Write a report that discusses the following:
1. Is this page an API?

- ✓ No.

  The web page does not conform to a set of rules that allow programs to talk to each other. For example, when we make a request to the URL, there is no data response

2. Does this page follow the RESTful API architecture?

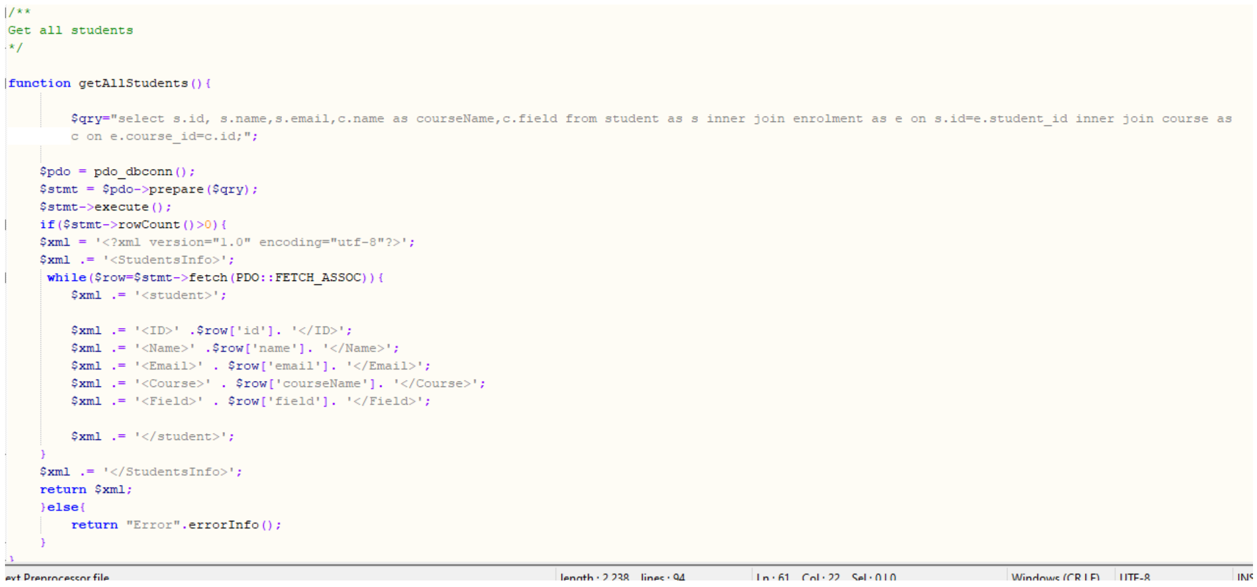
- ✓ No.

Explain in detail showing how exactly is it satisfying the RESTful architecture criteria (constraints + elements), and if not: state what needs to be changed in order to satisfy it, then make the necessary code amendments to turn it into a RESTful web service.

- ✓ A RESTful API contains the following architectural constraints: Uniform interface, Client–server, Stateless, Cacheable, Layered system and Code on demand (optional). Client-Server means that client application and server application MUST be able to evolve separately without any dependency on each other. Stateless means that the server will not store anything about the latest HTTP request the client made. It will treat every request as new. No session, no history. Caching brings performance improvement for the client-side and better scope for scalability for a server because the load has reduced.
- ✓ REST is based on the resource or noun instead of action or verb based. It means that a URI of a REST API should always end with a noun. Example: /api/students/ is a good example, but /api?id=1254 is a bad example of creating a REST API.

