

**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															
<p>Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.</p>															
<p>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.</p> <table border="1"> <thead> <tr> <th>Student Name</th> <th>Plymouth Index No</th> </tr> </thead> <tbody> <tr> <td>I.K.N.K Weerasinghe</td> <td>10674043</td> </tr> <tr> <td>G.L.I Karunananayake</td> <td>10673980</td> </tr> <tr> <td>Kely Weerasooriya</td> <td>10674044</td> </tr> <tr> <td>H.W.C Waduge</td> <td>10674040</td> </tr> <tr> <td>M.M.P.M Bandara</td> <td>10673074</td> </tr> <tr> <td>R.P.L Wanasinghe</td> <td>10674041</td> </tr> </tbody> </table>		Student Name	Plymouth Index No	I.K.N.K Weerasinghe	10674043	G.L.I Karunananayake	10673980	Kely Weerasooriya	10674044	H.W.C Waduge	10674040	M.M.P.M Bandara	10673074	R.P.L Wanasinghe	10674041
Student Name	Plymouth Index No														
I.K.N.K Weerasinghe	10674043														
G.L.I Karunananayake	10673980														
Kely Weerasooriya	10674044														
H.W.C Waduge	10674040														
M.M.P.M Bandara	10673074														
R.P.L Wanasinghe	10674041														
<p><i>We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.</i></p>															
<p>Signed on behalf of the group: I.K.N.K Weerasinghe</p>															
<p>Individual assignment: <i>I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.</i></p>															
<p>Signed :</p>															
<p>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.</p>															
<p>I *have used/not used translation software.</p>															
<p>If used, please state name of software.....</p>															
<p>Overall mark _____ % Assessors Initials _____ Date _____</p>															

*Please delete as appropriateSci/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	K.Nipuni.
Kely Weerasooriya	10674044	Kely
G.L.I Karunananayake	10673980	T.L
H.W.C. Waduge	10674040	Hegash
M.M.P.M Bandara	10673074	Poggani
R.P.L Wanasinghe	10674041	R.P.Wanasinghe

Table of Contents

Table of Contents

1. Introduction	4
API Documentation.....	5
2. Getting Started.....	5
a) Controller 1: Admin	6
b) Controller 2: PHI	10
c) Controller 3: Citizen.....	17
3. Tools and Technologies.....	30
4. Client Application.....	32
a) Web application.....	32
b) Mobile application.....	42
Individual Contribution.....	53
5. References	59
6. Appendices.....	60
a) ER Diagram.....	60
b) Controller Classes	61

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 Disease 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

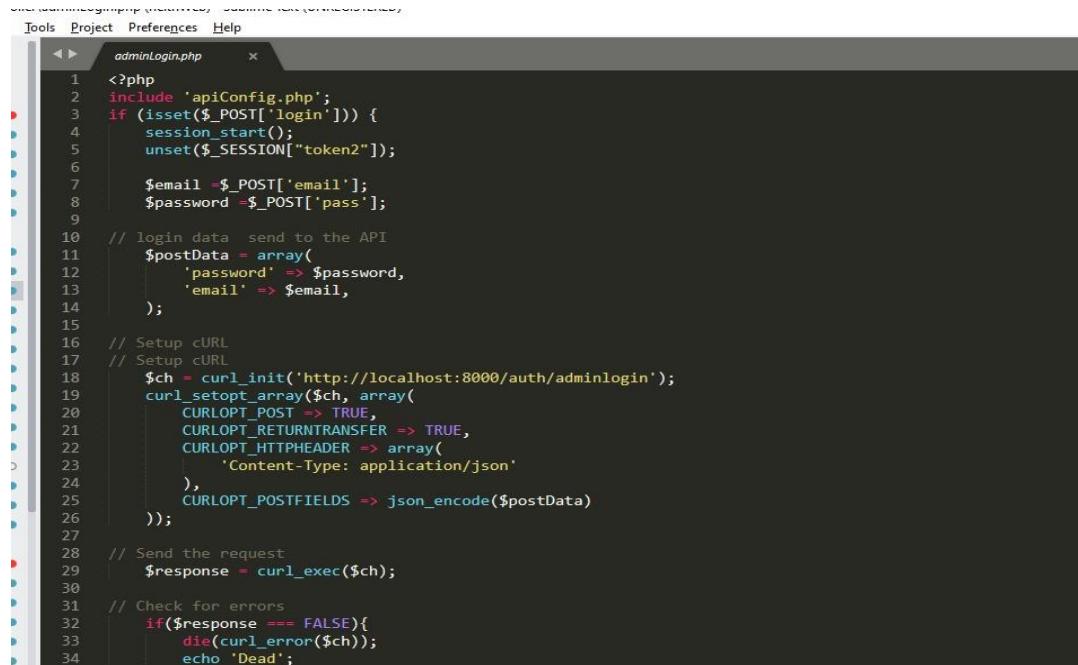
	Lists of APIs
adminLogin	<code>http://localhost:8000/auth/adminlogin</code>
addContact	<code>http://localhost:8000/contact/AddMessage</code>
citizen_signup	<code>http://localhost:8000/auth/citizen/signup</code>
citizenLogin	<code>http://localhost:8000/auth/citizenlogin</code>
deleteCitizen	<code>http://localhost:8000/citizen/updatestate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

a) Controller 1: Admin

1. Admin Login :

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

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



The screenshot shows a code editor window with the file 'adminLogin.php' open. The code is written in PHP and performs an admin login. It includes an include statement for 'apiConfig.php', checks if a 'login' key is set in the POST data, starts a session, and unsets a session variable. It then extracts email and password from the POST data. The code then sets up a cURL handle to send a POST request to 'http://localhost:8000/auth/adminlogin'. The cURL options include setting the post data as JSON, setting the content type to 'application/json', and setting the return transfer option. Finally, it sends the request using curl_exec, checks for errors, and outputs 'Dead' if there are any errors.

```
<?php
include 'apiConfig.php';
if (isset($_POST['login'])) {
    session_start();
    unset($_SESSION["token2"]);
}

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

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

// Setup cURL
// Setup cURL
$ch = curl_init('http://localhost:8000/auth/adminlogin');
curl_setopt_array($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';
```

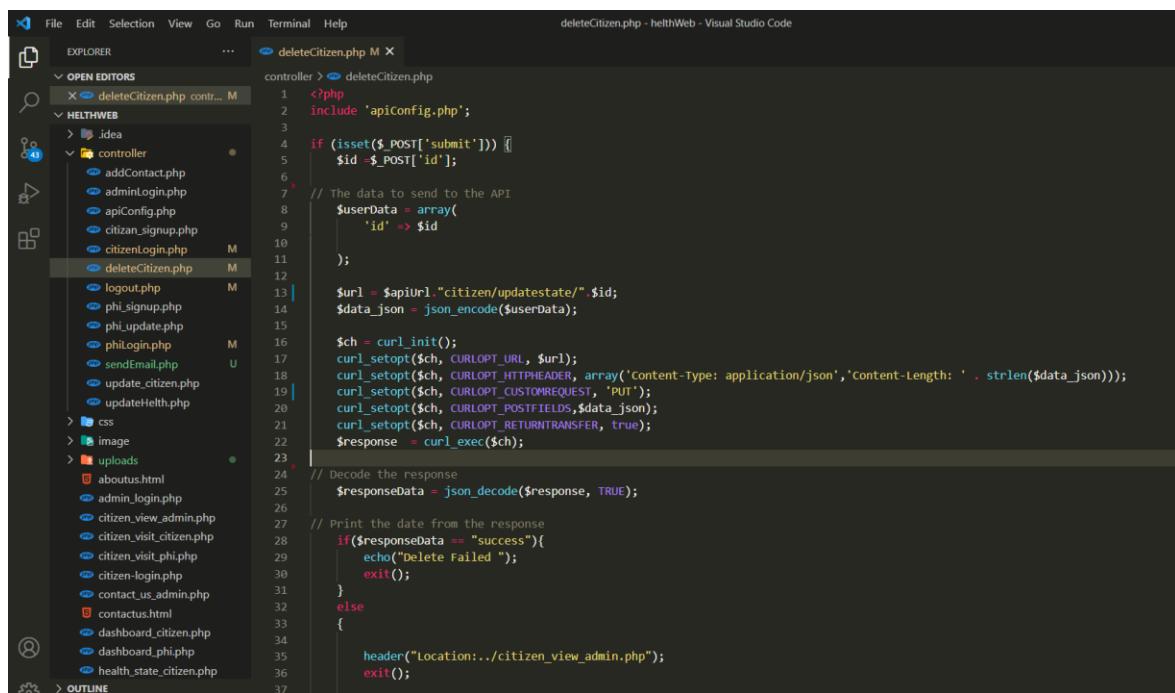
```

273     });
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       res.header('auth-token').send({
287         id: admin._id,
288         email: admin.email,
289         token: token
290       });
291     });
292   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>



```

File Edit Selection View Go Run Terminal Help
EXPLORE controller > deleteCitizen.php M
HEALTHWEB
controller > deleteCitizen.php
1 <?php
2 include 'apiConfig.php';
3
4 if (isset($_POST['submit'])) {
5   $id = $_POST['id'];
6
7   // The data to send to the API
8   $userData = array(
9     'id' => $id
10   );
11
12
13   $url = $apiUrl."citizen/updatestate/".$id;
14   $data_json = json_encode($userData);
15
16   $ch = curl_init();
17   curl_setopt($ch, CURLOPT_URL, $url);
18   curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json','Content-Length: ' . strlen($data_json)));
19   curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "PUT");
20   curl_setopt($ch, CURLOPT_POSTFIELDS,$data_json);
21   curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
22   $response = curl_exec($ch);
23
24   // Decode the response
25   $responseData = json_decode($response, TRUE);
26
27   // Print the date from the response
28   if($responseData == "success"){
29     echo("Delete Failed ");
30     exit();
31   }
32   else
33   {
34
35     header("Location:../citizen_view_admin.php");
36     exit();
37

```

The screenshot shows the Visual Studio Code interface with the file `citizen.js` open in the editor. The code is a Node.js module for updating a citizen's account status. It uses the `router.put` method to handle the `/updatestate/:id` endpoint. The code attempts to find and update the citizen by ID, setting the status to "Deactivated" and marking it as a new record with `new: true`. If successful, it returns the updated citizen object in JSON format. If there's an error, it catches it and passes it to the next middleware.

```
routes > citizen.js > router.put("/updatestate/:id") callback
164
165
166
167 //deactive account
168 router.put("/updatestate/:id", async(req, res, next) => {
169   try {
170     const citizen = await Citizen.findByIdAndUpdate({ _id: req.params.id }, {
171       status: "Deactivated",
172     },
173     { new: true,
174       userfindAndModify: false,
175       upsert: true
176     })
177   }
178   res.json(citizen)
179 } catch (e) {
180   next(e)
181 }
182
183 });
184 );
```

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>

The screenshot shows the Visual Studio Code interface with the file `sendEmail.php` open in the editor. This PHP script sends an email to a citizen using an API. It includes an `apiConfig.php` file for configuration. The script checks if a form submission was made. If so, it extracts the email address from the POST data and creates an array for user data with an email key. It then constructs a URL for the API endpoint and sends a PUT request with the user data as JSON. Finally, it decodes the response and prints it to the console. If the response indicates success, it exits; otherwise, it prints an error message.

```
controller > sendEmail.php
1 <?php
2 include 'apiConfig.php';
3
4 if (isset($_POST['submit'])) {
5
6   $email = $_POST['email'];
7
8   // The data to send to the API
9   $userData = array(
10     'email' => $email,
11   );
12
13
14   $url = $apiUrl."citizen/email/".$email;
15   $data_json = json_encode($userData);
16
17   $ch = curl_init();
18   curl_setopt($ch, CURLOPT_URL, $url);
19   curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json', 'Content-Length: ' . strlen($data_json)));
20   curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
21   curl_setopt($ch, CURLOPT_POSTFIELDS, $data_json);
22   curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
23   $response = curl_exec($ch);
24
25   // Decode the response
26   $responseData = json_decode($response, TRUE);
27
28   // Print the date from the response
29   if($responseData == "success"){
30     echo("Email Send failed");
31     exit();
32   }
33
34   header("Location:../citizen_view_admin.php");
35
36   exit();
37
38 }
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "E-HEALTH-API-MASTER".
- Open Editors:** Displays two files: "citizen.js" and "auth.js".
- Editor:** The "citizen.js" file is open, showing code for a PUT route to deactivate an account. It uses nodemailer to send an email to the provided email address.
- Search Bar:** Shows a search for "citizenjs" with no results found.

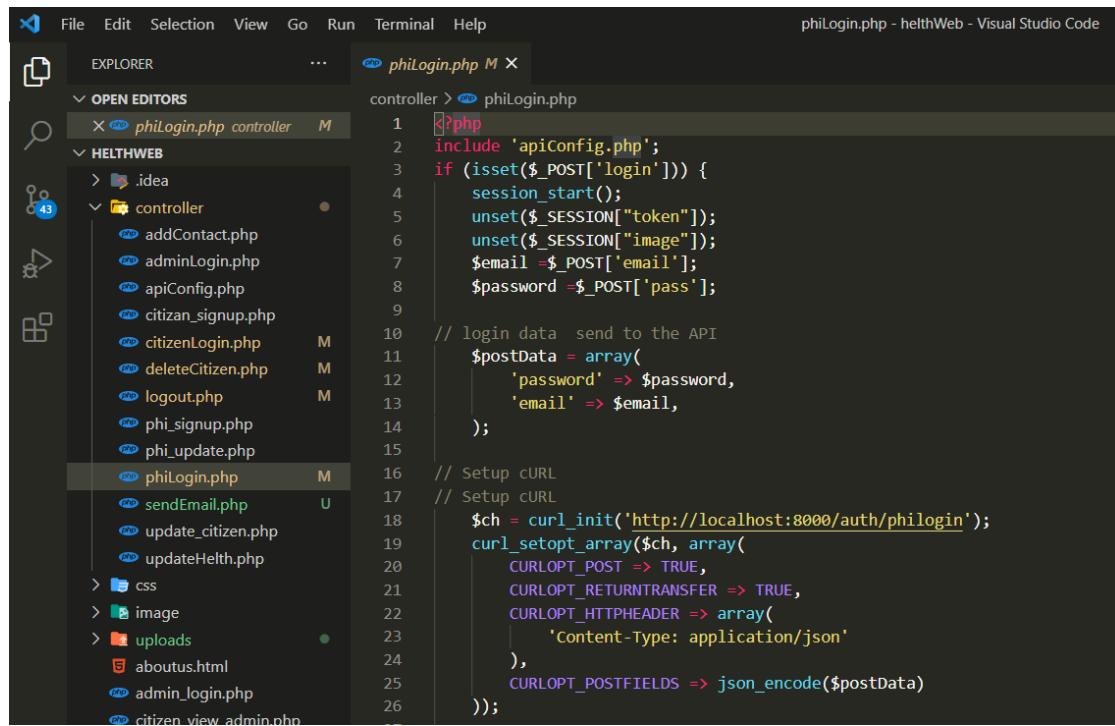
```
citizen.js - e-health-API-master - Visual Studio Code

File Edit Selection View Go Run Terminal Help
OPEN EDITORS
citizen.js x auth.js
routes > citizen.js > ...
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 Regards,\nE-Health.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;
```

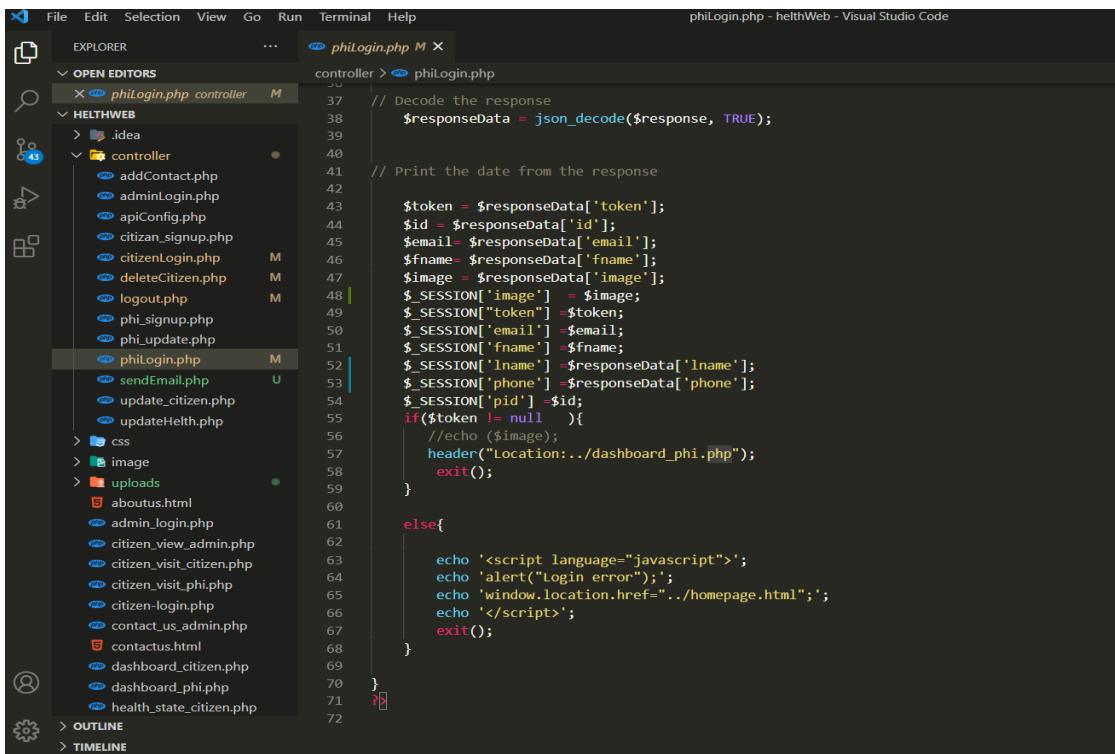
b) Controller 2: PHI

1. PHI Login: This is the login of PHI

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



```
<?php
include 'apiConfig.php';
if (isset($_POST['login'])) {
    session_start();
    unset($_SESSION["token"]);
    unset($_SESSION["image"]);
    $email = $_POST['email'];
    $password = $_POST['pass'];
}
// login data send to the API
$postData = array(
    'password' => $password,
    'email' => $email,
);
// Setup cURL
// Setup curl
$ch = curl_init('http://localhost:8000/auth/philogin');
curl_setopt_array($ch, array(
    CURLOPT_POST => TRUE,
    CURLOPT_RETURNTRANSFER => TRUE,
    CURLOPT_HTTPHEADER => array(
        'Content-Type: application/json'
    ),
    CURLOPT_POSTFIELDS => json_encode($postData)
));
```



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

// Print the date from the response

$token = $responseData['token'];
$id = $responseData['id'];
$email = $responseData['email'];
$fname = $responseData['fname'];
$image = $responseData['image'];
$_SESSION['image'] = $image;
$_SESSION['token'] = $token;
$_SESSION['email'] = $email;
$_SESSION['fname'] = $fname;
$_SESSION['lname'] = $responseData['lname'];
$_SESSION['phone'] = $responseData['phone'];
$_SESSION['pid'] = $id;
if ($token != null){
    //echo ($image);
    header("Location:../dashboard_phi.php");
    exit();
}
else{
    echo '<script language="javascript">';
    echo 'alert("login error");';
    echo 'window.location.href="..homepage.html";';
    echo '</script>';
    exit();
}
```

