IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: **I.K.N.K Weerasinghe**
Student Reference Number: **10674043**

Module Code: **PUSL3111**	Module Name: **API Software Development**
Coursework Title: **API Web and Mobile application**	
Deadline Date: **2021 April 23**	Member of staff responsible for coursework: **Mr. 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.

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.

Student Name	Plymouth Index No
I.K.N.K Weerasinghe	10674043
G.L.I Karunanayake	10673980
Kely Weerasooriya	10674044
H.W.C Waduge	10674040
M.M.P.M Bandara	10673074
R.P.L Wanasinghe	10674041

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: **I.K.N.K Weerasinghe**

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_____

*Please delete as appropriate Sci/ps/d:/students/cwkfrontcover/2013/14

Declaration

When forwarding this submission, we claim that our work contains no examples of abuses such as plagiarism or paraphrase. All content has been properly processed in accordance with Plymouth guidelines. Below is the list of members of the group.

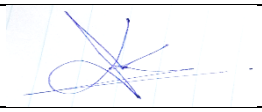
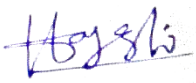
Team 15		
Member name	Plymouth ID	Signature
I.K.N.K Weerasinghe	10674043	
Kely Weerasooriya	10674044	
G.L.I Karunanayake	10673980	
H.W.C. Waduge	10674040	
M.M.P.M Bandara	10673074	
R.P.L Wanasinghe	10674041	

Table of Contents

Table of Contents

1. Introduction

This is the RESTful API, build for the ministry of health to gather information about the managing the pandemic situation, such as COVID-19. There have three main parts such as CDC (Center for Dicease Control Prevention), PHI and Citizen. CDC and PHI planning to establish internet service where citizens can report their location, when they are moving somewhere. There giving chance to citizens, free online registration to anyone. After citizen can update their location using QR code and PHI can monitor their antigen or PCR test result. CDC can traced by the covid positive patients. After that CDC can deactivate the covid positive patients accounts from the system. Then CDC can sent an email to the positive patients about the account deactivation. Most of methods written by POST, PUT, DELETE and GET methods for helping to developers.

API Documentation

2. Getting Started

The existing edition of built API lives at:
https://drive.google.com/file/d/1iFOGdAe8BFsSixHicHCMTDGAxrhRs0l0/view?usp=sharing

API Calls

The developed API supports Json , XML format data response and including with three main controllers.

We create the API using node.js and we use PHP to connect that API with the web application.

Follow are the,

1. Lists of APIs
2. Routes
3. Evidence
4. Discription

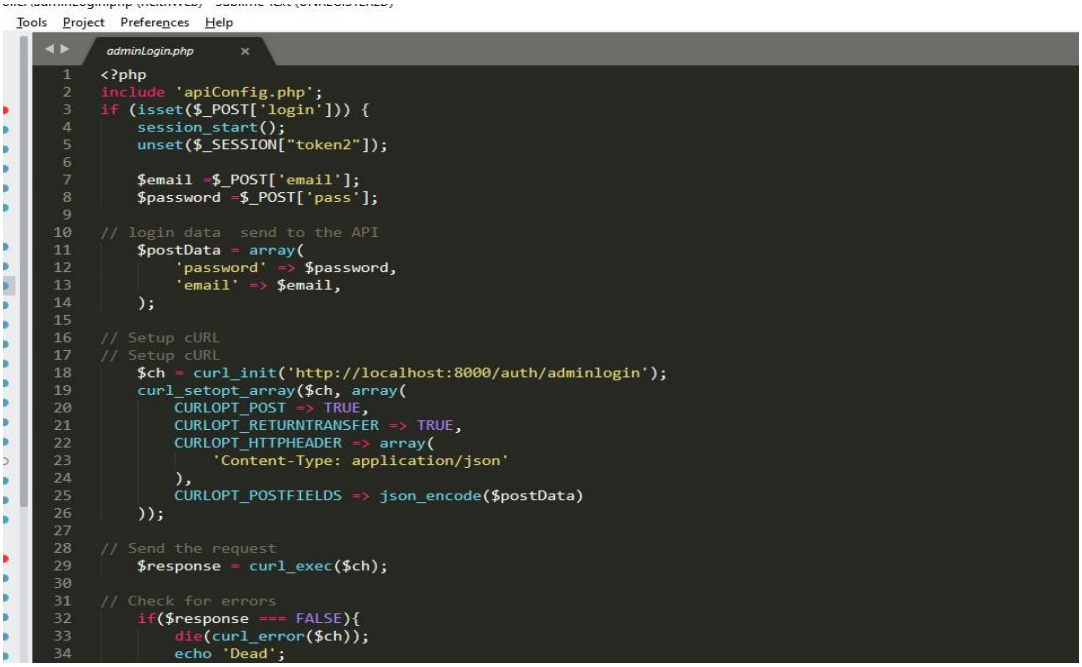
	Lists of APIs
adminLogin	http://localhost:8000/auth/adminlogin
addContact	http://localhost:8000/contact/AddMessage
citizen_signup	http://localhost:8000/auth/citizen/signup
citizenLogin	http://localhost:8000/auth/citizenlogin
deleteCitizen	http://localhost:8000/citizen/updatestate
phi_signup	http://localhost:8000/auth/phi/signup
phi_update	http://localhost:8000/auth/updatePhi
phi_Login	http://localhost:8000/auth/philogin
sendEmail	http://localhost:8000/citizen/email
update_citizen	http://localhost:8000/auth/updateCitizen
updateHealth	http://localhost:8000/citizen/addCitizenReport

a) Controller 1: Admin

1. Admin Login :

Routes: `http://localhost:8000/auth/adminlogin`

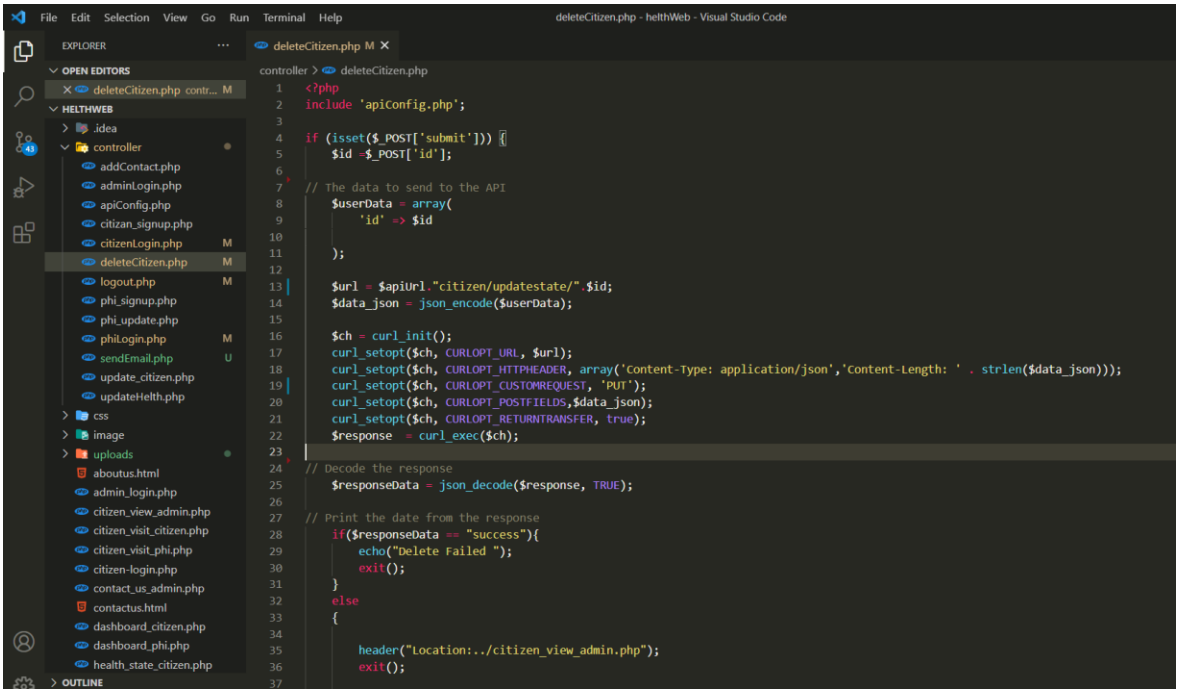
Discription: This is the admins login part. There are the API part and the Connection part of admins login.

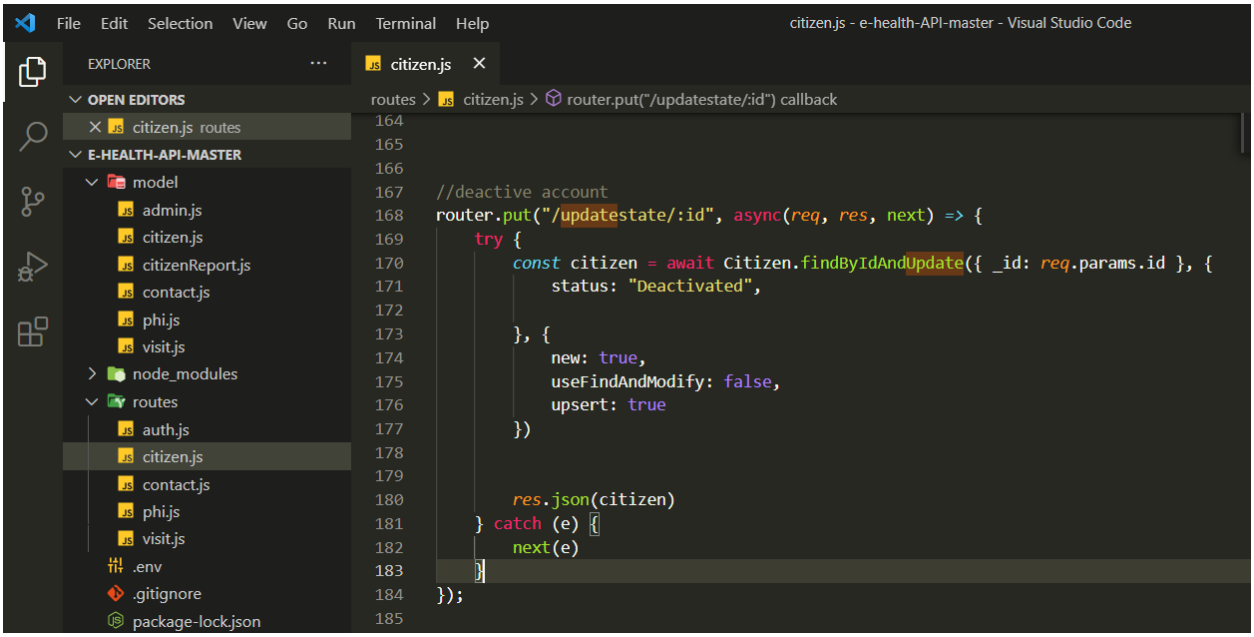


```
274
275   //login Admin
276   router.post('/adminlogin', async(req, res) => {
277       const admin = await Admin.findOne({ email: req.body.email });
278       if (!admin) return res.status(400).send({ message: 'Email does Not Exist' });
279
280       const validPass = await bcrypt.compare(req.body.password, admin.password);
281
282       if (!validPass) return res.status(400).send({ message: 'Password is Wrong' });
283
284       const token = jwt.sign({ _id: admin._id }, process.env.JWT_KEY);
285
286
287       res.header('auth-token').send({
288           id: admin._id,
289           email: admin.email,
290           token: token
291       });
292   });
293   module.exports = router;
```

2.Account Deactivation : This is the part of, the admin can deactivate the positive citizens profiles.

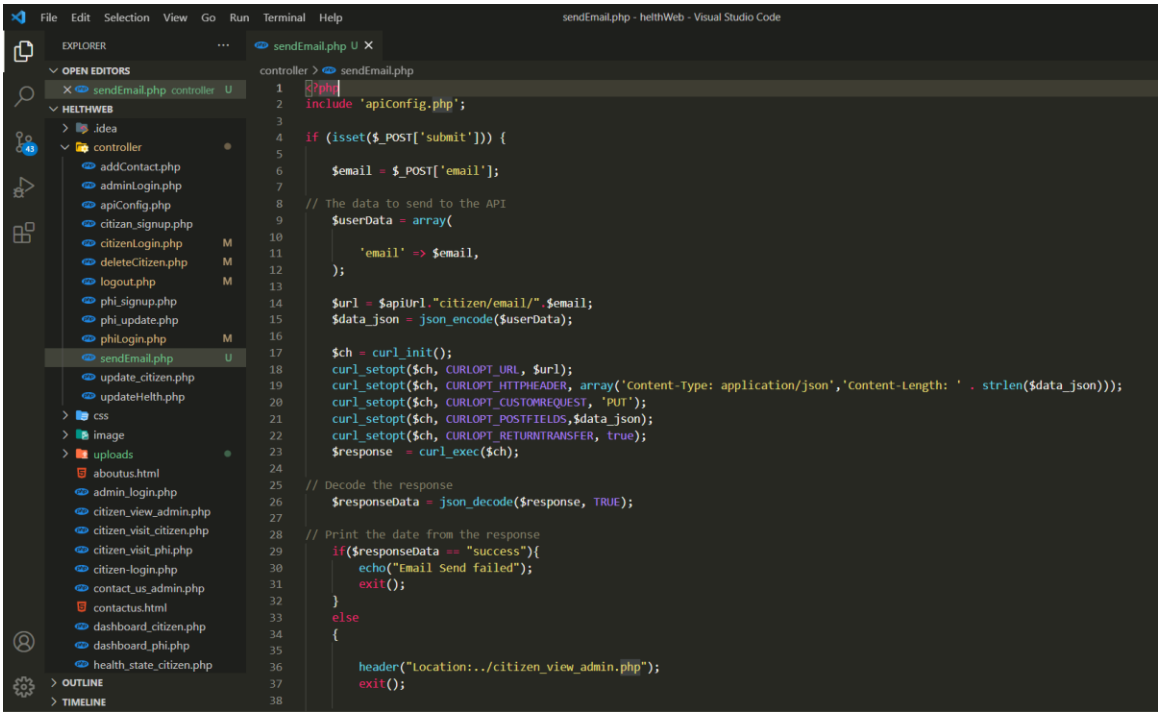
Routes: http://localhost:8000/citizen/updatestate

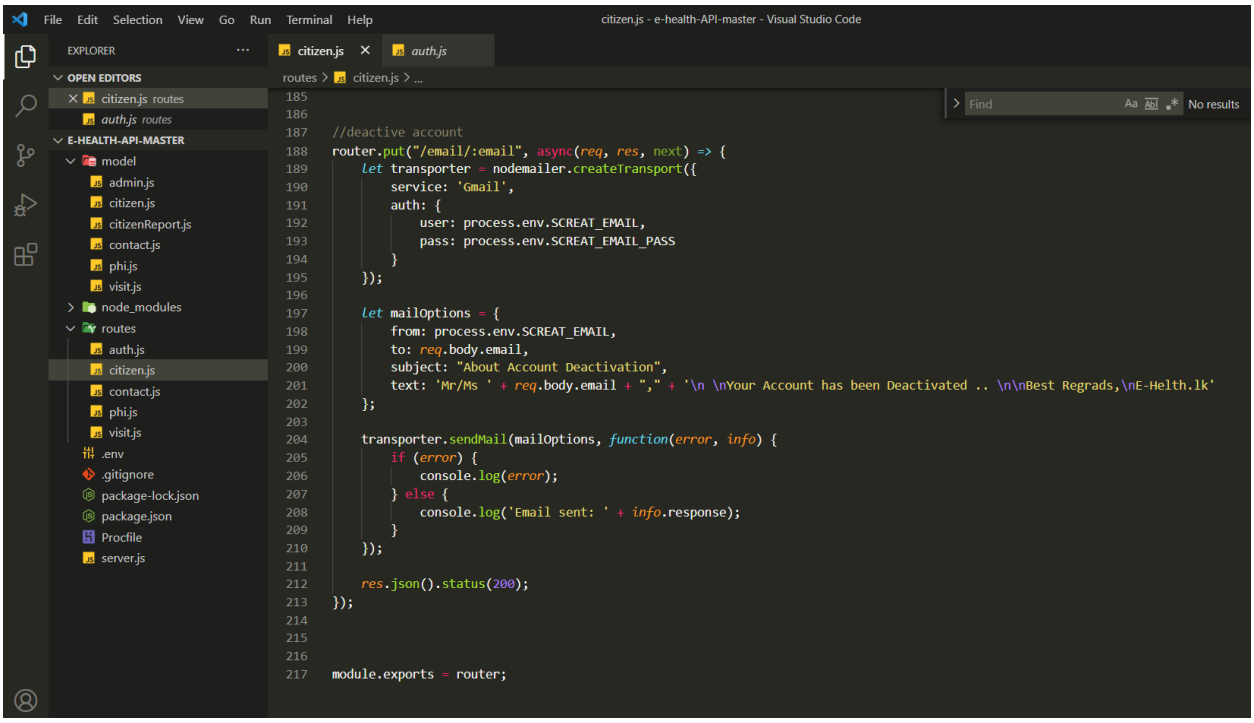




3.Send Email for Positive Patients : After the account deactivation, admin can send an email to the citizen. So this is that sending email part.

Routes : http://localhost:8000/citizen/email

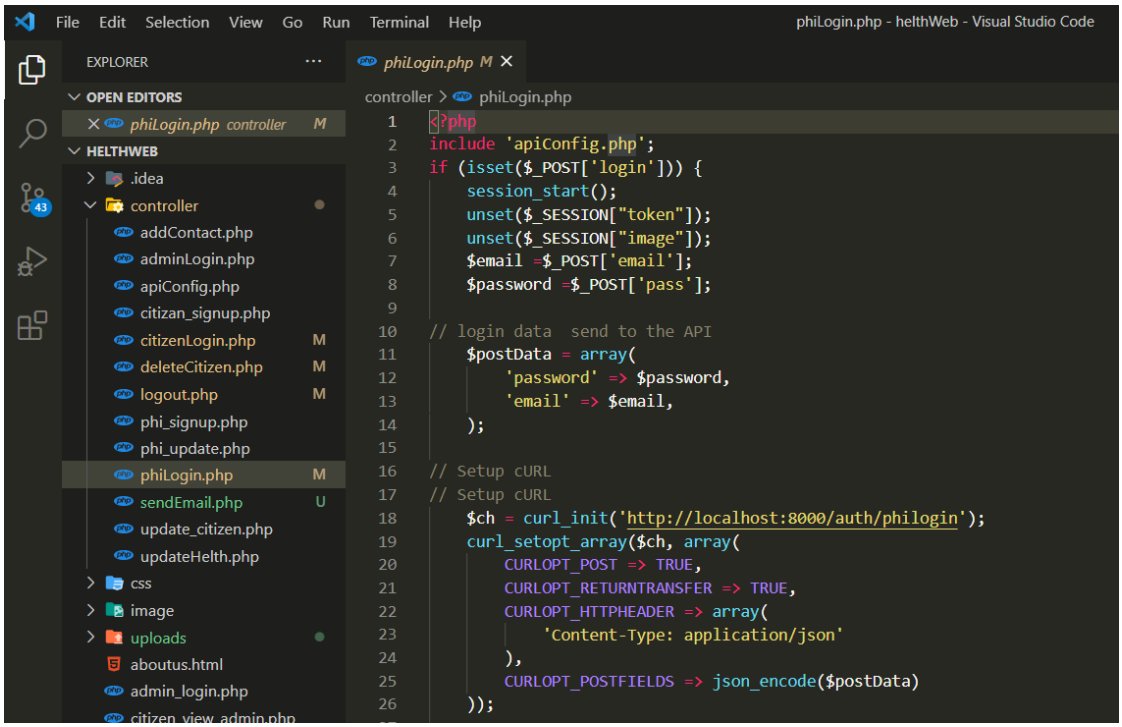


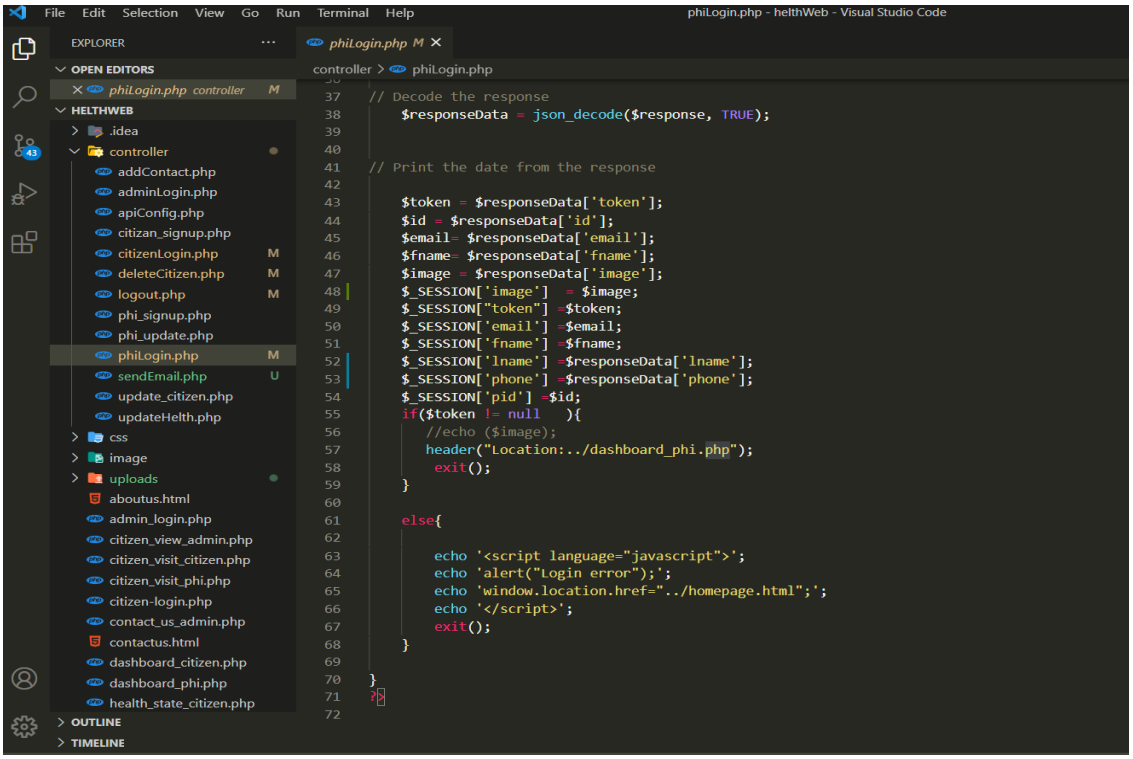

File Edit Selection View Go Run Terminal Help
citizen.js - e-health-API-master - Visual Studio Code
EXPLORER
citizen.js
auth.js
routes > citizen.js > ...
OPEN EDITORS
citizen.js routes
auth.js routes
E-HEALTH-API-MASTER
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
Procfile
server.js
Find
No results
185
186
187 //deactive account
188 router.put("/email/:email", async(req, res, next) => {
189 let transporter = nodemailer.createTransport({
190 service: 'Gmail',
191 auth: {
192 user: process.env.SCREAT_EMAIL,
193 pass: process.env.SCREAT_EMAIL_PASS
194 }
195 });
196
197 let mailOptions = {
198 from: process.env.SCREAT_EMAIL,
199 to: req.body.email,
200 subject: "About Account Deactivation",
201 text: 'Mr/Ms ' + req.body.email + "," + '\n \nYour Account has been Deactivated .. \n\nBest Regrads,\nE-Helth.lk'
202 };
203
204 transporter.sendMail(mailOptions, function(error, info) {
205 if (error) {
206 console.log(error);
207 } else {
208 console.log('Email sent: ' + info.response);
209 }
210 });
211
212 res.json().status(200);
213 });
214
215
216
217 module.exports = router;