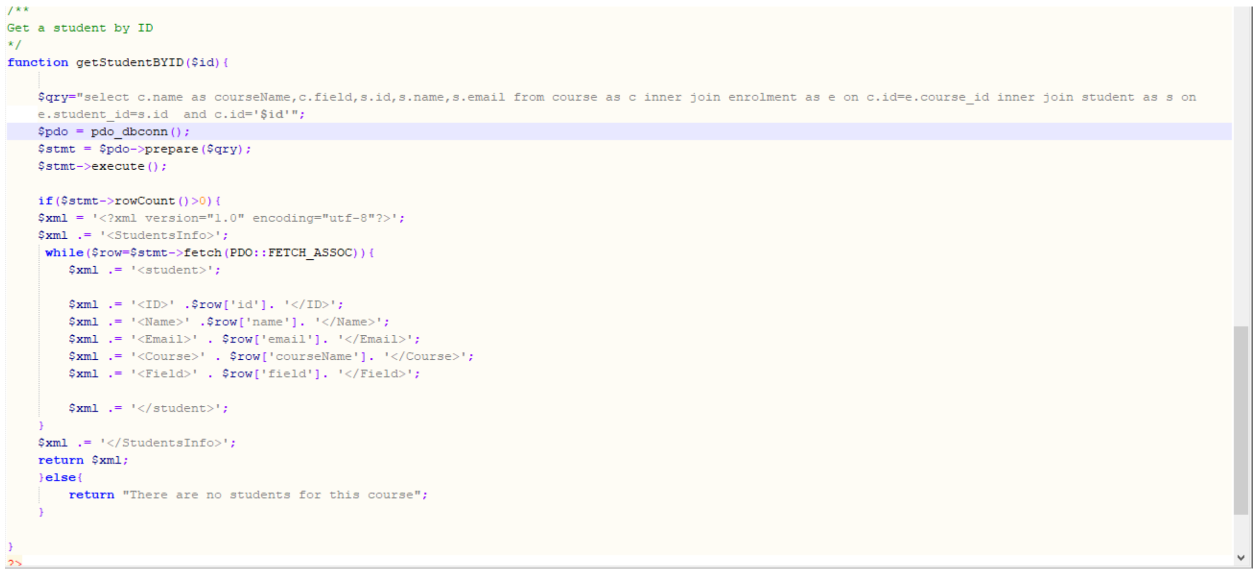
Changes:

The file has been changed to include 2 functions. These functions are referred to by the resources.



```
/**
Get all students
*/

function getAllStudents(){

        $qry="select s.id, s.name,s.email,c.name as courseName,c.field from student as s inner join enrolment as e on s.id=e.student_id inner join course as
        c on e.course_id=c.id;";

    $pdo = pdo_dbconn();
    $stmt = $pdo->prepare($qry);
    $stmt->execute();
    if($stmt->rowCount()>0){
    $xml = '<?xml version="1.0" encoding="utf-8"?>';
    $xml .= '<StudentsInfo>';
     while($row=$stmt->fetch(PDO::FETCH_ASSOC)){
        $xml .= '<student>';

        $xml .= '<ID>' .$row['id']. '</ID>';
        $xml .= '<Name>' .$row['name']. '</Name>';
        $xml .= '<Email>' . $row['email']. '</Email>';
        $xml .= '<Course>' . $row['courseName']. '</Course>';
        $xml .= '<Field>' . $row['field']. '</Field>';

        $xml .= '</student>';
    }
    $xml .= '</StudentsInfo>';
    return $xml;
    }else{
        return "Error".errorInfo();
    }
```

ext Preprocessor file   length : 2,238   lines : 94   Ln : 61   Col : 22   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS



