

    app.route('/members/:id/data/:type')

    .get((req, res, next)=>{

        User.findOne({username:  req.params.id}).exec(function(error, member){

            if(error) throw error;

            if(req.params.type=='json'){

                res.set('content-type','application/json');

                res.end(JSON.stringify(member));

            } else if(req.params.type=='xml'){

                res.set('content-type','application/json');

                var xmldata = xml.json2xml(JSON.stringify(member), {compact: true, spaces: 4});

                res.send(xmldata);

            }

        });

    });

    // UPDATE selected user’s personal information

    app.route('/members/:id/update/:coloumn')

    .get((req, res, next)=>{

        res.render('edit\_input', {id: req.params.id, column: req.params.coloumn});

    })

    .post((req, res, next)=>{

        var col = req.params.coloumn;

        var value = req.body.update\_val;

        if(col=="fullname"){

            User.update({username: req.params.id}, {fullname: value}, function(error, com, resp){

                if(error) throw error;

                console.log(com);

                res.redirect('/members/'+req.params.id);

            });

        } else if(col=="address"){

            User.update({username: req.params.id}, {address: value}, function(error, com, resp){

                if(error) throw error;

                console.log(com);

                res.redirect('/members/'+req.params.id);

            });

        } else if(col=="email"){

            User.update({username: req.params.id}, {email: value}, function(error, com, resp){

                if(error) throw error;

                console.log(com);

                res.redirect('/members/'+req.params.id);

            });

        } else if(col=="contact"){

            User.update({username: req.params.id}, {contact: value}, function(error, com, resp){

                if(error) throw error;

                console.log(com);

                res.redirect('/members/'+req.params.id);

            });

        }

    });

// Can’t accessed UPDATE or DELETE in HTML DOM but in request handlers like POSTMAN;

// PUT to update given user detail by accessing URL – PASSWORD has taken to prevent unauthorized acts

    app.route('/members/:id/:password/update/:coloumn/:value/:type')

    .put((req, res, next)=>{

        var column = req.params.coloumn;

        if(column==fullname){

            User.update({username: req.params.id, password: req.params.password}, {fullname: req.params.value}, function(error, com, resp){

                if(error) throw error;

                if(req.params.type=='json'){

                    res.set('content-type','application/json');

                    res.end(JSON.stringify({"Activity": "Completed", "Updated Value": req.params.value}));

                } else if(req.params.type=='xml'){

                    res.set('content-type','application/json');

                    var xmldata = xml.json2xml(JSON.stringify({"Activity": "Completed", "Updated Value": req.params.value}), {compact: true, spaces: 4});

                    res.send(xmldata);

                }

            });

        }

    });

// DELETE method to remove selected User

    app.route('/members/:id/delete')

    .get((req, res, next)=>{

        User.findOne({username: req.params.id}).remove().exec((error, user)=>{

            // res.json(true);

            if(error) throw error;

            res.redirect('/members');

        });

    });

// DELETE to delete given user detail by accessing URL – PASSWORD has taken to prevent unauthorized

// acts

    app.route('/members/:id/:password/delete/:type')

    .delete((req, res, next)=>{

        User.findOne({username: req.params.id, password:req.params.password}).remove().exec((error, user)=>{

            // res.json(true);

            if(req.params.type=='json'){

                res.set('content-type','application/json');

                res.end(JSON.stringify({"Activity": "Deletion Completed"}));

            } else if(req.params.type=='xml'){

                res.set('content-type','application/json');

                var xmldata = xml.json2xml(JSON.stringify({"Activity": "Deletion Completed"}), {compact: true, spaces: 4});

                res.send(xmldata);

            }

        });

    });

}

1. Industry Controller

const path = require('path');

const xml = require('xml-js');

const xml2 = require('jsontoxml');

// Data Models

const Industry = require('../models/industry');

module.exports = function(app){

    app.route('/industry')

    .get((req, res, next)=>{

        Industry.find({}, function(error, industries){

            if(req.session.username){

                return res.render('industries', {username: req.session.username, role: req.session.role, users: industries});

            }

            res.render('industries', {users: industries});

        });

    });

// New Industry adding Form

    app.route('/industry/form')

    .get((req, res, next)=>{

        Industry.find({}, function(error, industries){

            if(req.session.username){

                return res.render('industry\_form', {username: req.session.username, role: req.session.role, users: industries});

            }

            res.render('industry\_form', {users: industries});

        });

    })