```

File Edit Selection View Go Run Terminal Help
authjs - e-health-API-master - Visual Studio Code

OPEN EDITORS
auth.js
routes > auth.js > router.post('/addAdmin') callback > then() callback

//login PHI
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,
  });
});

```

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

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

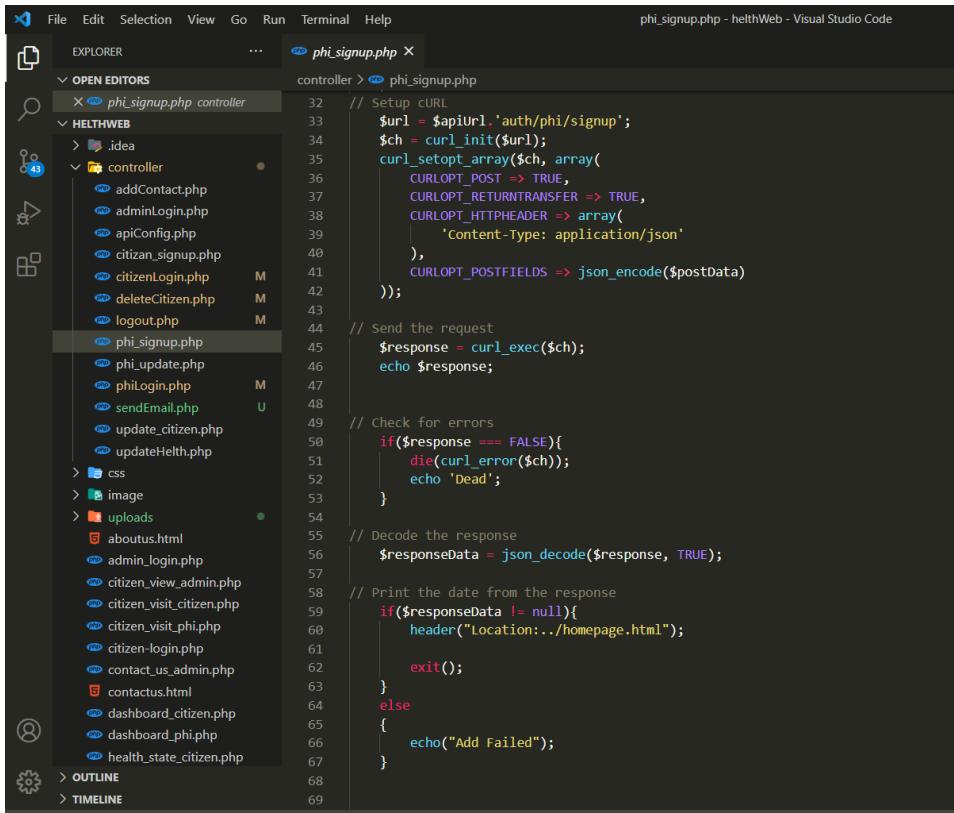
```

File Edit Selection View Go Run Terminal Help
phi_signup.php - helthWeb - Visual Studio Code

OPEN EDITORS
phi_signup.php
controller > phi_signup.php

1 <?php
2 include 'apiconfig.php';
3 if (isset($_POST['signup'])) {
4
5   $fname = $_POST['fname'];
6   $lname = $_POST['lname'];
7   $nic = $_POST['nic'];
8   $phone = $_POST['phone'];
9   $email = $_POST['email'];
10  $address = $_POST['address'];
11  $location = $_POST['location'];
12  $password = $_POST['password'];
13  $image_name = $_FILES["file"]["name"];
14  $image_type = $_FILES["file"]["type"];
15  $file_temp_name = $_FILES["file"]["tmp_name"];
16  $path = move_uploaded_file($file_temp_name, "C:/xampp/htdocs/helthWeb/uploads/".$image_name);
17
18 // The data to send to the API
19 $postData = array(
20   'fname' => $fname ,
21   'lname' => $lname,
22   'nic' => $nic,
23   'phone' => $phone,
24   'email' => $email,
25   'address' => $address,
26   'location' => $location,
27   'image' => $image_name,
28   'password' => $password,
29

```



```
controller > phi_signup.php
32 // Setup cURL
33 $url = $apiUrl.'auth/phi/signup';
34 $ch = curl_init($url);
35 curl_setopt_array($ch, array(
36     CURLOPT_POST => TRUE,
37     CURLOPT_RETURNTRANSFER => TRUE,
38     CURLOPT_HTTPHEADER => array(
39         'Content-Type: application/json'
40     ),
41     CURLOPT_POSTFIELDS => json_encode($postData)
42 ));
43
44 // Send the request
45 $response = curl_exec($ch);
46 echo $response;
47
48 // Check for errors
49 if($response === FALSE){
50     die(curl_error($ch));
51     echo 'Dead';
52 }
53
54 // Decode the response
55 $responseData = json_decode($response, TRUE);
56
57 // Print the date from the response
58 if($responseData != null){
59     header("Location:../homepage.html");
60
61     exit();
62 }
63 else
64 {
65     echo("Add Failed");
66 }
67
68
69
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Path:** auth.js - e-health-API-master - Visual Studio Code
- Explorer View:** Shows the project structure under "E-HEALTH-API-MASTER".
- Code Editor:** Displays the content of the auth.js file.

```
routes > auth.js > router.post('/addAdmin') callback > then() callback
14 //phi signup route
15 router.post('/phi/signup', (req, res, next) => {
16   Phi.find({ email: req.body.email })
17     .exec()
18     .then(phi => { //check if the email is allready taken
19       if (phi.length >= 1) {
20         return res.status(409).json({
21           message: 'Email Not Available.'
22         });
23       } else {
24         bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
25           if (err) {
26             return res.status(500).json({
27               error: err
28             });
29           } else {
30             const phi = new Phi({
31               _id: new mongoose.Types.ObjectId(),
32               fname: req.body.fname,
33               lname: req.body.lname,
34               nic: req.body.nic,
35               email: req.body.email,
36               phone: req.body.phone,
37               address: req.body.address,
38               location: req.body.location,
39               image: req.body.image,
40               password: hash
41             });
42             phi.save()
43               .then(result => {
44                 console.log(result);
45                 res.status(201).json({
46                   message: "Sign up Sucessfully",
47                   createdUser: result
48                 });
49               });
50           }
51         });
52       });
53     });
54   });
55   });
56   });
57   });
58   });
59   });
60   });
61   });
62   });
63   });

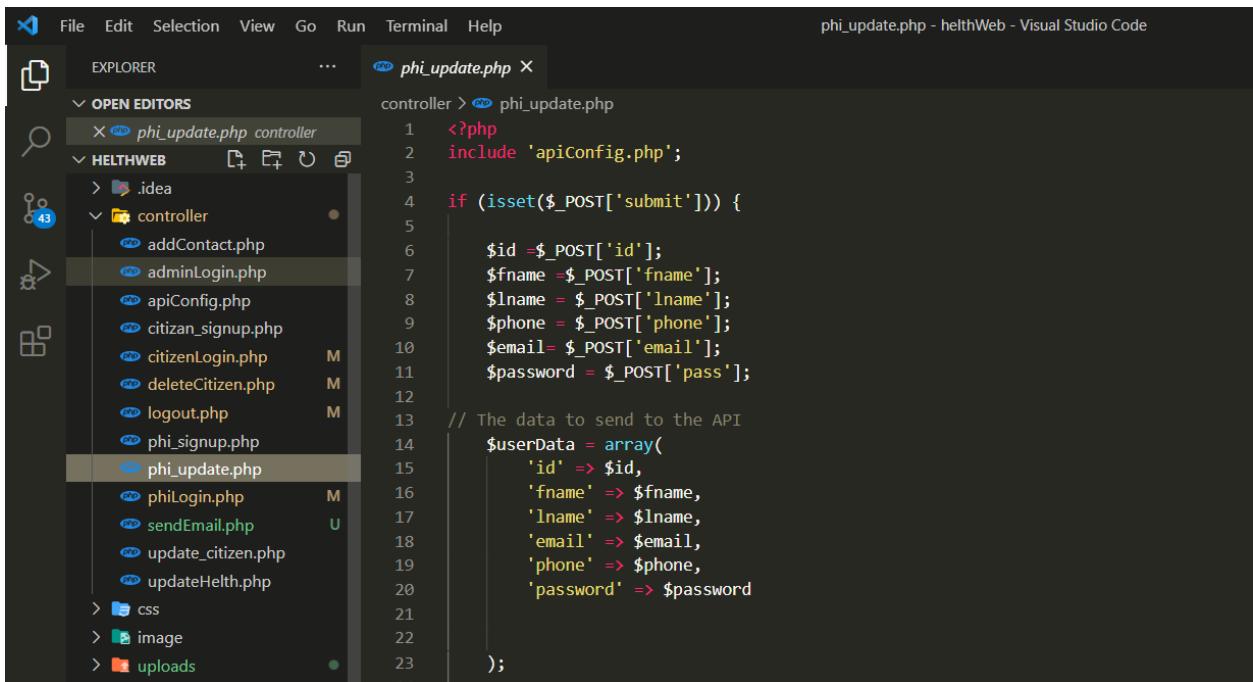
});
```

The screenshot shows the continuation of the auth.js file code from the previous screenshot, with the cursor positioned at the end of the file.

```
});  
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
  });
});
```

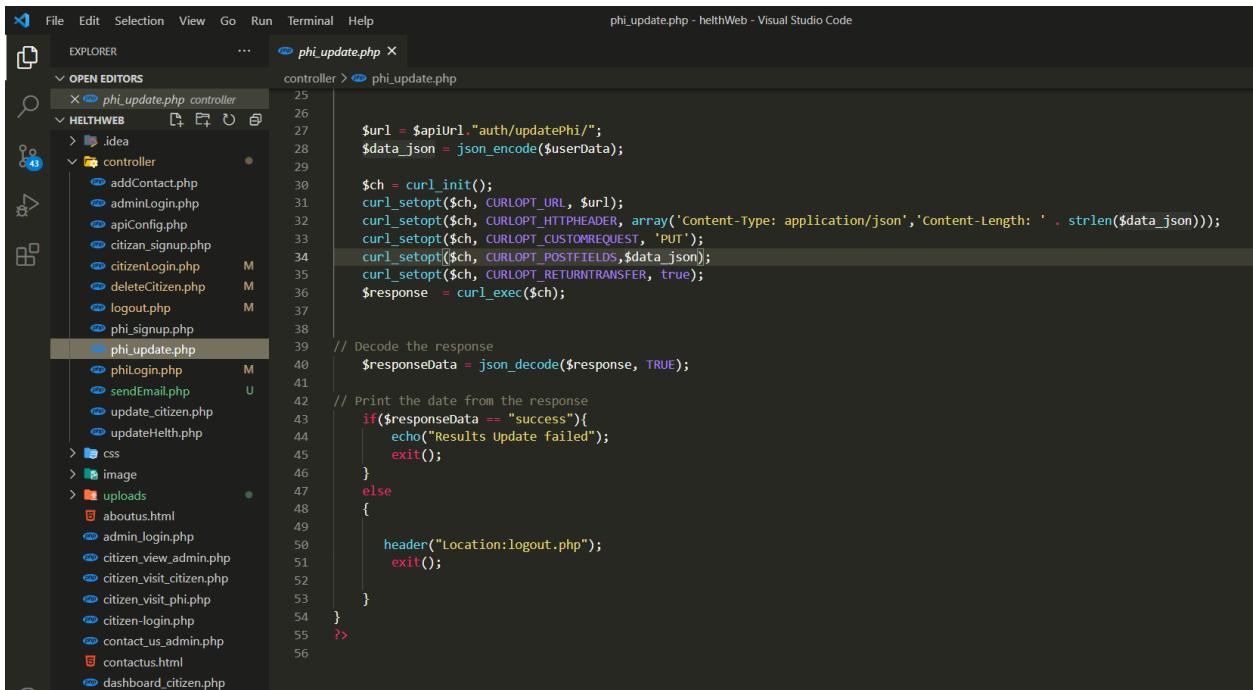
3.PHI Update Profile : PHI can update there profile. This is the profile updating part.

Routes: <http://localhost:8000/auth/updatePhi>



The screenshot shows the Visual Studio Code interface with the file `phi_update.php` open in the editor. The code handles a POST request to update a user's profile. It includes an include statement for `apiConfig.php` and a conditional block to check if the `submit` button was pressed. If so, it extracts various user input fields (`$id`, `$fname`, `$lname`, `$phone`, `$email`, `$password`) and prepares them for sending to an API. The `$userData` array maps each field to its corresponding POST value.

```
<?php
include 'apiConfig.php';
if (isset($_POST['submit'])) {
    $id = $_POST['id'];
    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $phone = $_POST['phone'];
    $email = $_POST['email'];
    $password = $_POST['pass'];
    // The data to send to the API
    $userData = array(
        'id' => $id,
        'fname' => $fname,
        'lname' => $lname,
        'email' => $email,
        'phone' => $phone,
        'password' => $password
    );
}
```



The screenshot shows the continuation of the `phi_update.php` file. It uses the `curl` library to send a PUT request to the API endpoint `$apiUrl."/auth/updatePhi"` with the JSON-encoded `$userData`. After executing the curl command, it decodes the response and prints it to the console. If the response is "success", it prints "Results update failed" and exits. Otherwise, it prints "header(Location:logout.php)" and exits.

```
$url = $apiUrl."/auth/updatePhi/";
$data_json = json_encode($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

OPEN EDITORS
  routes > auth.js > ...
  routes > auth.js routes
  routes > citizen.js routes
  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

174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

//update phi
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.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
updateHealth.php - helthWeb - Visual Studio Code

OPEN EDITORS
  updateHealth.php controller
  controller > updateHealth.php

HELTHWEB
  .idea
  controller
    addContact.php
    adminLogin.php
    apiConfig.php
    citizen_signup.php
    citizenLogin.php
    deleteCitizen.php
    logout.php
    phi_signup.php
    phi_update.php
    phiLogin.php
    sendEmail.php
    update_citizen.php
    updateHealth.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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34

?php
include 'apiConfig.php';
if (isset($_POST['submit'])) {

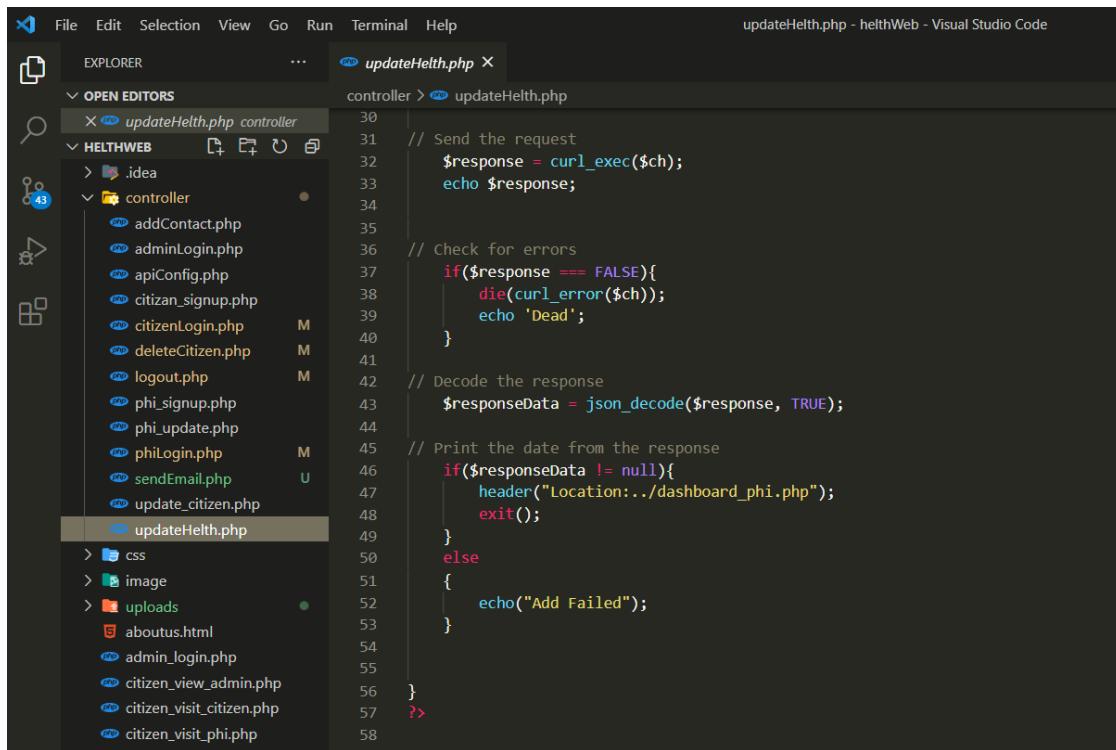
$id = $_POST['id'];
$test = $_POST['test'];
$results = $_POST['status'];
$nic = $_POST['nic'];

// The data to send to the API
$postData = array(
  'id' => $id ,
  'results' => $results,
  'test' => $test,
  'nic' => $nic
);

// Setup cURL
$url = $apiUrl.'citizen/addCitizenReport';
$ch = curl_init($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;
}

```



```
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
1 <?php
2 include 'apiconfig.php';
3 if (isset($_POST['login'])) {
4     session_start();
5     unset($_SESSION["token"]);
6     unset($_SESSION["image"]);
7
8     $email = $_POST['email'];
9     $password = $_POST['pass'];
10
11    // login data send to the API
12    $postData = array(
13        'password' => $password,
14        'email' => $email,
15    );
16
17    // Setup curl
18
19    $ch = curl_init('http://localhost:8000/auth/citizenlogin');
20    curl_setopt_array($ch, array(
21        CURLOPT_POST => TRUE,
22        CURLOPT_RETURNTRANSFER => TRUE,
23        CURLOPT_HTTPHEADER => array(
24            'Content-Type: application/json'
25        ),
26        CURLOPT_POSTFIELDS => json_encode($postData)
27    ));
28
29    // Send the request
30    $response = curl_exec($ch);
31
32    // Check for errors
33    if($response === FALSE){
34        die(curl_error($ch));
35        echo 'Dead';
36    }
37
38    // Decode the response
39    $responseData = json_decode($response, TRUE);
40    echo($responseData['token']);
41
42    // Print the date from the response
43
44    $token = $responseData['token'];
45    $id = $responseData['id'];
46    $email= $responseData['email'];
47    $fname= $responseData['name'];
48    $image = $responseData['image'];
49    $nic = $responseData['nic'];
50    $SESSION["token"] = $token;
51    $SESSION["email"] = $email;
52    $SESSION["fname"] = $fname;
53    $SESSION["image"] = $image;
54    $SESSION["nic"] = $nic;
55    $SESSION["cid"] = $id;
56    $SESSION["addr"] = $responseData['address'];
57    $SESSION["lct"] = $responseData['location'];
58    $SESSION["aff"] = $responseData['affiliation'];
59    $SESSION["prof"] = $responseData['profeson'];
60    $SESSION["lname"] = $responseData['name'];
61    $SESSION["phone"] = $responseData['phone'];
62    $SESSION["age"] = $responseData['age'];
63
64    if($token != null ){
65        //echo ($image);
66        header("Location:../dashboard_citizen.php");
67        exit();
68    }
69
70    else{
71        echo '<script language="javascript">';
72        echo 'alert("Login error");';
73        echo 'window.location.href="..homepage.html";';
74        echo '</script>';
75        exit();
76    }
77
78 }
79 ?>
```

```
37 // Decode the response
38 $responseData = json_decode($response, TRUE);
39 echo($responseData['token']);
40
41 // Print the date from the response
42
43 $token = $responseData['token'];
44 $id = $responseData['id'];
45 $email= $responseData['email'];
46 $fname= $responseData['name'];
47 $image = $responseData['image'];
48 $nic = $responseData['nic'];
49 $SESSION["token"] = $token;
50 $SESSION["email"] = $email;
51 $SESSION["fname"] = $fname;
52 $SESSION["image"] = $image;
53 $SESSION["nic"] = $nic;
54 $SESSION["cid"] = $id;
55 $SESSION["addr"] = $responseData['address'];
56 $SESSION["lct"] = $responseData['location'];
57 $SESSION["aff"] = $responseData['affiliation'];
58 $SESSION["prof"] = $responseData['profeson'];
59 $SESSION["lname"] = $responseData['name'];
60 $SESSION["phone"] = $responseData['phone'];
61 $SESSION["age"] = $responseData['age'];
62
63 if($token != null ){
64     //echo ($image);
65     header("Location:../dashboard_citizen.php");
66     exit();
67 }
68
69 else{
70     echo '<script language="javascript">';
71     echo 'alert("Login error");';
72     echo 'window.location.href="..homepage.html";';
73     echo '</script>';
74     exit();
75 }
76
77
78 }
79 ?>
```