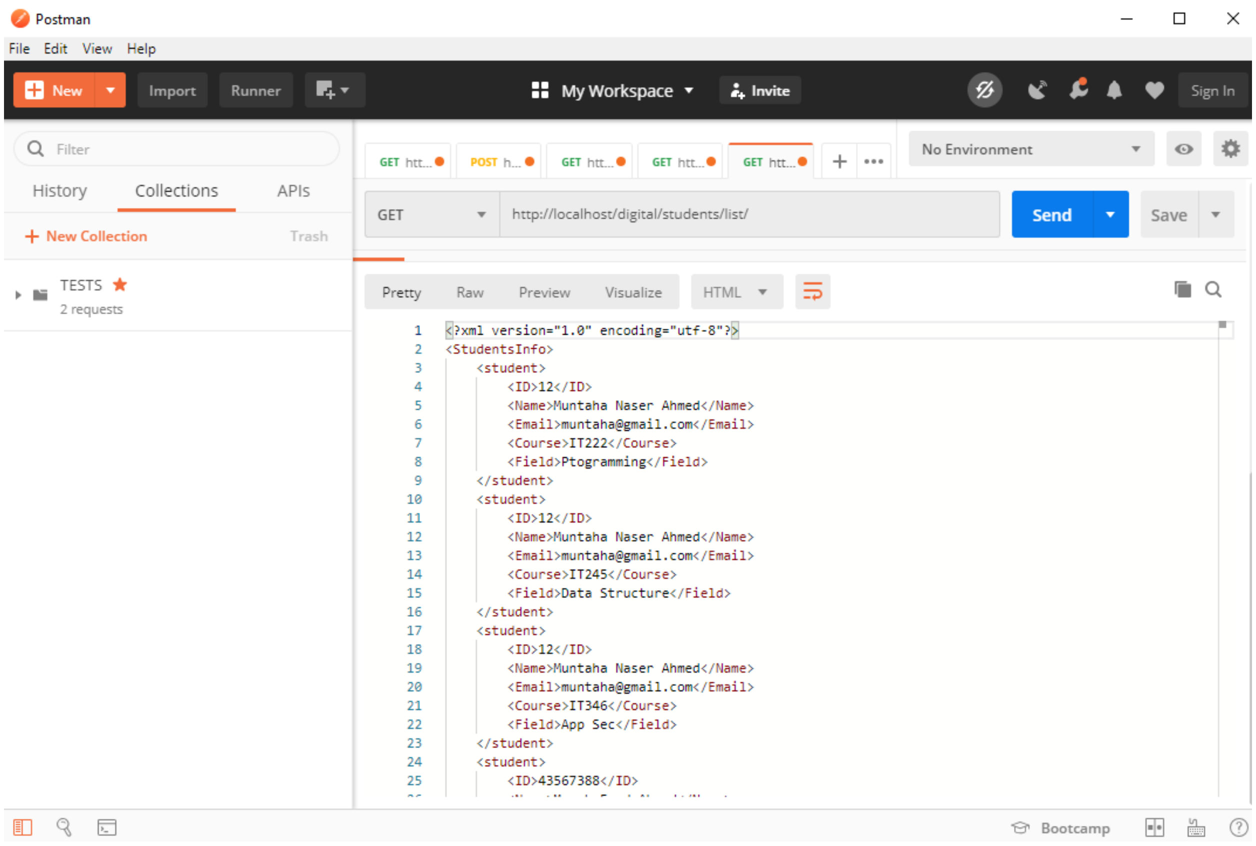
```
/**
Get a student by ID
*/
function getStudentBYID($id){

    $qry="select c.name as courseName,c.field,s.id,s.name,s.email from course as c inner join enrolment as e on c.id=e.course_id inner join student as s on
    e.student_id=s.id  and c.id='$id'";
    $pdo = pdo_dbconn();
    $stmt = $pdo->prepare($qry);
    $stmt->execute();

    if($stmt->rowCount()>0){
    $xml = '<?xml version="1.0" encoding="utf-8"?>';
    $xml .= '<StudentsInfo>';
     while($row=$stmt->fetch(PDO::FETCH_ASSOC)){
        $xml .= '<student>';

        $xml .= '<ID>' .$row['id']. '</ID>';
        $xml .= '<Name>' .$row['name']. '</Name>';
        $xml .= '<Email>' . $row['email']. '</Email>';
        $xml .= '<Course>' . $row['courseName']. '</Course>';
        $xml .= '<Field>' . $row['field']. '</Field>';

        $xml .= '</student>';
    }
    $xml .= '</StudentsInfo>';
    return $xml;
    }else{
        return "There are no students for this course";
    }

}
?>
```

A .htaccess file controls how requests are forwarded as shown below

```
# Turn rewrite engine on

Options +FollowSymlinks

RewriteEngine on

# map neat URL to internal URL

RewriteRule ^students/list/$   getStudent.php?view=all [nc,qsa]
RewriteRule ^students/list/([0-9]+)/$
getStudent.php?view=single&id=$1 [nc,qsa]
```

