UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 5, Examination in Engineering, July 2017

Module Number: EE5207 **Module Name: Internet Technologies Part - II**

[2 hours]

[Answer all questions, each question carries 5 marks]

Q1. a) What are the benefits of using Node.js.

[1 mark]

b) i) Explain the functionality of the following code segment.

Listing 1: Node.js code

```
var express = require('express');
var bodyParser = require('body-parser');
var path = require('path');

var app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended:false}));

app.use(express.static(path.join(__dirname,'public')));

app.get('/',function(req,res){
  res.send('Hello World');
});

app.listen(3000,function(){
  console.log('Server started on Port 3000');
});
```

[2 marks]

ii) What is the output of the request `http://localhost:3000/users` ?

[1 mark]

iii) Modify the code in Listing 1 to display "Welcome to Node.js" when you send the request to `http://localhost:3000/welcome` ?

[1 mark]

Q2. The following Express.js application is configured to use handlebars view engine. The router in Listing 2 is used to handle routing and the code for index.hbs is given in Listing 3. The application runs on port 3000.

Listing 2: index.js code for routing

```
var router = express.Router();


router.get('/', function(req, res, next) {
console.log(req.cookies);
console.log(req.session);
res.render('index', { title: 'Hello Express' ,
  success:req.session.success,errors:req.session.errors});
  req.errors = null;
});

router.post('/submit', function(req, res, next) {
req.check('email', 'Invalid email address').isEmail();
req.check('password', 'Password minimum length 4')
    .isLength({min: 4});
req.check('password', 'Password does not match')
    .equals(req.body.confirmPassword);

var errors = req.validationErrors();
if (errors) {
  req.session.errors = errors;
  req.session.success = false;
} else {
  req.session.success = true;
}
res.redirect('/');
});
```

Listing 3: index.hbs code

```
<h1>{{ title }}</h1>
{{# if success }}
  <section class="success">
    <h2>Successful Validation!</h2>
  </section>
{{ else }}
  {{# if errors }}
    <section class="errors">
    <ul>
```

```
    {{# each errors }}
      <li>{{ this.msg }}</li>
      {{/each}}
      </ul>
      </section>
  {{/if}}

  <form action="/submit" method="post">
    <div class="input">
      <label for="email">E-Mail</label>
      <input type="text" id="email" name="email">
    </div>
    <div class="input">
      <label for="password">Password</label>
      <input type="password" id="password" name="password">
    </div>
    <div class="input">
      <label for="confirmPassword">Confirm Password</label>
      <input type="password" id="confirmPassword" name="confirmPassword
        ">
    </div>
    <button type="submit">Sign up</button>
  </form>
  {{/if}}
```

a) Draw the web page rendered for the initial request to `http://localhost:3000/`?

[1.5 marks]

b) Explain the functionality of index.js in Listing 2.

[1.5 marks]

c) Explain what happens, when you press "Sign Up" button.

[2 marks]

Q3. a) What is the use of Middleware in Express.js?

[1 mark]

b) Write a Middleware function which print "Logging the request" in the console for each request it receives.

[1 mark]

c) How do you manage a session in Express.js?

[1 mark]

d) Explain the behavior of the code in Listing 4.

Listing 4: Node.js code

```
var express    = require('express');
var app        = express();
var bodyParser = require('body-parser');

var mongoose   = require('mongoose');
mongoose.connect('mongodb://localhost/test1');
var Bear     = require('./models/bear');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

var port = process.env.PORT || 8080;

var router = express.Router();


router.get('/', function(req, res) {
  res.json({ message: 'hooray! welcome to our api!' });
});

router.route('/bears')
  .post(function(req, res) {

    var bear = new Bear();
    bear.name = req.body.name;


    bear.save(function(err) {
      if (err)
        res.send(err);
      res.json({ message: 'Bear created!  : '+ bear.name });
    });

  })
  .get(function(req, res) {
    Bear.find(function(err, bears) {
      if (err)
        res.send(err);
    res.json(bears);
  });
```

```
});

app.use(router);
app.listen(port);
console.log('Magic happens on port ' + port);
```

[2 marks]

Q4. a) Explain the MVC design pattern in ASP.NET MVC.

[1 mark]

b) Write a controller name "Person" and write the following action methods

i) Write the `Greetings` action for `Person` controller, which takes *name* and *times* as parameters and display the "Hello " + *name* + " well come" message number equal to value *times*. (You need to write `Greetings` View with Razor code and pass variables from controller to the View.)

ii) Explain how you can pass *name* and *times* parameters using URL.

iii) Explain what errors can happen if these variables are not passed to the action method. Explain a way to rectify the situation.

[3 marks]

c) The `Product` class list is used to store the product details. Explain how you pass the list to the corresponding view and explain how you retrieve the list in the view.

[1 mark]

Q5. a) What is the difference between ASP.NET core and ASP.NET MVC frameworks

[1 mark]

b) What is a Web Service? Mention advantages of RESTful web services?

[1 mark]

c) Explain the Object Relational mapping framework. Give two examples.

[1 mark]

d) `Product` class contains *product_id*, *price*, *product_name* and *available_quantity* properties. The data is stored using *Products* `DbSet` property using Entity Framework. Explain how you can create a HTTP GET method using ASP.NET WEB API framework to retrieve list of products in ascending order of price.

[1 mark]

e) Explain the functionality of the code segment in Listing 5. Here _context is the Entity Framework data context object.

[1 mark]

Listing 5: Node.js code

```
[HttpPost]
public IActionResult Create([FromBody] TodoItem item)
{
  if (item == null)
  {
    return BadRequest();
  }

  _context.TodoItems.Add(item);
  _context.SaveChanges();

}
```