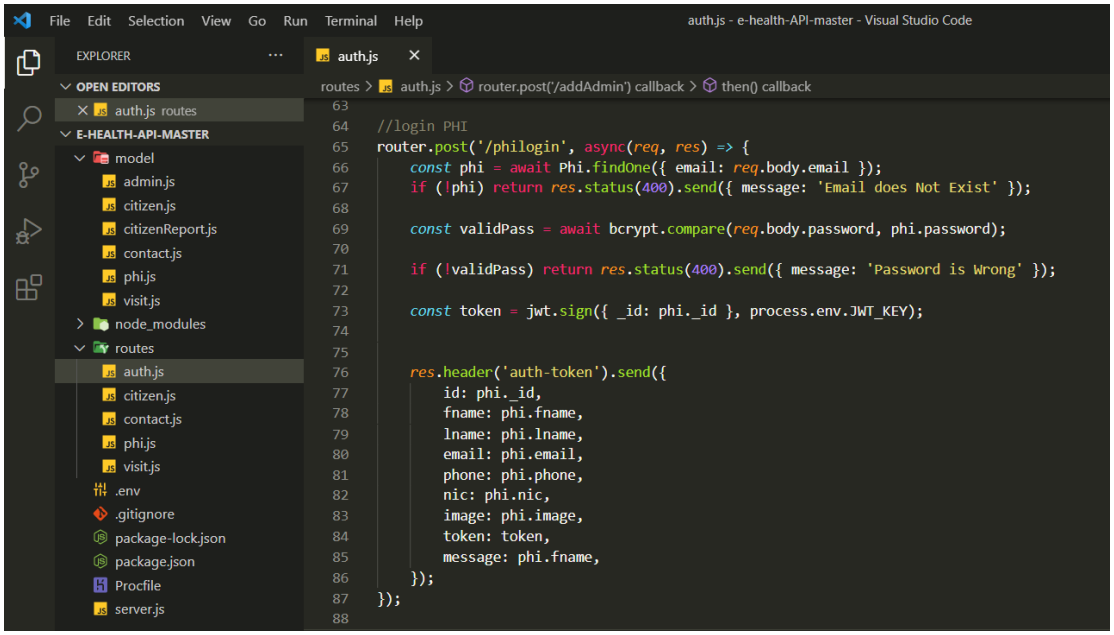
b) Controller 2: PHI

1. PHI Login: This is the login of PHI

Routes : http://localhost:8000/auth/philogin

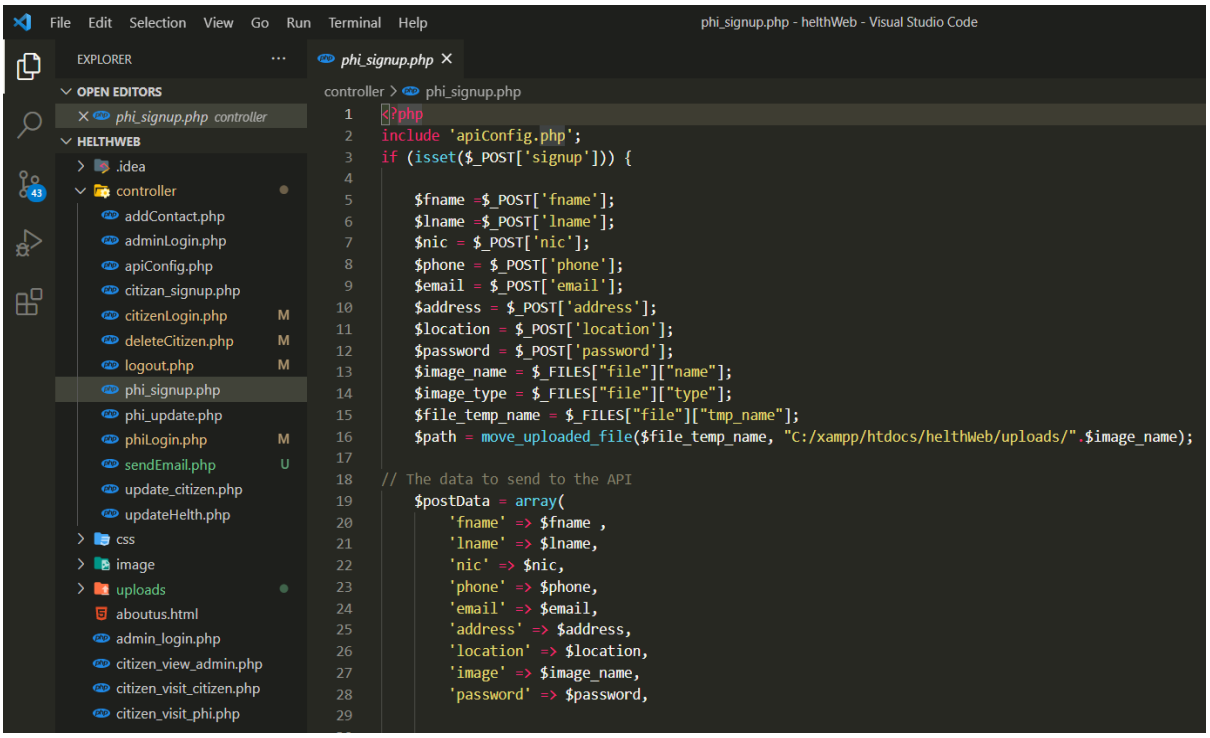


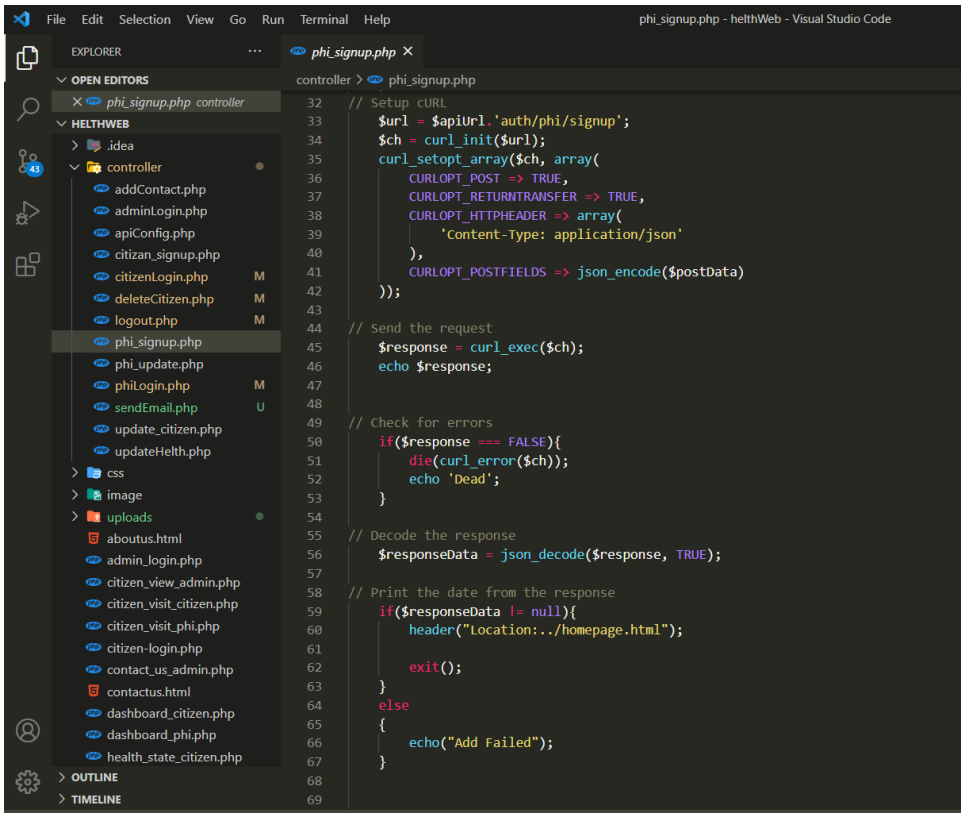


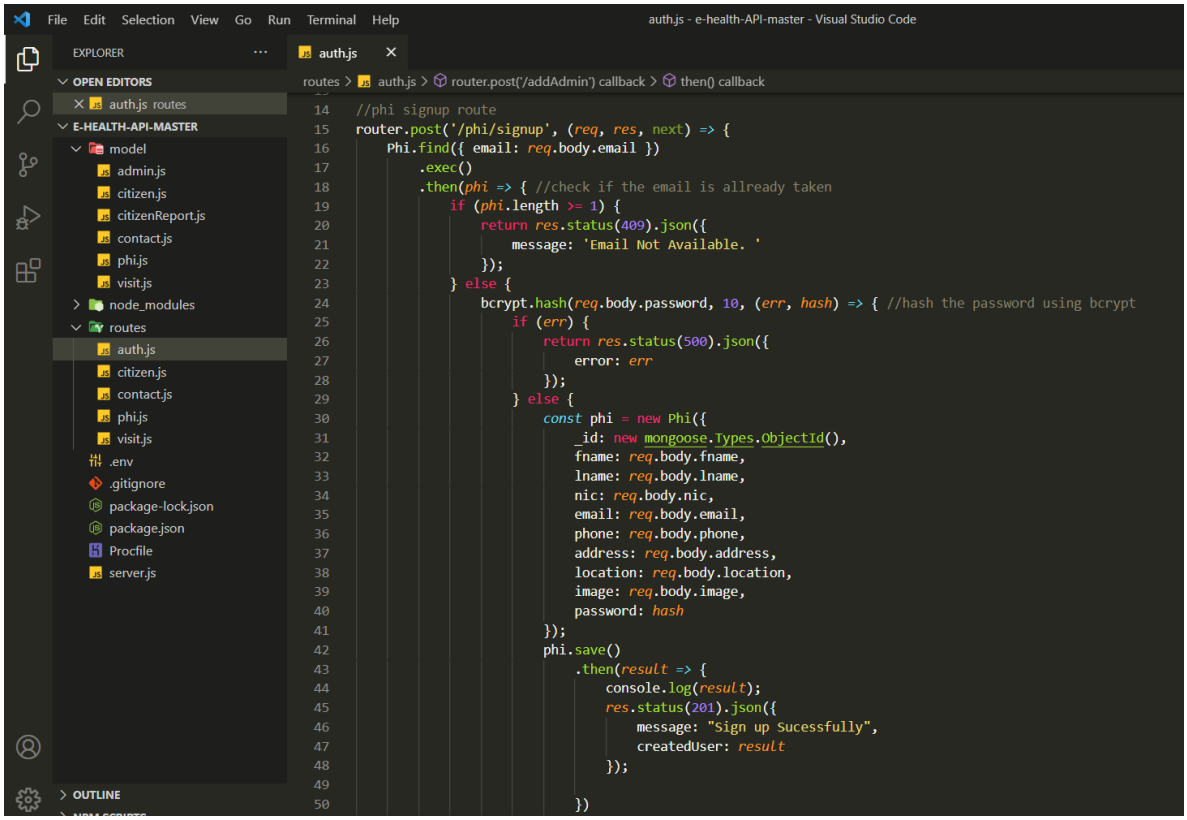


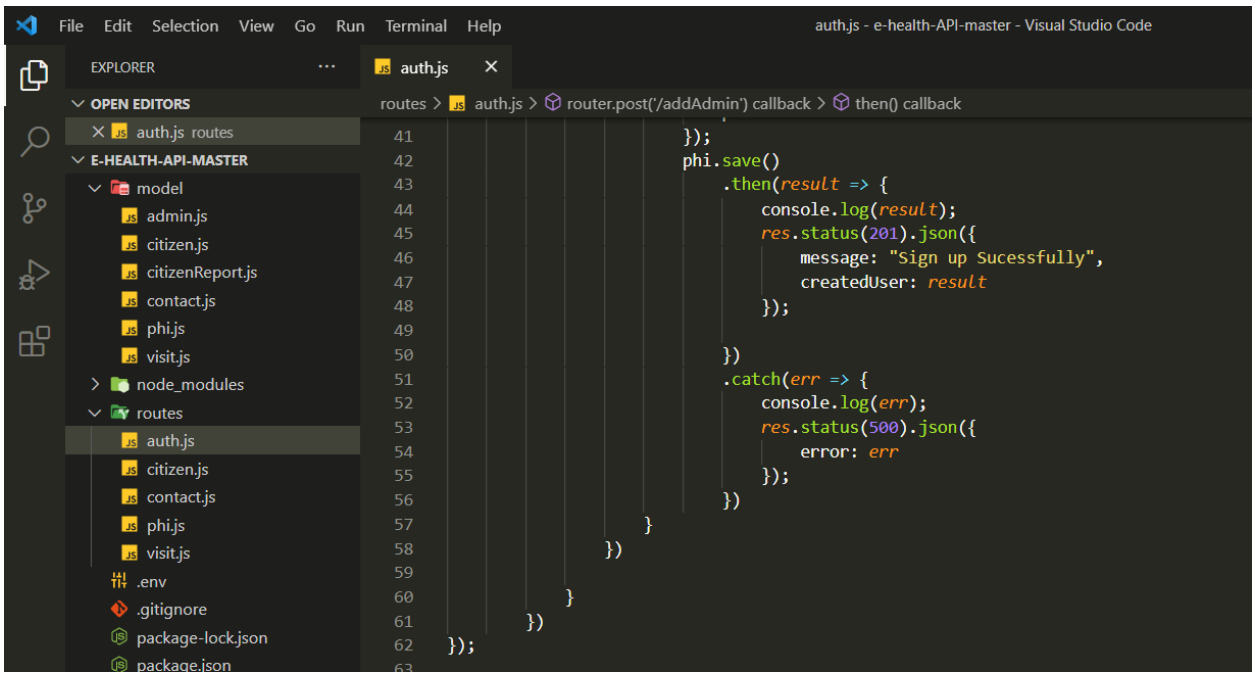
2.PHI Sign UP: This is the register part of the PHI

Routes: http://localhost:8000/auth/phi/signup



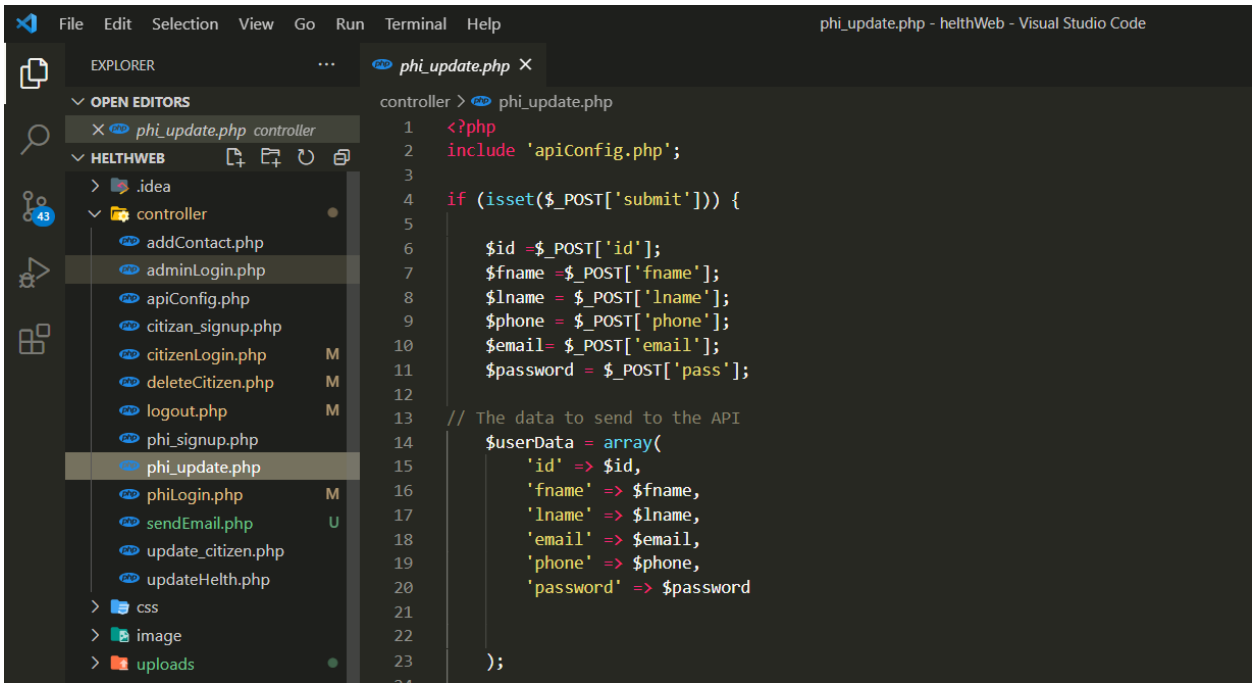

File Edit Selection View Go Run Terminal Help
phi_signup.php - helthWeb - Visual Studio Code
EXPLORER
phi_signup.php
OPEN EDITORS
phi_signup.php controller
HELTHWEB
.idea
controller
addContact.php
adminLogin.php
apiConfig.php
citizan_signup.php
citizenLogin.php M
deleteCitizen.php M
logout.php M
phi_signup.php
phi_update.php
phiLogin.php M
sendEmail.php U
update_citizen.php
updateHelth.php
css
image
uploads
aboutus.html
admin_login.php
citizen_view_admin.php
citizen_visit_citizen.php
citizen_visit_phi.php
citizen-login.php
contact_us_admin.php
contactus.html
dashboard_citizen.php
dashboard_phi.php
health_state_citizen.php
OUTLINE
TIMELINE
controller > phi_signup.php
// Setup cURL
$url = $apiUrl.'auth/phi/signup';
$ch = curl_init($url);
curl_setopt_array($ch, array(
CURLOPT_POST => TRUE,
CURLOPT_RETURNTRANSFER => TRUE,
CURLOPT_HTTPHEADER => array(
'Content-Type: application/json'
),
CURLOPT_POSTFIELDS => json_encode($postData)
));
// Send the request
$response = curl_exec($ch);
echo $response;
// Check for errors
if($response === FALSE){
die(curl_error($ch));
echo 'Dead';
}
// Decode the response
$responseData = json_decode($response, TRUE);
// Print the date from the response
if($responseData != null){
header("Location:../homepage.html");
exit();
}
else
{
echo("Add Failed");
}

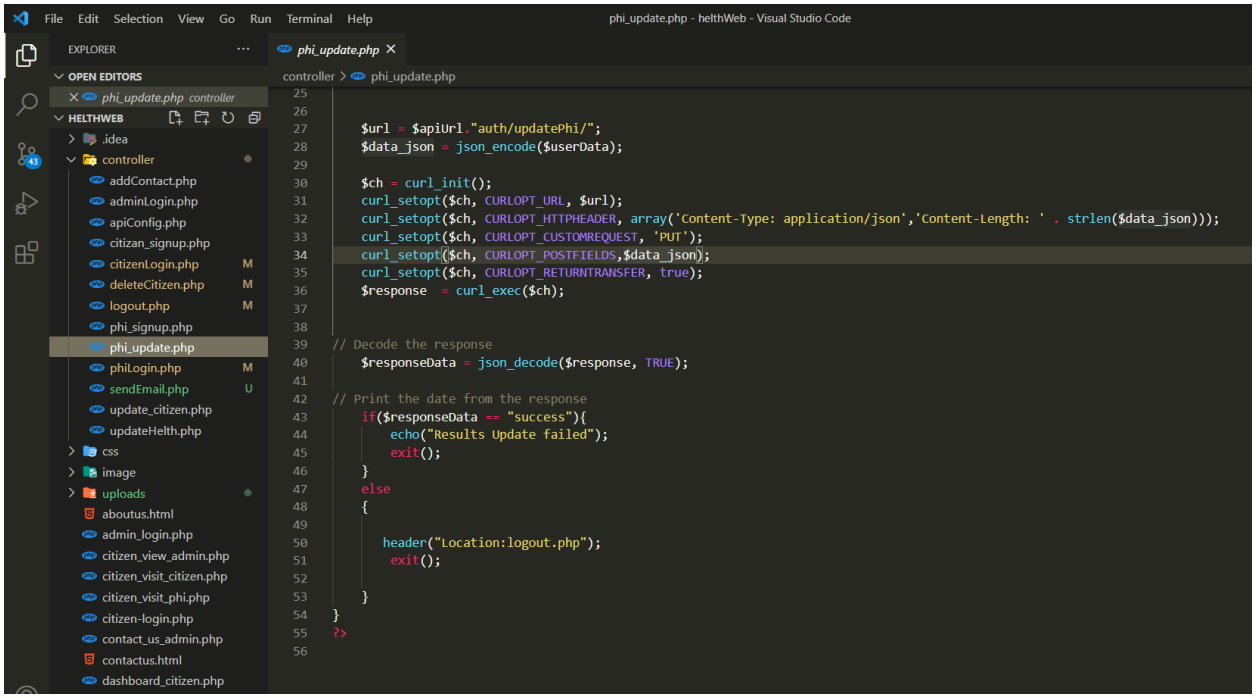

File Edit Selection View Go Run Terminal Help
auth.js - e-health-API-master - Visual Studio Code
EXPLORER
auth.js
OPEN EDITORS
auth.js routes
E-HEALTH-API-MASTER
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
Procfile
server.js
OUTLINE
routes > auth.js > router.post('/addAdmin') callback > then() callback
//phi signup route
router.post('/phi/signup', (req, res, next) => {
Phi.find({ email: req.body.email })
.exec()
.then(phi => { //check if the email is allready taken
if (phi.length >= 1) {
return res.status(409).json({
message: 'Email Not Available. '
});
} else {
bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
if (err) {
return res.status(500).json({
error: err
});
} else {
const phi = new Phi({
_id: new mongoose.Types.ObjectId(),
fname: req.body.fname,
lname: req.body.lname,
nic: req.body.nic,
email: req.body.email,
phone: req.body.phone,
address: req.body.address,
location: req.body.location,
image: req.body.image,
password: hash
});
phi.save()
.then(result => {
console.log(result);
res.status(201).json({
message: "Sign up Sucessfully",
createdUser: result
});
})

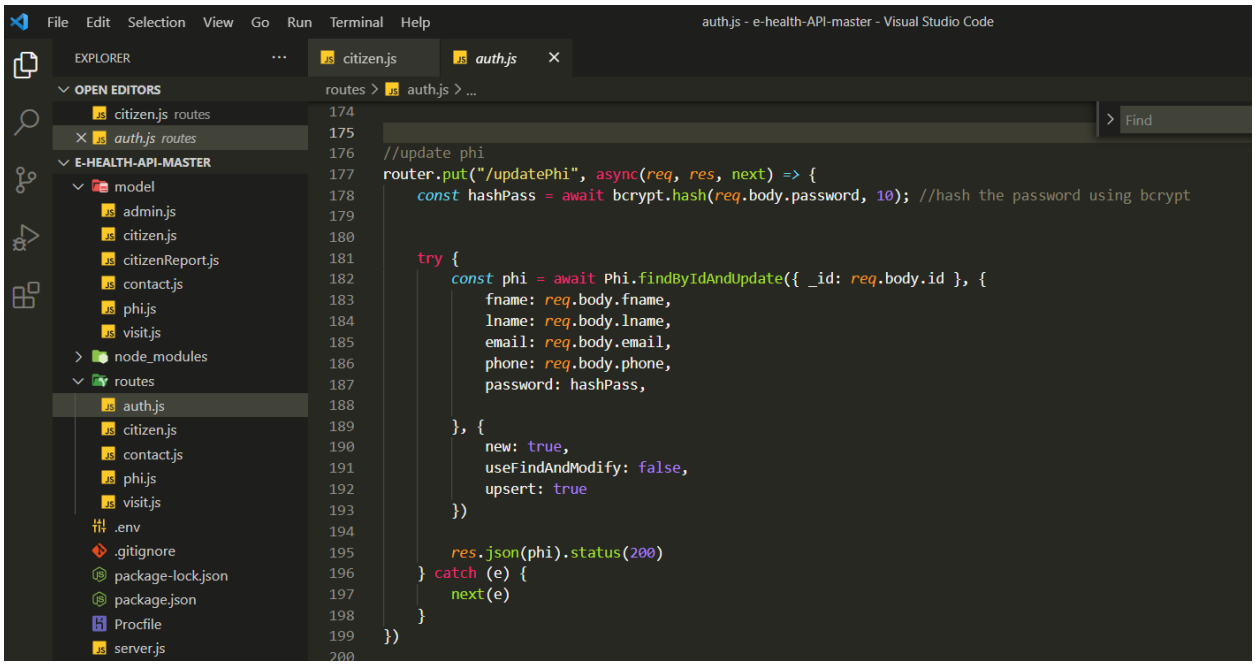

File Edit Selection View Go Run Terminal Help
auth.js - e-health-API-master - Visual Studio Code
EXPLORER
auth.js
OPEN EDITORS
auth.js routes
E-HEALTH-API-MASTER
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
routes > auth.js > router.post('/addAdmin') callback > then() callback
});
phi.save()
.then(result => {
console.log(result);
res.status(201).json({
message: "Sign up Sucessfully",
createdUser: result
});
})
.catch(err => {
console.log(err);
res.status(500).json({
error: err
});
})
}
})
}
})
});

3.PHI Update Profile : PHI can update there profile. This is the profile updating part.

Routes: `http://localhost:8000/auth/updatePhi`

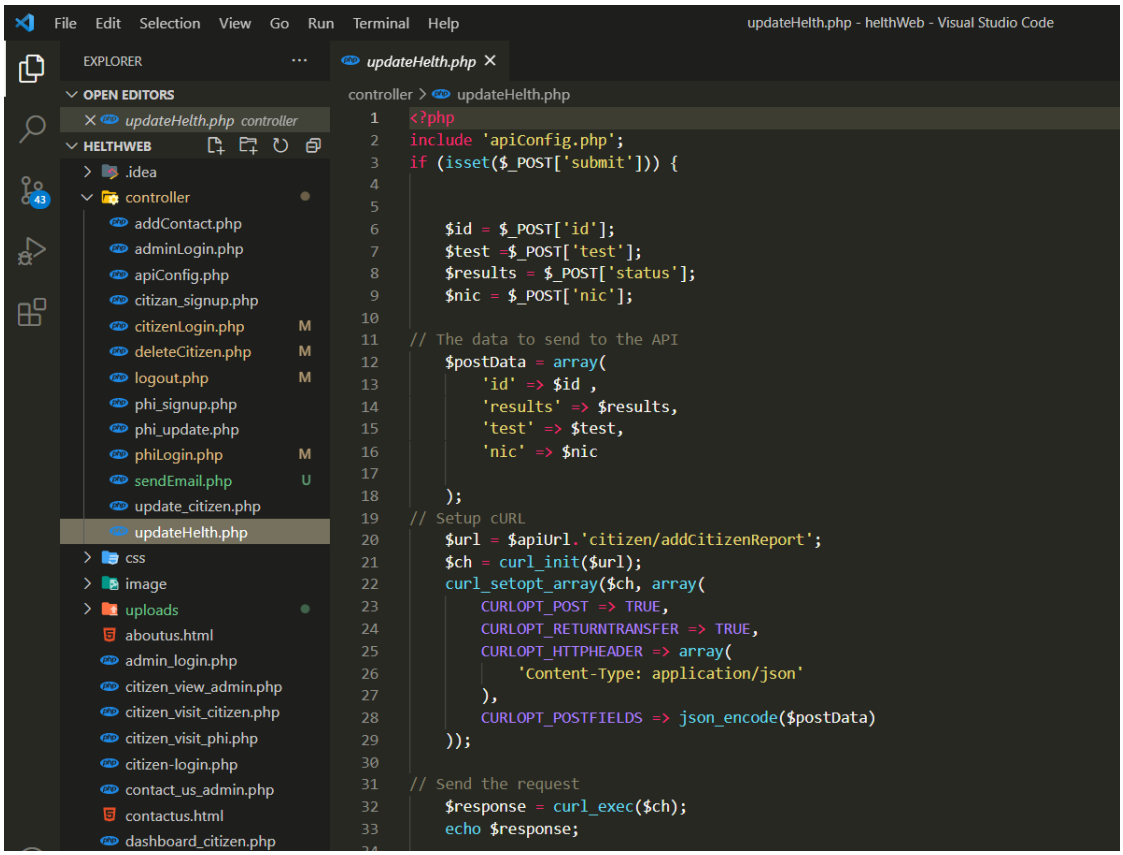


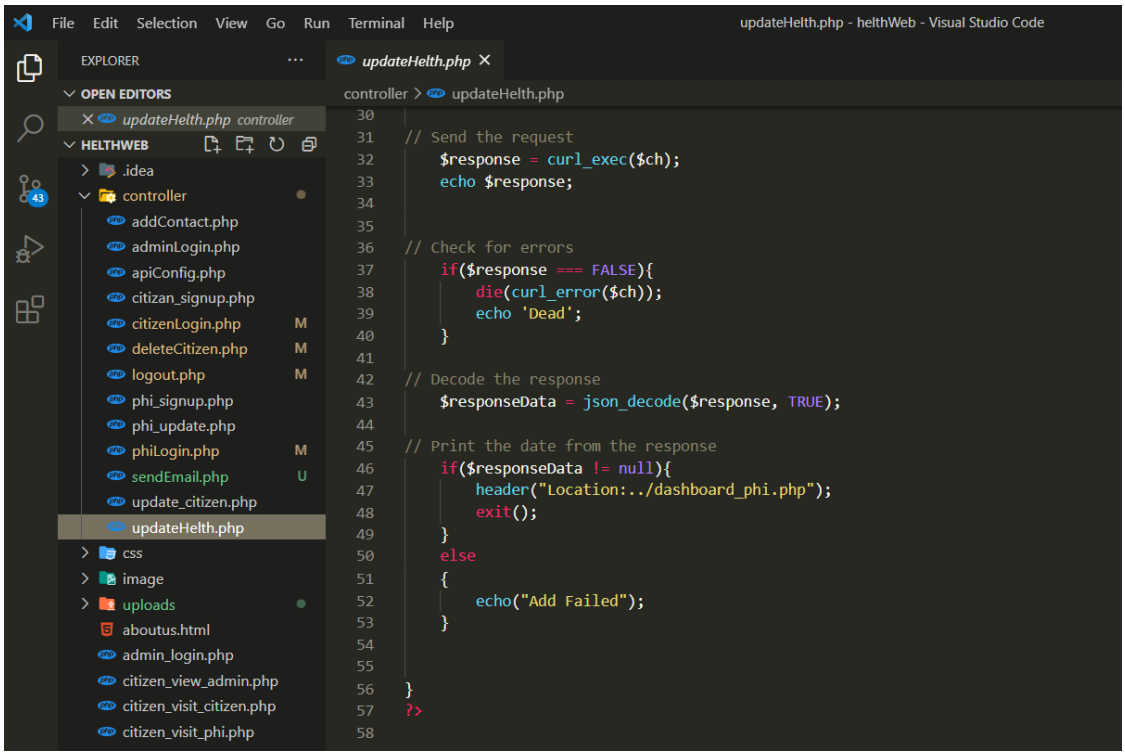




4.Update citizen health : PHI can update the health status of the citizen. (Positive, Negative). So this is the citizen health updating part.