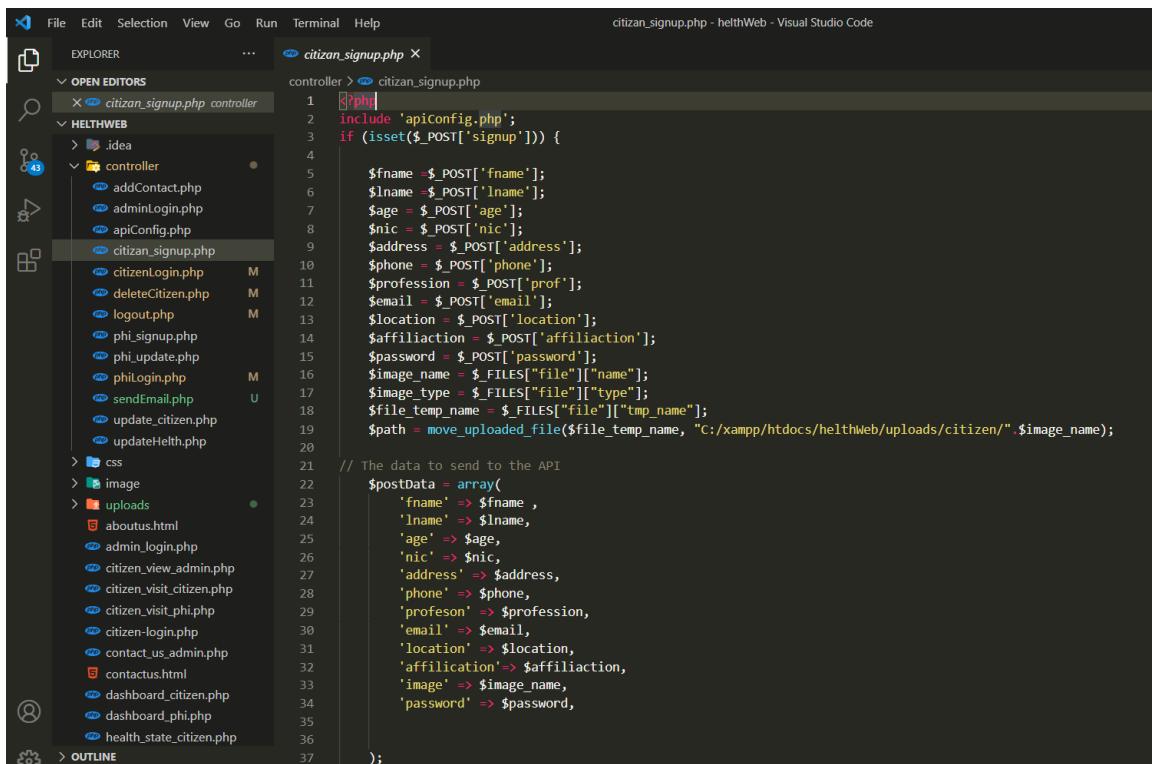
```

144 //login Citizen
145 router.post('/citizenlogin', async(req, res) => {
146   const citizen = await Citizen.findone({ email: req.body.email });
147   if (!citizen) return res.status(400).send({ message: 'Email does Not Exist' });
148
149   const validPass = await bcrypt.compare(req.body.password, citizen.password);
150
151   if (!validPass) return res.status(400).send({ message: 'Password is Wrong' });
152
153   const token = jwt.sign({ _id: citizen._id }, process.env.JWT_KEY);
154
155
156   res.header('auth-token').send({
157     id: citizen.id,
158     fname: citizen.fname,
159     email: citizen.email,
160     nic: citizen.nic,
161     image: citizen.image,
162     token,
163     lname: citizen.lname,
164     phone: citizen.phone,
165     age: citizen.age,
166     affiliation: citizen.affiliation,
167     profesion: citizen.profeson,
168     address: citizen.address,
169     message: citizen.fname,
170     status: citizen.status
171   });
172 });
173
174
175

```

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

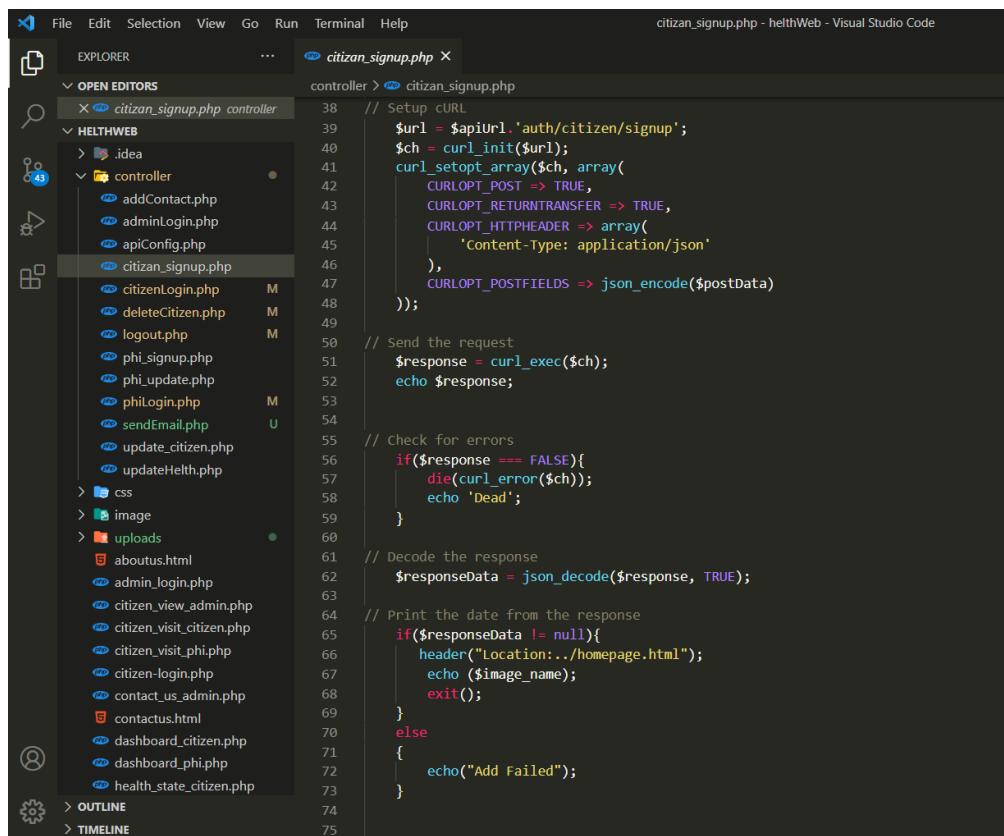
Routes: <http://localhost:8000/auth/citizen/signup>



```

File Edit Selection View Go Run Terminal Help
citizen_signup.php - helthWeb - Visual Studio Code
EXPLORER OPEN EDITORS controller > citizen_signup.php
HELTHWEB
> idea
controller
  addContact.php
  admintLogin.php
  apiConfig.php
  citizen_signup.php
  citizen_login.php M
  deleteCitizen.php M
  logout.php M
  phi_signup.php
  phi_update.php
  phi_login.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
controller > citizen_signup.php X
1 <?php
2 include 'apiconfig.php';
3 if (isset($_POST['signup'])) {
4
5   $fname = $_POST['fname'];
6   $lname = $_POST['lname'];
7   $age = $_POST['age'];
8   $nic = $_POST['nic'];
9   $address = $_POST['address'];
10  $phone = $_POST['phone'];
11  $profession = $_POST['prof'];
12  $email = $_POST['email'];
13  $location = $_POST['location'];
14  $affiliation = $_POST['affiliation'];
15  $password = $_POST['password'];
16  $image_name = $_FILES["file"]["name"];
17  $image_type = $_FILES["file"]["type"];
18  $file_temp_name = $_FILES["file"]["tmp_name"];
19  $path = move_uploaded_file($file_temp_name, "C:/xampp/htdocs/helthWeb/uploads/citizen/".$image_name);
20
21 // The data to send to the API
22 $postData = array(
23   'fname' => $fname ,
24   'lname' => $lname,
25   'age' => $age,
26   'nic' => $nic,
27   'address' => $address,
28   'phone' => $phone,
29   'profeson' => $profession,
30   'email' => $email,
31   'location' => $location,
32   'affiliation'=> $affiliation,
33   'image' => $image_name,
34   'password' => $password,
35
36 );

```



The screenshot shows the Visual Studio Code interface with the file `citizen_signup.php` open in the editor. The code implements a `citizen_signup.php` controller using cURL to interact with an API. It includes error handling, decoding the response, and printing the date from the response.

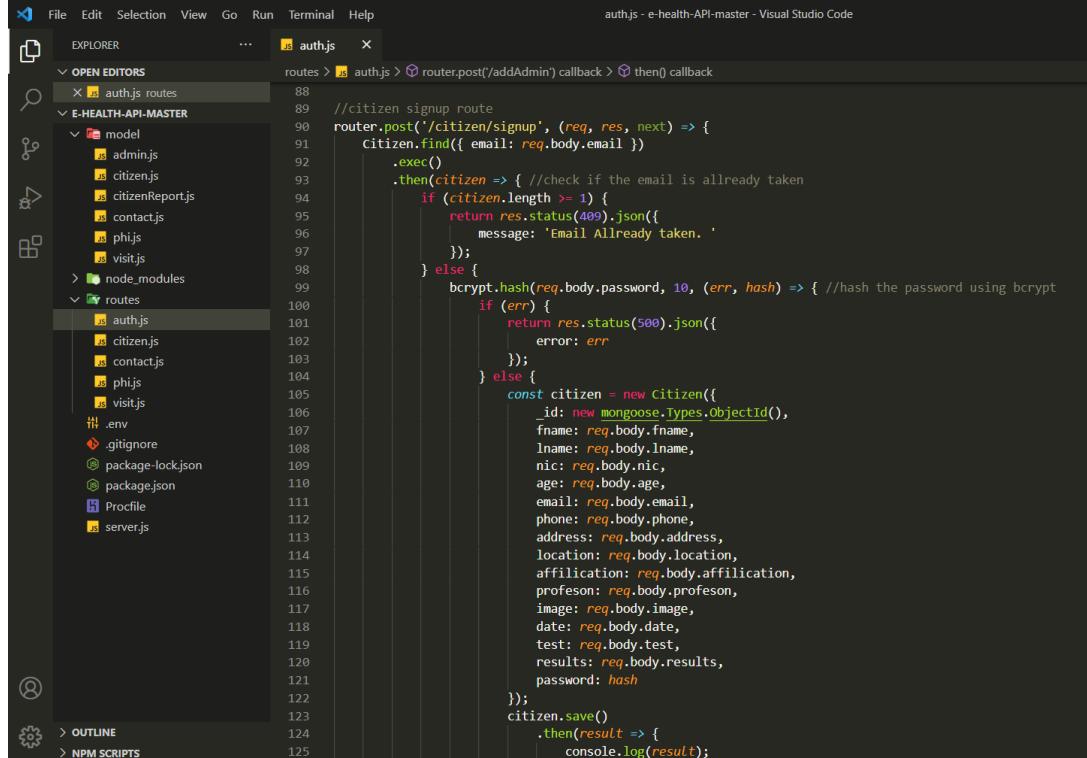
```
// Setup cURL
$url = $apiUrl.'auth/citizen/signup';
$ch = curl_init($url);
curl_setopt_array($ch, array(
    CURLOPT_POST => TRUE,
    CURLOPT_RETURNTRANSFER => TRUE,
    CURLOPT_HTTPHEADER => array(
        'Content-Type: application/json'
    ),
    CURLOPT_POSTFIELDS => json_encode($postData)
));

// 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");
}
```



The screenshot shows the Visual Studio Code interface with the file `auth.js` open in the editor. The code defines a route for citizen signup and handles the password hashing and citizen creation logic using Mongoose.

```
//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 brypt
                    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,
                            affiliation: req.body.affiliation,
                            profesion: req.body.profesion,
                            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);
                            })
                    }
                });
            }
        })
});
```

```

File Edit Selection View Go Run Terminal Help
auth.js - e-health-API-master - Visual Studio Code

OPEN EDITORS
auth.js
E-HEALTH-API-MASTER
model
  admin.js
  citizen.js
  citizenReport.js
  contact.js
  phi.js
  visit.js
routes
  auth.js
  citizen.js
  contact.js
  phi.js
  visit.js
  .env
  .gitignore
  package-lock.json
  package.json
  Profile
  server.js

auth.js (143 lines)
100 routes > auth.js > router.post('/addAdmin') callback > then() callback
101   10. new mongoose.Types.ObjectId(),
102     fname: req.body.fname,
103     lname: req.body.lname,
104     nic: req.body.nic,
105     age: req.body.age,
106     email: req.body.email,
107     phone: req.body.phone,
108     address: req.body.address,
109     location: req.body.location,
110     affiliation: req.body.affiliation,
111     profesion: req.body.profesion,
112     image: req.body.image,
113     date: req.body.date,
114     test: req.body.test,
115     results: req.body.results,
116     password: hash
117   );
118   citizen.save()
119     .then(result => {
120       console.log(result);
121       res.status(201).json({
122         message: "Sign up Sucessfully",
123         createdUser: result
124       });
125     })
126     .catch(err => {
127       console.log(err);
128       res.status(500).json({
129         error: err
130       });
131     })
132   }
133 }
134 )
135 )
136 )
137 )
138 )
139 )
140   }
141   }
142   }
143 });

}

```

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 - Visual Studio Code

OPEN EDITORS
addContact.php controller
HELTHWEB
  .idea
  controller
    addContact.php
    adminLogin.php
    apiConfig.php
    citizan_signup.php
    citizenLogin.php
    deleteCitizen.php
    logout.php
    phi_signup.php
    phi_update.php
    philogin.php
    sendEmail.php
    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

addContact.php (29 lines)
1 <?php
2 include 'apiConfig.php';
3 if (isset($_POST['contact'])) {
4
5   $name = $_POST['name'];
6   $email = $_POST['email'];
7   $message = $_POST['message'];
8
9   // The data to send to the API
10  $postData = array(
11    'name' => $name ,
12    'email' => $email,
13    'message' => $message,
14
15  );
16
17
18  );
19  // Setup cURL
20  $url = $apiUrl.'contact/AddMessage';
21  $ch = curl_init($url);
22  curl_setopt_array($ch, array(
23    CURLOPT_POST => TRUE,
24    CURLOPT_RETURNTRANSFER => TRUE,
25    CURLOPT_HTTPHEADER => array(
26      'Content-Type: application/json'
27    ),
28    CURLOPT_POSTFIELDS => json_encode($postData)
29  ));

}

```

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Editor Title:** addContact.php - helthWeb
- Explorer:** Shows the project structure under "OPEN EDITORS". The "controller" folder is expanded, showing files like addContact.php, adminLogin.php, apiConfig.php, citizan_signup.php, citizenLogin.php, deleteCitizen.php, logout.php, phi_signup.php, phi_update.php, phiLogin.php, sendEmail.php, update_citizen.php, and updateHelth.php.
- Code Editor:** The "addContact.php" file is open. The code handles a curl request to add a contact, checks for errors, decodes the response, and prints the date if successful, or echoes an error message if failed.

```
30 // Send the request
31 $response = curl_exec($ch);
32 echo $response;
33
34
35 // Check for errors
36 if($response === FALSE){
37     die(curl_error($ch));
38     echo 'Dead';
39 }
40
41 // Decode the response
42 $responseData = json_decode($response, TRUE);
43
44 // Print the date from the response
45 if($responseData != null){
46     header("Location:../homepage.html");
47     exit();
48 }
49 else
50 {
51     echo("Add Failed");
52 }
```

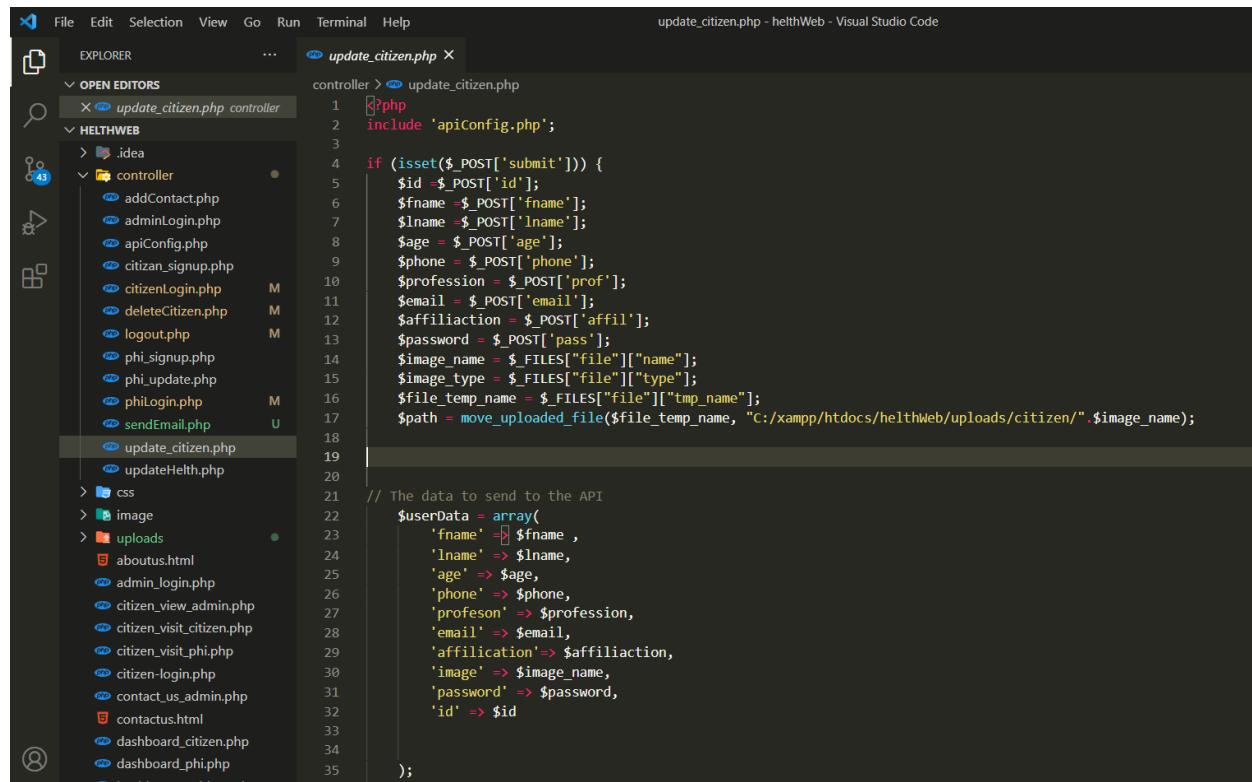
The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Editor Title:** contact.js - e-health-API-master - Visual Studio Code
- Explorer:** Shows the project structure under "OPEN EDITORS". The "contactjs routes" folder is expanded, showing files like citizen.js, contact.js, and contactRoutes.js.
- Code Editor:** The "contact.js" file is open. It defines an Express router for handling messages. It uses mongoose to create a new Contact document with name, email, and message fields, and then returns the created message as JSON.

