Name: Adithya M	SRN: PES1UG20CS621	Section: K
	Date: 13/12/2021	Exercise No: 5

PROBLEM STATEMENT(EVEN SRN's)

1. Create an API that has a collection of books having different fields (such as book_id, book_name, book_price,book_authetc)for each book.
   Using HTTP methods GET method extract the data of book using book_id, Using PUT method update the price, Using POST method insert a new data and display the same. (Use MongoDb database)

2. Create student resume with details (such as name, dob, qualification, nationality etc…) using formdata and upload the student photo to the form.

OBJECTIVE

The objective of this exercise is to test the student on ExpressJS framework.
It evaluates the student's knowledge of http request, respose objects. Creating RestFul API and web services

PREREQUISITE

In order to write this program, the student needs to understand the fundamentals of HTML and CSS. The student must be familiar with basic Javascript and express module.

PROGRAM

1)

```
const express = require("express");
const MongoClient = require("mongodb").MongoClient;
const app = express();
const dburl = "mongodb://localhost:27017/";

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

app.get("/book", (req, res) => {
  MongoClient.connect(dburl, (err, db) => {
    if (err) {
      res.send(err);
```

```
    }
    let dbo = db.db("book");
    dbo
      .collection("books")
      .find({})
      .toArray((err, result) => {
        if (err) throw err;
        res.send(result);
        db.close();
      });
  });
});
app.get("/book/:id", (req, res) => {
  bookId = req.params.id;
  MongoClient.connect(dburl, (err, db) => {
    if (err) {
      res.send(err);
    }
    let dbo = db.db("book");
    dbo.collection("books").findOne({ book_id: bookId }, (err, result) => {
      if (err) throw err;
      res.send(result);
      db.close();
    });
  });
});

app.post("/book", (req, res) => {
  data = {
    book_id: req.body.book_id,
    book_name: req.body.book_name,
    book_price: req.body.book_price,
    book_auth: req.body.book_auth,
  };
  try {
    MongoClient.connect(dburl, (err, db) => {
      if (err) {
        res.send(err);
      }
      let dbo = db.db("book");
      dbo.collection("books").insertOne(data, (err, result) => {
```

```
        if (err) throw err;
        res.send({ message: "Data added successfully", data: data });
        db.close();
      });
    });
  } catch (error) {
    res.send(error);
  }
});

app.put("/book/:id", (req, res) => {
  bookId = req.params.id;
  MongoClient.connect(dburl, (err, db) => {
    if (err) {
      res.send(err);
    }
    let query = { book_id: bookId };
    let newValues = { $set: req.body };
    let dbo = db.db("book");
    dbo.collection("books").updateOne(query, newValues, { upsert: true }, (err, result) => {
      if (err) throw err;
      console.log("Updated document successfully");
      res.send(result);
      db.close();
    });
  });
});

app.listen(8081, () => {
  console.log("Server started");
});
```

2)

```
const express = require("express");
const bodyParser = require("body-parser");
const fileUpload = require("express-fileupload");
const fs = require("fs");

const app = express();

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

app.get("/", (req, res) => {
  res.sendFile(__dirname + "/index.html");
});

app.post("/", (req, res) => {
  if (req.files) {
    console.log(req.files);
  }

  let fileObject = req.files.image;
  let fileName = fileObject.name;
  let fileSize = fileObject.size;

  fileObject.mv("./uploads/" + fileName, (err) => {
    if (err) {
      console.log("Error : " + err);
    } else {
      res.send("FileName : " + fileName + " Uploaded Successfully to ./uploads
Directory\n");
    }
  });
});
app.listen(8081, () => {
  console.log("Server started");
});
```

TEST CASES

1) GET request

GET http://localhost:8081/book

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "61b75354330c3864a7dc3a5a",
        "book_id": "abcd",
        "book_name": "defg",
        "book_price": "420",
        "book_auth": "somebody"
    },
    {
        "_id": "61b755ec330c3864a7dc3a5b",
        "book_id": "yes",
        "book_name": "no",
        "book_price": "50",
        "book_auth": "yo"
    },
    {
        "_id": "61b75637330c3864a7dc3a5c",
        "book_id": "hmm",
        "book_name": "xyz",
        "book_price": "bcvhd",
        "book_auth": "mnop"
    }
```


2) GET request to /book/yes

GET http://localhost:8081/book/yes

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "_id": "61b755ec330c3864a7dc3a5b",
    "book_id": "yes",
    "book_name": "no",
    "book_price": "50",
    "book_auth": "yo"
}
```


3) POST request

POST http://localhost:8081/book

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "book_id":"qwerty",
    "book_name":"idk",
    "book_price":"234",
    "book_auth":"right"
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "message": "Data added successfully",
    "data": {
        "book_id": "qwerty",
        "book_name": "idk",
        "book_price": "234",
        "book_auth": "right",
        "_id": "61b7583f6894dad4b850e392"
    }
}
```

4) PUT request to /book/abcd

PUT http://localhost:8081/book/abcd

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

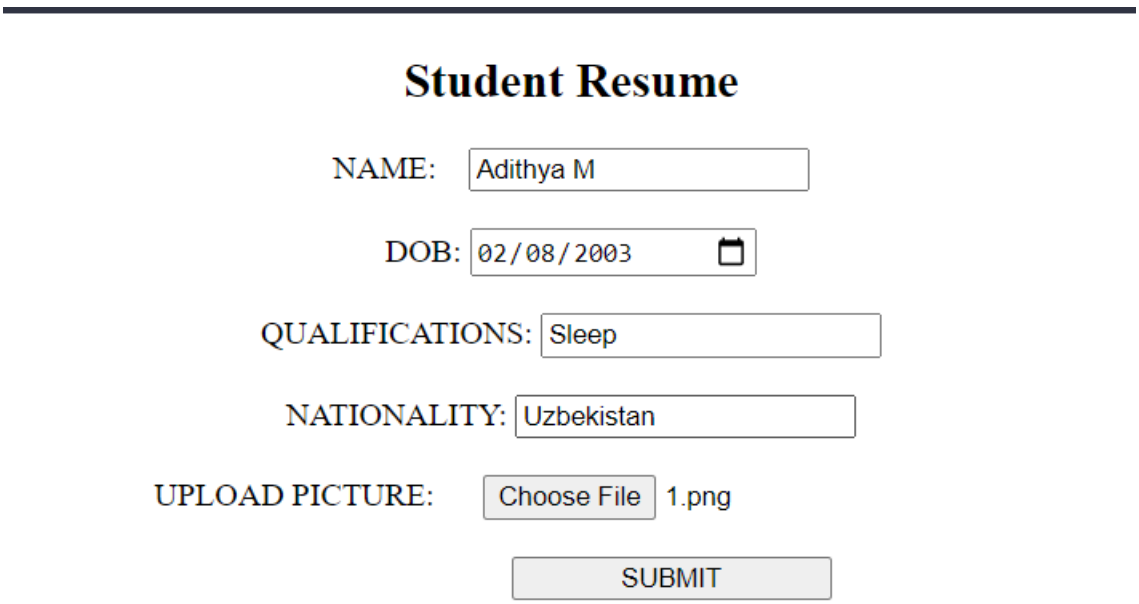
```
{
    "book_price":"678"
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "acknowledged": true,
    "modifiedCount": 1,
    "upsertedId": null,
    "upsertedCount": 0,
    "matchedCount": 1
}
```

SCREENSHOT OF OUTPUT



FileName : 1.png Uploaded Successfully to ./uploads Directory