Routes: http://localhost:8000/citizen/addCitizenReport




File Edit Selection View Go Run Terminal Help
updateHelth.php - helthWeb - Visual Studio Code
EXPLORER
updateHelth.php
OPEN EDITORS
updateHelth.php controller
HELTHWEB
.idea
controller
addContact.php
adminLogin.php
apiConfig.php
citizan_signup.php
citizenLogin.php M
deleteCitizen.php M
logout.php M
phi_signup.php
phi_update.php
phiLogin.php M
sendEmail.php U
update_citizen.php
updateHelth.php
css
image
uploads
aboutus.html
admin_login.php
citizen_view_admin.php
citizen_visit_citizen.php
citizen_visit_phi.php
controller > updateHelth.php
30
31 // Send the request
32 $response = curl_exec($ch);
33 echo $response;
34
35
36 // Check for errors
37 if($response === FALSE){
38 die(curl_error($ch));
39 echo 'Dead';
40 }
41
42 // Decode the response
43 $responseData = json_decode($response, TRUE);
44
45 // Print the date from the response
46 if($responseData != null){
47 header("Location:../dashboard_phi.php");
48 exit();
49 }
50 else
51 {
52 echo("Add Failed");
53 }
54
55
56 }
57 ?>
58

c) Controller 3: Citizen

1. Citizen Login : This is the citizen login part.

Routes: http://localhost:8000/auth/citizenlogin

citizenLogin.php

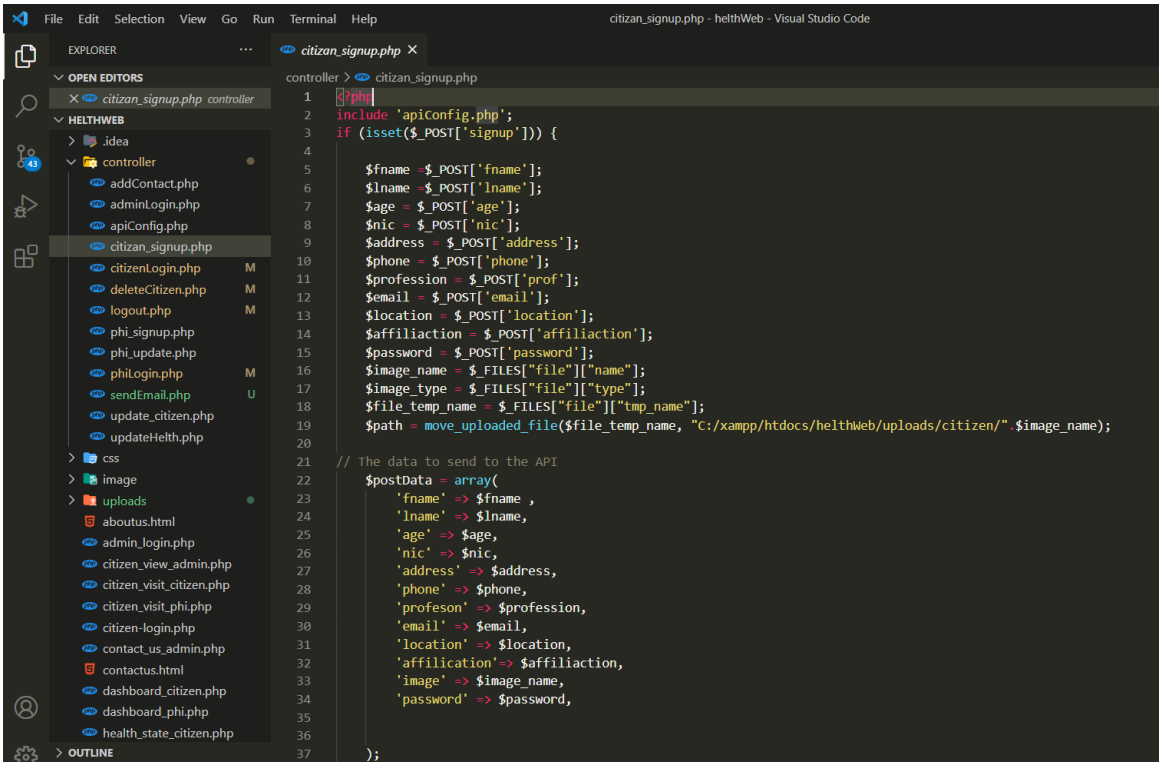
```php
<?php
include 'apiConfig.php';
if (isset($_POST['login'])) {
    session_start();
    unset($_SESSION["token1"]);
    unset($_SESSION["image1"]);

    $email =$_POST['email'];
    $password =$_POST['pass'];

// login data  send to the API
    $postData = array(
        'password' => $password,
        'email' => $email,
    );

// Setup cURL

    $ch = curl_init('http://localhost:8000/auth/citizenlogin');
    curl_setopt_array($ch, array(
        CURLOPT_POST => TRUE,
        CURLOPT_RETURNTRANSFER => TRUE,
        CURLOPT_HTTPHEADER => array(
            'Content-Type: application/json'
        ),
        CURLOPT_POSTFIELDS => json_encode($postData)
    ));

// Send the request
    $response = curl_exec($ch);

// Check for errors
    if($response === FALSE){
        die(curl_error($ch));
        echo 'Dead';
    }

// Decode the response
    $responseData = json_decode($response, TRUE);
    echo($responseData['token'] );

// Print the date from the response

    $token = $responseData['token'];
    $id = $responseData['id'];
    $email= $responseData['email'];
    $fname= $responseData['fname'];
```

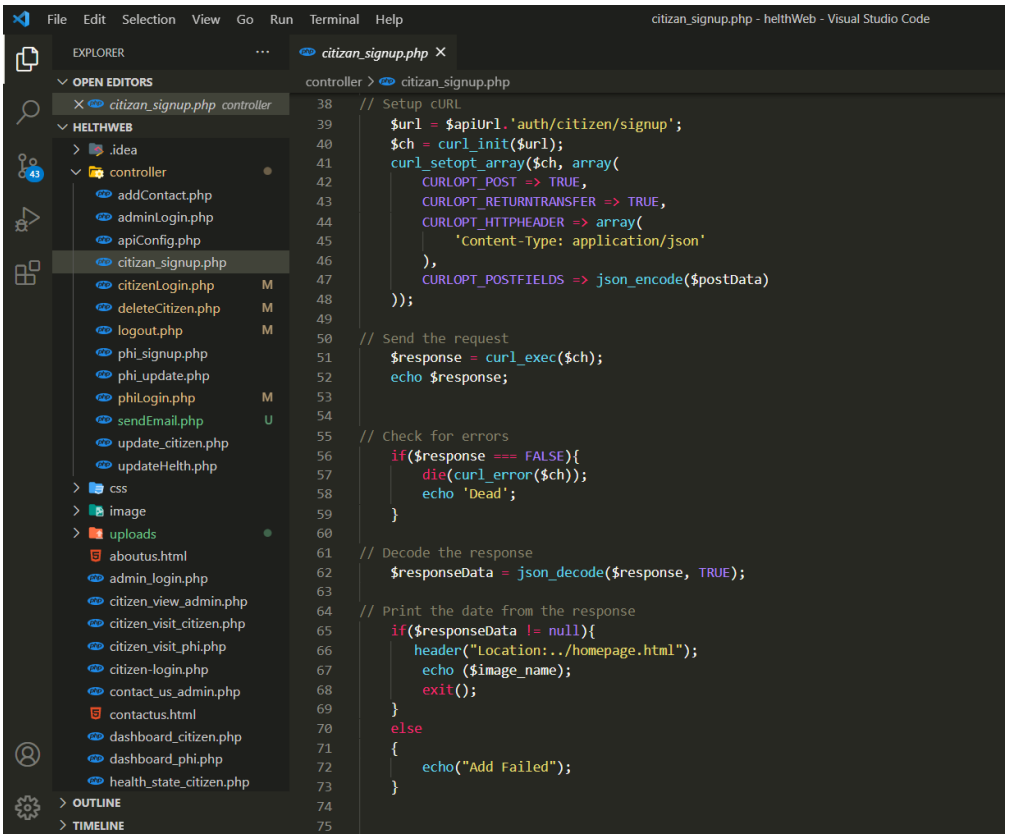
```php
// Decode the response
    $responseData = json_decode($response, TRUE);
    echo($responseData['token'] );

// Print the date from the response

    $token = $responseData['token'];
    $id = $responseData['id'];
    $email= $responseData['email'];
    $fname= $responseData['fname'];
    $image = $responseData['image'];
    $nic = $responseData['nic'];
    $_SESSION["token1"] =$token;
    $_SESSION['email1'] =$email;
    $_SESSION['fname1'] =$fname;
    $_SESSION['image1'] =$image;
    $_SESSION['nic'] =$nic;
    $_SESSION['cid'] =$id;
    $_SESSION['addr'] =$responseData['address'];
    $_SESSION['lct'] =$responseData['location'];
    $_SESSION['aff'] =$responseData['affilication'];
    $_SESSION['prof'] =$responseData['profeson'];
    $_SESSION['lname1'] =$responseData['lname'];
    $_SESSION['phone1'] =$responseData['phone'];
    $_SESSION['age1'] =$responseData['age'];
    if($token != null   ){
        //echo ($image);
        header("Location:../dashboard_citizen.php");
        exit();
    }

    else{

        echo '<script language="javascript">';
        echo 'alert("Login error");';
        echo 'window.location.href="../homepage.html";';
        echo '</script>';
        exit();
    }

}
?>
```

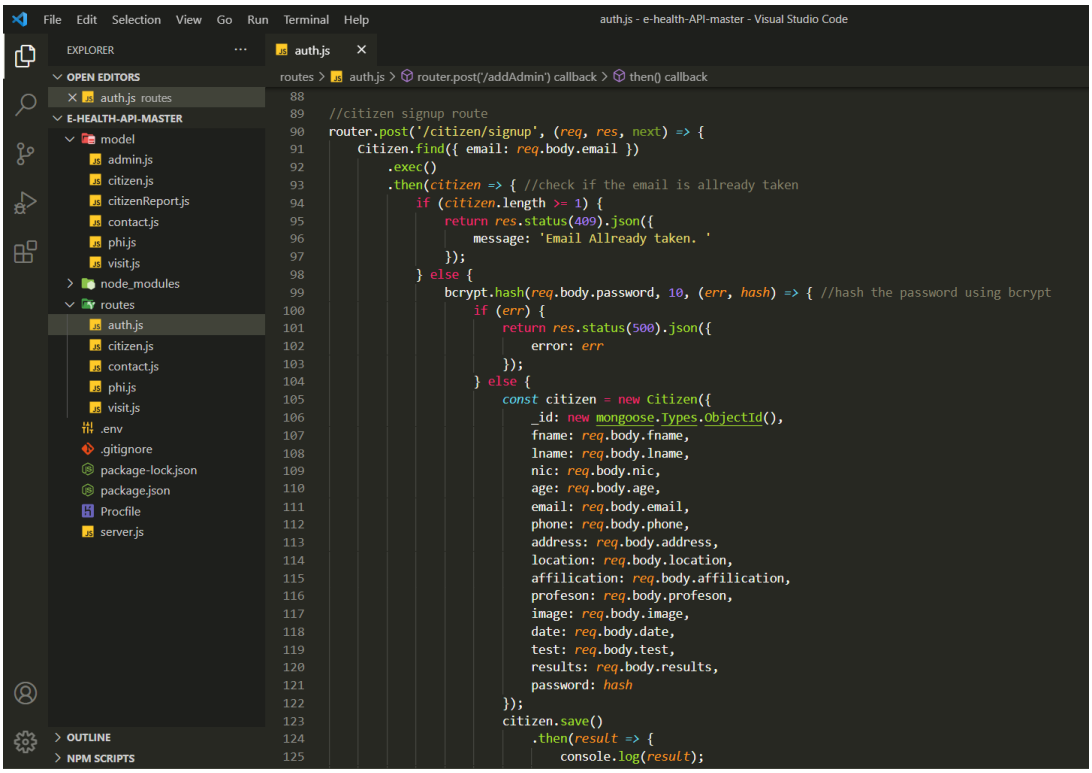
```
//login Citizen
router.post('/citizenlogin', async(req, res) => {
    const citizen = await Citizen.findOne({ email: req.body.email });
    if (!citizen) return res.status(400).send({ message: 'Email does Not Exist' });

    const validPass = await bcrypt.compare(req.body.password, citizen.password);

    if (!validPass) return res.status(400).send({ message: 'Password is Wrong' });

    const token = jwt.sign({ _id: citizen._id }, process.env.JWT_KEY);


    res.header('auth-token').send({
        id: citizen._id,
        fname: citizen.fname,
        email: citizen.email,
        nic: citizen.nic,
        image: citizen.image,
        token: token,
        lname: citizen.lname,
        phone: citizen.phone,
        age: citizen.age,
        affilication: citizen.affilication,
        profeson: citizen.profeson,
        address: citizen.address,
        message: citizen.fname,
        status: citizen.status
    });
});
```

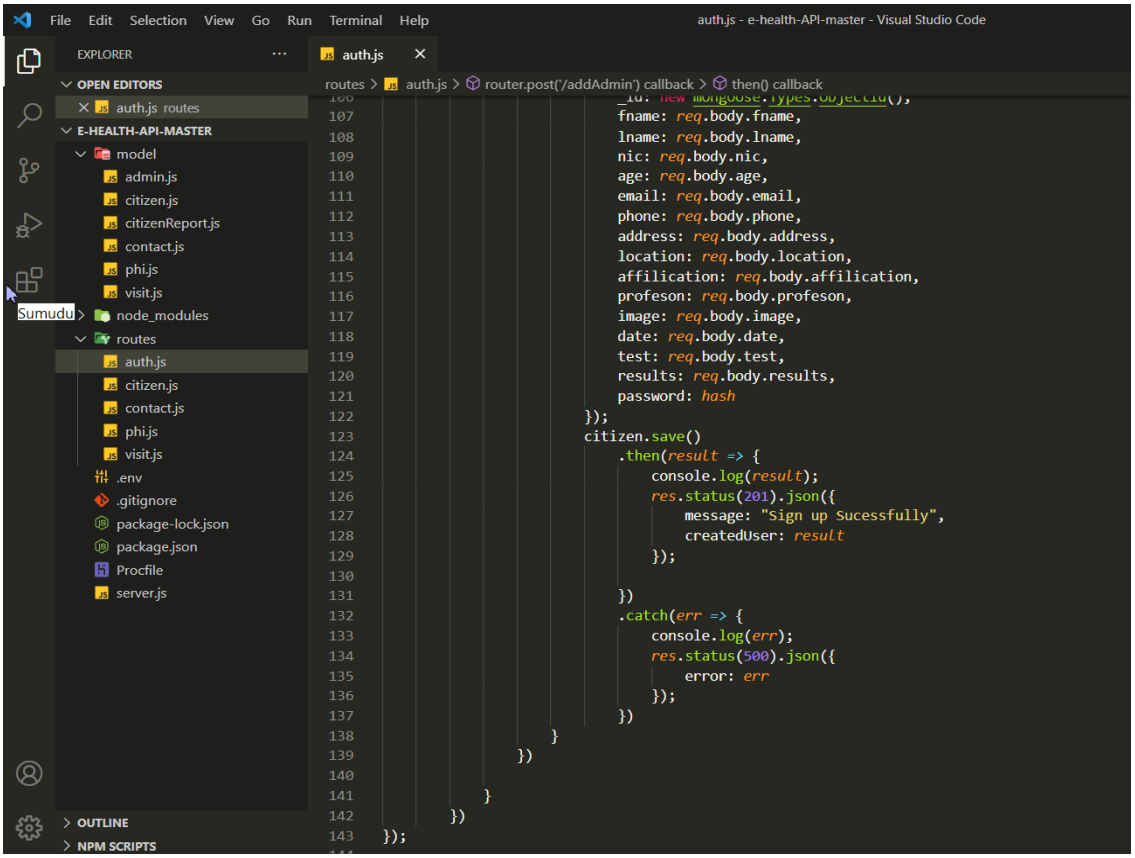
2. Citizen Sign Up : Ths is the citizen register part.

Routes: http://localhost:8000/auth/citizen/signup



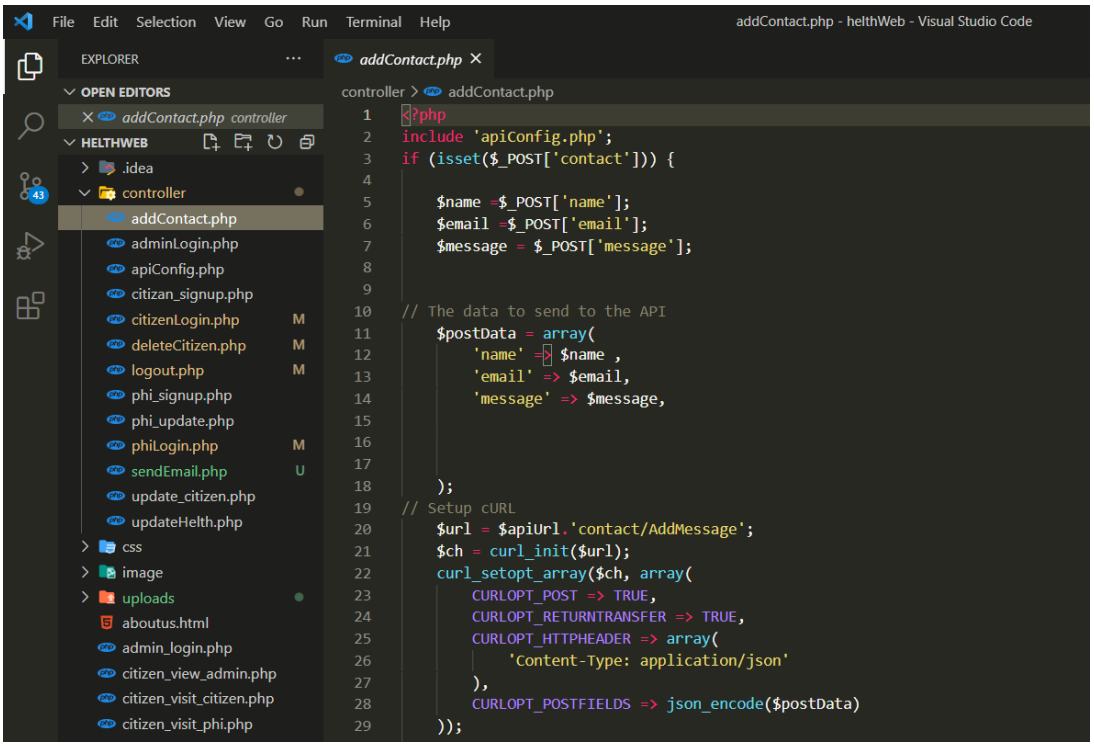

File Edit Selection View Go Run Terminal Help
citizan_signup.php - helthWeb - Visual Studio Code
EXPLORER
citizan_signup.php
OPEN EDITORS
citizan_signup.php controller
HELTHWEB
.idea
controller
addContact.php
adminLogin.php
apiConfig.php
citizan_signup.php
citizenLogin.php M
deleteCitizen.php M
logout.php M
phi_signup.php
phi_update.php
phiLogin.php M
sendEmail.php U
update_citizen.php
updateHelth.php
css
image
uploads
aboutus.html
admin_login.php
citizen_view_admin.php
citizen_visit_citizen.php
citizen_visit_phi.php
citizen-login.php
contact_us_admin.php
contactus.html
dashboard_citizen.php
dashboard_phi.php
health_state_citizen.php
OUTLINE
TIMELINE
controller > citizan_signup.php
// Setup cURL
$url = $apiUrl.'auth/citizen/signup';
$ch = curl_init($url);
curl_setopt_array($ch, array(
CURLOPT_POST => TRUE,
CURLOPT_RETURNTRANSFER => TRUE,
CURLOPT_HTTPHEADER => array(
'Content-Type: application/json'
),
CURLOPT_POSTFIELDS => json_encode($postData)
));
// Send the request
$response = curl_exec($ch);
echo $response;
// Check for errors
if($response === FALSE){
die(curl_error($ch));
echo 'Dead';
}
// Decode the response
$responseData = json_decode($response, TRUE);
// Print the date from the response
if($responseData != null){
header("Location:../homepage.html");
echo ($image_name);
exit();
}
else
{
echo("Add Failed");
}

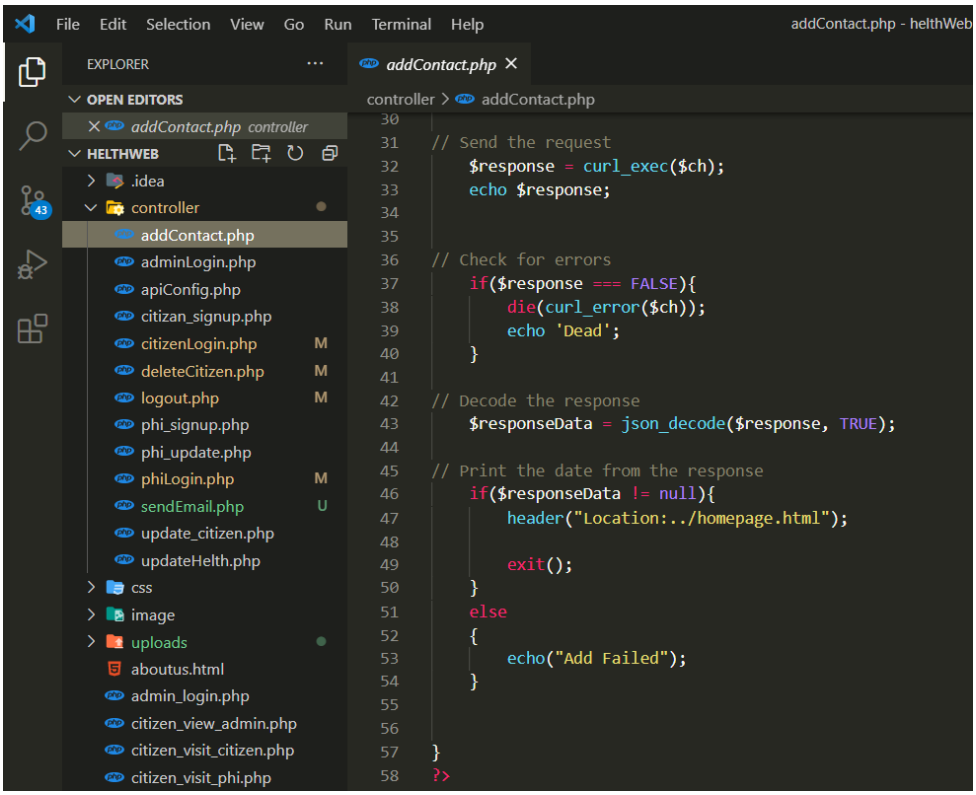

File Edit Selection View Go Run Terminal Help
auth.js - e-health-API-master - Visual Studio Code
EXPLORER
auth.js
OPEN EDITORS
auth.js routes
E-HEALTH-API-MASTER
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
Procfile
server.js
OUTLINE
NPM SCRIPTS
routes > auth.js > router.post('/addAdmin') callback > then() callback
//citizen signup route
router.post('/citizen/signup', (req, res, next) => {
Citizen.find({ email: req.body.email })
.exec()
.then(citizen => { //check if the email is allready taken
if (citizen.length >= 1) {
return res.status(409).json({
message: 'Email Allready taken. '
});
} else {
bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
if (err) {
return res.status(500).json({
error: err
});
} else {
const citizen = new Citizen({
_id: new mongoose.Types.ObjectId(),
fname: req.body.fname,
lname: req.body.lname,
nic: req.body.nic,
age: req.body.age,
email: req.body.email,
phone: req.body.phone,
address: req.body.address,
location: req.body.location,
affilication: req.body.affilication,
profeson: req.body.profeson,
image: req.body.image,
date: req.body.date,
test: req.body.test,
results: req.body.results,
password: hash
});
citizen.save()
.then(result => {
console.log(result);