```
1 const express = require('express');
2 const router = express.Router();
3 const mongoose = require('mongoose');
4 const Contact = require('../model/contact');
5 const xml = require('xml');
6
7 // add message
8
9 router.post("/AddMessage", async(req, res, next) => {
10     try {
11         const message = await Contact.create({
12             _id: new mongoose.Types.ObjectId(),
13             name: req.body.name,
14             email: req.body.email,
15             message: req.body.message
16         })
17
18         res.json(message)
19     } catch (e) {
20         next(e)
21     }
22 }
23 );
24 );
```

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

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



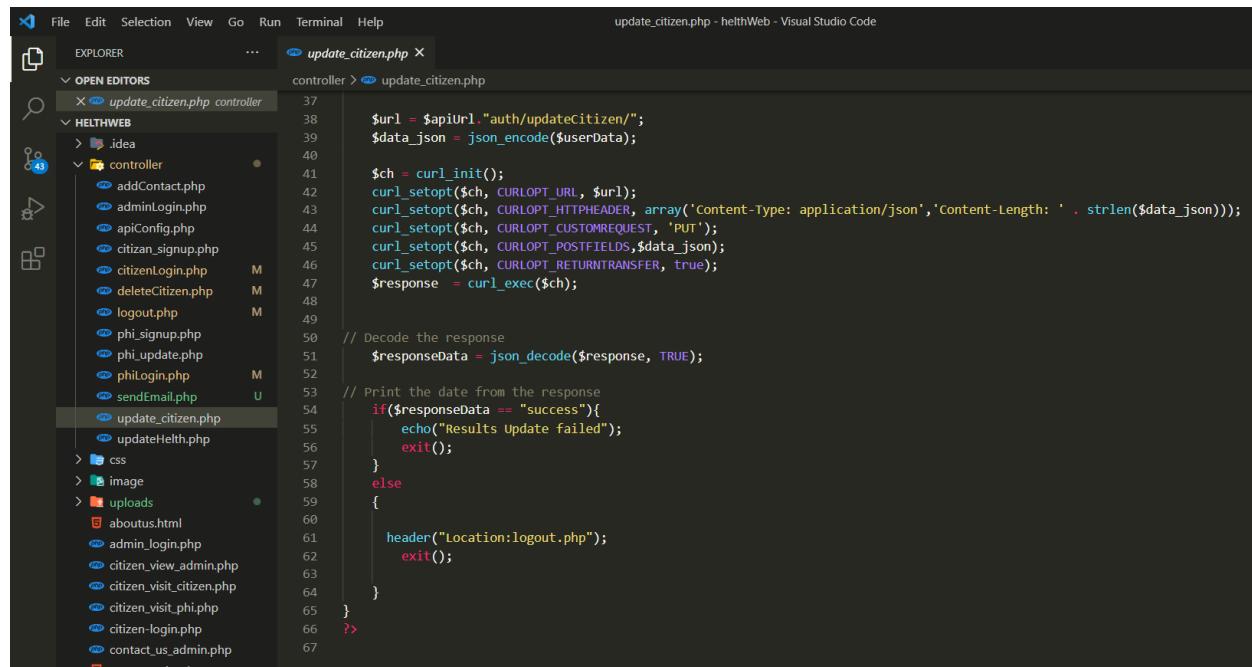
```

File Edit Selection View Go Run Terminal Help
update_citizen.php - helthWeb - Visual Studio Code

EXPLORER OPEN EDITORS controller > update_citizen.php
HEALTHWEB .idea controller
controller addContact.php M
controller adminLogin.php M
controller apiConfig.php M
controller citizen_signup.php M
controller citizenLogin.php M
controller deleteCitizen.php M
controller logout.php M
controller phi_signup.php M
controller phi_update.php M
controller phiLogin.php M
controller sendEmail.php U
controller update_citizen.php M
controller updateHealth.php M
css image uploads
uploads aboutus.html
uploads admin_login.php
uploads citizen_view_admin.php
uploads citizen_visit_citizen.php
uploads citizen_visit_phi.php
uploads citizen_login.php
uploads contact_us_admin.php
uploads contactus.html
uploads dashboard_citizen.php
uploads dashboard_phi.php

update_citizen.php
1 <?php
2 include 'apiConfig.php';
3
4 if (isset($_POST['submit'])) {
5     $id = $_POST['id'];
6     $fname = $_POST['fname'];
7     $lname = $_POST['lname'];
8     $age = $_POST['age'];
9     $phone = $_POST['phone'];
10    $profession = $_POST['prof'];
11    $email = $_POST['email'];
12    $affiliation = $_POST['affil'];
13    $password = $_POST['pass'];
14    $image_name = $_FILES["file"]["name"];
15    $image_type = $_FILES["file"]["type"];
16    $file_temp_name = $_FILES["file"]["tmp_name"];
17    $path = move_uploaded_file($file_temp_name, "c:/xampp/htdocs/helthWeb/uploads/citizen/".$image_name);
18
19
20 // The data to send to the API
21 $userData = array(
22     'fname' => $fname,
23     'lname' => $lname,
24     'age' => $age,
25     'phone' => $phone,
26     'profesn' => $profession,
27     'email' => $email,
28     'affiliation' => $affiliation,
29     'image' => $image_name,
30     'password' => $password,
31     'id' => $id
32 );
33
34
35 );

```



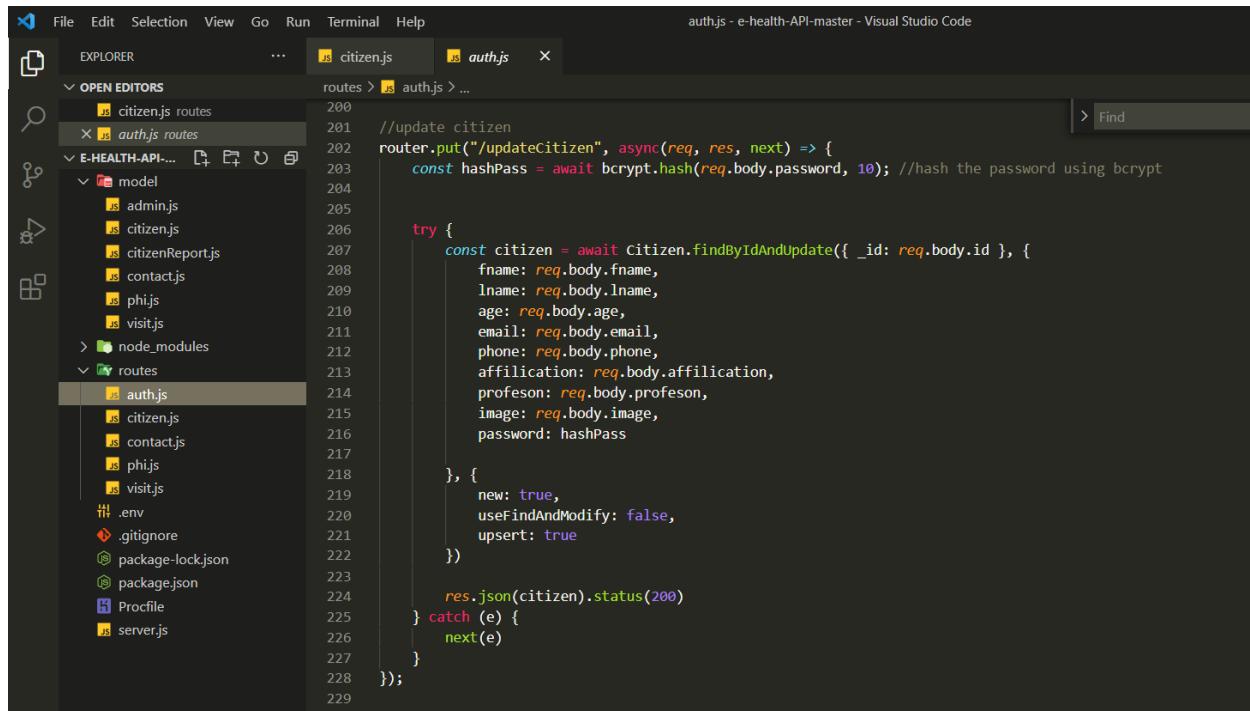
```

File Edit Selection View Go Run Terminal Help
update_citizen.php - helthWeb - Visual Studio Code

EXPLORER OPEN EDITORS controller > update_citizen.php
HEALTHWEB .idea controller
controller addContact.php M
controller adminLogin.php M
controller apiConfig.php M
controller citizen_signup.php M
controller citizenLogin.php M
controller deleteCitizen.php M
controller logout.php M
controller phi_signup.php M
controller phi_update.php M
controller phiLogin.php M
controller sendEmail.php U
controller update_citizen.php M
controller updateHealth.php M
css image uploads
uploads aboutus.html
uploads admin_login.php
uploads citizen_view_admin.php
uploads citizen_visit_citizen.php
uploads citizen_visit_phi.php
uploads citizen_login.php
uploads contact_us_admin.php
uploads contactus.html
uploads dashboard_citizen.php
uploads dashboard_phi.php

update_citizen.php
37
38     $url = $apiUrl."auth/updateCitizen/";
39     $data_json = json_encode($userData);
40
41     $ch = curl_init();
42     curl_setopt($ch, CURLOPT_URL, $url);
43     curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json', 'Content-Length: ' . strlen($data_json)));
44     curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
45     curl_setopt($ch, CURLOPT_POSTFIELDS, $data_json);
46     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
47     $response = curl_exec($ch);
48
49 // Decode the response
50     $responseData = json_decode($response, TRUE);
51
52 // Print the date from the response
53     if($responseData == "success"){
54         echo("Results Update failed");
55         exit();
56     }
57     else{
58
59         header("Location:logout.php");
60         exit();
61     }
62
63
64
65
66 ?>
67

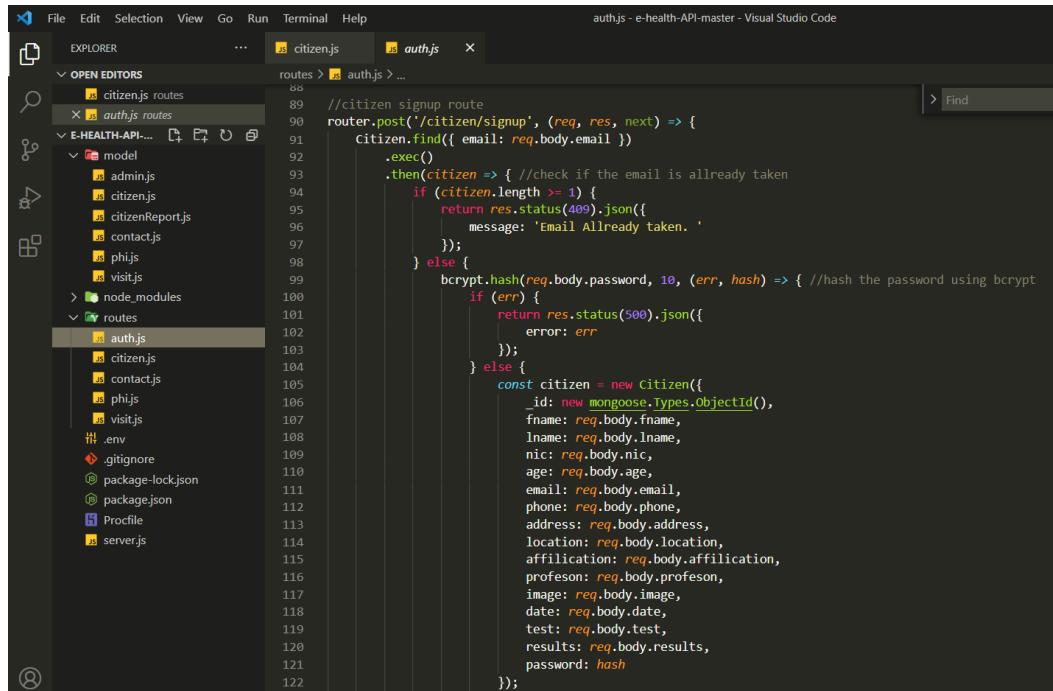
```



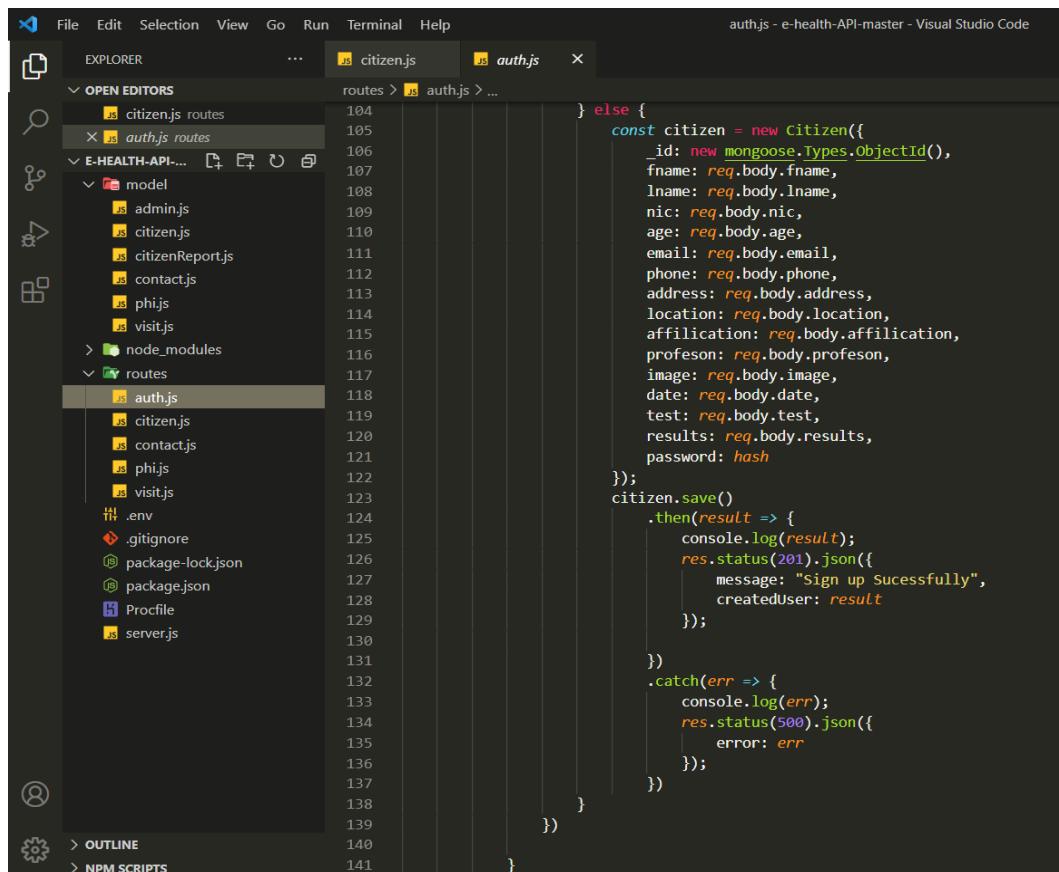
```
citizen.js auth.js
routes > auth.js > ...
200 //update citizen
201 router.put("/updateCitizen", async(req, res, next) => {
202     const hashPass = await bcrypt.hash(req.body.password, 10); //hash the password using bcrypt
203
204     try {
205         const citizen = await Citizen.findByIdAndUpdate({ _id: req.body.id }, {
206             fname: req.body.fname,
207             lname: req.body.lname,
208             age: req.body.age,
209             email: req.body.email,
210             phone: req.body.phone,
211             affiliation: req.body.affiliation,
212             profesion: req.body.profesion,
213             image: req.body.image,
214             password: hashPass
215         }, {
216             new: true,
217             useFindAndModify: false,
218             upsert: true
219         })
220
221         res.json(citizen).status(200)
222     } catch (e) {
223         next(e)
224     }
225 }
226
227 });
228
229 );
```

Minimal Required APIs

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