    .post((req, res, next)=>{

        var ind = new Industry();

                ind.name = req.body.name;

                ind.brief = req.body.brief;

                ind.description = req.body.description;

                ind.contact = req.body.contact;

                ind.email = req.body.email;

                ind.save(function(error){

                    if(error) throw error;

                    res.redirect('/industry');

                });

    });

//

    app.route('/members/data/:type')

    .get((req, res, next)=>{

        User.find({}, function(error, members){

            if(error) throw error;

                if(req.params.type=='json'){

                res.set('content-type','application/json');

                res.end(JSON.stringify(members));

            } else if(req.params.type=='xml'){

                res.set('content-type','application/json');

                var xmldata = xml.json2xml(JSON.stringify(members), {compact: true, spaces: 4});

                res.send(xmldata);

            }

        });

    });

    app.route('/members/:id')

    .get((req, res, next)=>{

        User.findOne({username:  req.params.id}).exec(function(error, member){

            if(error) throw error;

            if(req.session.username){

                return res.render('user\_profile', {username: req.session.username, role: req.session.role, user: member});

            }

            res.render('user\_profile', {user: member});

        });

    });

    app.route('/members/:id/data/:type')

    .get((req, res, next)=>{

        User.findOne({username:  req.params.id}).exec(function(error, member){

            if(error) throw error;

            if(req.params.type=='json'){

                res.set('content-type','application/json');

                res.end(JSON.stringify(member));

            } else if(req.params.type=='xml'){

                res.set('content-type','application/json');

                var xmldata = xml.json2xml(JSON.stringify(member), {compact: true, spaces: 4});

                res.send(xmldata);