3. Test your API using Postman and show screenshots of the results.

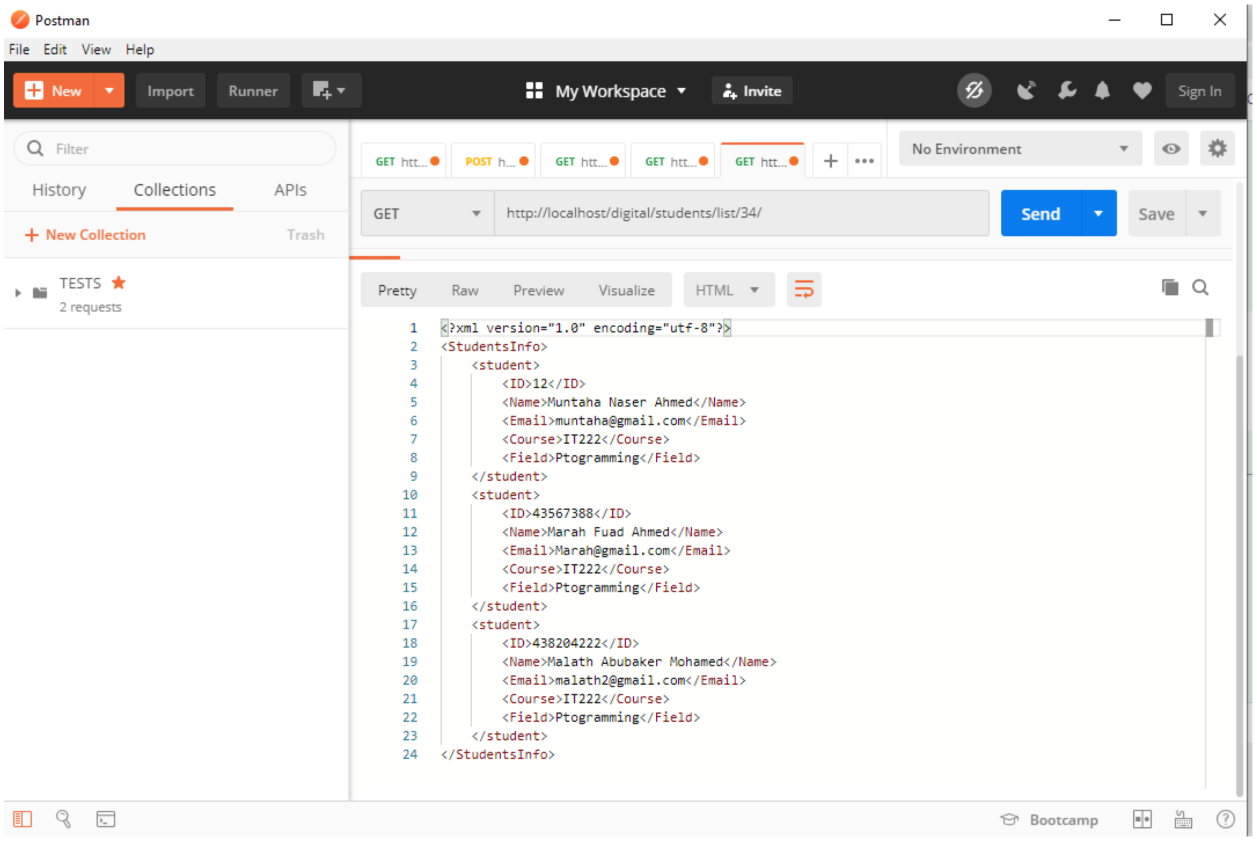
i. Get a list of all students



GET http://localhost/digital/students/list/

```
<?xml version="1.0" encoding="utf-8"?>
<StudentsInfo>
    <student>
        <ID>12</ID>
        <Name>Muntaha Naser Ahmed</Name>
        <Email>muntaha@gmail.com</Email>
        <Course>IT222</Course>
        <Field>Ptogramming</Field>
    </student>
    <student>
        <ID>12</ID>
        <Name>Muntaha Naser Ahmed</Name>
        <Email>muntaha@gmail.com</Email>
        <Course>IT245</Course>
        <Field>Data Structure</Field>
    </student>
    <student>
        <ID>12</ID>
        <Name>Muntaha Naser Ahmed</Name>
        <Email>muntaha@gmail.com</Email>
        <Course>IT346</Course>
        <Field>App Sec</Field>
    </student>
    <student>
        <ID>43567388</ID>
```

ii. Get a student by ID



GET http://localhost/digital/students/list/34/

```
<?xml version="1.0" encoding="utf-8"?>
<StudentsInfo>
    <student>
        <ID>12</ID>
        <Name>Muntaha Naser Ahmed</Name>
        <Email>muntaha@gmail.com</Email>
        <Course>IT222</Course>
        <Field>Ptogramming</Field>
    </student>
    <student>
        <ID>43567388</ID>
        <Name>Marah Fuad Ahmed</Name>
        <Email>Marah@gmail.com</Email>
        <Course>IT222</Course>
        <Field>Ptogramming</Field>
    </student>
    <student>
        <ID>438204222</ID>
        <Name>Malath Abubaker Mohamed</Name>
        <Email>malath2@gmail.com</Email>
        <Course>IT222</Course>
        <Field>Ptogramming</Field>
    </student>
</StudentsInfo>
```

4. Write the documentation of your API so that it is usable by 3rd party developers.

The RESTful provides 2 resources: get a list of all students and get a specific course by ID. Every resource is identified via a URI (Uniform Resource Identifier).

URI	**Type**	**Method**	**Description**
http://localhost/digital/students/list/	GET	XML	Get a list of all students and their course
http://localhost/digital/students/list/{ID}/	GET	XML	Get a specific course and a list of the student takes