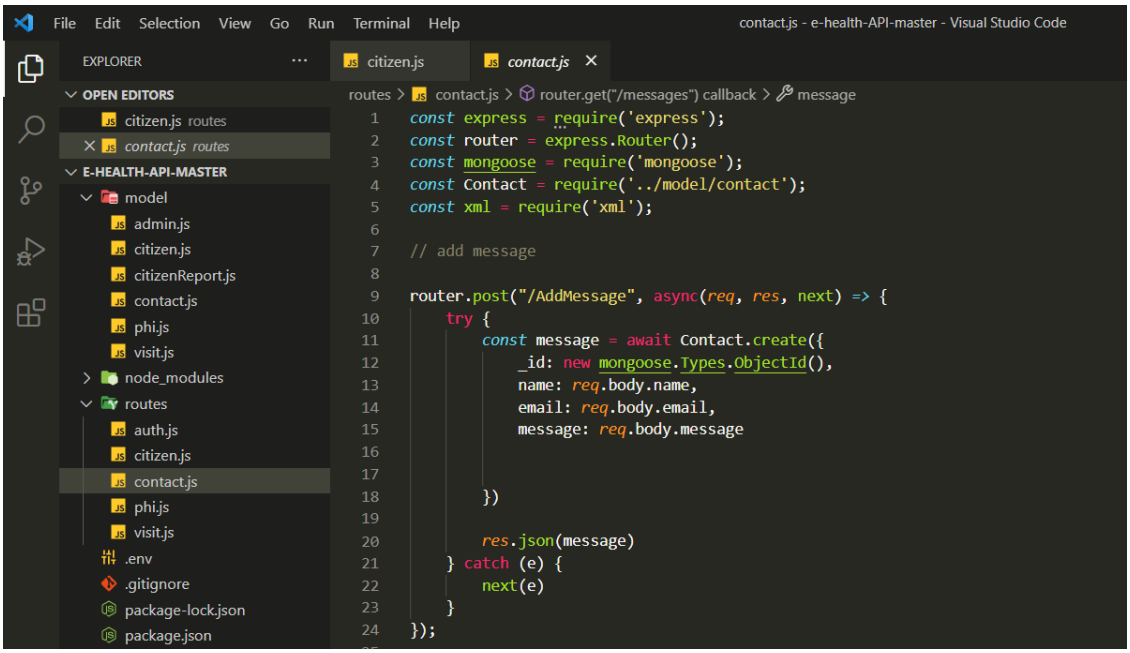


3. Contact Us: Citizen can contact admin by sending any message using the contact us part.

Routes: http://localhost:8000/contact/AddMessage

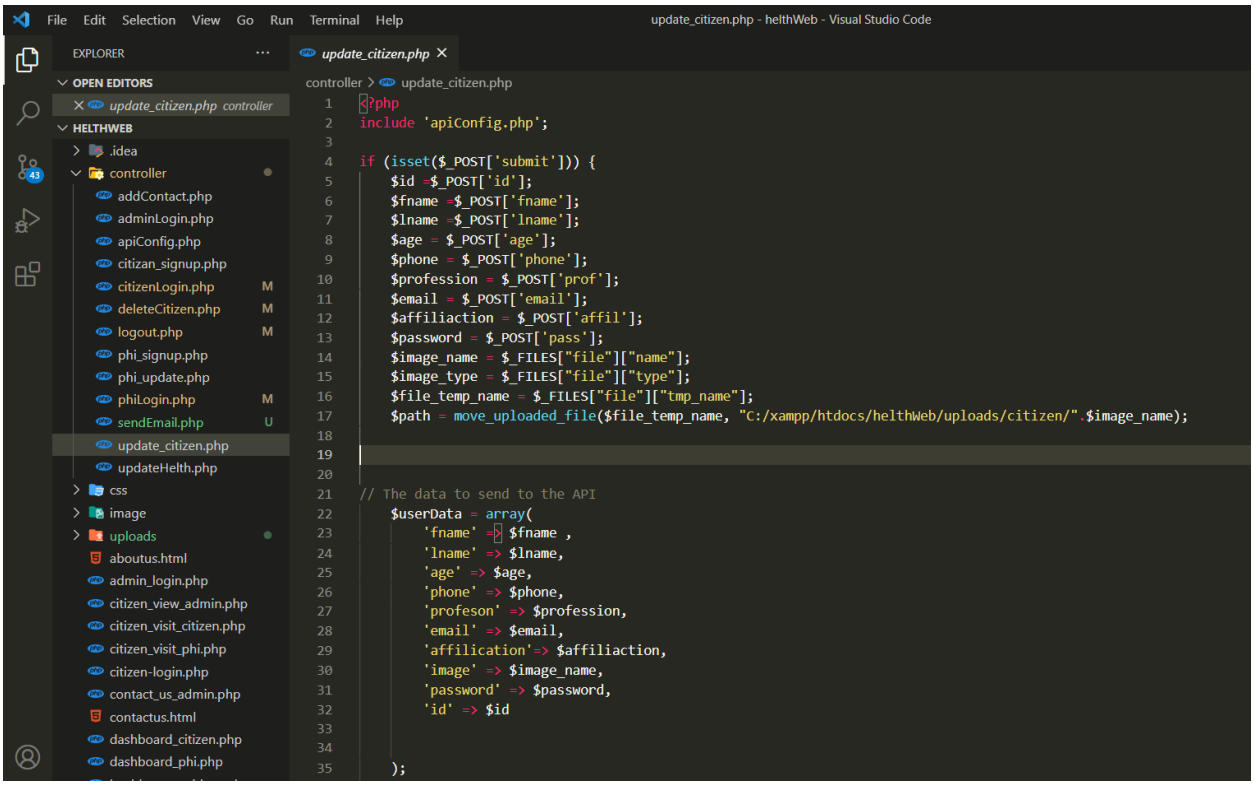



File Edit Selection View Go Run Terminal Help
addContact.php - helthWeb
EXPLORER
addContact.php
OPEN EDITORS
addContact.php controller
HELTHWEB
.idea
controller
addContact.php
adminLogin.php
apiConfig.php
citizan_signup.php
citizenLogin.php M
deleteCitizen.php M
logout.php M
phi_signup.php
phi_update.php
phiLogin.php M
sendEmail.php U
update_citizen.php
updateHelth.php
css
image
uploads
aboutus.html
admin_login.php
citizen_view_admin.php
citizen_visit_citizen.php
citizen_visit_phi.php
controller > addContact.php
// Send the request
$response = curl_exec($ch);
echo $response;
// Check for errors
if($response === FALSE){
die(curl_error($ch));
echo 'Dead';
}
// Decode the response
$responseData = json_decode($response, TRUE);
// Print the date from the response
if($responseData != null){
header("Location:../homepage.html");
exit();
}
else
{
echo("Add Failed");
}
}
?>

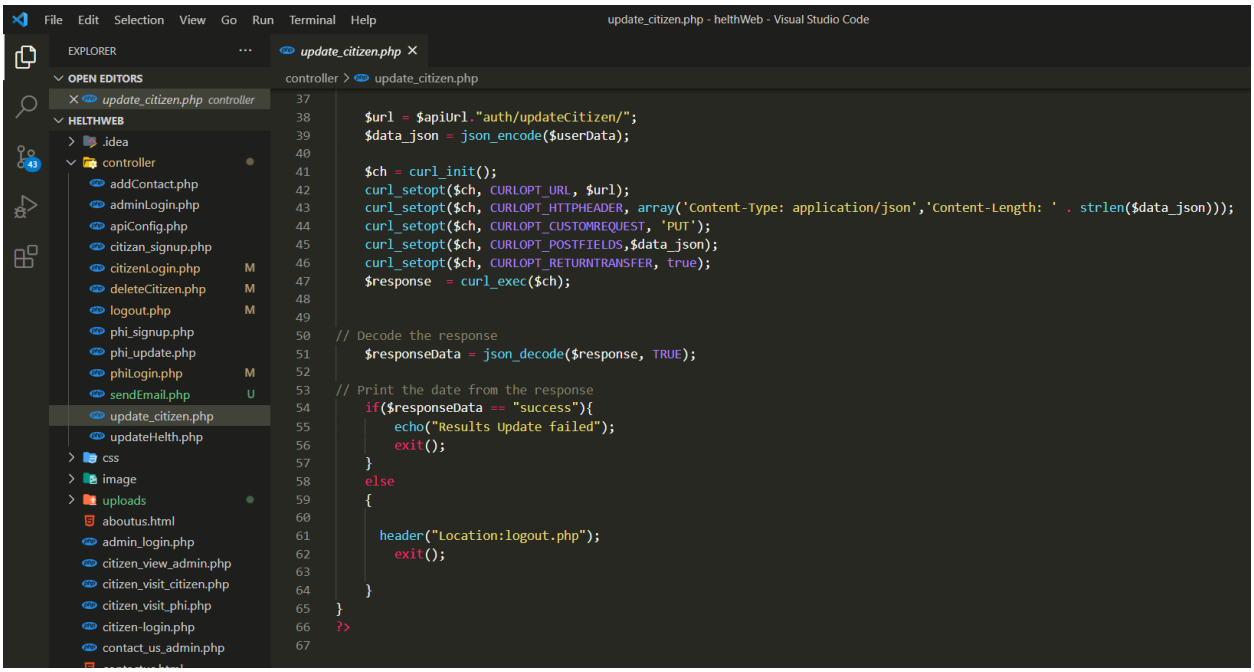

File Edit Selection View Go Run Terminal Help
contact.js - e-health-API-master - Visual Studio Code
EXPLORER
citizen.js
contact.js
OPEN EDITORS
citizen.js routes
contact.js routes
E-HEALTH-API-MASTER
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
routes > contact.js > router.get("/messages") callback > message
const express = require('express');
const router = express.Router();
const mongoose = require('mongoose');
const Contact = require('../model/contact');
const xml = require('xml');
// add message
router.post("/AddMessage", async(req, res, next) => {
try {
const message = await Contact.create({
_id: new mongoose.Types.ObjectId(),
name: req.body.name,
email: req.body.email,
message: req.body.message
})
res.json(message)
} catch (e) {
next(e)
}
});

4. Update Citizen : Citizen can update there profile. This is the profile updating part.

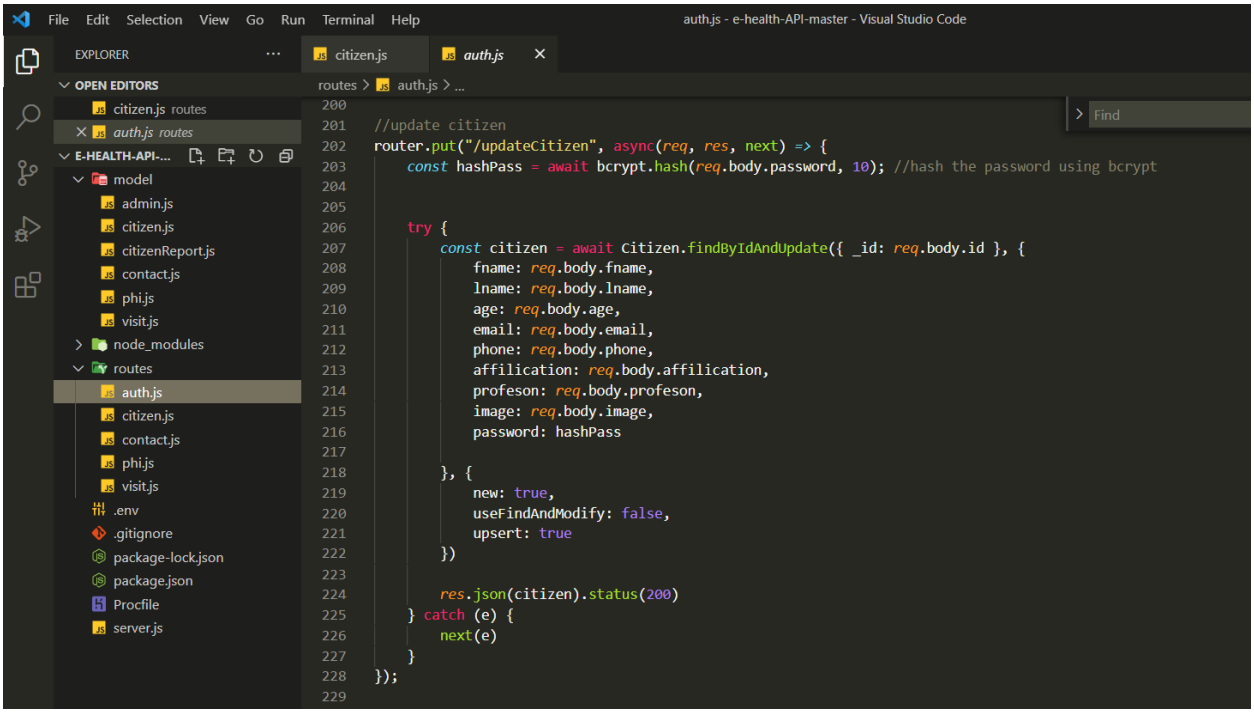
Routes: http://localhost:8000/auth/updateCitizen



```php
<?php
include 'apiConfig.php';

if (isset($_POST['submit'])) {
    $id =$_POST['id'];
    $fname =$_POST['fname'];
    $lname =$_POST['lname'];
    $age = $_POST['age'];
    $phone = $_POST['phone'];
    $profession = $_POST['prof'];
    $email = $_POST['email'];
    $affiliaction = $_POST['affil'];
    $password = $_POST['pass'];
    $image_name = $_FILES["file"]["name"];
    $image_type = $_FILES["file"]["type"];
    $file_temp_name = $_FILES["file"]["tmp_name"];
    $path = move_uploaded_file($file_temp_name, "C:/xampp/htdocs/helthWeb/uploads/citizen/".$image_name);


// The data to send to the API
    $userData = array(
        'fname' => $fname ,
        'lname' => $lname,
        'age' => $age,
        'phone' => $phone,
        'profeson' => $profession,
        'email' => $email,
        'affilication'=> $affiliaction,
        'image' => $image_name,
        'password' => $password,
        'id' => $id


    );
```

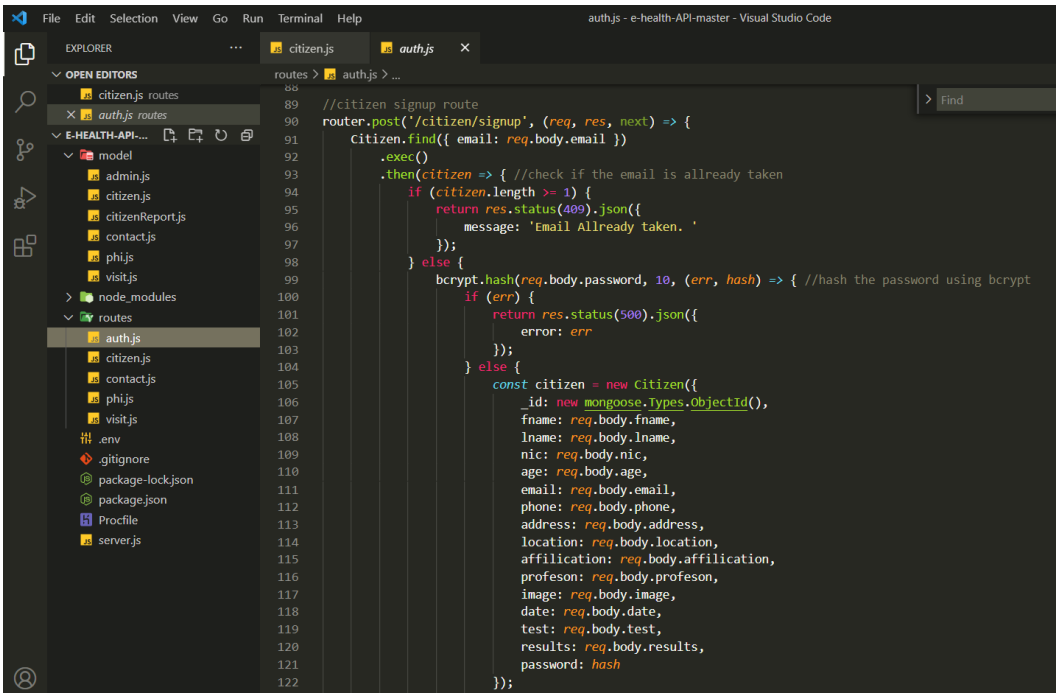


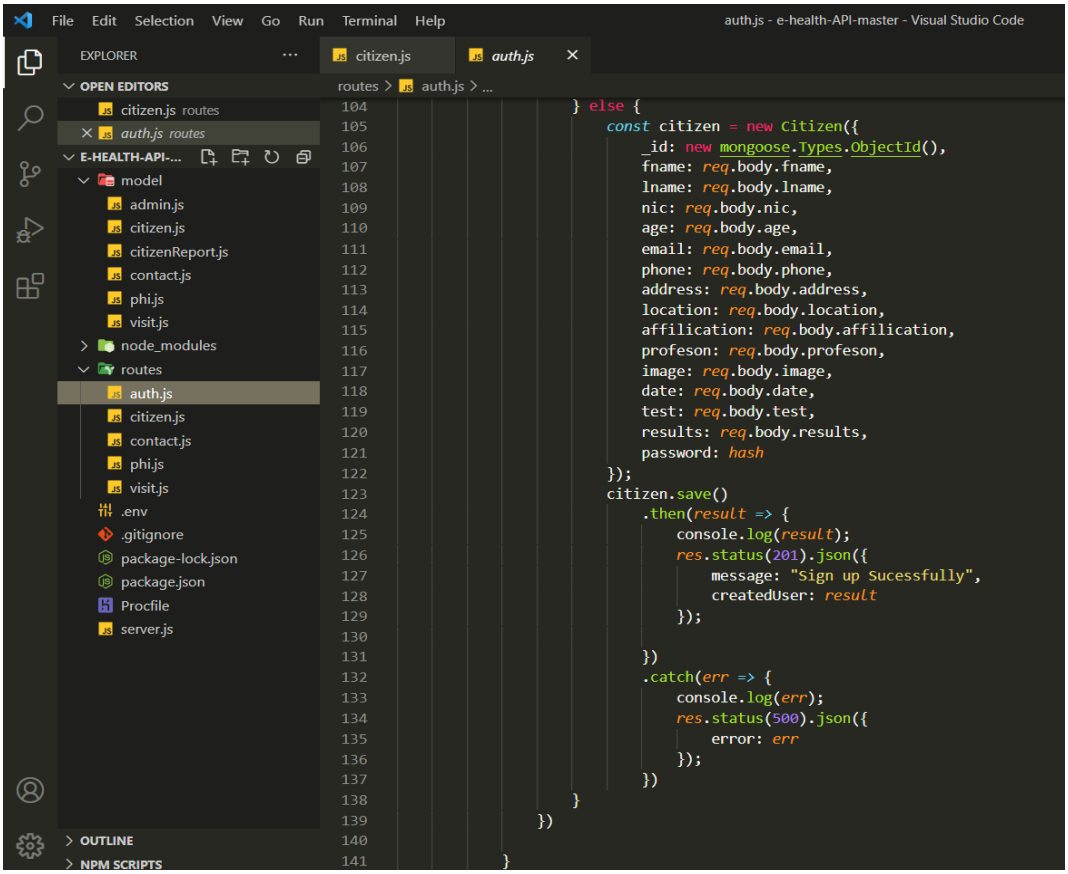
```php
    $url = $apiUrl."auth/updateCitizen/";
    $data_json = json_encode($userData);

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json','Content-Length: ' . strlen($data_json)));
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
    curl_setopt($ch, CURLOPT_POSTFIELDS,$data_json);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $response  = curl_exec($ch);


// Decode the response
    $responseData = json_decode($response, TRUE);

// Print the date from the response
    if($responseData == "success"){
        echo("Results Update failed");
        exit();
    }
    else
    {

      header("Location:logout.php");
        exit();

    }
}
?>
```

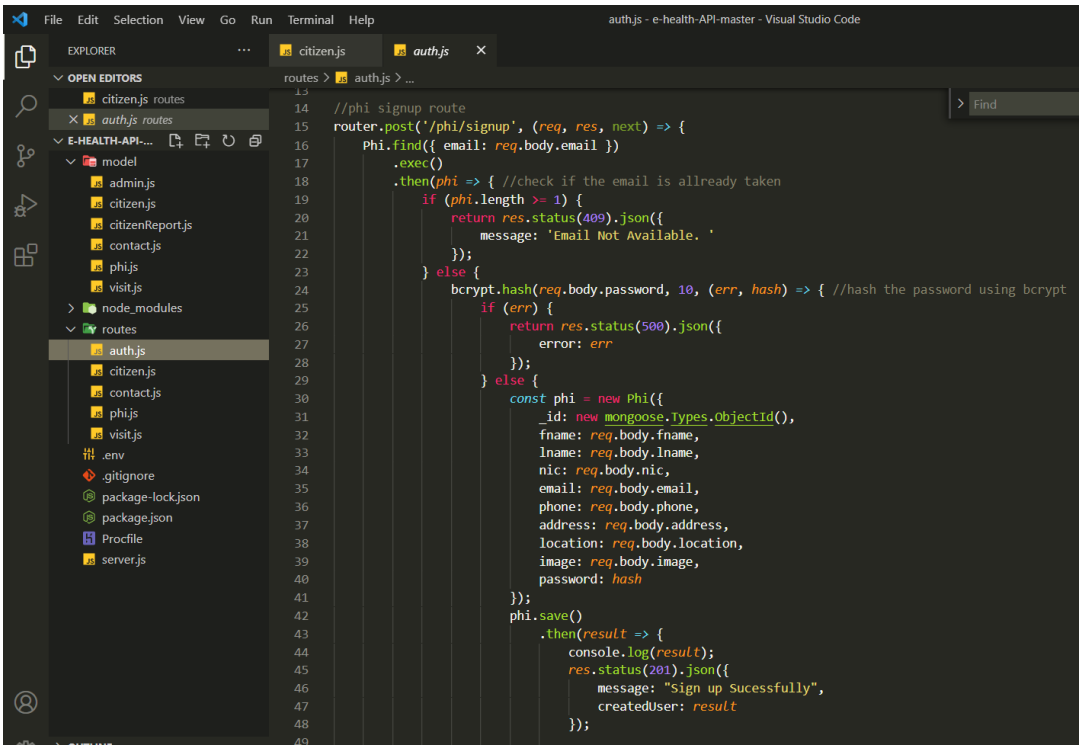

File Edit Selection View Go Run Terminal Help
auth.js - e-health-API-master - Visual Studio Code
EXPLORER
OPEN EDITORS
citizen.js routes
auth.js routes
E-HEALTH-API-...
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
Procfile
server.js
citizen.js
auth.js
routes > auth.js > ...
Find
200
201 //update citizen
202 router.put("/updateCitizen", async(req, res, next) => {
203 const hashPass = await bcrypt.hash(req.body.password, 10); //hash the password using bcrypt
204
205
206 try {
207 const citizen = await Citizen.findByIdAndUpdate({ _id: req.body.id }, {
208 fname: req.body.fname,
209 lname: req.body.lname,
210 age: req.body.age,
211 email: req.body.email,
212 phone: req.body.phone,
213 affilication: req.body.affilication,
214 profeson: req.body.profeson,
215 image: req.body.image,
216 password: hashPass
217
218 }, {
219 new: true,
220 useFindAndModify: false,
221 upsert: true
222 })
223
224 res.json(citizen).status(200)
225 } catch (e) {
226 next(e)
227 }
228 });
229

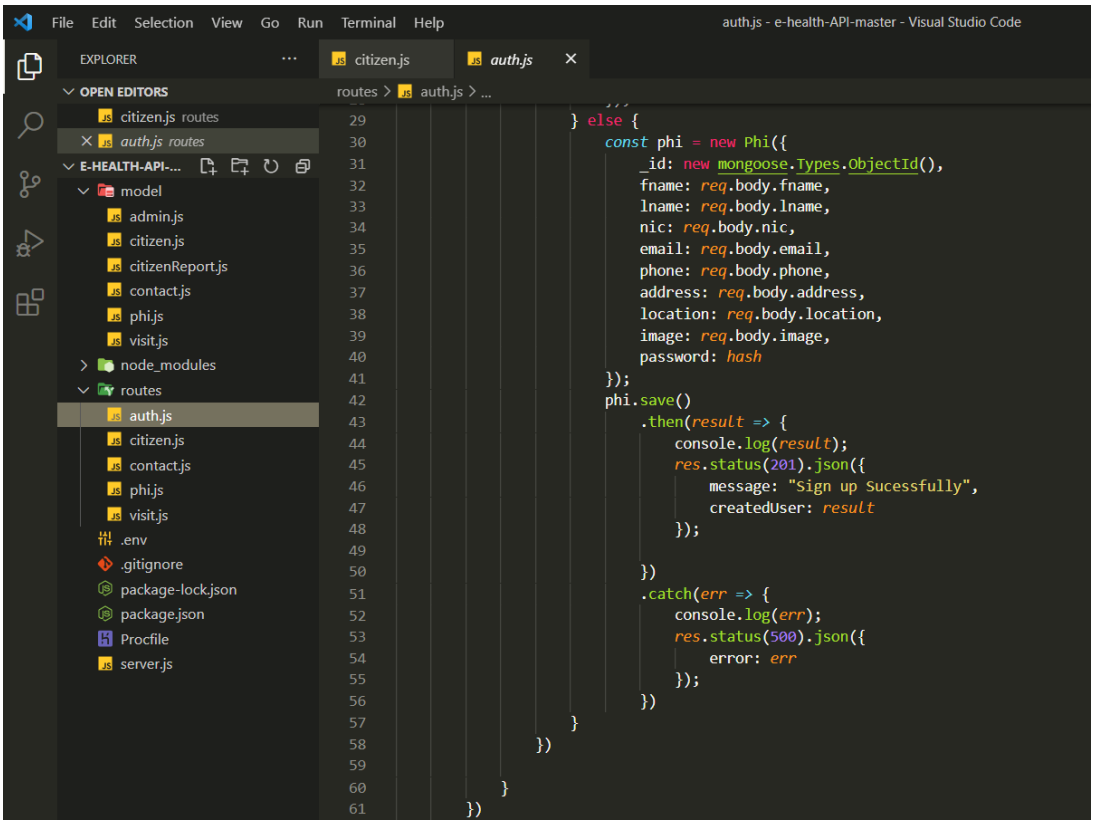
Minimal Required APIs

1. POST (citizens and PHIs can register them self)



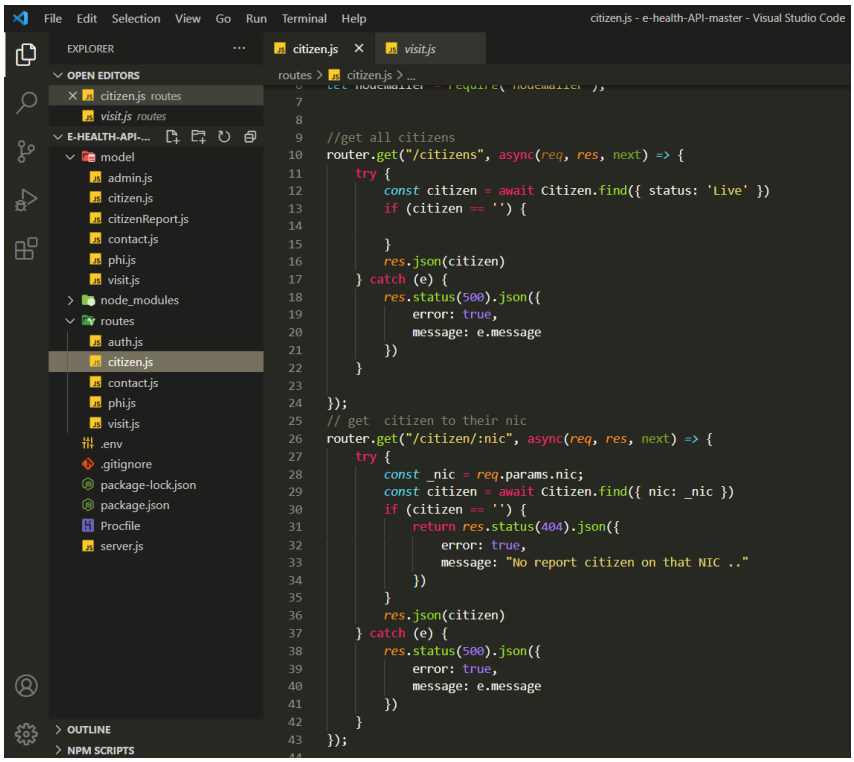



File Edit Selection View Go Run Terminal Help
auth.js - e-health-API-master - Visual Studio Code
EXPLORER
citizen.js
auth.js
routes > auth.js > ...
OPEN EDITORS
citizen.js routes
auth.js routes
E-HEALTH-API-...
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
Procfile
server.js
Find
//phi signup route
router.post('/phi/signup', (req, res, next) => {
Phi.find({ email: req.body.email })
.exec()
.then(phi => { //check if the email is allready taken
if (phi.length >= 1) {
return res.status(409).json({
message: 'Email Not Available. '
});
} else {
bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
if (err) {
return res.status(500).json({
error: err
});
} else {
const phi = new Phi({
_id: new mongoose.Types.ObjectId(),
fname: req.body.fname,
lname: req.body.lname,
nic: req.body.nic,
email: req.body.email,
phone: req.body.phone,
address: req.body.address,
location: req.body.location,
image: req.body.image,
password: hash
});
phi.save()
.then(result => {
console.log(result);
res.status(201).json({
message: "Sign up Sucessfully",
createdUser: result
});