```
File Edit Selection View Go Run Terminal Help
EXPLORER routes > auth.js > ...
OPEN EDITORS citizen.js routes
E-HEALTH-API... routes > auth.js > ...
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
auth.js
routes > auth.js > ...
88 //citizen signup route
89 router.post('/citizen/signup', (req, res, next) => {
90   Citizen.find({ email: req.body.email })
91     .exec()
92       .then(citizen => { //check if the email is allready taken
93         if (citizen.length > 1) {
94           return res.status(400).json({
95             message: 'Email Allready taken.'
96           });
97         } else {
98           bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
99             if (err) {
100               return res.status(500).json({
101                 error: err
102               });
103             } else {
104               const citizen = new Citizen({
105                 _id: new mongoose.Types.ObjectId(),
106                 fname: req.body.fname,
107                 lname: req.body.lname,
108                 nic: req.body.nic,
109                 age: req.body.age,
110                 email: req.body.email,
111                 phone: req.body.phone,
112                 address: req.body.address,
113                 location: req.body.location,
114                 affiliation: req.body.affiliation,
115                 profesion: req.body.profesion,
116                 image: req.body.image,
117                 date: req.body.date,
118                 test: req.body.test,
119                 results: req.body.results,
120                 password: hash
121               });
122             citizen.save()
123               .then(result => {
124                 console.log(result);
125                 res.status(201).json({
126                   message: "Sign up Sucessfully",
127                   createdUser: result
128                 });
129               });
130             }
131           }
132         }
133       }
134     );
135   }
136   }
137 }
138 }
139 }
140 }
141 );
```



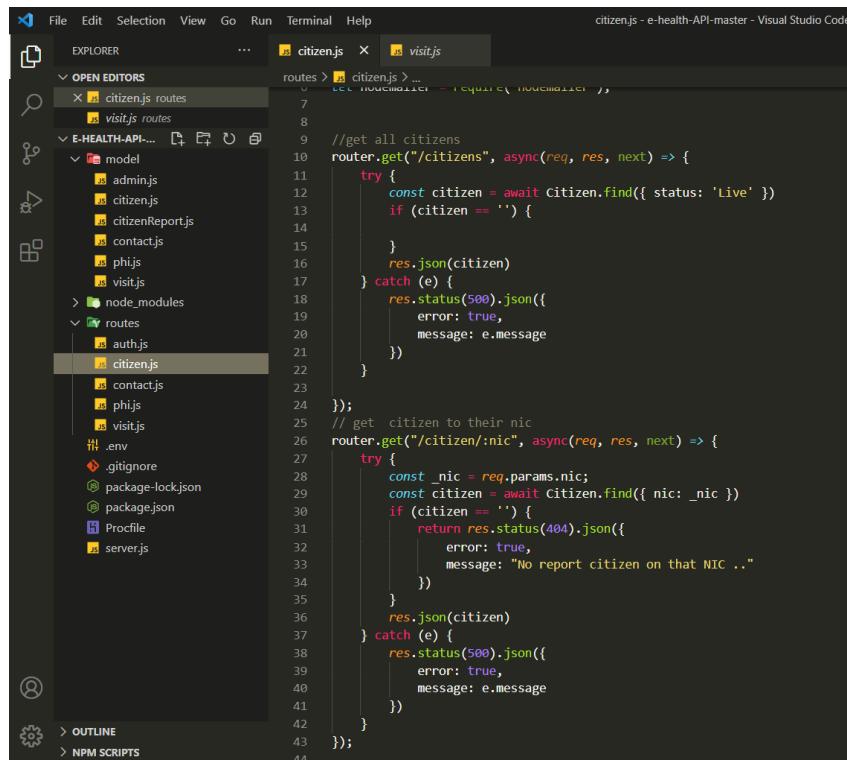
```
File Edit Selection View Go Run Terminal Help
EXPLORER routes > auth.js > ...
OPEN EDITORS citizen.js routes
E-HEALTH-API... routes > auth.js > ...
model
  admin.js
  citizen.js
  citizenReport.js
  contact.js
  phi.js
  visit.js
node_modules
routes
  auth.js
  citizen.js
  contact.js
  phi.js
  visit.js
.env
.gitignore
package-lock.json
package.json
Procfile
server.js
auth.js
routes > auth.js > ...
104 } else {
105   const citizen = new Citizen({
106     _id: new mongoose.Types.ObjectId(),
107     fname: req.body.fname,
108     lname: req.body.lname,
109     nic: req.body.nic,
110     age: req.body.age,
111     email: req.body.email,
112     phone: req.body.phone,
113     address: req.body.address,
114     location: req.body.location,
115     affiliation: req.body.affiliation,
116     profesion: req.body.profesion,
117     image: req.body.image,
118     date: req.body.date,
119     test: req.body.test,
120     results: req.body.results,
121     password: hash
122   });
123   citizen.save()
124     .then(result => {
125       console.log(result);
126       res.status(201).json({
127         message: "Sign up Sucessfully",
128         createdUser: result
129       });
130     });
131   }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
```

```
citizen.js auth.js
routes > auth.js > ...
13
14 //phi signup route
15 router.post('/phi/signup', (req, res, next) => {
16   Phi.find({ email: req.body.email })
17     .exec()
18     .then(phi => { //check if the email is allready taken
19       if (phi.length >= 1) {
20         return res.status(409).json({
21           message: 'Email Not Available.'
22         });
23     } else {
24       bcrypt.hash(req.body.password, 10, (err, hash) => { //hash the password using bcrypt
25         if (err) {
26           return res.status(500).json({
27             error: err
28           });
29         } else {
30           const phi = new Phi({
31             _id: new mongoose.Types.ObjectId(),
32             fname: req.body.fname,
33             lname: req.body.lname,
34             nic: req.body.nic,
35             email: req.body.email,
36             phone: req.body.phone,
37             address: req.body.address,
38             location: req.body.location,
39             image: req.body.image,
40             password: hash
41           });
42           phi.save()
43             .then(result => {
44               console.log(result);
45               res.status(201).json({
46                 message: "Sign up Sucessfully",
47                 createdUser: result
48               });
49             });
50           });
51         });
52       });
53     });
54   });
55   });
56   });
57   });
58   });
59   });
60   });
61   });
});
```

```
citizen.js auth.js
routes > auth.js > ...
29
30 } else {
31   const phi = new Phi({
32     _id: new mongoose.Types.ObjectId(),
33     fname: req.body.fname,
34     lname: req.body.lname,
35     nic: req.body.nic,
36     email: req.body.email,
37     phone: req.body.phone,
38     address: req.body.address,
39     location: req.body.location,
40     image: req.body.image,
41     password: hash
42   });
43   phi.save()
44     .then(result => {
45       console.log(result);
46       res.status(201).json({
47         message: "Sign up Sucessfully",
48         createdUser: result
49       });
50     });
51   });
52   .catch(err => {
53     console.log(err);
54     res.status(500).json({
55       error: err
56     });
57   });
58   });
59   });
60   });
61   });
});
```

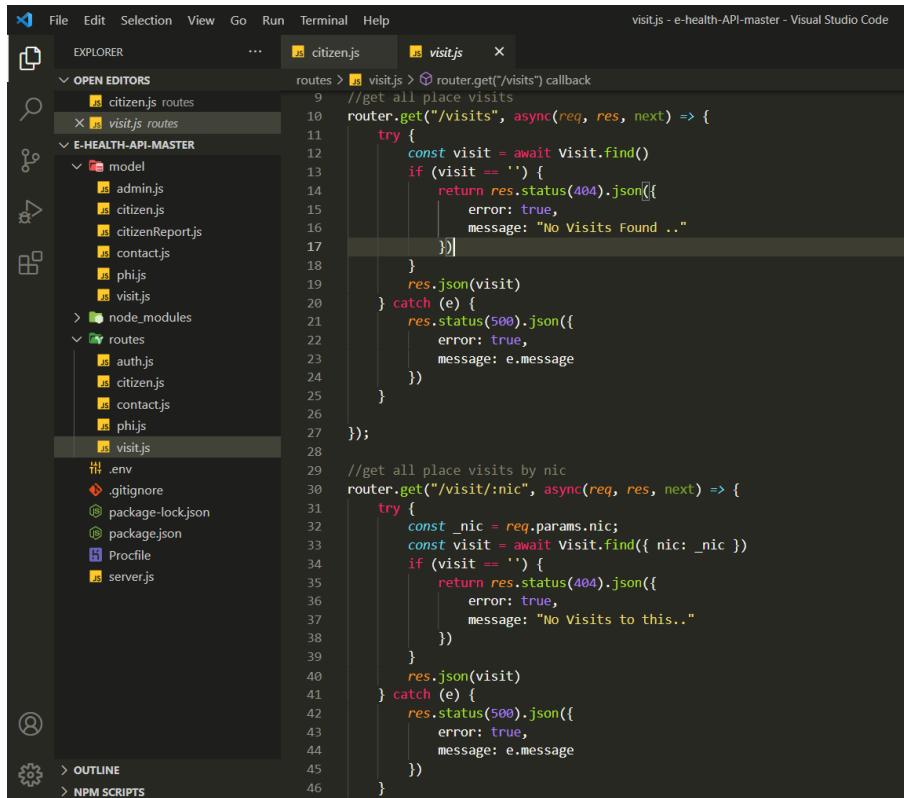
2.GET (CDC or PHI can collect details of any citizen)

CDC



```
citizen.js - e-health-API-master - Visual Studio Code
File Edit Selection View Go Run Terminal Help
citizen.js routes
routes > citizen.js > ...
  7
  8
  9 //get all citizens
 10 router.get("/citizens", async(req, res, next) => {
 11   try {
 12     const citizen = await Citizen.find({ status: 'Live' })
 13     if (citizen == '') {
 14       }
 15       res.json(citizen)
 16     } catch (e) {
 17       res.status(500).json({
 18         error: true,
 19         message: e.message
 20       })
 21     }
 22   });
 23
 24   // get citizen to their nic
 25   router.get("/citizen/:nic", async(req, res, next) => {
 26     try {
 27       const _nic = req.params.nic;
 28       const citizen = await Citizen.find({ nic: _nic })
 29       if (citizen == '') {
 30         return res.status(404).json({
 31           error: true,
 32           message: "No report citizen on that NIC .."
 33         })
 34       }
 35       res.json(citizen)
 36     } catch (e) {
 37       res.status(500).json({
 38         error: true,
 39         message: e.message
 40       })
 41     }
 42   });
 43 });
 44
 45
 46
```

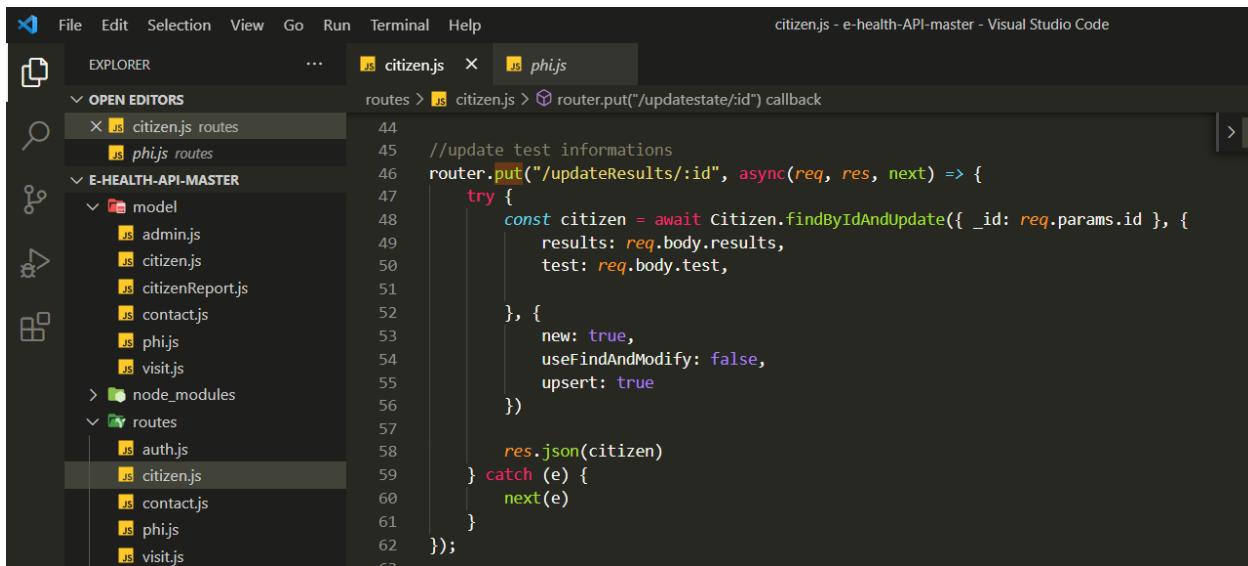
PHI



```
visit.js - e-health-API-master - Visual Studio Code
File Edit Selection View Go Run Terminal Help
citizen.js routes
routes > visit.js > ...
  9 //get all place visits
 10 router.get("/visits", async(req, res, next) => {
 11   try {
 12     const visit = await Visit.find()
 13     if (visit == '') {
 14       return res.status(404).json([
 15         {
 16           error: true,
 17           message: "No Visits Found .."
 18         }
 19       ])
 20     }
 21     res.json(visit)
 22   } catch (e) {
 23     res.status(500).json({
 24       error: true,
 25       message: e.message
 26     })
 27   });
 28
 29 //get all place visits by nic
 30 router.get("/visit/:nic", async(req, res, next) => {
 31   try {
 32     const _nic = req.params.nic;
 33     const visit = await Visit.find({ nic: _nic })
 34     if (visit == '') {
 35       return res.status(404).json({
 36         error: true,
 37         message: "No Visits to this.."
 38       })
 39     }
 40     res.json(visit)
 41   } catch (e) {
 42     res.status(500).json({
 43       error: true,
 44       message: e.message
 45     })
 46   }
 47 }
```

3. PUT

PHI should update clients health status

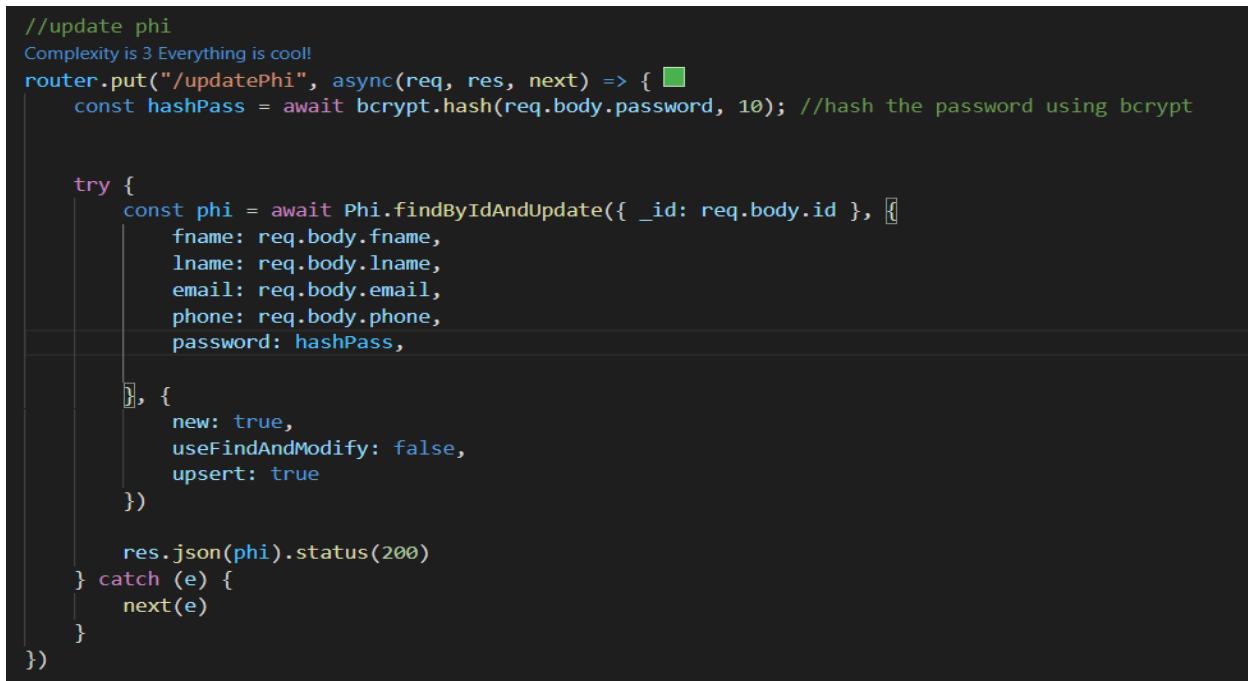


The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "E-HEALTH-API-MASTER".
 - routes:** citizen.js (selected), auth.js, contact.js, phi.js, visit.js.
 - model:** admin.js, citizen.js, citizenReport.js, contact.js, phi.js, visit.js.
 - node_modules:** routes.
- Editor:** Displays the code for `citizen.js`. The current line is highlighted at line 62, which contains `});`.
- Status Bar:** Shows "citizen.js - e-health-API-master - Visual Studio Code".

```
//update test informations
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)
  }
});
```

PHI can update profile details



The screenshot shows the Visual Studio Code interface with the following details:

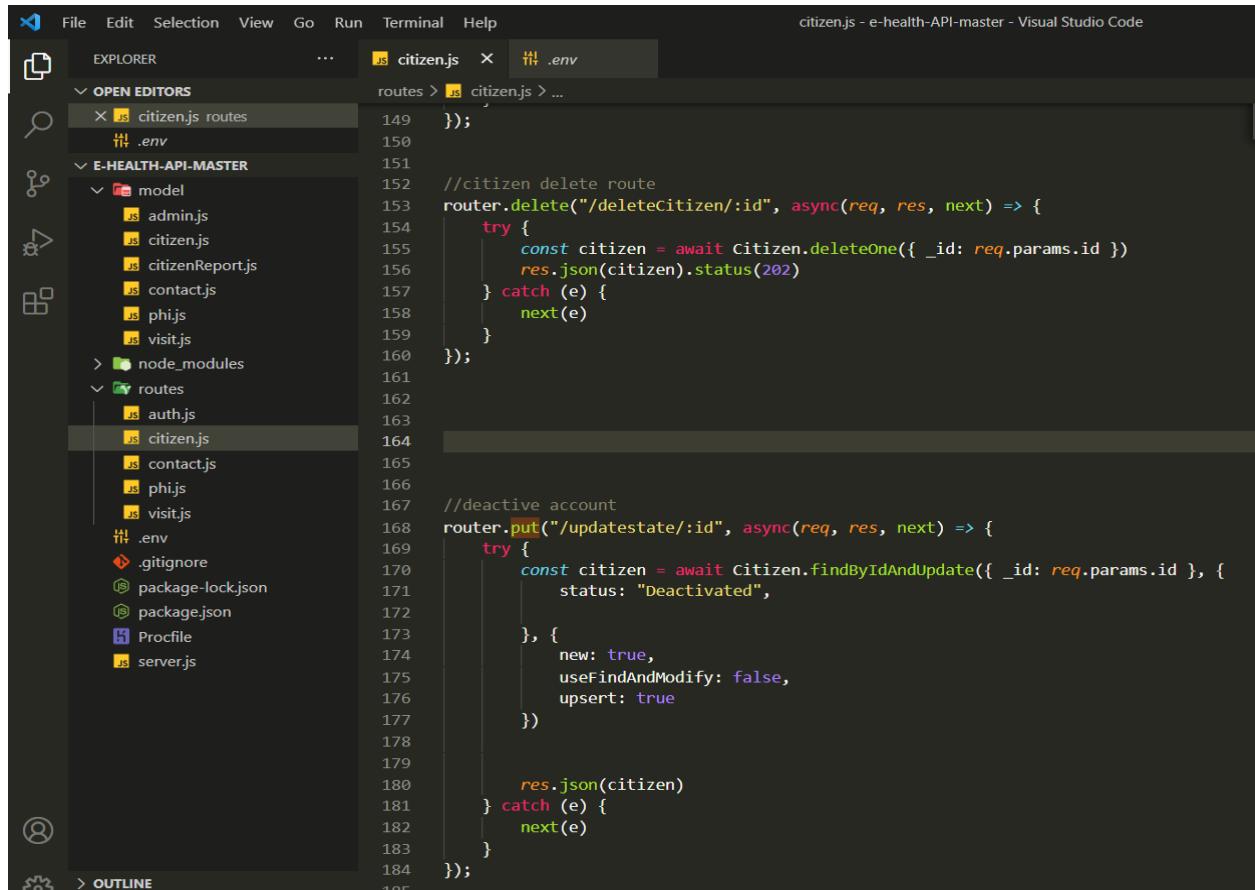
- File Explorer:** Shows the project structure under "E-HEALTH-API-MASTER".
 - routes:** auth.js, citizen.js (selected), contact.js, phi.js, visit.js.
- Editor:** Displays the code for `phi.js`. The current line is highlighted at line 62, which contains `});`.
- Status Bar:** Shows "phi.js - e-health-API-master - Visual Studio Code".