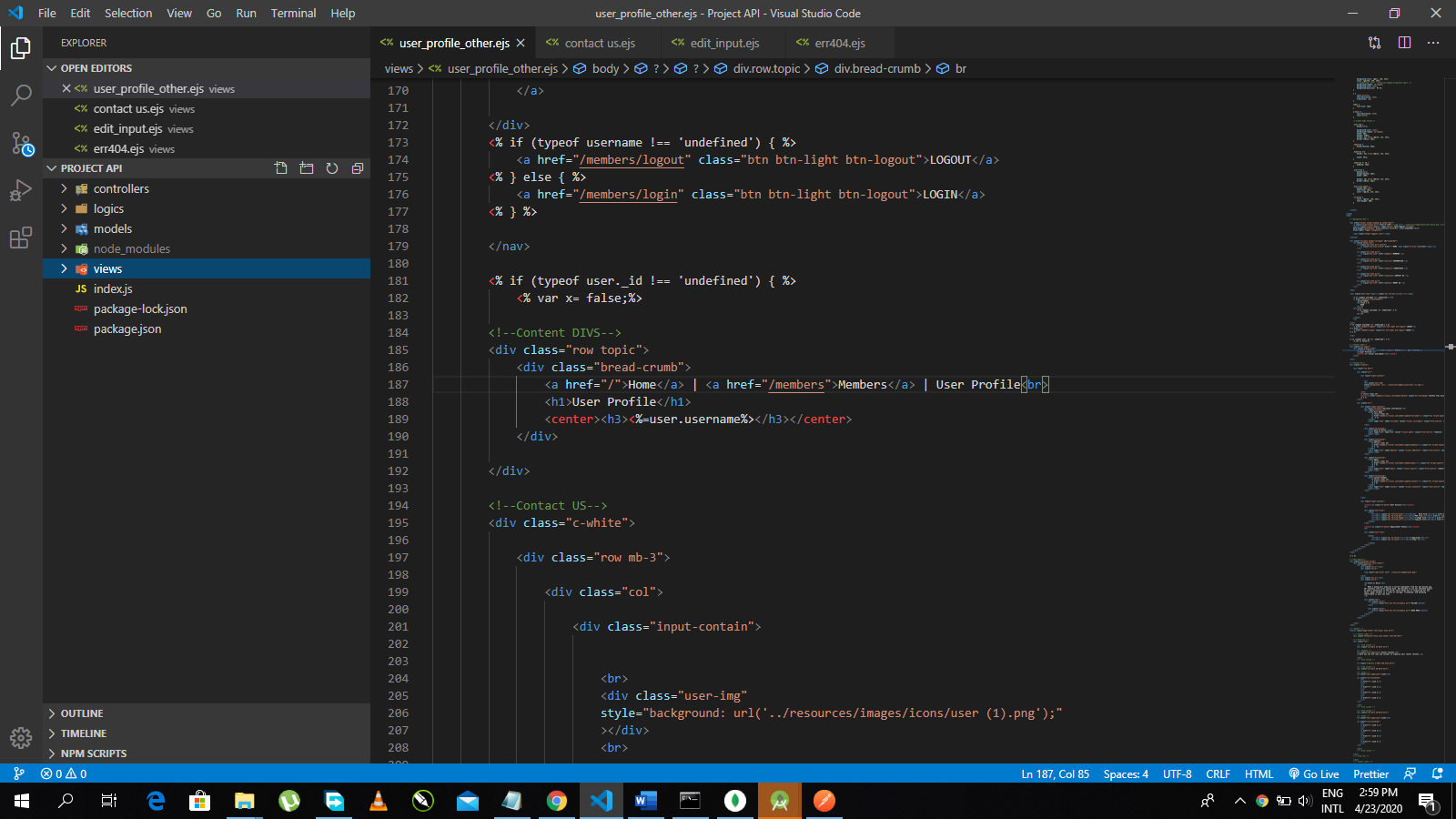
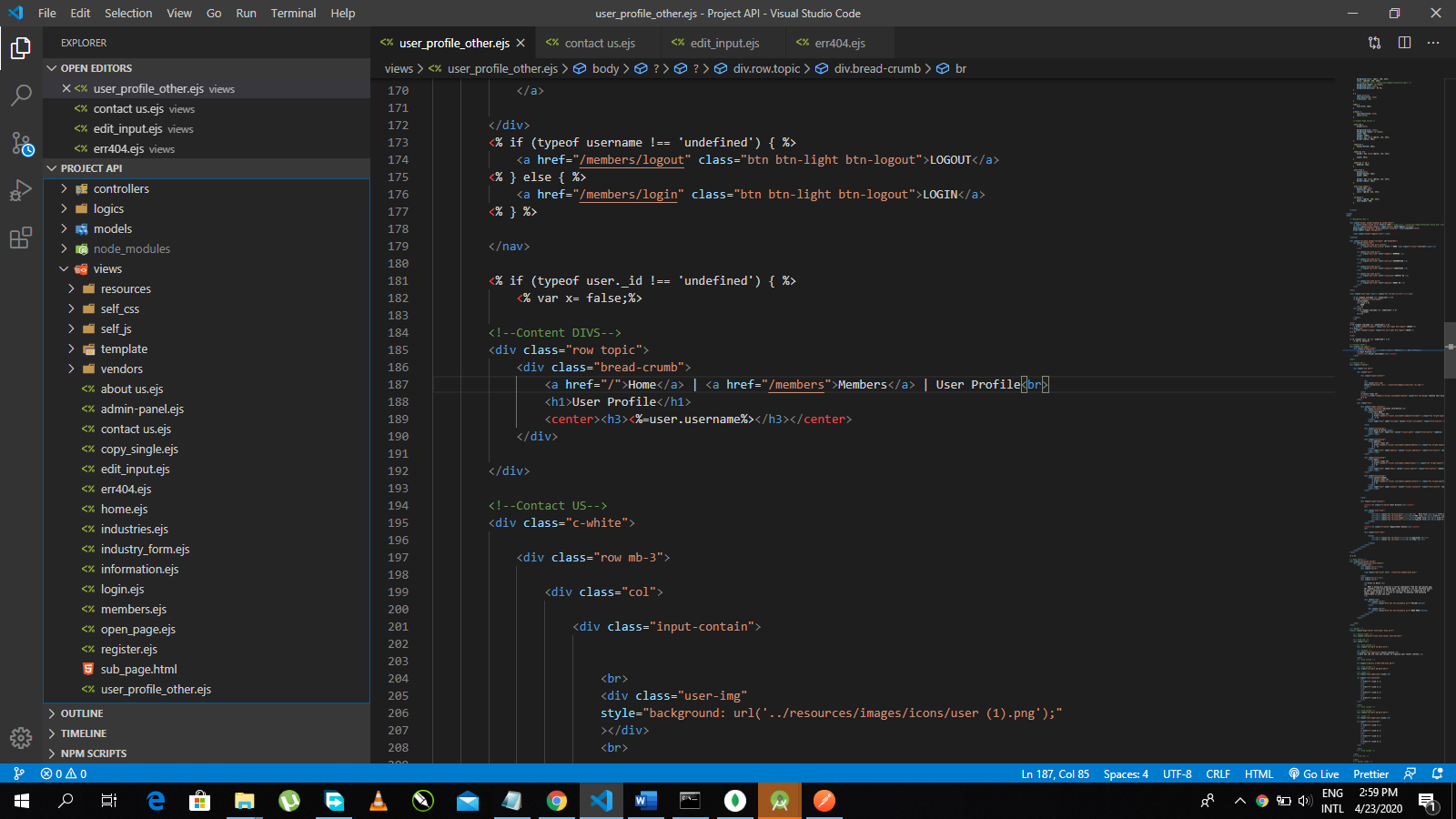
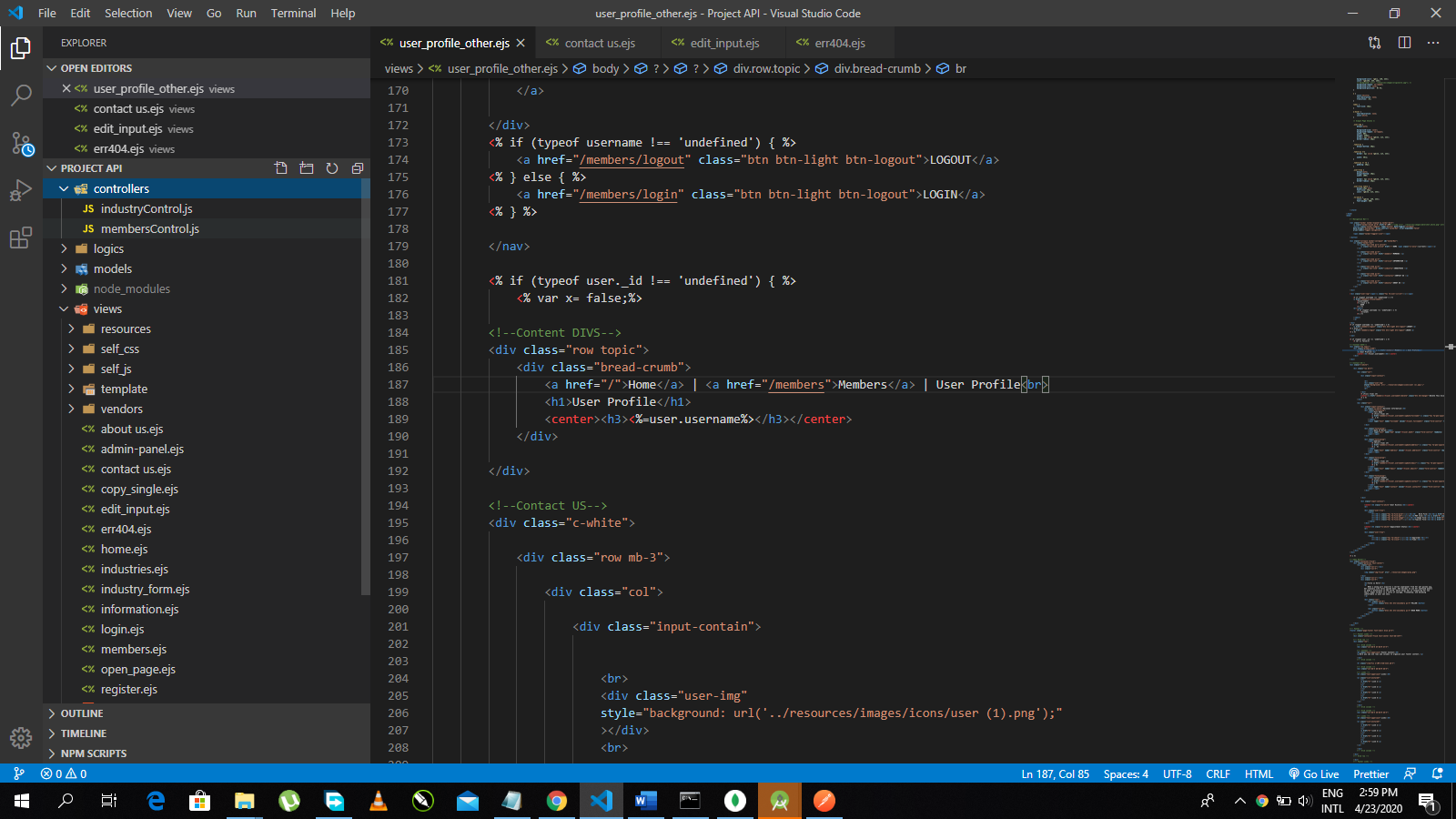
            }

        });

    });

}

# Clearly Paged Project Files

Files are structured according to MVC architecture.

modals directory => Modals Layer

controllers + logics directories => Controllers Layer

views directory => Views Layer