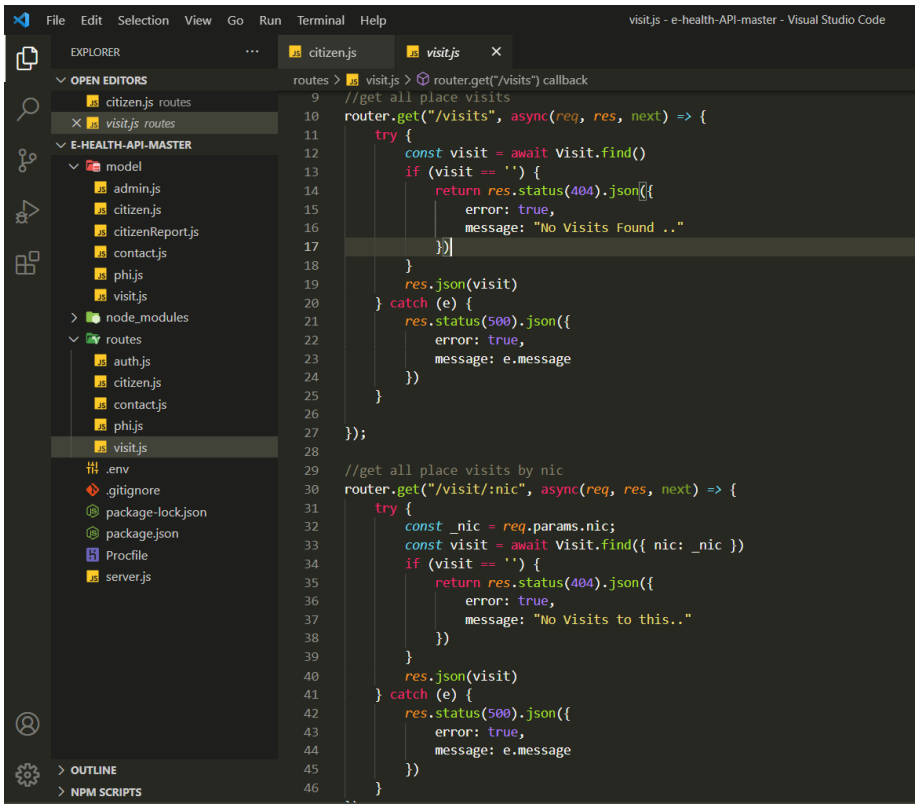

File Edit Selection View Go Run Terminal Help
auth.js - e-health-API-master - Visual Studio Code
EXPLORER
citizen.js
auth.js
routes > auth.js > ...
OPEN EDITORS
citizen.js routes
auth.js routes
E-HEALTH-API-...
model
admin.js
citizen.js
citizenReport.js
contact.js
phi.js
visit.js
node_modules
routes
auth.js
citizen.js
contact.js
phi.js
visit.js
.env
.gitignore
package-lock.json
package.json
Procfile
server.js
} else {
const phi = new Phi({
_id: new mongoose.Types.ObjectId(),
fname: req.body.fname,
lname: req.body.lname,
nic: req.body.nic,
email: req.body.email,
phone: req.body.phone,
address: req.body.address,
location: req.body.location,
image: req.body.image,
password: hash
});
phi.save()
.then(result => {
console.log(result);
res.status(201).json({
message: "Sign up Sucessfully",
createdUser: result
});
})
.catch(err => {
console.log(err);
res.status(500).json({
error: err
});
})
}
})
}
})

2.GET (CDC or PHI can collect details of any citizen)

CDC

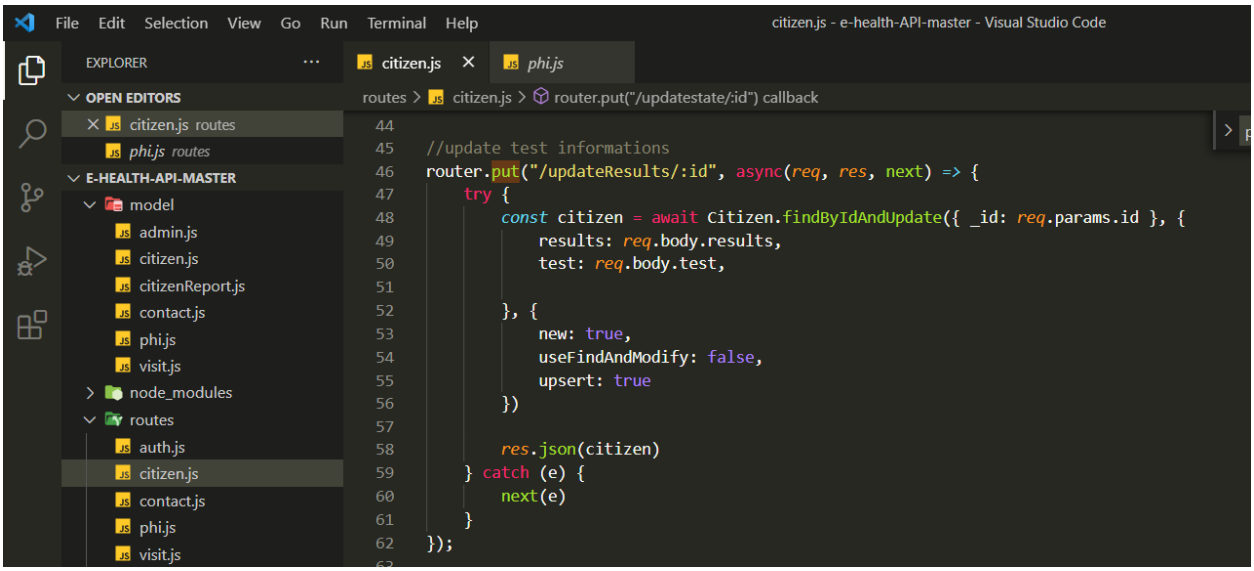


PHI



3. PUT

PHI should update clients health status

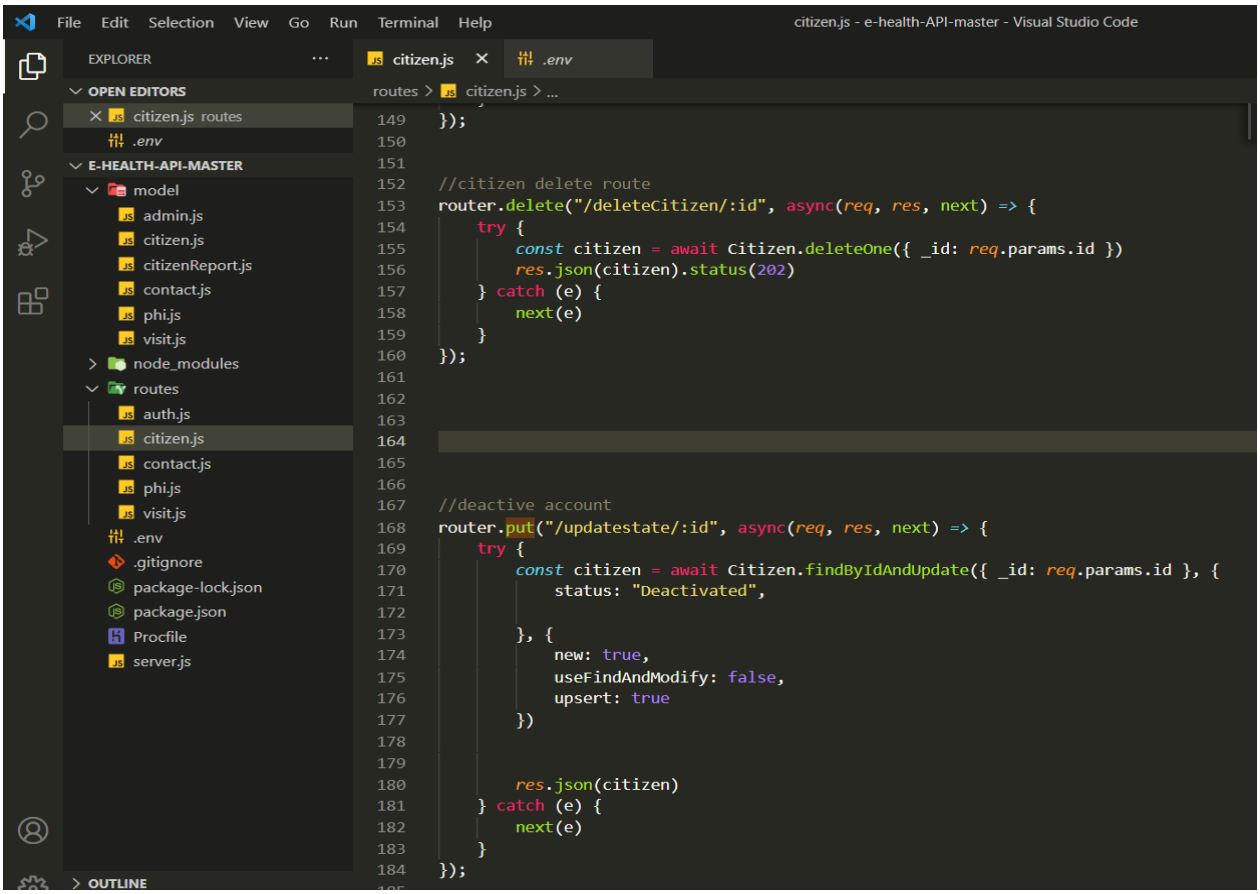


PHI can update profile details

```
//update phi
Complexity is 3 Everything is cool!
router.put("/updatePhi", async(req, res, next) => {
    const hashPass = await bcrypt.hash(req.body.password, 10); //hash the password using bcrypt


    try {
        const phi = await Phi.findByIdAndUpdate({ _id: req.body.id }, {
            fname: req.body.fname,
            lname: req.body.lname,
            email: req.body.email,
            phone: req.body.phone,
            password: hashPass,

        }, {
            new: true,
            useFindAndModify: false,
            upsert: true
        })

        res.json(phi).status(200)
    } catch (e) {
        next(e)
    }
})
```

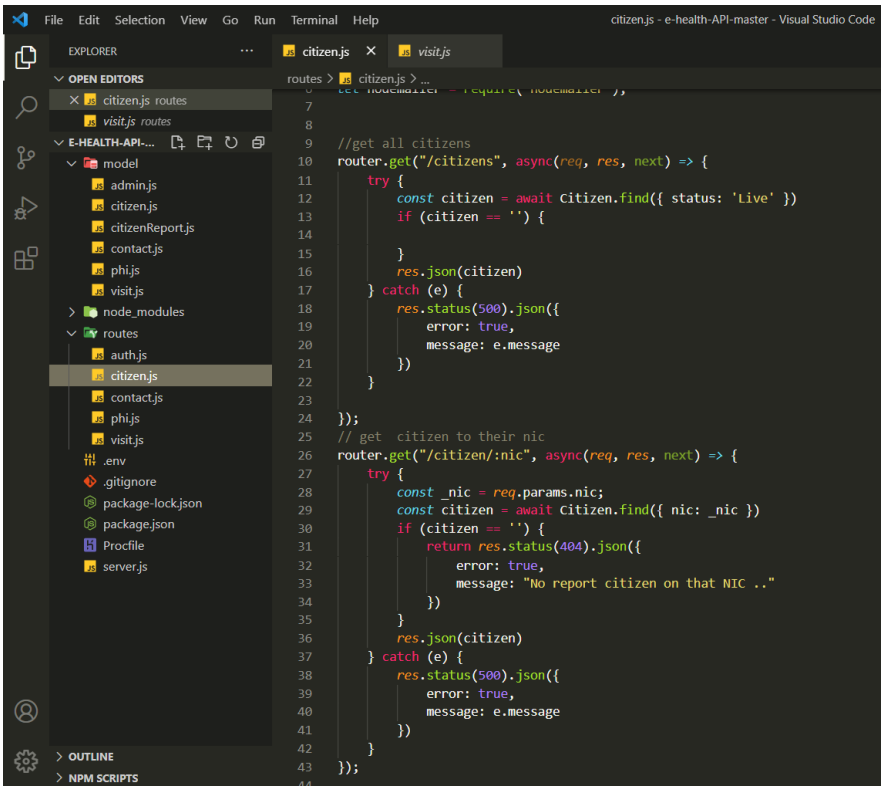
4. DELETE

Delete the positive patients accounts



5. GET

CDC can get any citizens contacts informations.



3. Tools and Technologies

These are the tools & technologies that we used to develop the E- health system.

We used, PHP, HTML, BOOTSTRAP, NODE JS and Flutter with dart language to do the front end development part by creating a web application and the mobile application.

1) (Mobile application) flutter with dart language
   We are using flutter with dart language to develop our E- health mobile application. The reason that we choose flutter is,
   - The mobile application is run on both android and iOS. (can developed cross platform apps)
   - Hot reload
   - Ready to use widgets.
   - Simple platform.
   - Attractive UI designs.
2) (web application)
   We use PHP, HTML, CSS, JavaScript and Bootstrap to create the E-health web application.
3) Node JS
   We are using Node.js to develop the application programing interface (API). Because, there are huge number of free tools, high performance, easy to learn and use etc.
   We are using express js, core js, bcrypt.js etc as the supported middleware technologies.
   Those are explaining in follow.

Reasons for developing this API with Node Js

- Reusable code
- Accessible technology
- Scalability
- Largest eco system of tools
- High performance

Express. Js

This is an open source framework for node.js, this is reliable and have a good processing speed.

This is used for robust APIs.

Bcrypt. Js

Bcrypt.js is a good way to hash and store passwords.

Core.js

This is a client-side JS library. This is used for creating event-driven JS codes and object-oriented codes. This is very useful to enable to web access.

Body-parser

This body-parser object exposes various kind of factories to create middleware. This is used to read HTTP POST data, and this read a form's input and stores it as a JavaScript object.

Json web token

This is an open standard for passing claims in web application environments. Also, this is an internet proposed standard in creating data through optional signature or optional encryption which the payload holds JSON which asserts some amount of the claims.

Mongoose

This is a type of Mongo DB object which is designed to work in an asynchronous environment. This provides a direct schema based to model the user's application data.

Multer

This is a node.js middleware. This is mainly used for uploading files and also this would never process any form which is not multipart.

Node mailer

This is a module in Node.js to send e-mails and this is licensed under MIT license.

XML

This is a mark-up language which can be read by human language and machine language. This is designed commonly to be use friendly and also to emphasize simplicity and usability through the internet. This language is widely used for the representation of data structures.

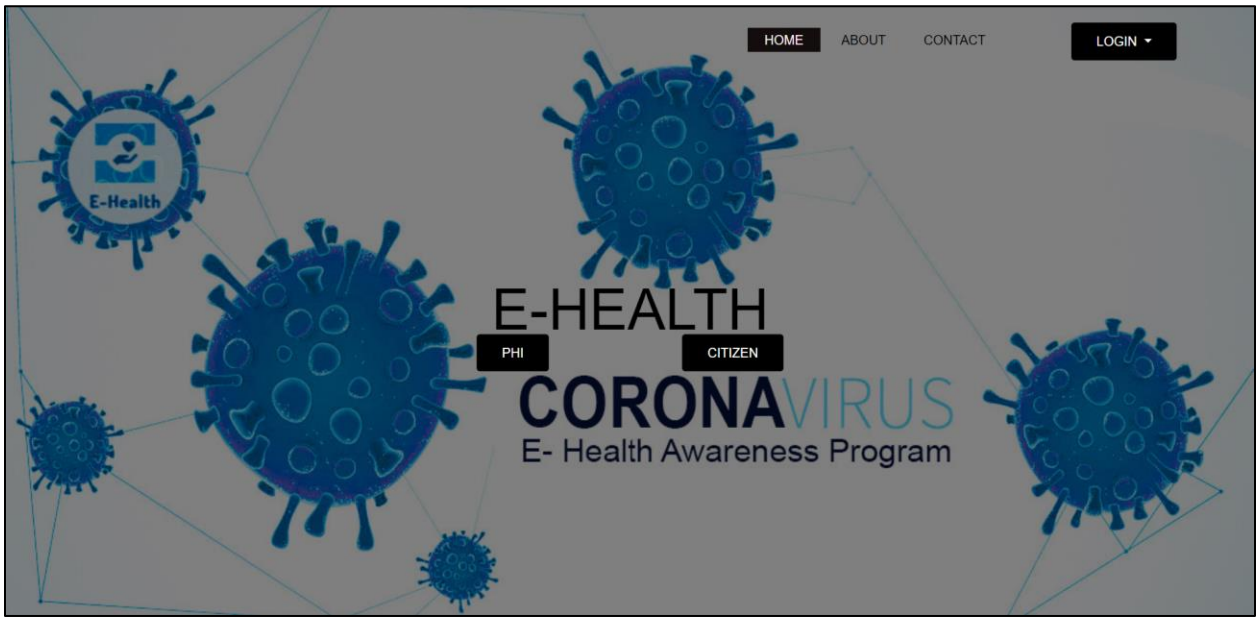
4. Client Application

a) Web application

The web application was developed using html, css, javascript and bootstrap framework. All the functionalities are performed by using web API. Here mention below screenshots of web application.

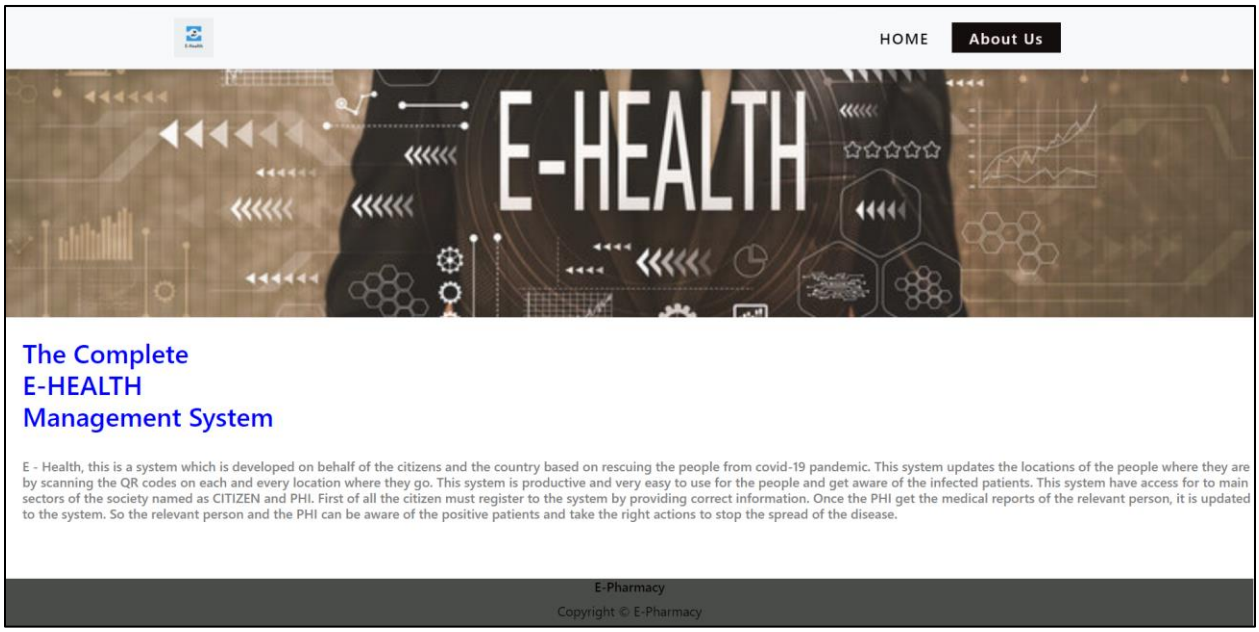
Home Page

This is E-health website home page. There have option for PHI and citizen to login and register account. Also anybody can contact them and can watch their details.



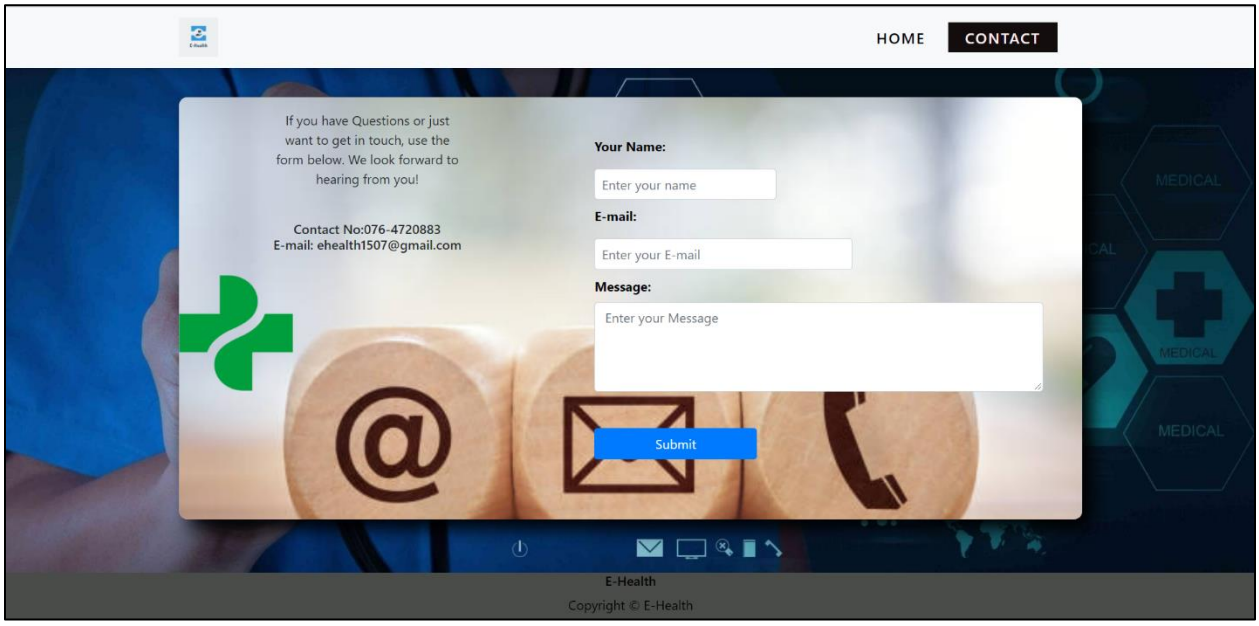
About us

There can have option to get E-health management system informations.



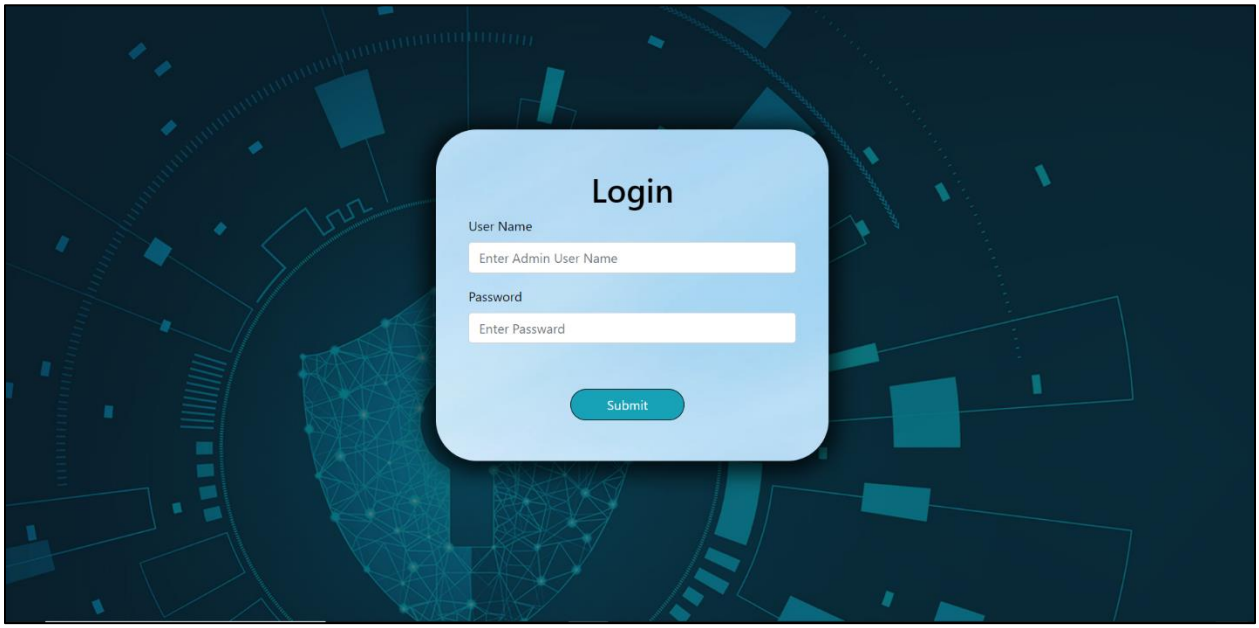
Contact page

Here can Contact them anyone, PHI or citizen. After this request receive to their email.