```
//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



The screenshot shows the Visual Studio Code interface with the following details:

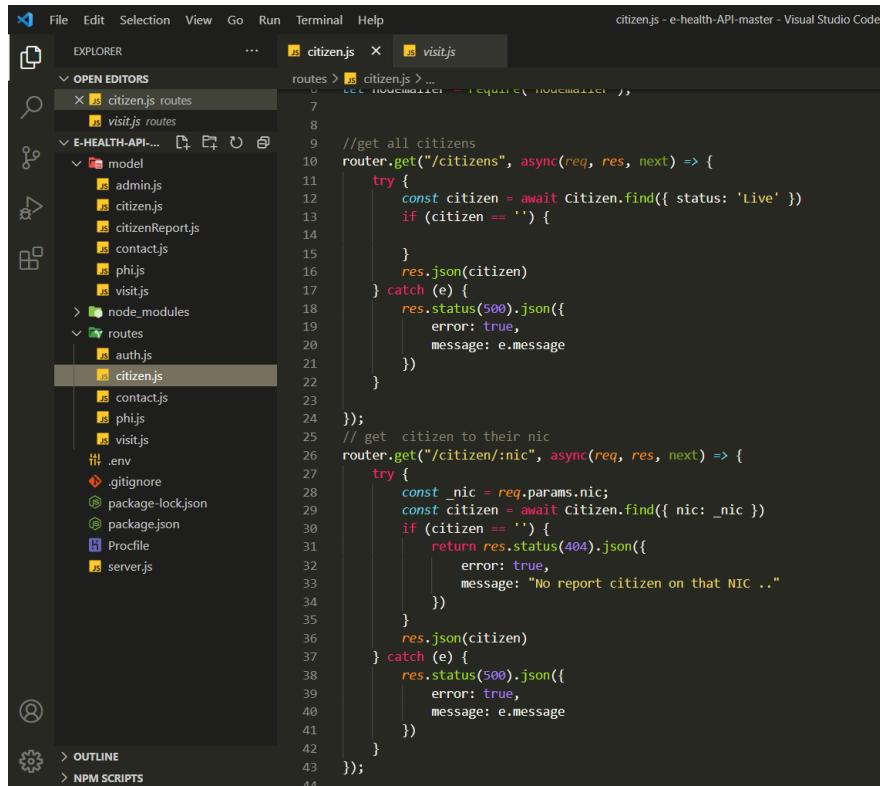
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** citizen.js - e-health-API-master - Visual Studio Code.
- Explorer:** Shows the project structure:
 - OPEN EDITORS: citizen.js, routes, .env
 - E-HEALTH-API-MASTER:
 - model: admin.js, citizen.js, citizenReport.js, contact.js, phi.js, visit.js
 - routes: auth.js, citizen.js, contact.js, phi.js, visit.js, .env, .gitignore, package-lock.json, package.json, Procfile, server.js
- Editor:** The citizen.js file is open, showing code for handling DELETE and PUT requests. The code uses async/await syntax and imports from the Citizen model.

```
//citizen delete route
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
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)
  }
});
```

5. GET

CDC can get any citizens contacts informations.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "E-HEALTH-API...".
 - OPEN EDITORS:** citizen.js (highlighted)
 - routes:** citizen.routes, visit.routes
 - model:** admin.js, citizen.js, citizenReport.js, contact.js, phi.js, visit.js
 - node_modules**
 - routes:** auth.js, citizen.js (highlighted), contact.js, phi.js, visit.js
 - env**
 - gitignore**
 - package-lock.json**
 - package.json**
 - Profile**
 - server.js**
- Editor:** Displays the content of citizen.js.

```
routes > citizen.js > ...
...
9 //get all citizens
10 router.get("/citizens", async(req, res, next) => {
11   try {
12     const citizen = await Citizen.find({ status: 'Live' })
13     if (citizen == '') {
14       }
15     res.json(citizen)
16   } catch (e) {
17     res.status(500).json({
18       error: true,
19       message: e.message
20     })
21   }
22 });
23 // get citizen to their nic
24 router.get("/citizen/:nic", async(req, res, next) => {
25   try {
26     const _nic = req.params.nic;
27     const citizen = await Citizen.find({ nic: _nic })
28     if (citizen == '') {
29       return res.status(404).json({
30         error: true,
31         message: "No report citizen on that NIC .."
32       })
33     }
34     res.json(citizen)
35   } catch (e) {
36     res.status(500).json({
37       error: true,
38       message: e.message
39     })
40   }
41 });
42 });
43 });
44 );
```

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 programming 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 web access.

Body-parser

This body-parser object exposes various kinds of factories to create middleware. This is used to read HTTP POST data, and this reads 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 MongoDB 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 user friendly and also to emphasize simplicity and usability through the internet. This language is widely used for the representation of data structures.

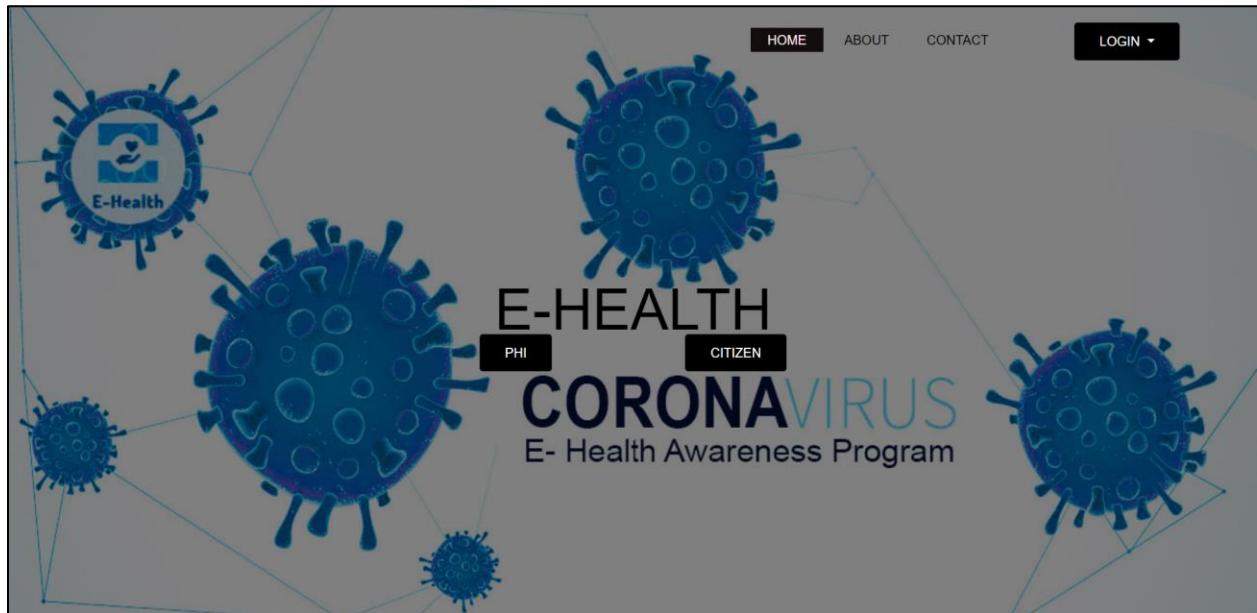
4. Client Application

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.

The screenshot shows the 'About Us' page of the E-Health Management System. At the top, there is a navigation bar with 'HOME' and 'About Us' buttons. The main header features the word 'E-HEALTH' in large, bold, white letters against a dark, futuristic background with various icons like graphs, arrows, and hexagons. Below the header, the text 'The Complete E-HEALTH Management System' is displayed in blue. A detailed description follows, explaining the system's purpose: 'E - Health, this is a system which is developed on behalf of the citizens and the country based on rescuing the people from covid-19 pandemic. This system updates the locations of the people where they are by scanning the QR codes on each and every location where they go. This system is productive and very easy to use for the people and get aware of the infected patients. This system have access for to main sectors of the society named as CITIZEN and PHI. First of all the citizen must register to the system by providing correct information. Once the PHI get the medical reports of the relevant person, it is updated to the system. So the relevant person and the PHI can be aware of the positive patients and take the right actions to stop the spread of the disease.' At the bottom of the page, there is a footer with 'E-Pharmacy' and 'Copyright © E-Pharmacy'.

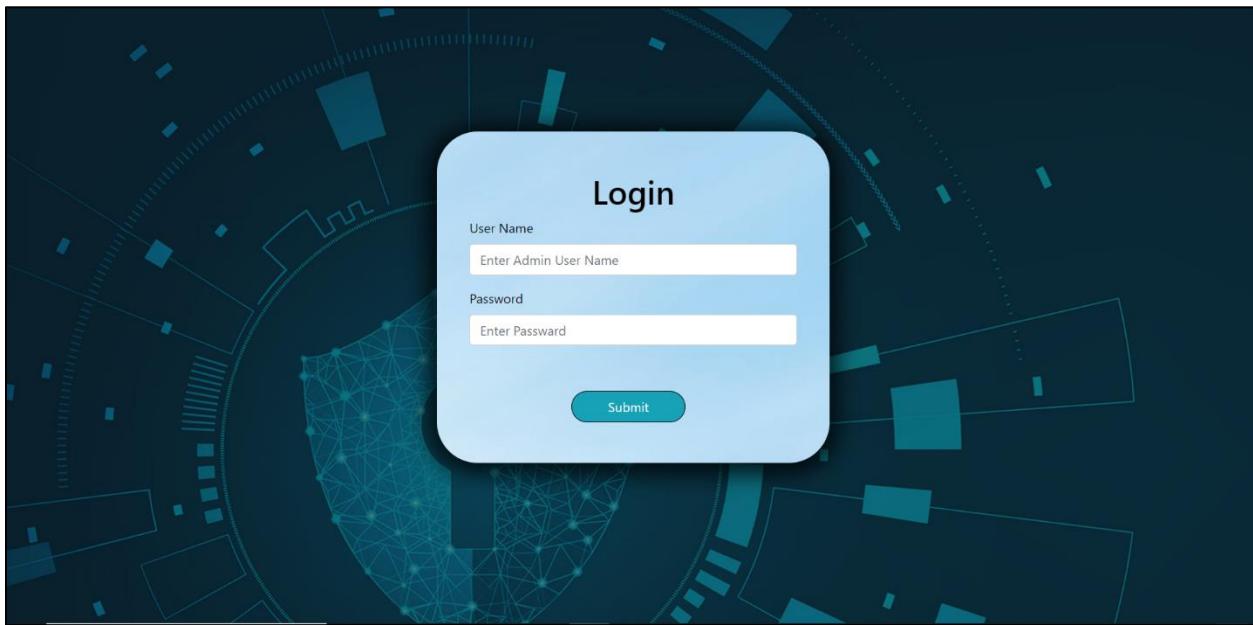
Contact page

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

The screenshot shows the 'CONTACT' page of the E-Health Management System. The top navigation bar includes 'HOME' and 'CONTACT' buttons. The page features a contact form with fields for 'Your Name' (with placeholder 'Enter your name'), 'E-mail' (with placeholder 'Enter your E-mail'), and 'Message' (with placeholder 'Enter your Message'). To the left of the form, there is a message: 'If you have Questions or just want to get in touch, use the form below. We look forward to hearing from you!' and contact details: 'Contact No:076-4720883' and 'E-mail: ehealth1507@gmail.com'. On the right side of the form, there are icons for a green cross (Medical), an '@' symbol (Email), an envelope (Email), and a telephone receiver (Phone). Below the form, there is a footer with 'E-Health' and 'Copyright © E-Health'.

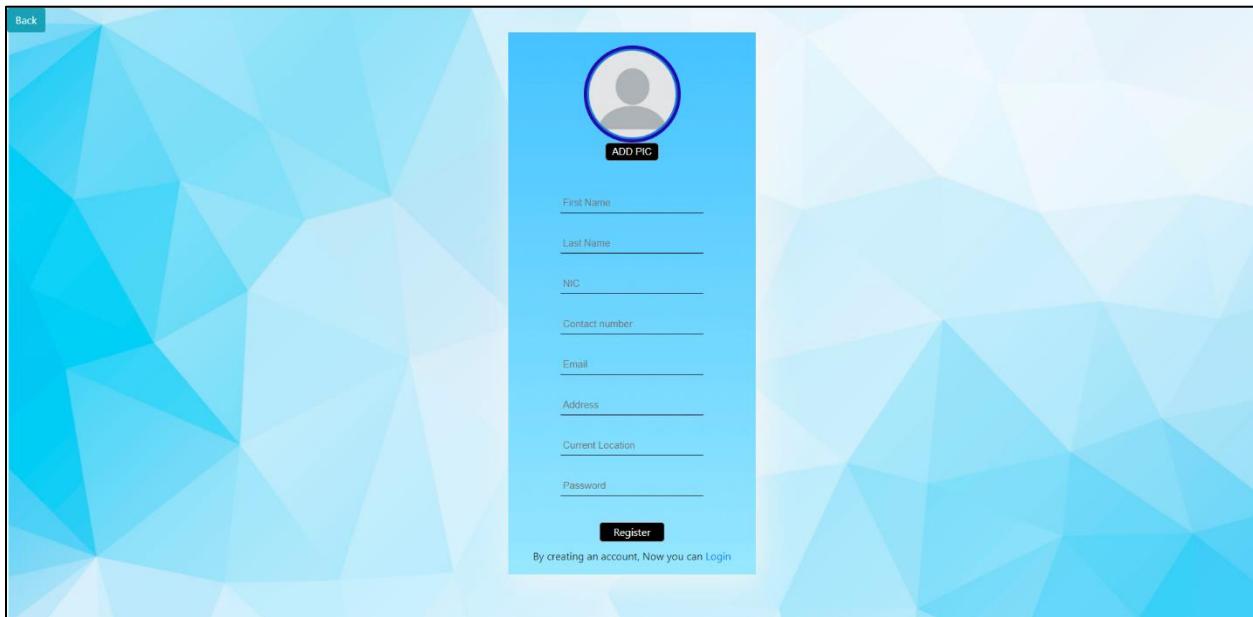
Login page

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



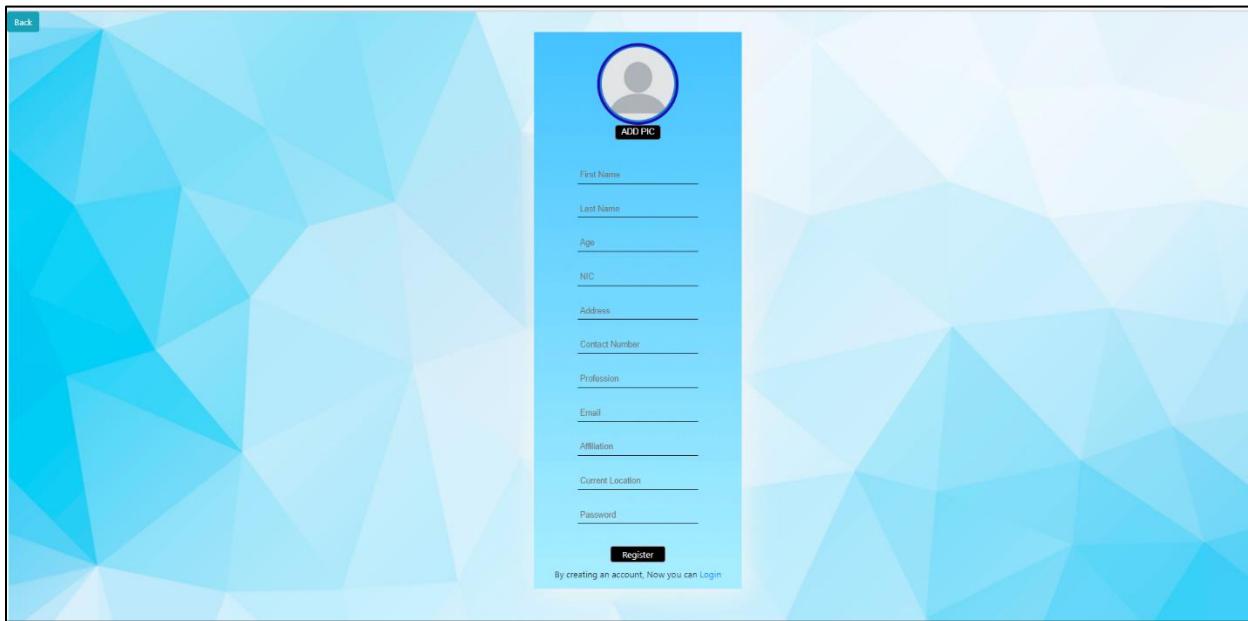
PHI register page

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

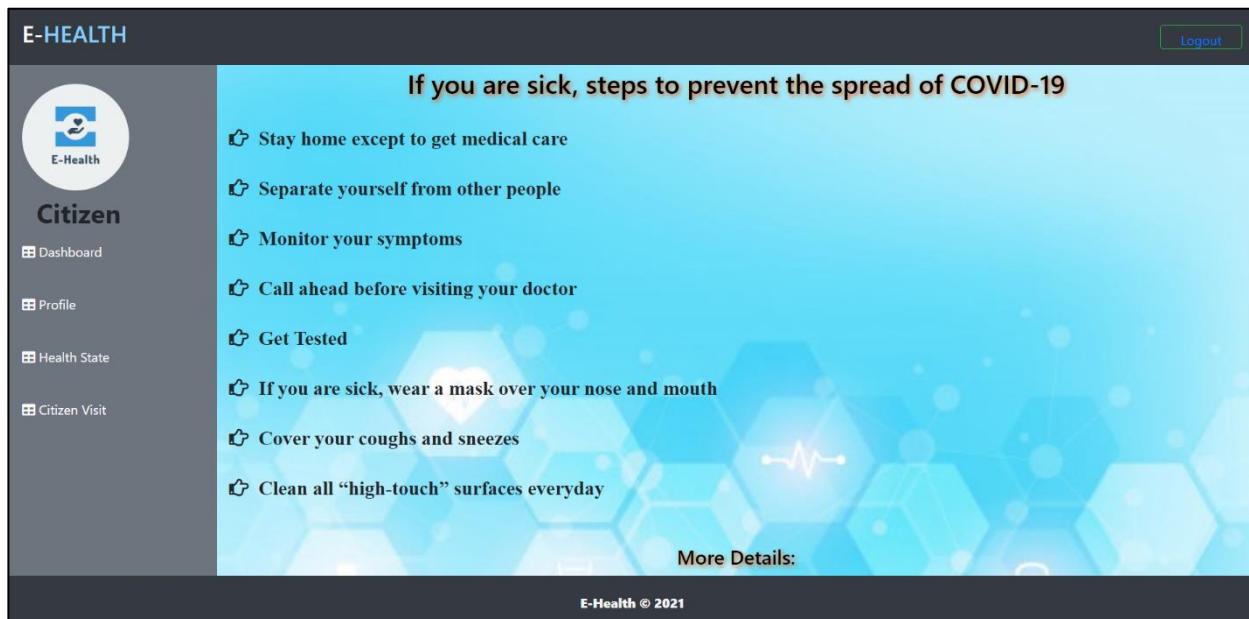
Here can have option to register citizen those who would like to join our system. So they have to add their details such as, first name, last name, email, NIC, address, contact no, profession, email, affiliation, current location and password.



The image shows a registration form titled "Back". It features a large blue polygonal background. On the right side, there is a circular placeholder for a profile picture with the text "ADD PIC" below it. Below the placeholder are several input fields: "First Name", "Last Name", "Age", "NIC", "Address", "Contact Number", "Profession", "Email", "Affiliation", "Current Location", and "Password". At the bottom of the form is a "Register" button and a note: "By creating an account, Now you can Login".

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.



