Unit 13.3 Highlights

Separation of concerns (SoC) - Design principle for separating a computer program into distinct sections, such that each section addresses a separate concern or set of information which affects the code of a computer program (i.e., separate routing concern from app/server logic for node express apps)

Ex. 1

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

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

// Sets up the Express app to handle data parsing
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

require("./routes/index.js")(app); //pass a reference to the express app to a separate file

app.listen(PORT, function() {
  console.log("App listening on PORT: " + PORT);
});
//--------------------------------meanwhile, in the ./routes/index.js file...

module.exports = function(app) {
   app.get("/user", function(req,res) {
      res.end(“GET user”);
   });
   app.post(“/user", function(req,res) {
      res.end("POST user");
   });
}
```

express.Router() - Method which produces a new router object, which is an isolated instance of middleware and routes

Ex. 2

```
var app = require("express")();
var bodyParser = require("body-parser");
var users = require("./routes");

var PORT = 8080;

// Sets up the Express app to handle data parsing
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use('/user', users);

app.listen(PORT, function() {
  console.log("App listening on PORT: " + PORT);
});
//--------------------------------meanwhile, in the ./routes/index.js file...

var router = require("express").Router();

module.exports = function() {
   router.get("/name", function(req,res) {
      res.send("GET user");
   });
   router.post("/name", function(req,res) {
      res.send("POST user");
   });
}
```