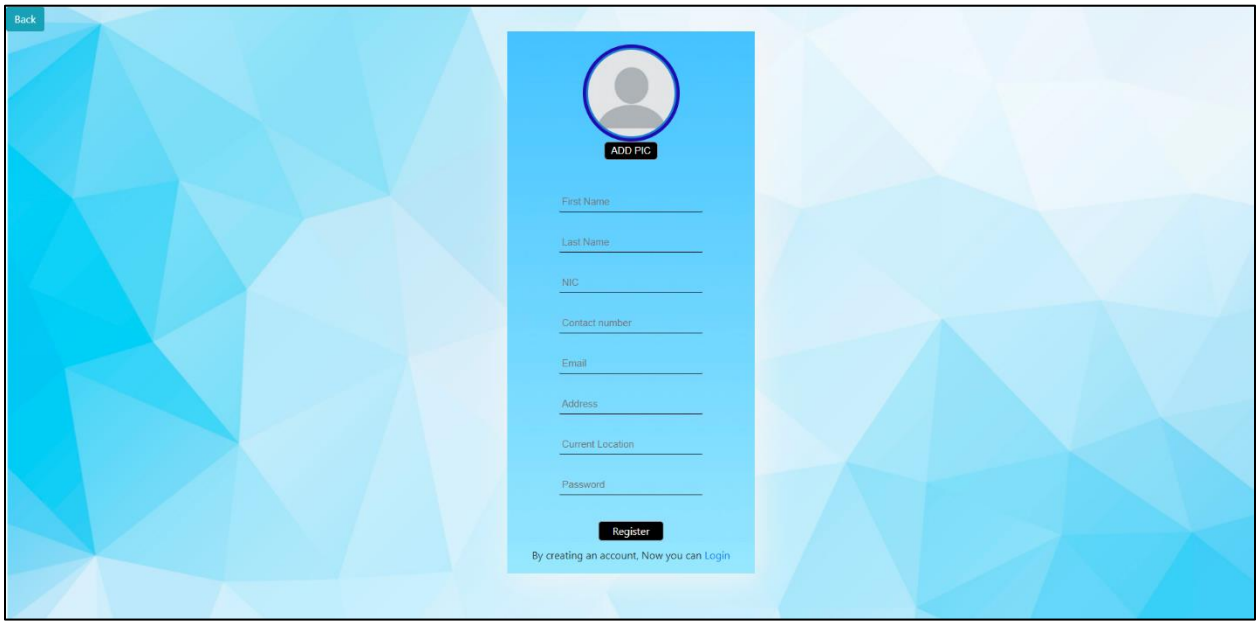
Login page

There have option to login their account for PHI and Citizen, who register with system .



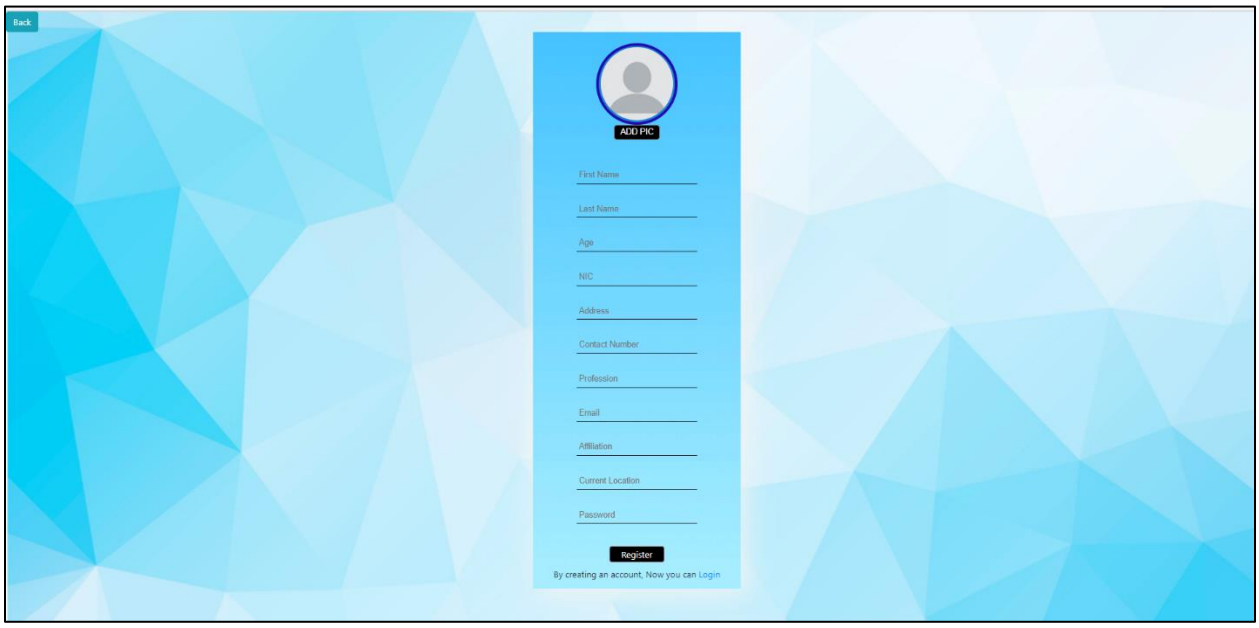
PHI register page

There can have to chance register for PHI, those who would like to join our system, they have to add their details such as, first name, last name, NIC, contact no, email, address, current location and password.



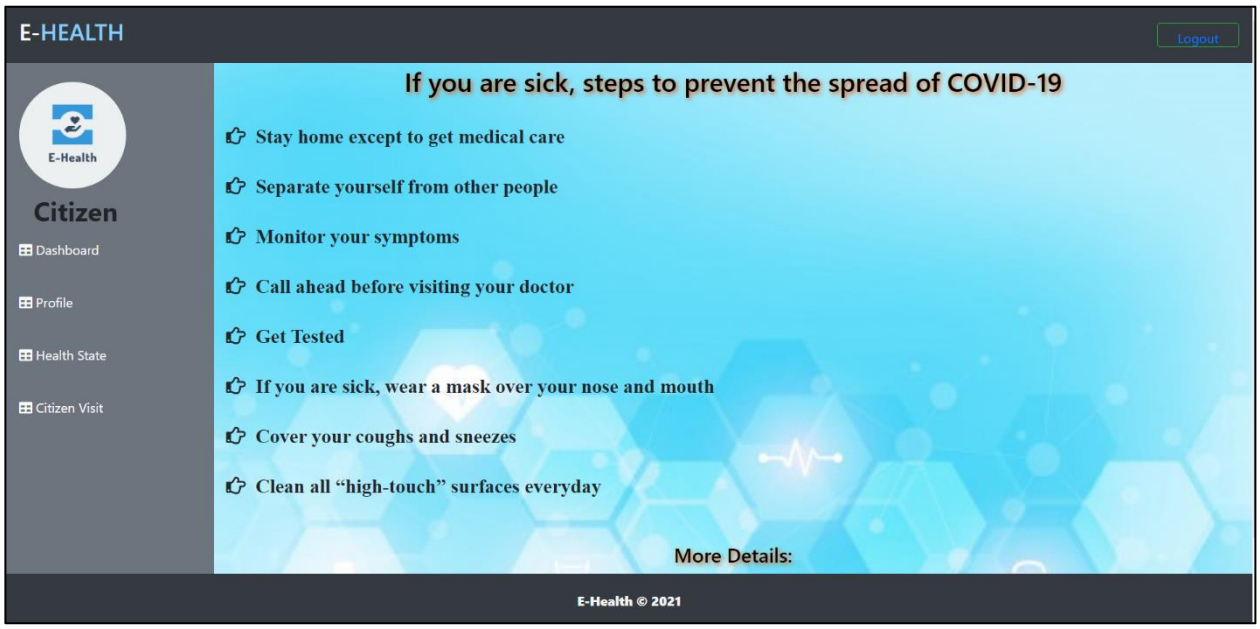
Citizen register page

Here can have option to register citizen those who would like to join our system. So they have to add their details such as, first name, last name, email, NIC, address, contact no, profession, email, affiliation, current location and password.



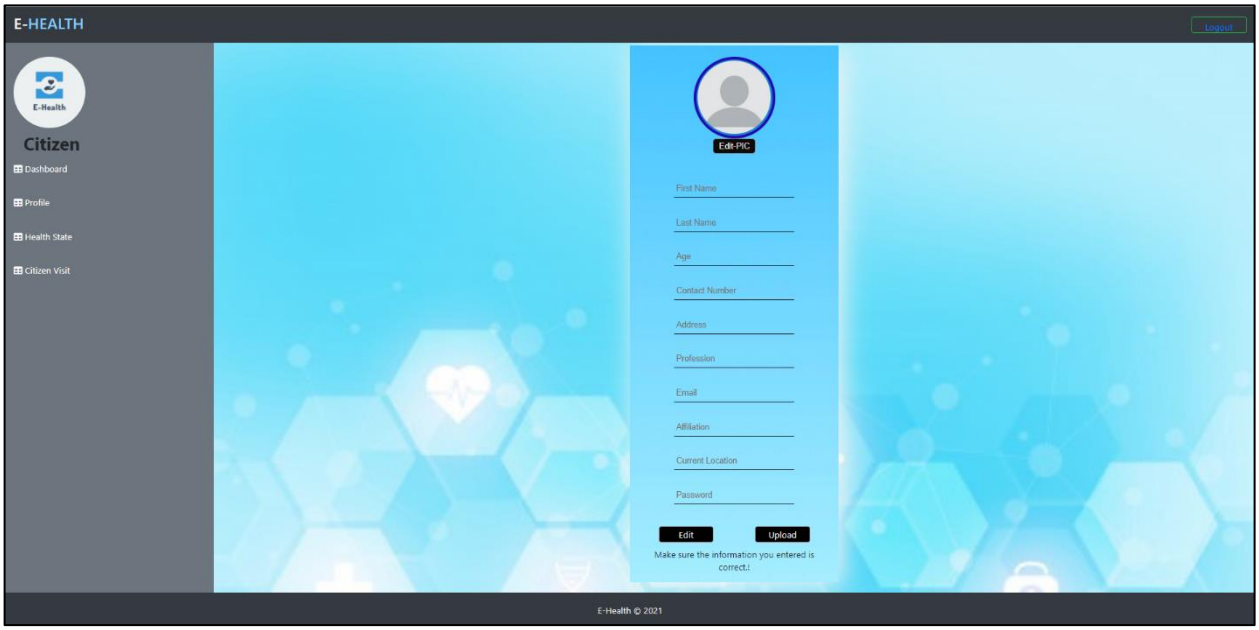
Citizen dashboard

This is citizen dashboard page. This will access, who have register with our system. This menu give you options for update your profile settings details, health status and citizen visit.



Citizen profile page

This is citizen profile page. There can have option to update their details such as, first name, last name age, contact no, profession, email, affiliation, current location.

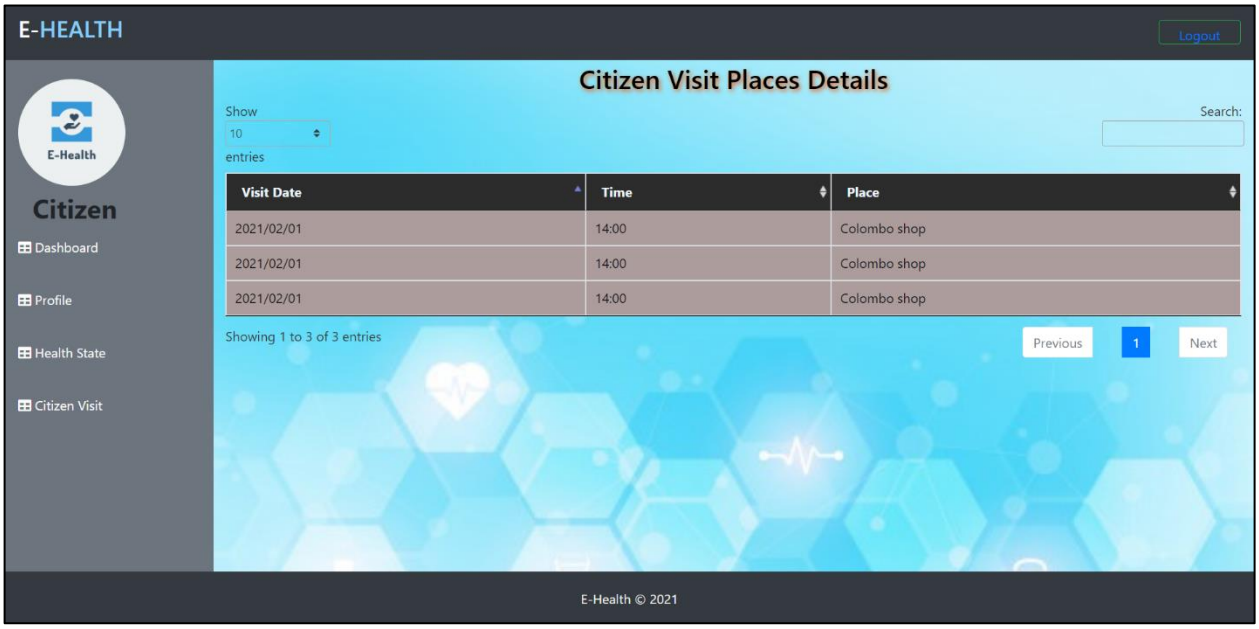


Citizen health state page

This is citizen health status details page. Here citizen can check their PCR or Antigen test result details.

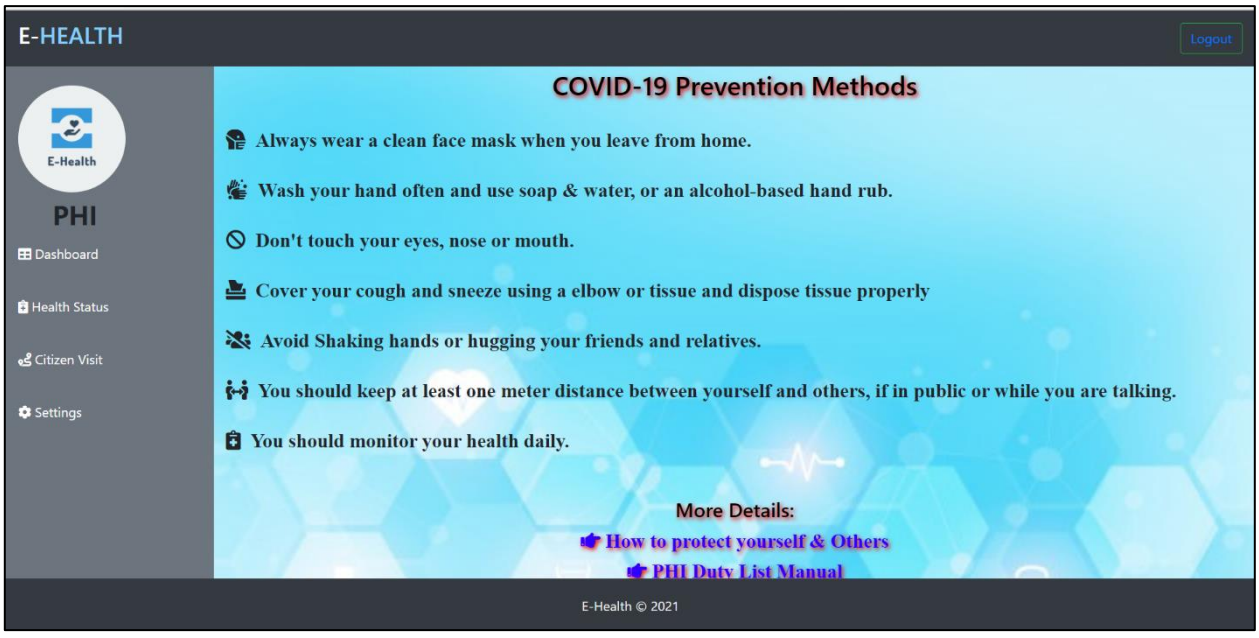
Citizen visit page

This is citizen visit details page. Here can watch where visited and visit date and time.



PHI Dashboard

This is PHI dashboard. There can have access to those who was register with our system. Here can have to option to watch, dashboard, health status page, citizen visit and PHI profile details update pages.

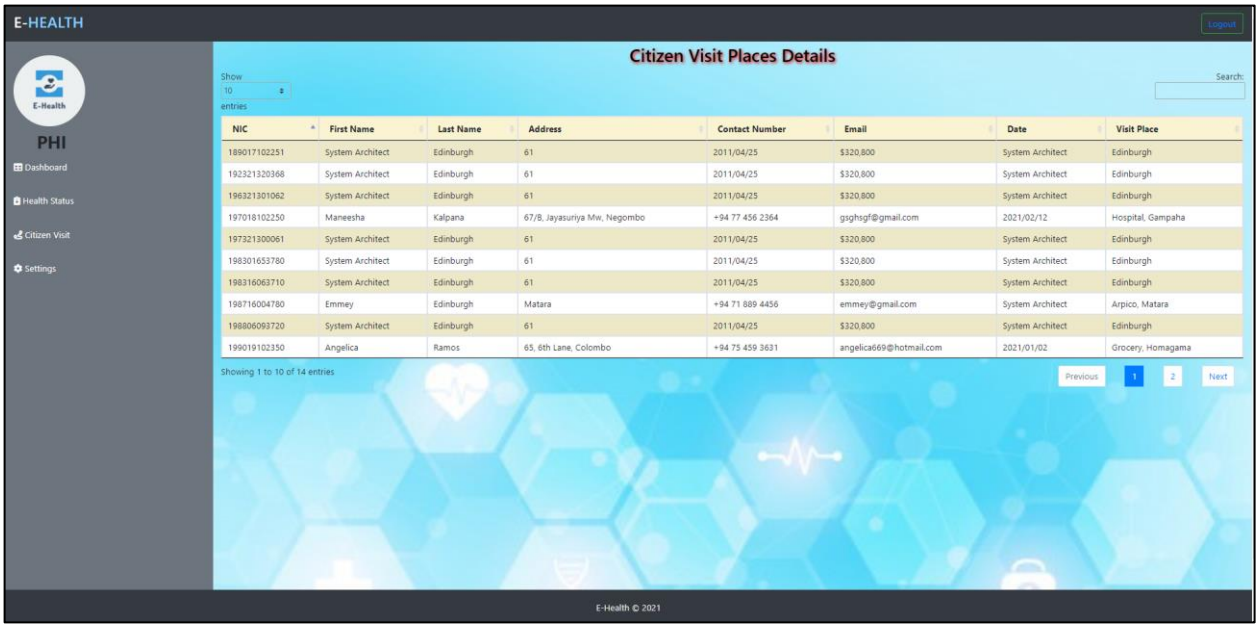


PHI Health state page

Here we given to option for update citizen health status. There include citizen's NIC, firstname and last name, age, address, profession, email, affiliate, current location, date and test type and result.

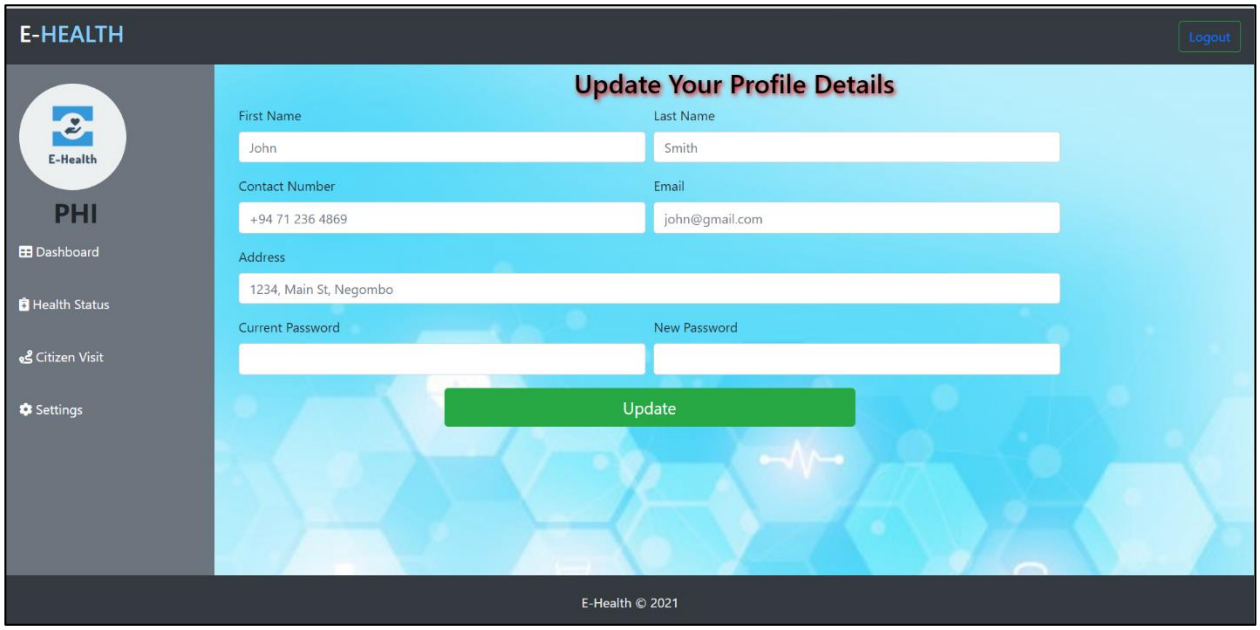
PHI citizen visit page

There can check citizen visit places details. It helps to find details this covid situation.



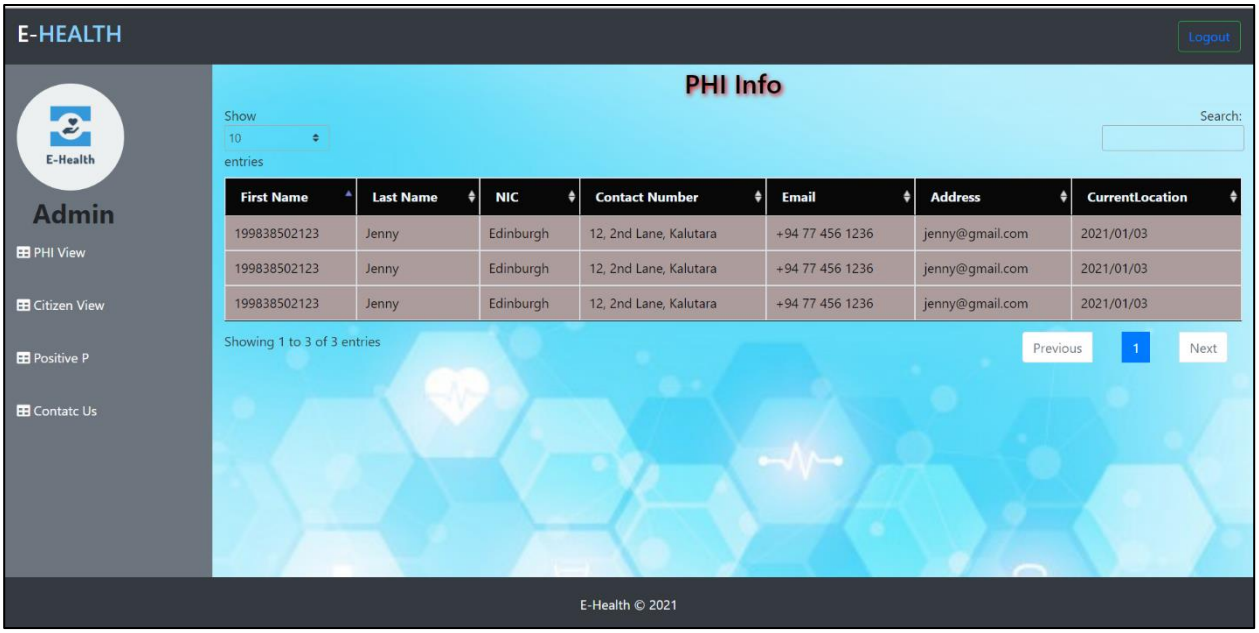
PHI profile settings page

Here PHI can upate their account information.



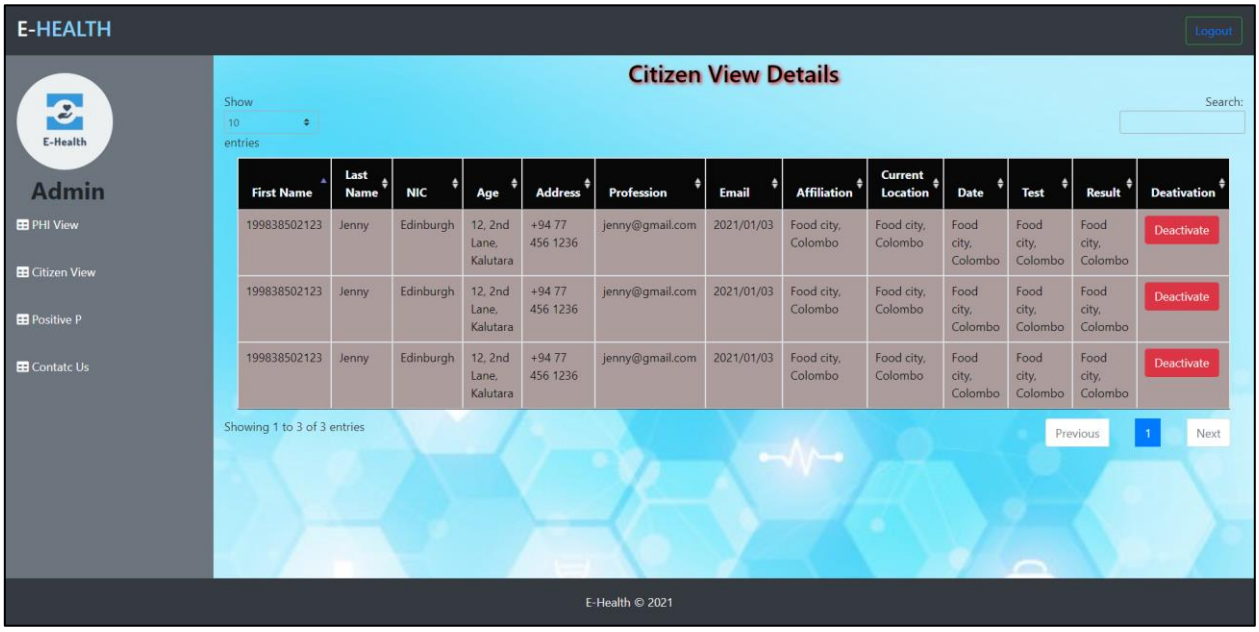
PHI View (admin)

This is Admin panel. This page given to option for check PHI register details.