The image shows the citizen dashboard. The top navigation bar includes "E-HEALTH" on the left and "Logout" on the right. The main content area has a title "If you are sick, steps to prevent the spread of COVID-19" followed by a list of nine steps with icons: Stay home except to get medical care, Separate yourself from other people, Monitor your symptoms, Call ahead before visiting your doctor, Get Tested, If you are sick, wear a mask over your nose and mouth, Cover your coughs and sneezes, and Clean all "high-touch" surfaces everyday. To the right of the list is a small heart rate monitor icon. At the bottom of the content area is a "More Details:" link. The left sidebar, titled "Citizen", contains links for "Dashboard", "Profile", "Health State", 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.

E-HEALTH

Citizen

Logout

First Name _____

Last Name _____

Age _____

Contact Number _____

Address _____

Profession _____

Email _____

Affiliation _____

Current Location _____

Edit Upload

Make sure the information you entered is correct!

E-Health © 2021

Citizen health state page

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

E-HEALTH

Citizen

Logout

Show 10 entries

Search:

Date Test Result

2021/02/02	PCR	Negative
2021/02/02	Antigen	Negative
2021/02/02	PCR	Negative
2021/02/03	PCR	Negative

Showing 1 to 4 of 4 entries

Previous 1 Next

E-Health © 2021

Citizen visit page

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

The screenshot shows the 'Citizen Visit Places Details' section of the E-Health application. On the left, there's a sidebar with a logo and navigation links for Dashboard, Profile, Health State, and Citizen Visit. The main area has a title 'Citizen Visit Places Details' and a search bar. It displays a table with three entries showing visit dates, times, and locations. The table has columns for Visit Date, Time, and Place. The data is as follows:

Visit Date	Time	Place
2021/02/01	14:00	Colombo shop
2021/02/01	14:00	Colombo shop
2021/02/01	14:00	Colombo shop

Below the table, it says 'Showing 1 to 3 of 3 entries'. There are navigation buttons for Previous, Next, and a page number '1'. The footer of the page includes the copyright notice 'E-Health © 2021'.

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.

The screenshot shows the 'COVID-19 Prevention Methods' section of the E-Health application. On the left, there's a sidebar with a logo and navigation links for Dashboard, Health Status, Citizen Visit, and Settings. The main area has a title 'COVID-19 Prevention Methods' and a list of prevention methods with icons. The methods are:

- Always wear a clean face mask when you leave from home.
- Wash your hand often and use soap & water, or an alcohol-based hand rub.
- Don't touch your eyes, nose or mouth.
- Cover your cough and sneeze using a elbow or tissue and dispose tissue properly
- Avoid Shaking hands or hugging your friends and relatives.
- You should keep at least one meter distance between yourself and others, if in public or while you are talking.
- You should monitor your health daily.

Below the list, there's a 'More Details:' section with two purple links: 'How to protect yourself & Others' and 'PHI Duty List Manual'. The footer of the page includes the copyright notice 'E-Health © 2021'.

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.

The screenshot shows a table titled "Health Status" with 14 columns. The columns are: NIC, First Name, Last Name, Age, Address, Profession, Email, Affiliation, Current Location, Date, Test, Result, and Update Health Status. The table contains 10 entries. Each entry includes dropdown menus for "Test" and "Result". A "Update" button is located at the bottom right of each row. The table has a light blue background with alternating row colors. At the top left of the main content area, there is a sidebar with a logo and navigation links: Dashboard, Health Status, Citizen Visit, and Settings. At the bottom of the page, it says "E-Health © 2021".

PHI citizen visit page

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

The screenshot shows a table titled "Citizen Visit Places Details" with 8 columns. The columns are: NIC, First Name, Last Name, Address, Contact Number, Email, Date, and Visit Place. The table contains 14 entries. The "Visit Place" column includes some descriptive text like "Hospital, Gampaha" and "Arpico, Matara". A "Logout" button is located at the top right of the main content area. The table has a light blue background with alternating row colors. At the top left of the main content area, there is a sidebar with a logo and navigation links: Dashboard, Health Status, Citizen Visit, and Settings. At the bottom of the page, it says "E-Health © 2021".

PHI profile settings page

Here PHI can update their account information.

E-HEALTH

Logout

Update Your Profile Details

First Name: John Last Name: Smith

Contact Number: +94 71 236 4869 Email: john@gmail.com

Address: 1234, Main St, Negombo

Current Password: [redacted] New Password: [redacted]

Update

E-Health © 2021

PHI View (admin)

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

E-HEALTH

Logout

PHI Info

Show 10 entries

Search:

First Name	Last Name	NIC	Contact Number	Email	Address	CurrentLocation
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03

Showing 1 to 3 of 3 entries

Previous 1 Next

E-Health © 2021

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**

E-HEALTH

Logout



Admin

PHI View

Citizen View

Positive P

Contact Us

Citizen View Details

Show 10 entries

First Name	Last Name	NIC	Age	Address	Profession	Email	Affiliation	Current Location	Date	Test	Result	Deactivation
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03	Food city, Colombo	<button>Deactivate</button>				
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03	Food city, Colombo	<button>Deactivate</button>				
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03	Food city, Colombo	<button>Deactivate</button>				

Showing 1 to 3 of 3 entries

Previous 1 Next

E-Health © 2021

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.

E-HEALTH

Logout



Admin

PHI View

Citizen View

Positive P

Contact Us

Positive Patients

Show 10 entries

First Name	Last Name	NIC	Address	Email	Current Location	Health State	Email
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03	<button>Email</button>
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03	<button>Email</button>
199838502123	Jenny	Edinburgh	12, 2nd Lane, Kalutara	+94 77 456 1236	jenny@gmail.com	2021/01/03	<button>Email</button>

Showing 1 to 3 of 3 entries

Previous 1 Next

E-Health © 2021

Contact us page (admin)

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

E-HEALTH

 E-Health

Admin

- PHI View
- Citizen View
- Positive P
- Contact Us

Contatc Us Info

Show
10 entries

Search:

Name	Email	Message
199838502123	Jenny	Edinburgh
199838502123	Jenny	Edinburgh
199838502123	Jenny	Edinburgh

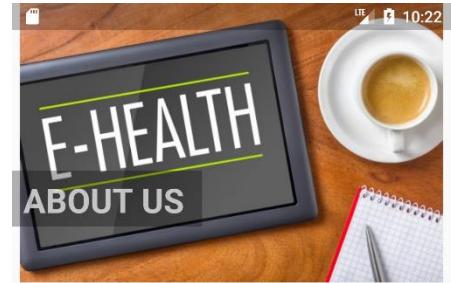
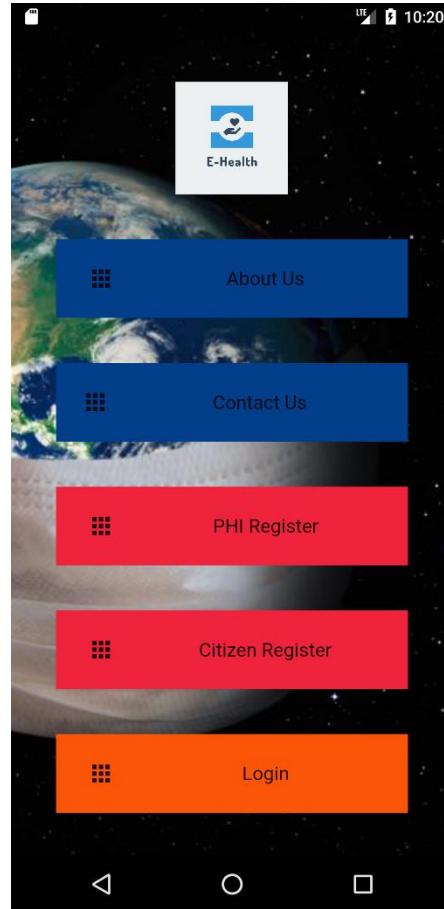
Showing 1 to 3 of 3 entries

Previous 1 Next

E-Health © 2021

b) Mobile application

- Mobile application was created using flutter framework.

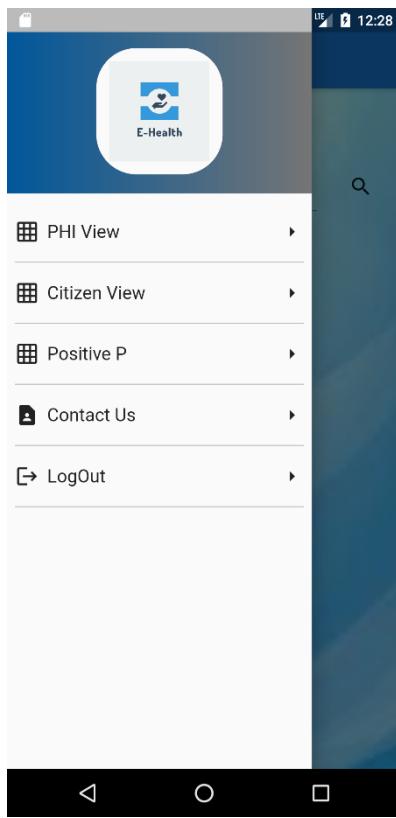


The Complete E-Health Management System

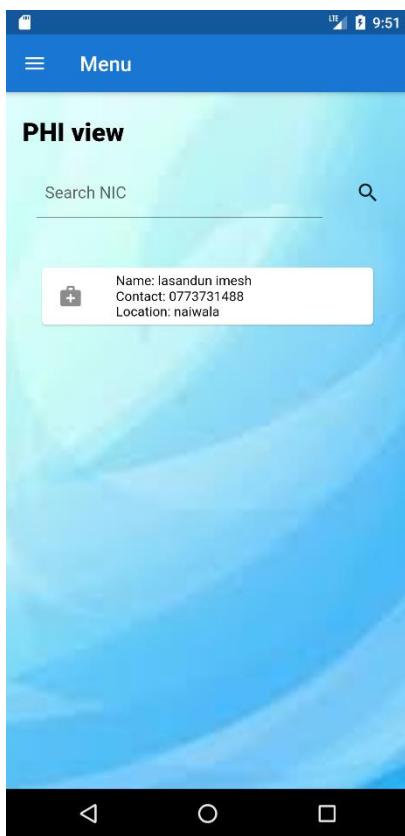
E - Health, this is a system which is developed on behalf of the citizens and the country based on rescuing the people from covid-19 pandemic. This system updates the locations of the people where they are by scanning the QR codes on each and every location where they go. This system is productive and very easy to use for the people and get aware of the infected patients. This system have access for to main sectors of the society named as CITIZEN and PHI. First of all the citizen must register to the system by providing correct information. Once the PHI get the medical reports of the relevant person, it is updated to the system. So the relevant person and the PHI can be aware of the positive patients and take the right actions to stop the spread of the disease.

- 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.
- 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.
- When enter the about us button, it will goes to the about us page. There are some info about our overall system.

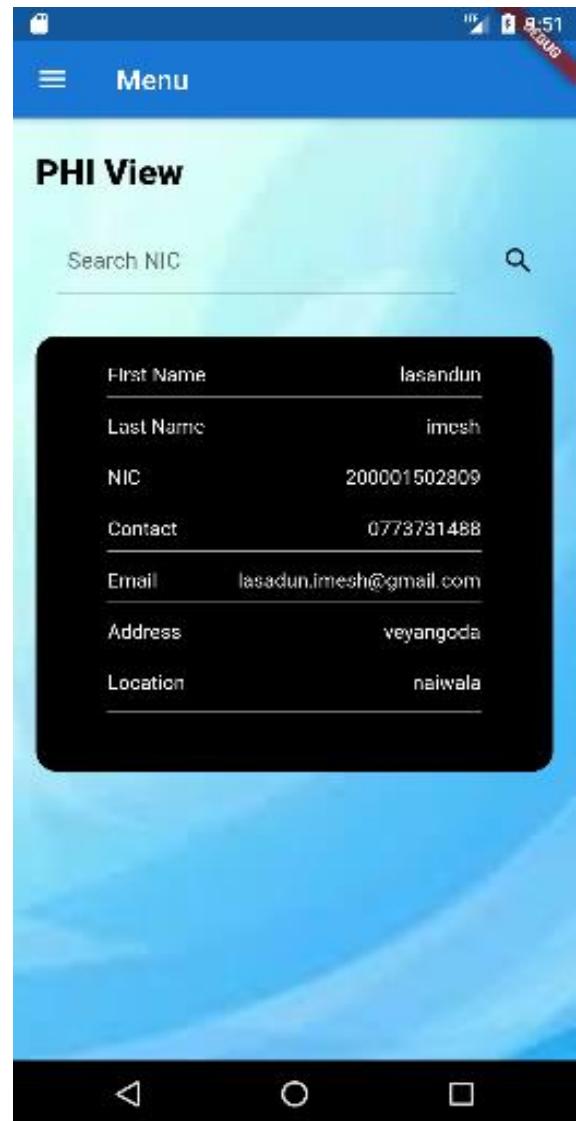
1. Admin part (CDC)

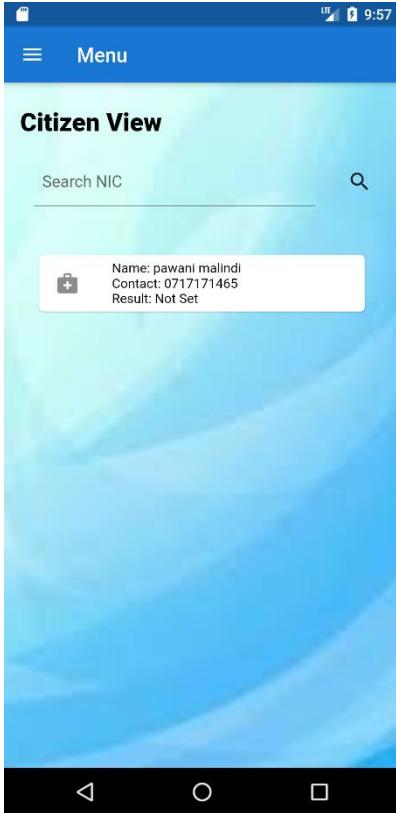


- This is the admin dashboard page. There are sub pages, PHI view, citizen view, positive p, contact us and there is the log out button.

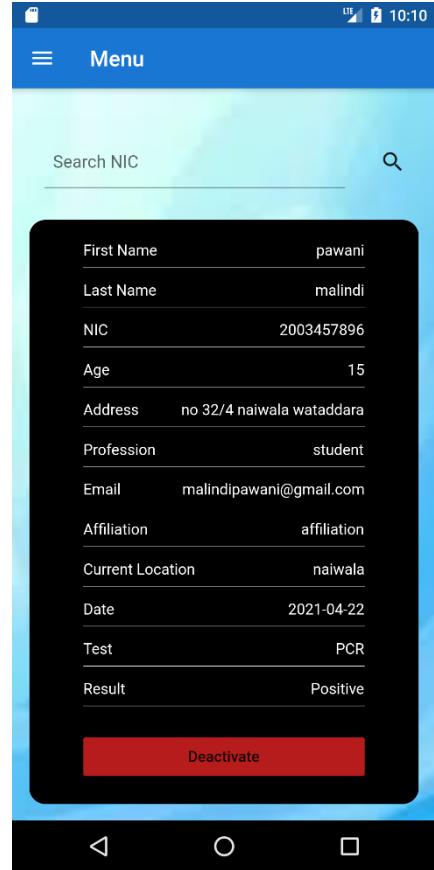


- This is the PHI view page.
Admin can view the PHI details that are connected with this system.

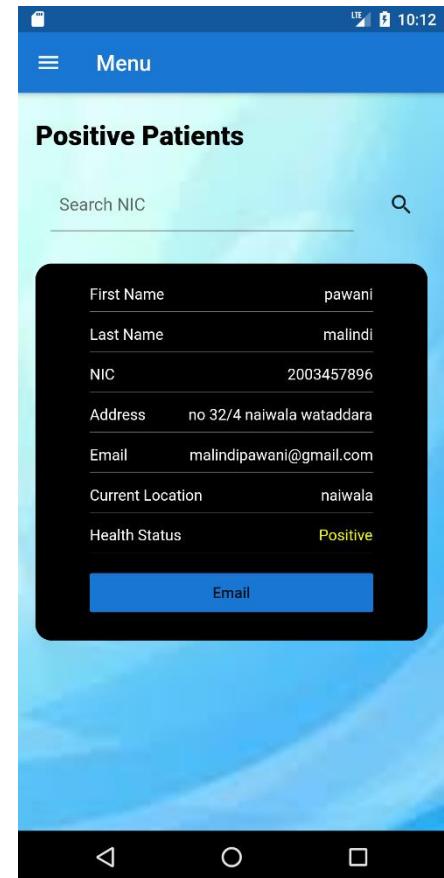




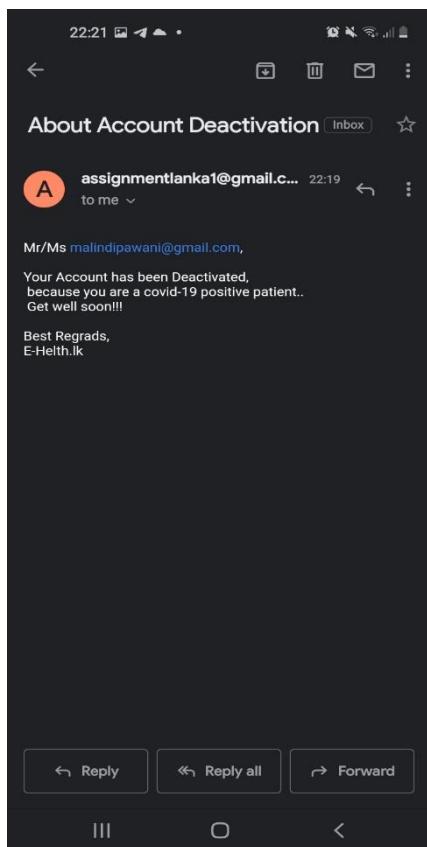
● This is the citizen view page.
Admin can view all the details of the citizens.
If the citizen is a covid 19 positive patient, admin can deactivate the citizens account.



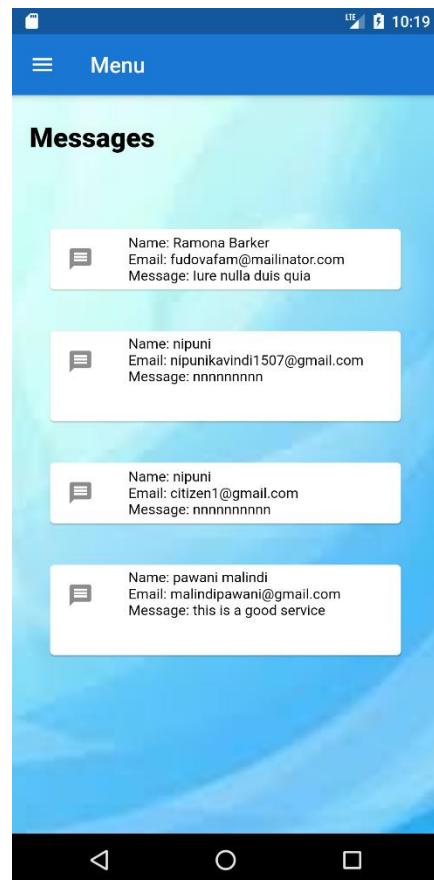
- After the deactivation admin can send an email to the positive patient.



- The email is like this.



- Admin can view the contact us messages.