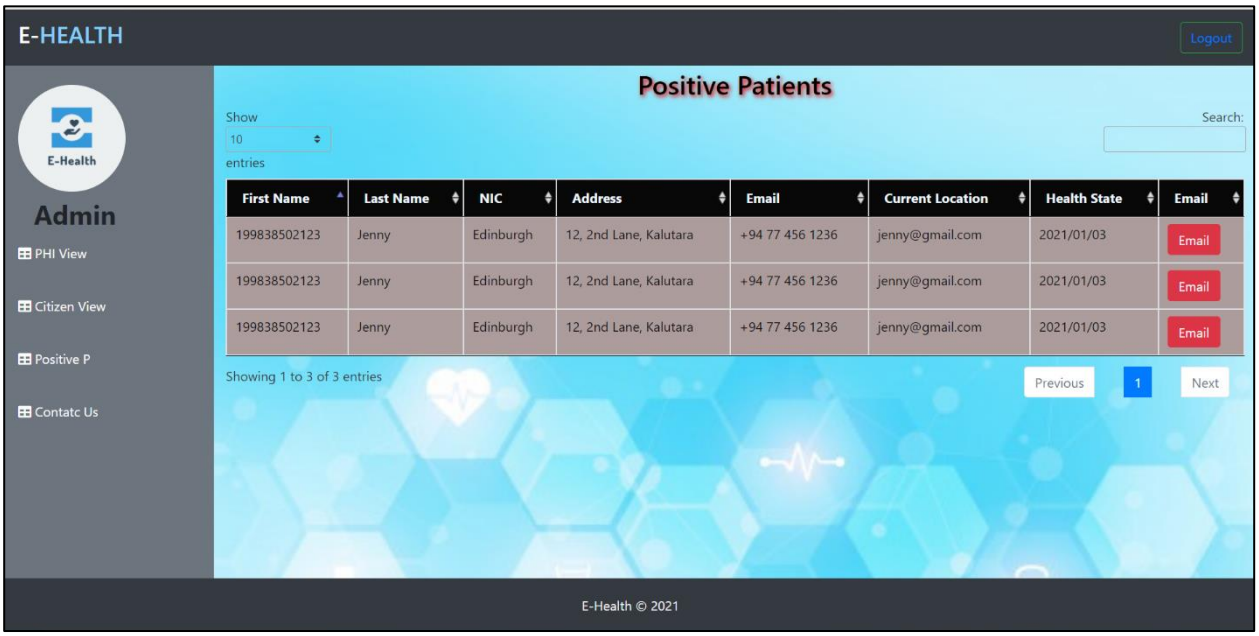
Citizen view (admin)

This is Citizen view details page. There have to option for check citizen details including with their test results. If someone find with positive result, admin can able to deactivate their account.**Positive**



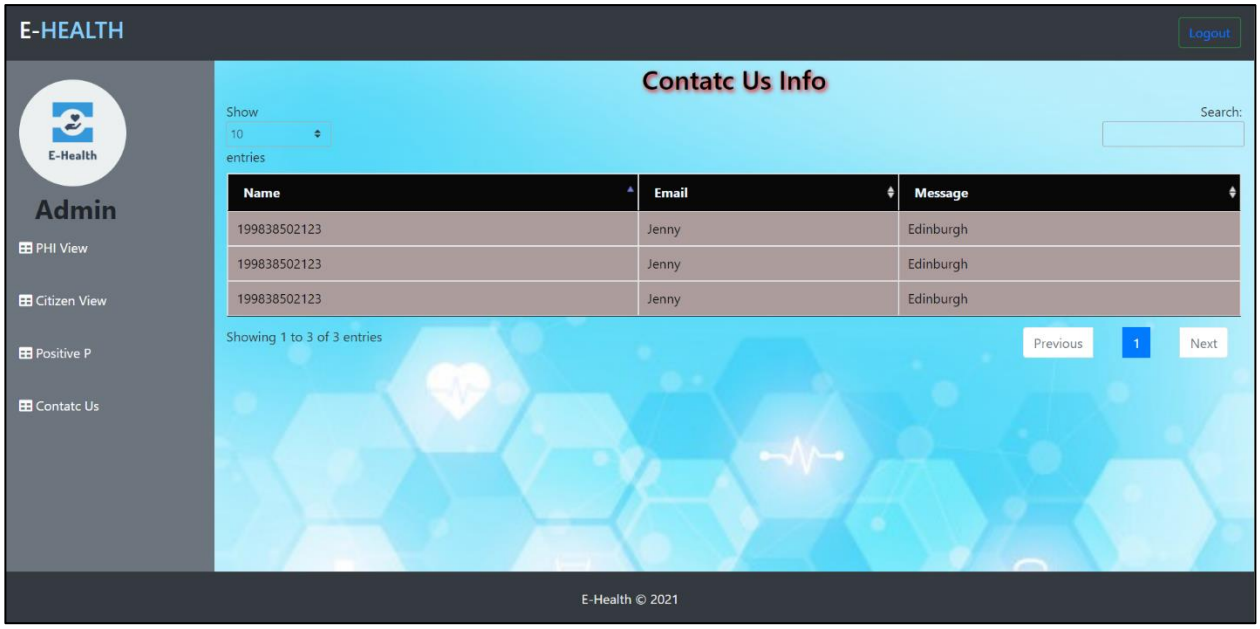
patients page (admin)

This is positive patients page. There admin can check those who have covid positive result and admin can send them email too.



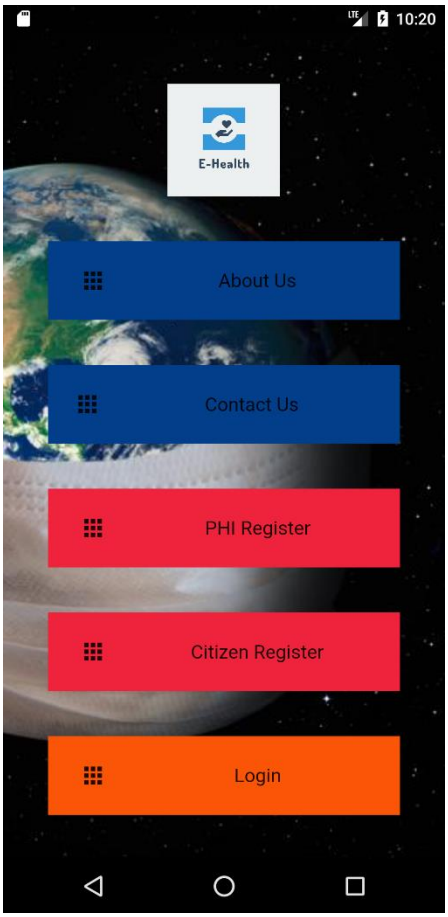
Contact us page (admin)

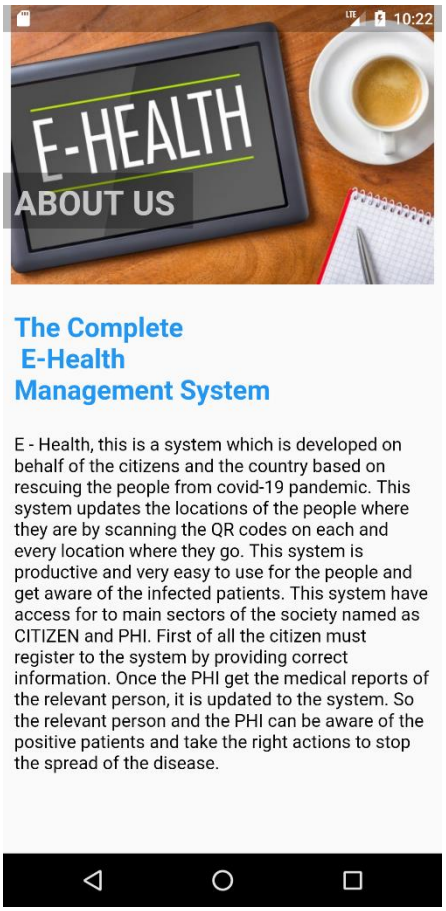
This is contact us info page. Admin can check contact details here.



b) Mobile application

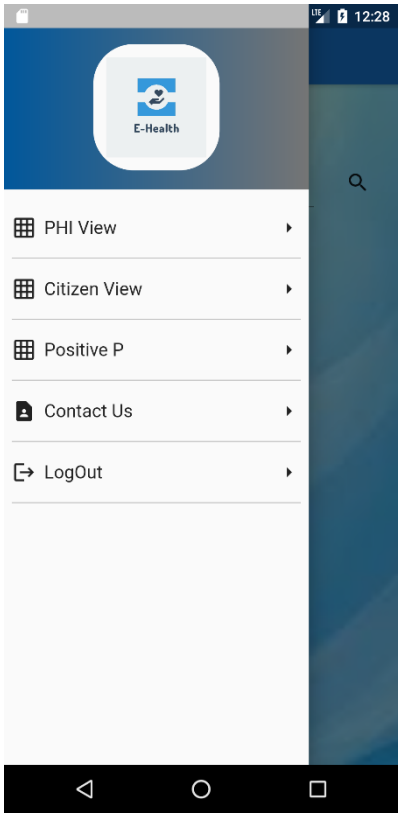
- Mobile application was created using flutter framework.



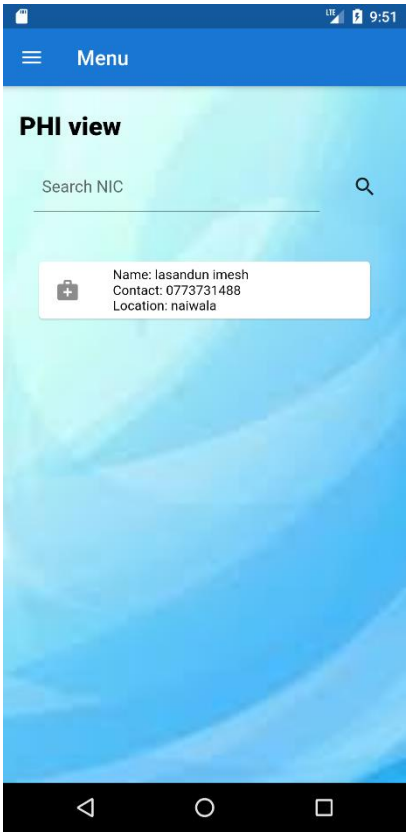


1. **This is the loading page of our mobile application. Its loads 3s and then load the second page(according to the screen shots). The logo of our system is in the middle of the interface.**
2. **This is the menu page. There are 5 buttons. About us, contact us, PHI register, Citizen Register and the Login. There is also the logo in the top.**
3. **When enter the about us button, it will goes to the about us page. There are some info about our overall system.**

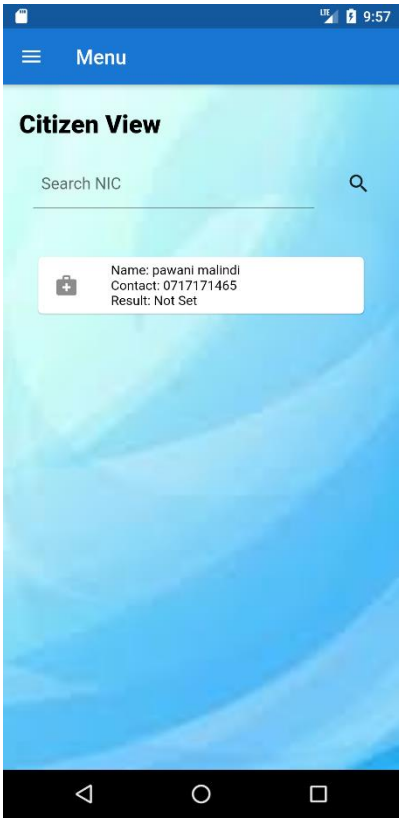
1. **Admin part (CDC)**



- **This is the admin dashboard page. There are sub pages, PHI view, citizen view, positive p, contact us and there is the log out button.**



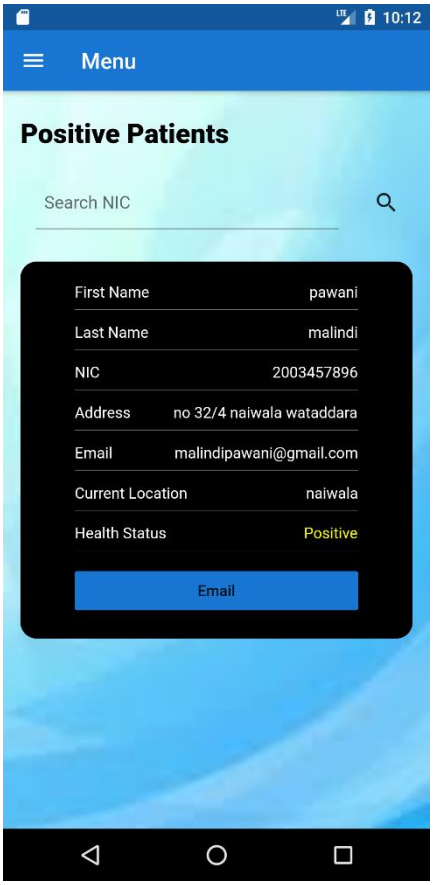
- **This is the PHI view page. Admin can view the PHI details that are connected with this system.**

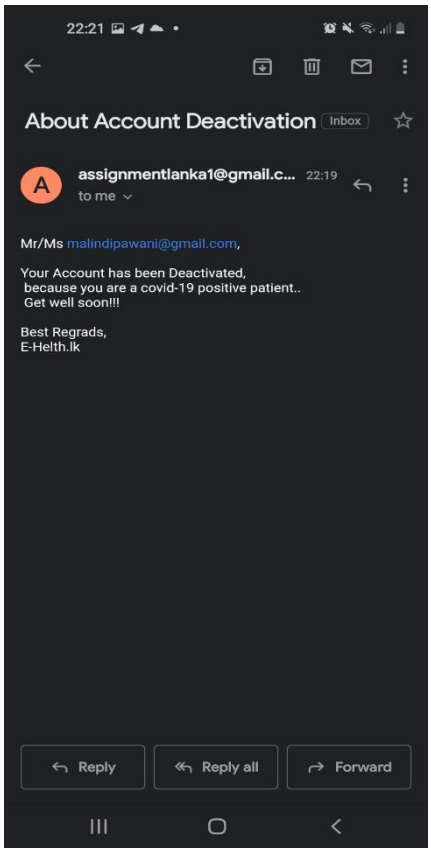


- **This is the citizen view page. Admin can view all the details of the citizens.**

If the citizen is a covid 19 positive patient, admin can deactivate the citizens account.

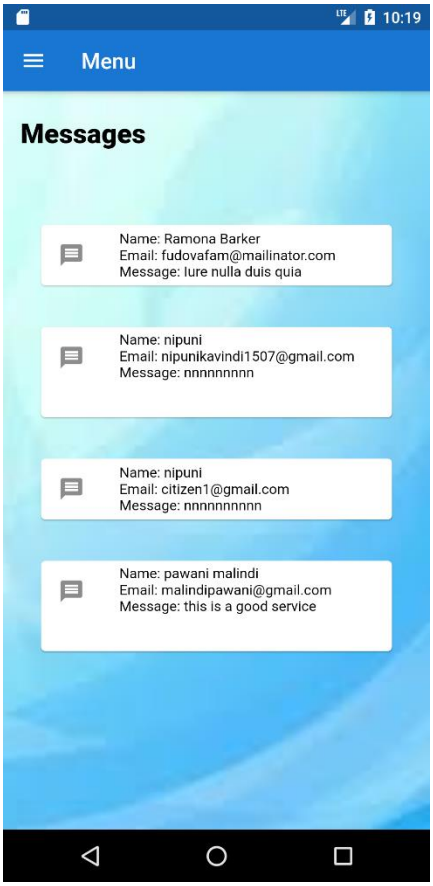
- **After the deactivation admin can send an email to the positive patient.**





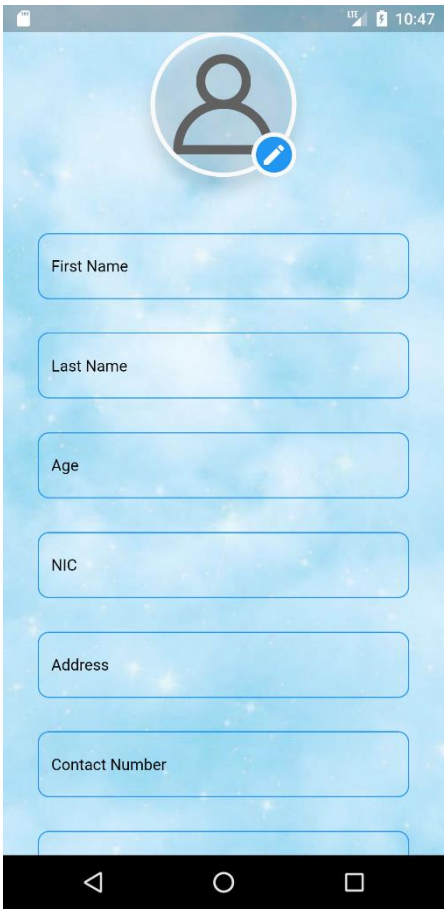
- **The email is like this.**

- **Admin can view the contact us messages.**

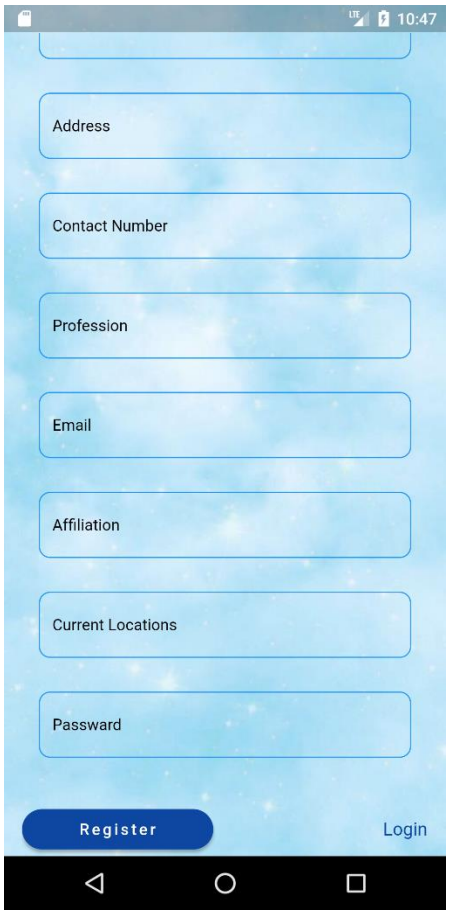


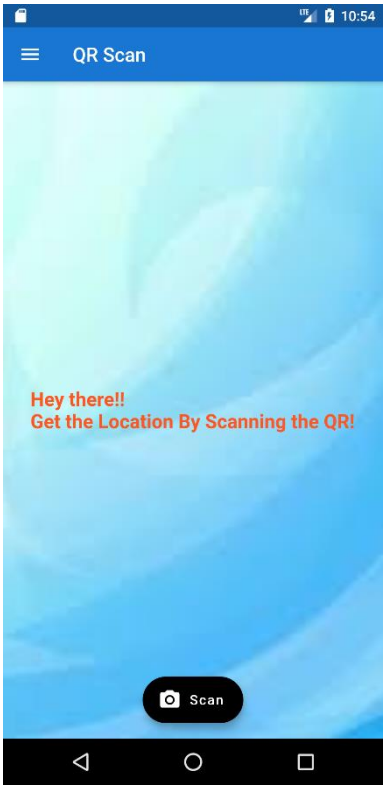
2. Citizen part

- **This is the citizen dashboard**

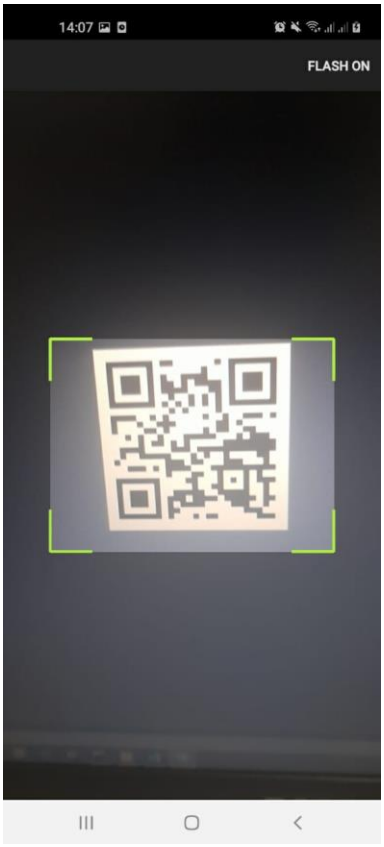


- **This is the citizen registration part.**

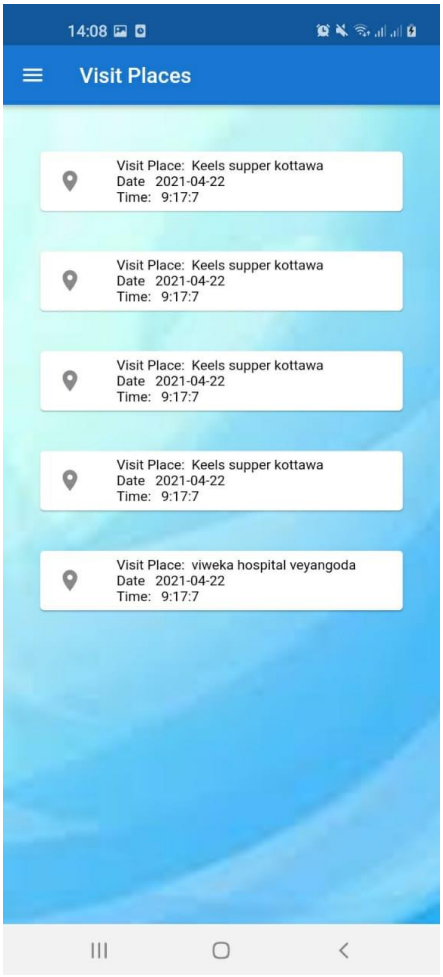




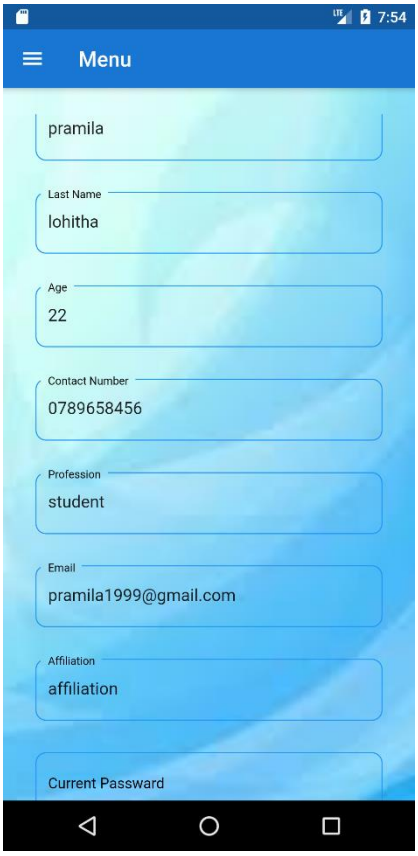
- This is the QR code scanner
- There are the QR codes in the folder



The data that scanning from the QR code

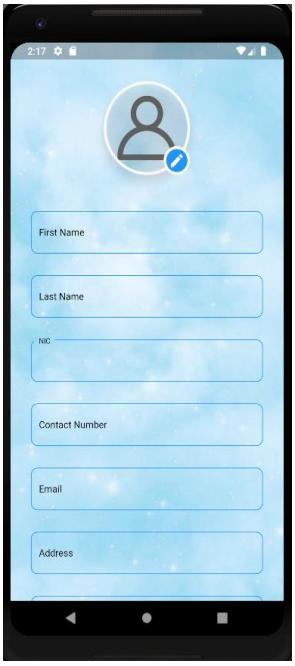


- The result of the PCR Or antigen that sending by The PHI.

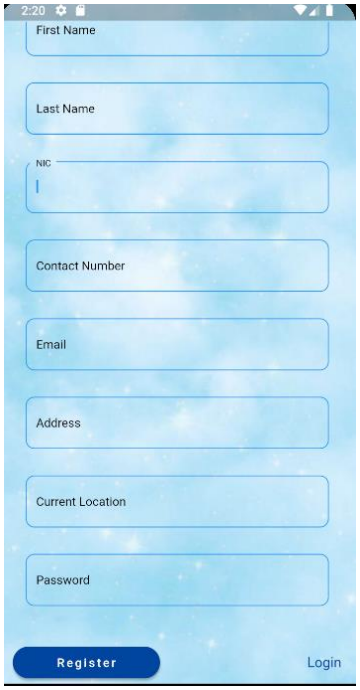


- This is the citizen edit profile page

3. PHI Part

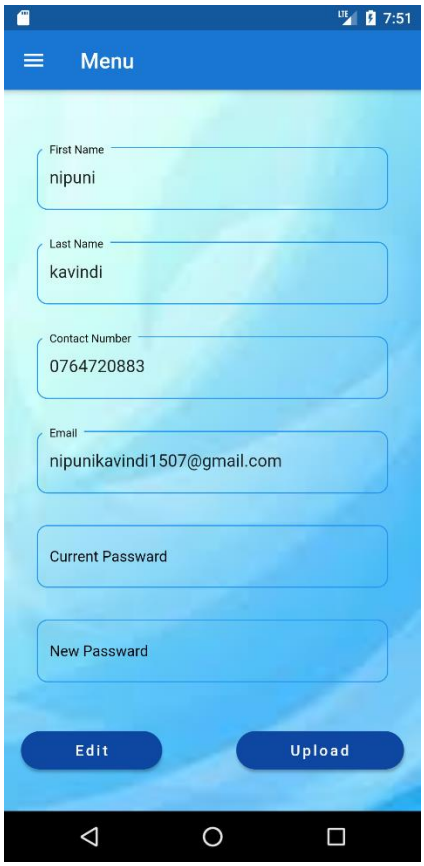


- This is the PHI registration page



- This is the PHI's Dashboard.

- This is the citizen visit page. Phi can see the places that citizen visit and the date.

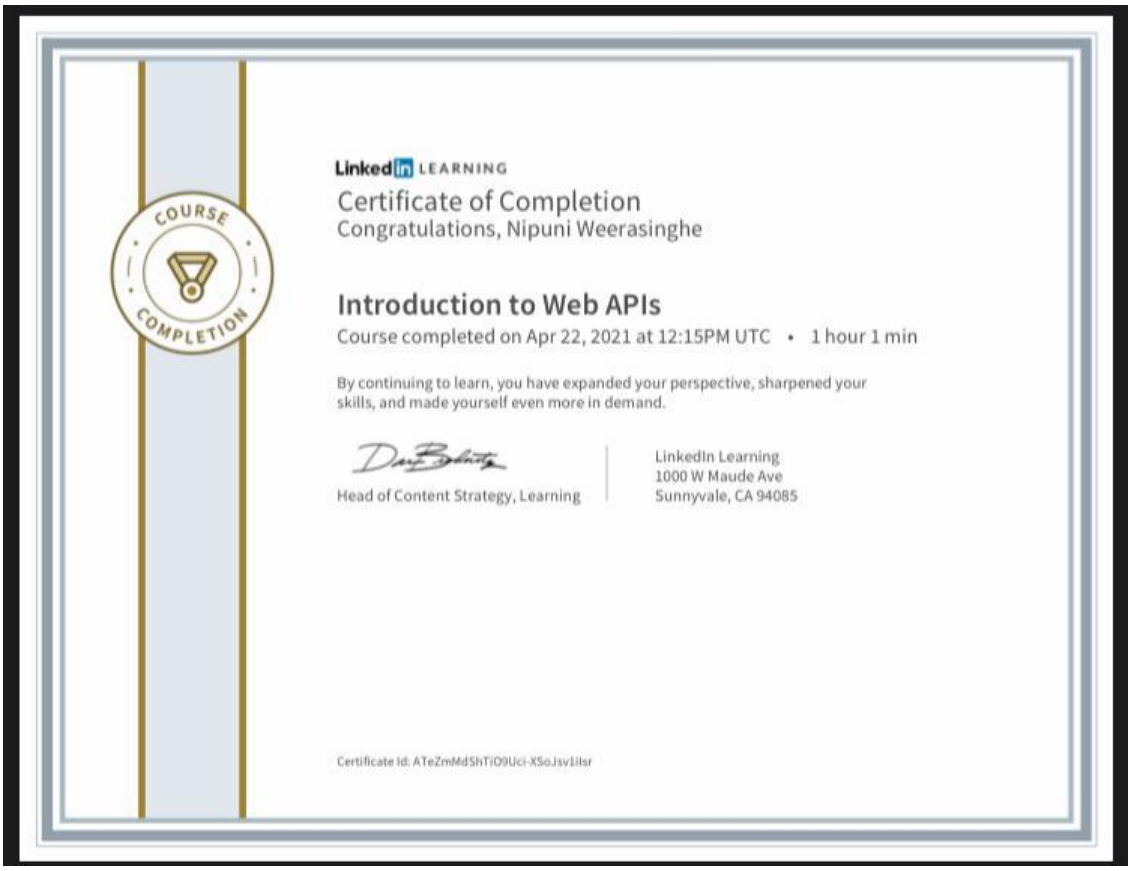


- This page is the PHI profile update page.