2. Citizen part

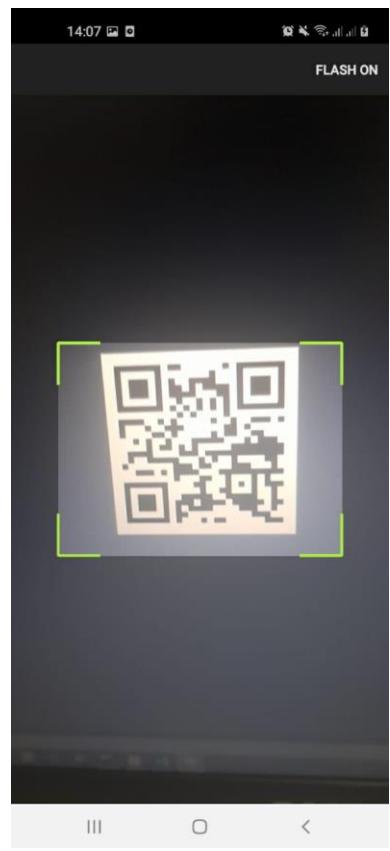


- This is the citizen dashboard

- This is the citizen registration part.



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

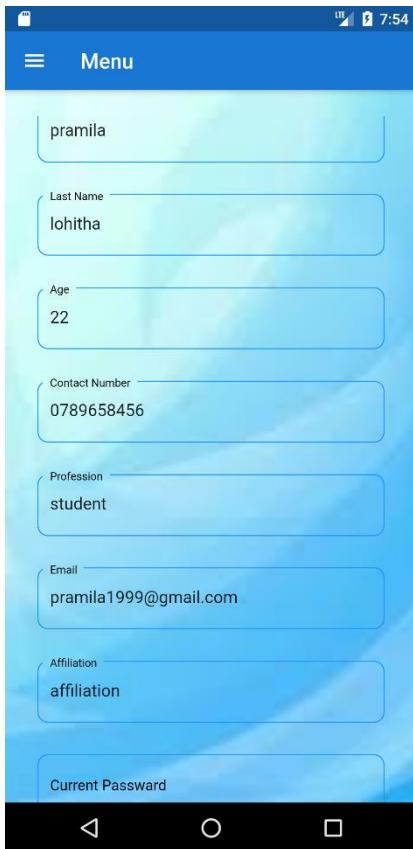


The data that scanning from the QR code



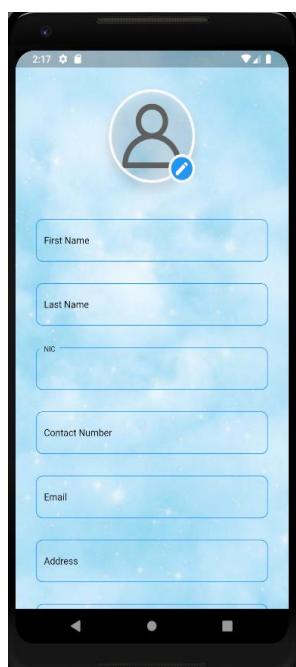
- The result of the PCR Or antigen that sending by The PHI.



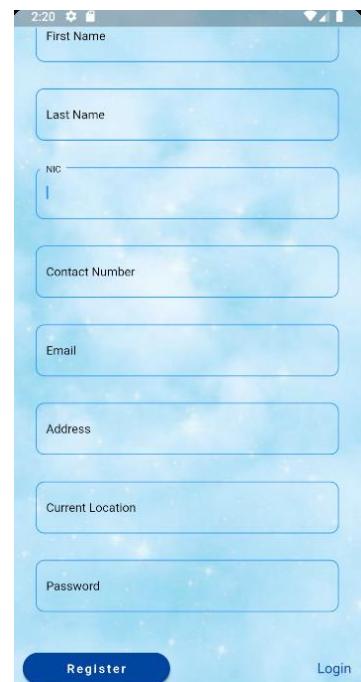


- This is the citizen edit profile page

3. PHI Part



- This is the PHI registration page





- This is the PHI's Dashboard.

Date	Visit Place
2021-04-14	maharagama bus stand
2021-04-19	Keels supper kottawa
2021-04-19	pettah station
2021-04-22	Keels supper kottawa
2021-04-22	viwaka hospital veiyangoda

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

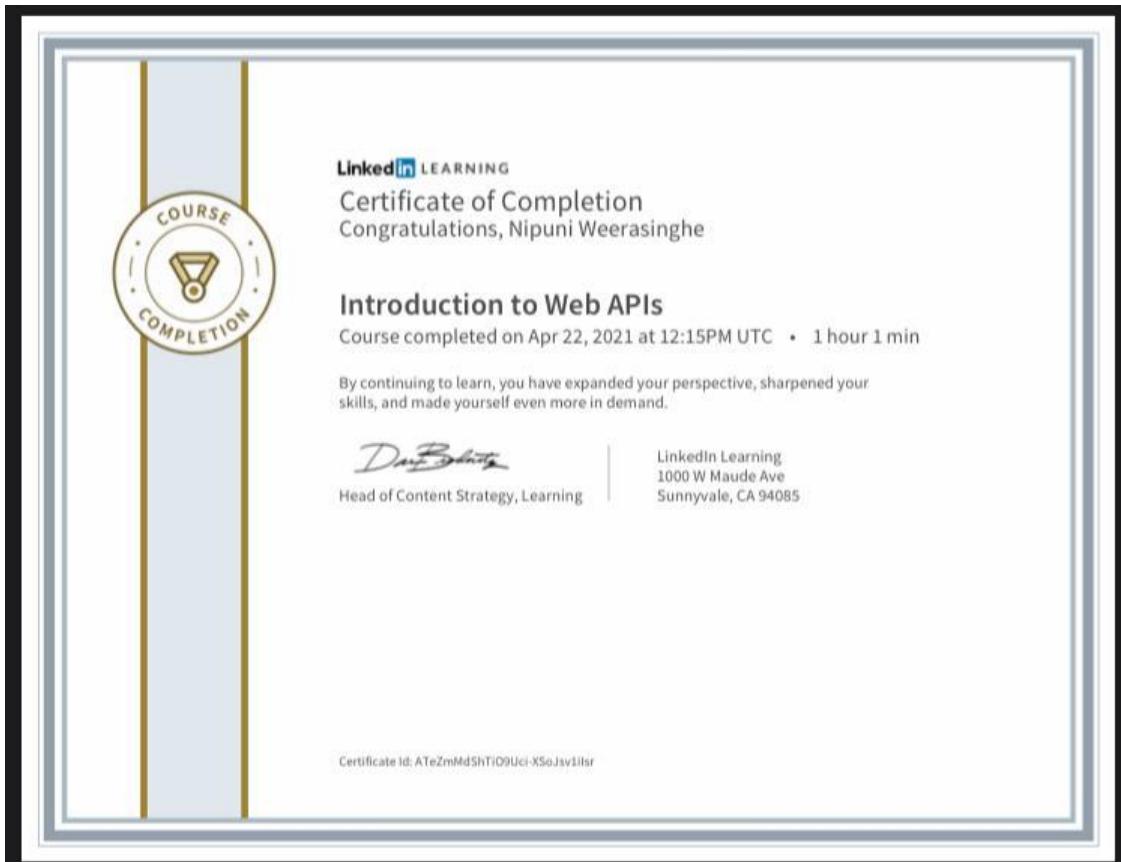


- This page is the PHI profile update page.

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)

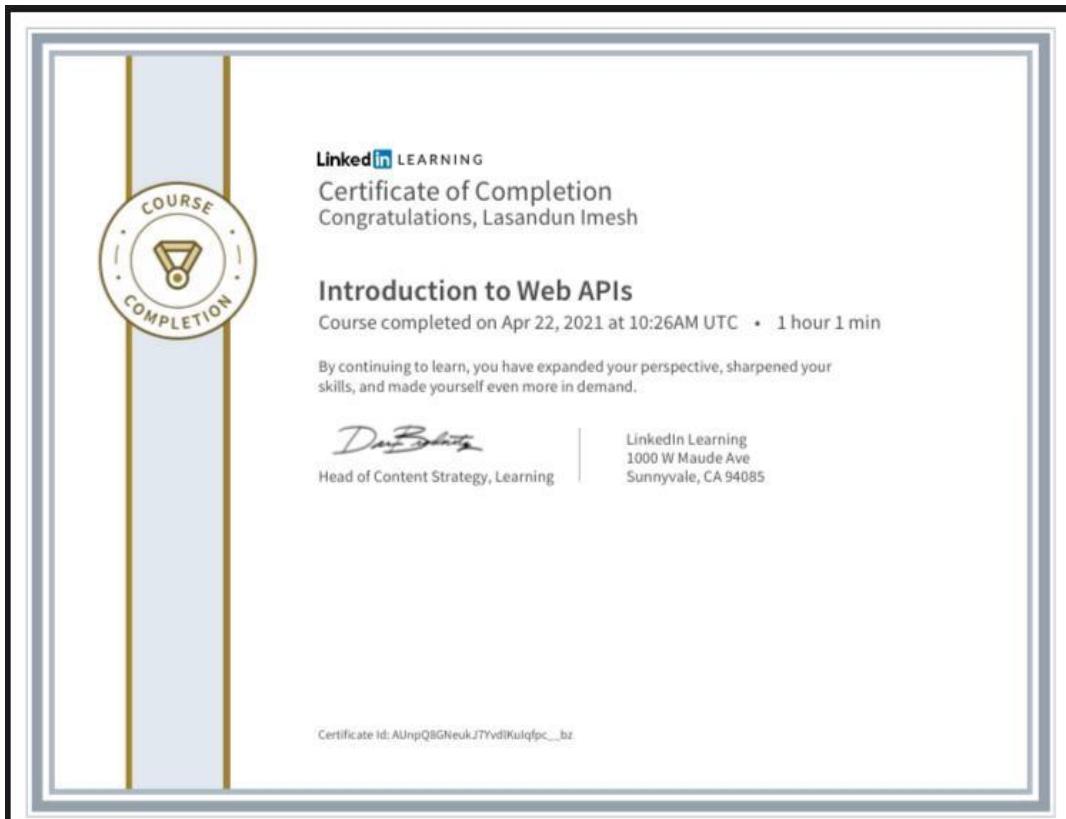
PHI Part

- 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 Karunananayake

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)

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 hon Software Engineering (Plymouth)

My Contribution

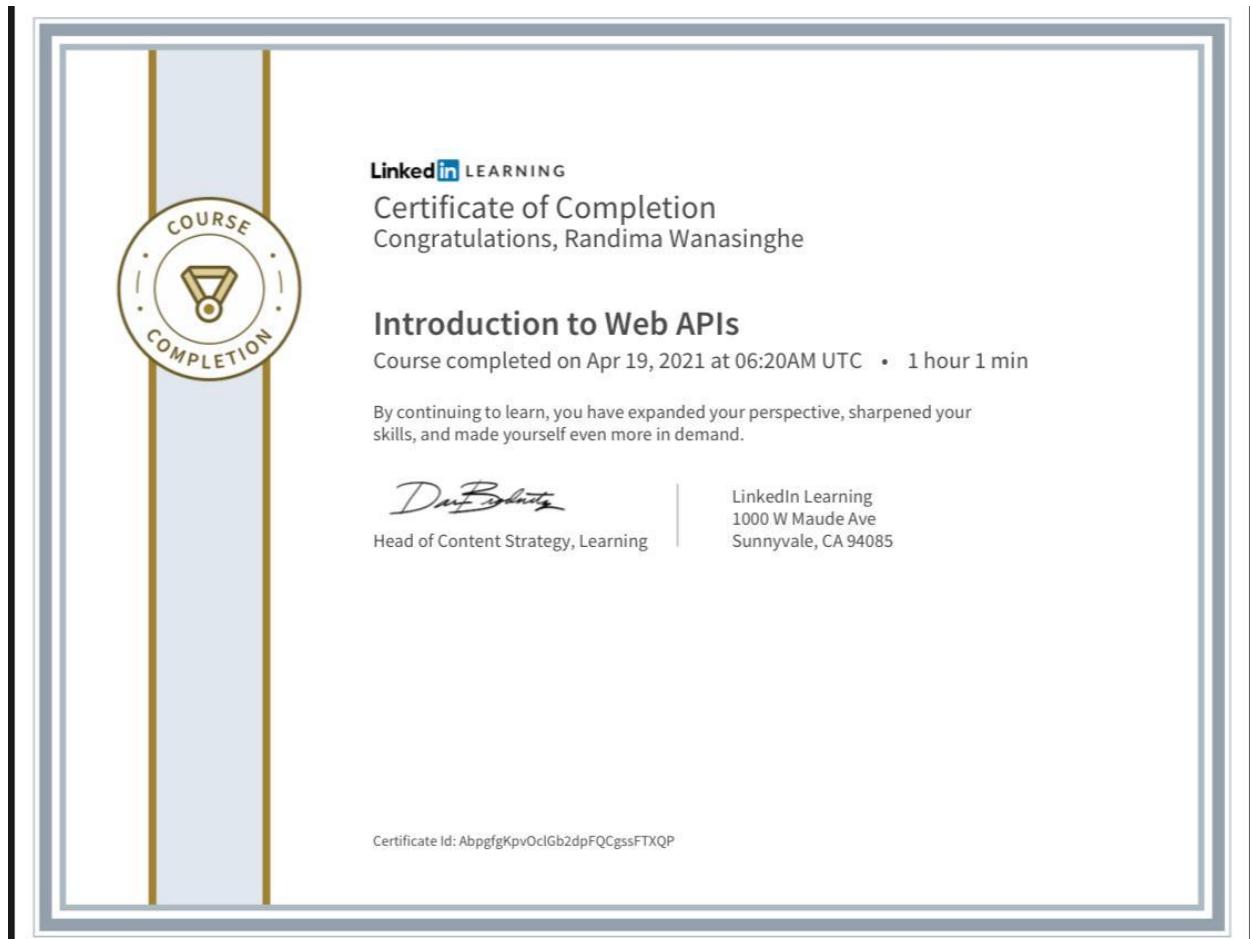
In this API software development project scenario given to create web application and mobile application for Center for Disease Control and Prevention (CDC) and with the ministry of health and Public Health Inspectors (PHI) to provide support for ongoing covid – 19 situations. This application is created 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

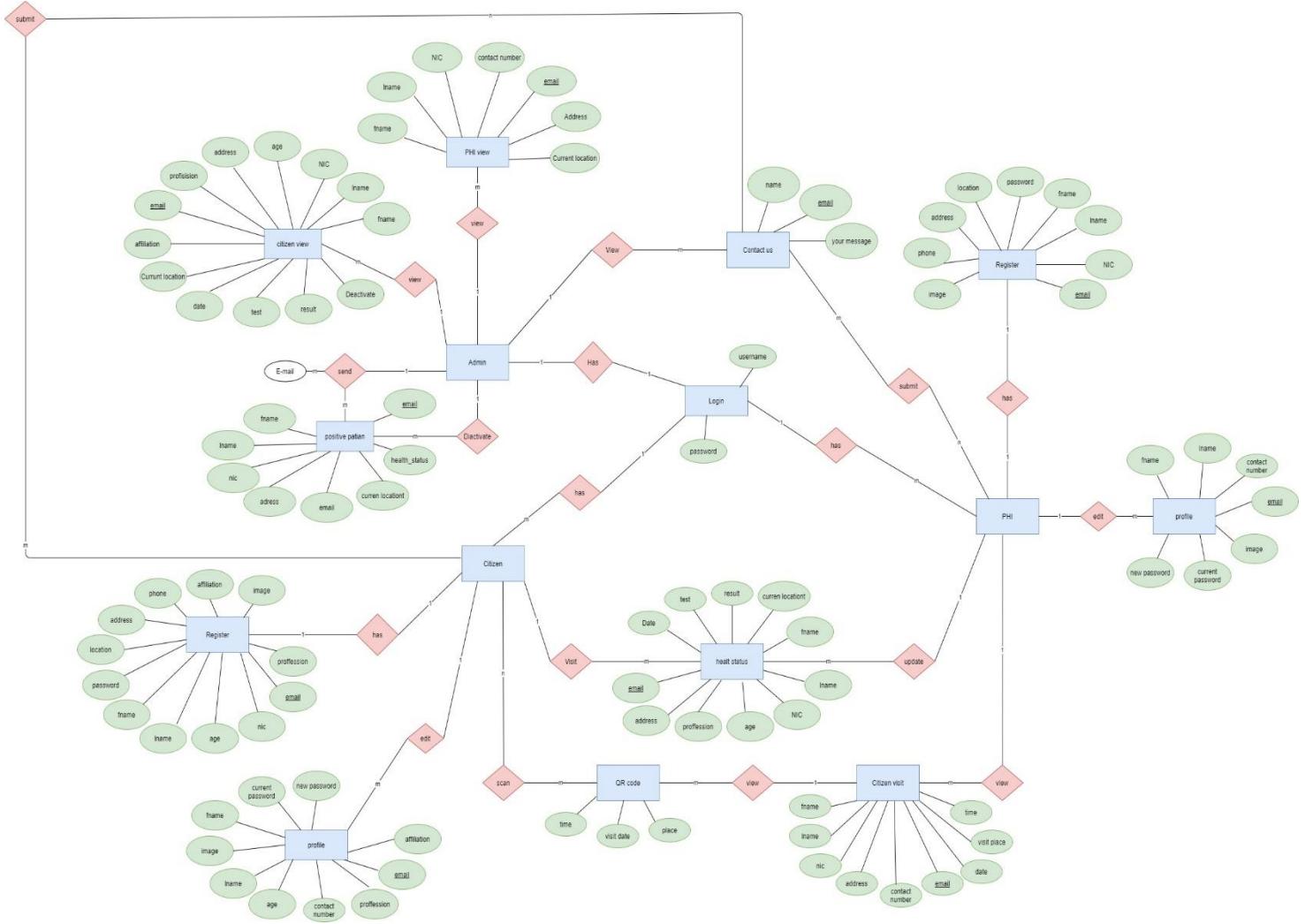


5. References

- 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,
                            affiliation: req.body.affiliation,
                            profesion: 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,
    affiliation: citizen.affilication,
    profesor: 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,
            affiliation: req.body.affiliation,
            profesor: req.body.profesor,
            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 reports 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 Your Account has been Deactivated, \n because you are a covid-19 positive patient..\\n Get well soon!!! \\n\\nBest Regards,\\nE-Health.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;
```

```
5 //get all Phis
Complexity is 5 Everything is cool!
7 router.get("/phis", async(req, res, next) => {
8   try {
9     const phi = await Phi.find()
10    if (phi == '') {
11      return res.status(404).json({
12        error: true,
13        message: "No Phis Found ."
14      })
15    }
16    res.json(phi)
17  } catch (e) {
18    res.status(500).json({
19      error: true,
20      message: e.message
21    })
22  }
23
24});
```

```

// 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;

```

```

9 //get all place visits
Complexity is 5 Everything is cool!
0 router.get("/visits", async(req, res, next) => {
1     try {
2         const visit = await Visit.find()
3         if (visit == '') {
4             return res.status(404).json({
5                 error: true,
6                 message: "No Visits Found .."
7             })
8         }
9         res.json(visit)
0     } catch (e) {
1         res.status(500).json({
2             error: true,
3             message: e.message
4         })
5     }
6
7 });
8

```

```
//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;
```