Individual Contribution

I.K.N.K Weerasinghe

INDEX:-10674043



Admin part

(CDC) Center for Diseases Control (Admin part)

- Contact us main page: POST , addContact.php
- Contact Us Admin page :GET
- Citizen account Deactivation : DELETE, deleteCitizen.php
- PHI view : GET
- Designing part (Web & mobile) – contact us page(Admin, Main) , account deactivation page, Phi view page(Mobile Application), Dashboard(Mobile application)

Kely Weerasooriya – 10674044

PHI Part

- Account creation – POST
- Update PHI profile details – PUT
- View citizen visit places – GET
- Created PHI part web & mobile pages – PHI Registration, PHI profile settings update, PHI dashboard page, PHI citizen visit pages.

G.L.I Karunanayake

INDEX NO: - 10673980

Admin part

(CDC) Center for Diseases Control (Admin part)

- Admin Login : POST , adminLogin.php
- Sending the Email : POST, sendEmail.php
- PHI view: GET
- Designing part (Web & mobile) – Admin login page, sending email page, PHI view page(web application), Dashboard(Web application)

H.W.C. Waduge – 10674040

PHI part

- PHI login - POST
- Health status update – PUT
- View citizen visit places – GET
- Created PHI part web & mobile pages – Health status update page, login, citizen visit pages.

M.M.P.M Bandara

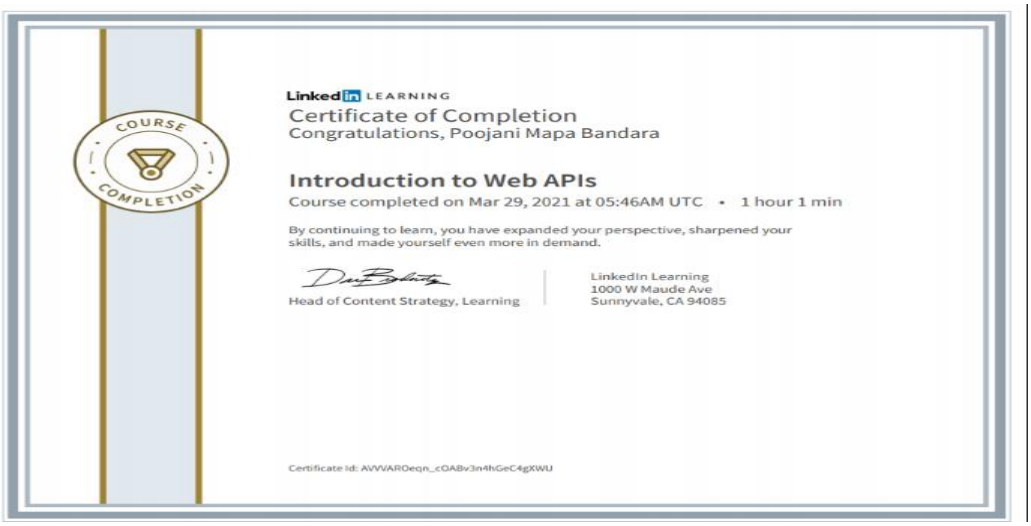
Citizen part

Index Number: 10673074

Degree Program: BSc hons Software Engineering (Plymouth)

My Contribution

In this API software development project scenario given to create web application and mobile application for Center for Disease Control and Prevention (CDC) and with the ministry of health and Public Health Inspectors (PHI) to provide support for ongoing covid – 19 situations. This application is created to know locations when people are moving from one location to another. This system contains PHI, Admin and Citizen part which have web application and mobile application connected through API. I have created / contributed for half of Citizen part. Half from both web application and mobile application of Citizen which connected through API is created by me. I have contributed to login of citizen, about us Page, Personnel profile of citizen, Citizen Visit Page and Qr code scanner of citizen which can track locations of people where they visit and can update those locations' part also, I have contributed this Qr code scanner part contain result of the PCR or Antigens which have been send by PHI which contain in web application. Mobile application also contains these parts which contain Citizen edit page, login, Qr code scanner, about us page. Personnel profile page have option to update their details such as, first name, last name age, contact no, profession, email, affiliation, current location, password and photo can be edited, login contain user name and password, from about us page have option to get E-health management system information's and through citizen visit page can get details of citizen who have visited and can get details were visited and visit date and time. Login – POST, Update Citizen Profile – PUT.



W A R P L Wanasinghe – 10026924

Citizen part

- Registration - POST
- Health state – GET
- Citizen visit – GET
- Home page

5. References

Marczydło, D., 2019. *5 reasons to build your backend using node js.* [Online]
Available at: https://concisesoftware.com/5-reasons-to-build-your-backend-using-node-js/
[Accessed 07 April 2021].

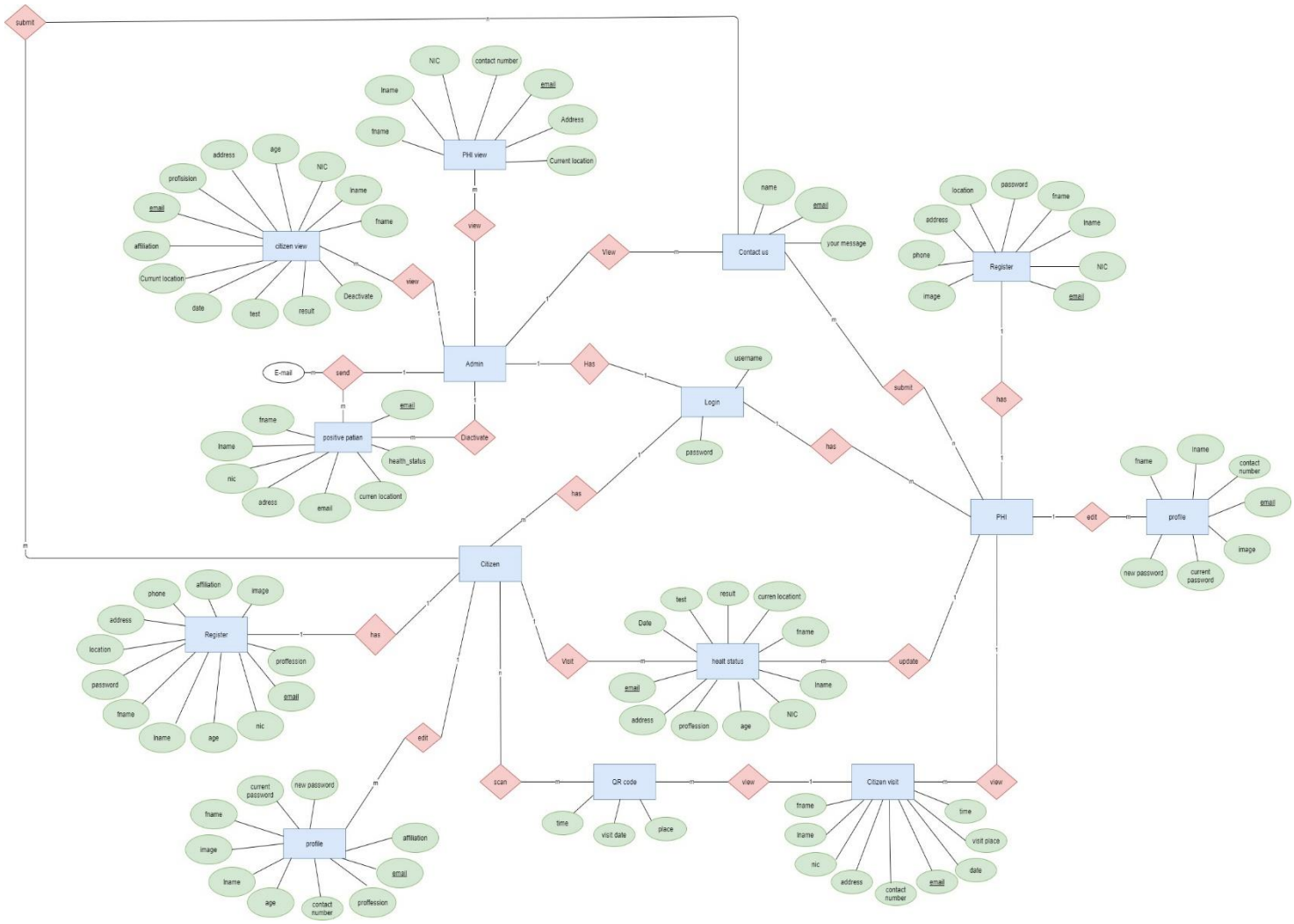
Nolle, T., n.d. *application program interface (API).* [Online]
Available at: https://searchapparchitecture.techtarget.com/definition/application-program-interface-API
[Accessed 7 April 2021].

valuecoders, 2021. *Some Essential And Amazing Web Development Tools And Resources -2021.* [Online]
Available at: https://www.valuecoders.com/blog/technology-and-apps/web-development-tools-resources/
[Accessed 18 april 2021].

w3school, n.d. *Node.js Introduction.* [Online]
Available at: https://www.w3schools.com/nodejs/nodejs_intro.asp
[Accessed 18 april 2021].

6. Appendices

a) ER Diagram



b) Controller Classes

```
//phi signup route
Complexity is 11 You must be kidding
router.post('/phi/signup', (req, res, next) => {
    Phi.find({ email: req.body.email })
        .exec()
        Complexity is 10 It's time to do something...
        .then(phi => { //check if the email is allready taken
            if (phi.length >= 1) {
                return res.status(409).json({
                    message: 'Email Not Available. '
                });
            } else {
                Complexity is 6 It's time to do something...
                bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
                    if (err) {
                        return res.status(500).json({
                            error: err
                        });
                    } else {
                        const phi = new Phi({
                            _id: new mongoose.Types.ObjectId(),
                            fname: req.body.fname,
                            lname: req.body.lname,
                            nic: req.body.nic,
                            email: req.body.email,
                            phone: req.body.phone,
                            address: req.body.address,
                            location: req.body.location,
                            image: req.body.image,
                            password: hash
                        });
                        phi.save()
                            .then(result => {
                                console.log(result);
                                res.status(201).json({
                                    message: "Sign up Sucessfully",
                                    createdUser: result
                                });

                            })
                            .catch(err => {
                                console.log(err);
                                res.status(500).json({
                                    error: err
                                });
                            })
                    }
                })

            }
        })
});
```

```
//login PHI
Complexity is 5 Everything is cool!
router.post('/philogin', async(req, res) => {
    const phi = await Phi.findOne({ email: req.body.email });
    if (!phi) return res.status(400).send({ message: 'Email does Not Exist' });

    const validPass = await bcrypt.compare(req.body.password, phi.password);

    if (!validPass) return res.status(400).send({ message: 'Password is Wrong' });

    const token = jwt.sign({ _id: phi._id }, process.env.JWT_KEY);


    res.header('auth-token').send({
        id: phi._id,
        fname: phi.fname,
        lname: phi.lname,
        email: phi.email,
        phone: phi.phone,
        nic: phi.nic,
        image: phi.image,
        token: token,
        message: phi.fname,
    });
});
```

```
//citizen signup route
Complexity is 11 You must be kidding
router.post('/citizen/signup', (req, res, next) => {
    Citizen.find({ email: req.body.email })
        .exec()
        Complexity is 10 It's time to do something...
        .then(citizen => { //check if the email is allready taken
            if (citizen.length >= 1) {
                return res.status(409).json({
                    message: 'Email Allready taken. '
                });
            } else {
                Complexity is 6 It's time to do something...
                bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
                    if (err) {
                        return res.status(500).json({
                            error: err
                        });
                    } else {
                        const citizen = new Citizen({
                            _id: new mongoose.Types.ObjectId(),
                            fname: req.body.fname,
                            lname: req.body.lname,
                            nic: req.body.nic,
                            age: req.body.age,
                            email: req.body.email,
                            phone: req.body.phone,
                            address: req.body.address,
                            location: req.body.location,
                            affilication: req.body.affilication,
                            profeson: req.body.profeson,
                            image: req.body.image,
                            date: req.body.date,
                            test: req.body.test,
                            results: req.body.results,
                            password: hash
                        });
                        citizen.save()
                            .then(result => {
                                console.log(result);
                                res.status(201).json({
                                    message: "Sign up Sucessfully",
                                    createdUser: result
                                });

                            })
                            .catch(err => {
                                console.log(err);
                                res.status(500).json({
                                    error: err
                                });
                            })
                    }
                })

            }
        })
```

```
//login Citizen
Complexity is 5 Everything is cool!
router.post('/citizenlogin', async(req, res) => {
    const citizen = await Citizen.findOne({ email: req.body.email });
    if (!citizen) return res.status(400).send({ message: 'Email does Not Exist' });

    const validPass = await bcrypt.compare(req.body.password, citizen.password);

    if (!validPass) return res.status(400).send({ message: 'Password is Wrong' });

    const token = jwt.sign({ _id: citizen._id }, process.env.JWT_KEY);


    res.header('auth-token').send({
        id: citizen._id,
        fname: citizen.fname,
        email: citizen.email,
        nic: citizen.nic,
        image: citizen.image,
        token: token,
        lname: citizen.lname,
        phone: citizen.phone,
        age: citizen.age,
        affilication: citizen.affilication,
        profeson: citizen.profeson,
        address: citizen.address,
        message: citizen.fname,
        status: citizen.status
    });
});
```

```
//update phi
Complexity is 3 Everything is cool!
router.put("/updatePhi", async(req, res, next) => {
    const hashPass = await bcrypt.hash(req.body.password, 10); //hash the password using bcrypt


    try {
        const phi = await Phi.findByIdAndUpdate({ _id: req.body.id }, {
            fname: req.body.fname,
            lname: req.body.lname,
            email: req.body.email,
            phone: req.body.phone,
            password: hashPass,

        }, {
            new: true,
            useFindAndModify: false,
            upsert: true
        })

        res.json(phi).status(200)
    } catch (e) {
        next(e)
    }
})
```

```
//update citizen
Complexity is 3 Everything is cool!
router.put("/updateCitizen", async(req, res, next) => {
    const hashPass = await bcrypt.hash(req.body.password, 10); //hash the password using bcrypt


    try {
        const citizen = await Citizen.findByIdAndUpdate({ _id: req.body.id }, {
            fname: req.body.fname,
            lname: req.body.lname,
            age: req.body.age,
            email: req.body.email,
            phone: req.body.phone,
            affilication: req.body.affilication,
            profeson: req.body.profeson,
            image: req.body.image,
            password: hashPass

        }, {
            new: true,
            useFindAndModify: false,
            upsert: true
        })

        res.json(citizen).status(200)
    } catch (e) {
        next(e)
    }
});
```

```
// add the admin
Complexity is 11 You must be kidding
router.post('/addAdmin', (req, res, next) => {
    Admin.find({ email: req.body.email })
        .exec()
        Complexity is 10 It's time to do something...
        .then(admin => { //check if the email is allready taken
            if (admin.length >= 1) {
                return res.status(409).json({
                    message: 'Email Already Taken. '
                });
            } else {
                Complexity is 6 It's time to do something...
                bcrypt.hash("12345678", 10, (err, hash) => { //hash the password using bcrypt
                    if (err) {
                        return res.status(500).json({
                            error: err
                        });
                    } else {
                        const admin = new Admin({
                            _id: new mongoose.Types.ObjectId(), //create id in mongodb
                            email: "admin@helth.lk",
                            password: hash,
                            type: req.body.type,
                        });
                        admin.save()
                            .then(result => {
                                console.log(result);
                                res.status(201).json({
                                    message: "Admin Added Successfully!!",
                                    createdAdmin: result
                                });

                            })
                            .catch(err => {
                                console.log(err);
                                res.status(500).json({
                                    error: err
                                });
                            })
                    }
                })

            }
        })
});
```

```
//login Admin
Complexity is 5 Everything is cool!
router.post('/adminlogin', async(req, res) => {
    const admin = await Admin.findOne({ email: req.body.email });
    if (!admin) return res.status(400).send({ message: 'Email does Not Exist' });

    const validPass = await bcrypt.compare(req.body.password, admin.password);

    if (!validPass) return res.status(400).send({ message: 'Password is Wrong' });

    const token = jwt.sign({ _id: admin._id }, process.env.JWT_KEY);


    res.header('auth-token').send({
        id: admin._id,
        email: admin.email,
        token: token
    });
});
module.exports = router;
```

```
//get all citizens
Complexity is 4 Everything is cool!
router.get("/citizens", async(req, res, next) => {
    try {
        const citizen = await Citizen.find({ status: 'Live' })
        if (citizen == '') {

        }
        res.json(citizen)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }

});
```

```
// get  citizen to their nic
Complexity is 5 Everything is cool!
router.get("/citizen/:nic", async(req, res, next) => {
    try {
        const _nic = req.params.nic;
        const citizen = await Citizen.find({ nic: _nic })
        if (citizen == '') {
            return res.status(404).json({
                error: true,
                message: "No report citizen on that NIC .."
            })
        }
        res.json(citizen)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }
});
```

```
//update test informations
Complexity is 3 Everything is cool!
router.put("/updateResults/:id", async(req, res, next) => {
    try {
        const citizen = await Citizen.findByIdAndUpdate({ _id: req.params.id }, {
            results: req.body.results,
            test: req.body.test,

        }, {
            new: true,
            useFindAndModify: false,
            upsert: true
        })

        res.json(citizen)
    } catch (e) {
        next(e)
    }
});
```

```
Complexity is 3 Everything is cool!
router.post("/addCitizenReport", async(req, res, next) => {
    try {
        const report = await CitizenReport.create({
            cid: req.body.id,
            date: req.body.date,
            test: req.body.test,
            results: req.body.results,
            nic: req.body.nic

        })


        const citizen = await Citizen.findByIdAndUpdate({ _id: req.body.id }, {
            results: req.body.results,
            test: req.body.test,

        }, {
            new: true,
            useFindAndModify: false,
            upsert: true
        })

        res.json(report).status(200)
    } catch (e) {
        next(e)
    }
})
```

```
// get  citizen roports to their nic
Complexity is 5 Everything is cool!
router.get("/citizenReportsNic/:nic", async(req, res, next) => {
    try {
        const _nic = req.params.nic;
        const report = await CitizenReport.find({ nic: _nic })
        if (report == '') {
            return res.status(404).json({
                error: true,
                message: "No report Found on that NIC .."
            })
        }
        res.json(report)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }
});
```

```
// citizenReports get to id
Complexity is 4 Everything is cool!
router.get("/citizenReports/:id", async(req, res, next) => {
    try {
        const id = req.params.id;
        const report = await CitizenReport.find({ cid: id })
        if (report == '') {

        }
        res.json(report)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }
});
```

```
// citizenReports get to positivity

Complexity is 4 Everything is cool!
router.get("/positivecitizen", async(req, res, next) => {
    try {

        const citizen = await Citizen.find({ status: 'Deactivated' })
        if (citizen == '') {

        }
        res.json(citizen)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }
});
```

```
//citizen delete route
Complexity is 3 Everything is cool!
router.delete("/deleteCitizen/:id", async(req, res, next) => {
    try {
        const citizen = await Citizen.deleteOne({ _id: req.params.id })
        res.json(citizen).status(202)
    } catch (e) {
        next(e)
    }
});
```

```
//deactive account
Complexity is 3 Everything is cool!
router.put("/updatestate/:id", async(req, res, next) => {
    try {
        const citizen = await Citizen.findByIdAndUpdate({ _id: req.params.id }, {
            status: "Deactivated",

        }, {
            new: true,
            useFindAndModify: false,
            upsert: true
        })


        res.json(citizen)
    } catch (e) {
        next(e)
    }
});
```

```
//deactive account
Complexity is 4 Everything is cool!
router.put("/email/:email", async(req, res, next) => {
    let transporter = nodemailer.createTransport({
        service: 'Gmail',
        auth: {
            user: process.env.SCREAT_EMAIL,
            pass: process.env.SCREAT_EMAIL_PASS
        }
    });

    let mailOptions = {
        from: process.env.SCREAT_EMAIL,
        to: req.body.email,
        subject: "About Account Deactivation",
        text: 'Mr/Ms ' + req.body.email + "," + '\n \nYour Account has been Deactivated, \n because you are a covid-19 positive patient..\n Get well soon!!! \n\nBest Regrads,\nE-Helth.lk'
    };

    Complexity is 3 Everything is cool!
    transporter.sendMail(mailOptions, function(error, info) {
        if (error) {
            console.log(error);
        } else {
            console.log('Email sent: ' + info.response);
        }
    });

    res.json().status(200);
});



module.exports = router;
```

```
// add message

Complexity is 3 Everything is cool!
router.post("/AddMessage", async(req, res, next) => {
    try {
        const message = await Contact.create({
            _id: new mongoose.Types.ObjectId(),
            name: req.body.name,
            email: req.body.email,
            message: req.body.message


        })

        res.json(message)
    } catch (e) {
        next(e)
    }
});
```

```
//get all messages
Complexity is 5 Everything is cool!
router.get("/messages", async(req, res, next) => {
    try {
        const message = await Contact.find()
        if (message == '') {
            return res.status(404).json({
                error: true,
                message: "No message Found .."
            })
        }
        res.json(message)

    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }

});
```

```
//get all messages to XML format
Complexity is 5 Everything is cool!
router.get("/messagesToXml", async(req, res, next) => {
    try {
        const message = await Contact.find()
        if (message == '') {
            return res.status(404).json({
                error: true,
                message: "No message Found .."
            })
        }
        res.set('Content-Type', 'text/xml');
        res.send(xml(message));

    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }

});
```

```
//get  messages to email
Complexity is 5 Everything is cool!
router.get("/messagesToEmail/:email", async(req, res, next) => {
    try {
        const _email = req.params.email;
        const message = await Contact.find({ email: _email })
        if (message == '') {
            return res.status(404).json({
                error: true,
                message: "No message Found .."
            })
        }
        res.json(message)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }
});

module.exports = router;
```

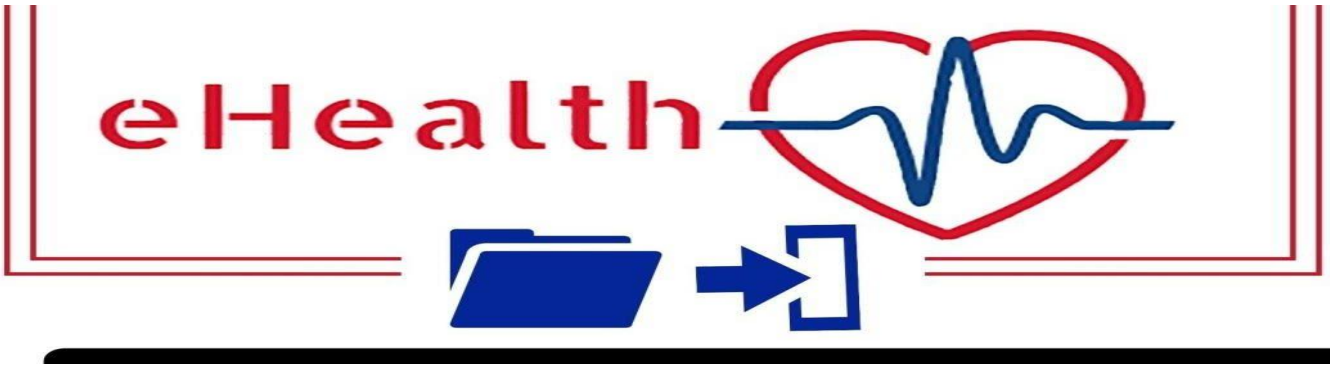
```
//get all Phis
Complexity is 5 Everything is cool!
router.get("/phis", async(req, res, next) => {
    try {
        const phi = await Phi.find()
        if (phi == '') {
            return res.status(404).json({
                error: true,
                message: "No Phis Found .."
            })
        }
        res.json(phi)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }

});
```

```
// get  phi to their nic
Complexity is 5 Everything is cool!
router.get("/phi/:nic", async(req, res, next) => {
    try {
        const _nic = req.params.nic;
        const phi = await Phi.find({ nic: _nic })
        if (phi == '') {
            return res.status(404).json({
                error: true,
                message: "No report phi on that NIC .."
            })
        }
        res.json(phi)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }
});


module.exports = router;
```

```
//get all place visits
Complexity is 5 Everything is cool!
router.get("/visits", async(req, res, next) => {
    try {
        const visit = await Visit.find()
        if (visit == '') {
            return res.status(404).json({
                error: true,
                message: "No Visits Found .."
            })
        }
        res.json(visit)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }

});
```

```
//get all place visits by nic
Complexity is 5 Everything is cool!
router.get("/visit/:nic", async(req, res, next) => {
    try {
        const _nic = req.params.nic;
        const visit = await Visit.find({ nic: _nic })
        if (visit == '') {
            return res.status(404).json({
                error: true,
                message: "No Visits to this.."
            })
        }
        res.json(visit)
    } catch (e) {
        res.status(500).json({
            error: true,
            message: e.message
        })
    }
});
```

```
// add new visit

Complexity is 3 Everything is cool!
router.post("/AddVisit", async(req, res, next) => {
    try {
        const visit = await Visit.create({
            _id: new mongoose.Types.ObjectId(),
            fname: req.body.fname,
            lname: req.body.lname,
            email: req.body.email,
            address: req.body.address,
            phone: req.body.phone,
            nic: req.body.nic,
            location: req.body.location,
            date: req.body.date,
            cTime: req.body.cTime


        })

        res.json(visit)
    } catch (e) {
        next(e)
    }
});


module.exports = router;
```


eHealth