1. SpringBoot:

1. Spring Initialzr: 2.5.6; Java version 8; Dependencies: Spring Web, Spring Data JPA, MySQL Driver.
2. localhost/phpmyadmin: Create database (id, author, publisher)
3. Downloaded folder: src-main-resources-application.properties: (paste)

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/*db_example*
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name =com.mysql.jdbc.Driver
```

4. Make sure pom.xml java version=1.8
5. src-main-java-com-example-*BookCrud*-controller:
   BookController.java (name of folder: BookCrud,

```
package com.example.BookCrud.controller;

import com.example.BookCrud.model.Book;
import com.example.BookCrud.service.BookService;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
public class BookController {
   private final BookService bookService;

   public BookController(BookService bookService) {
      this.bookService = bookService;
   }

   @GetMapping("/getAllBooks")
   public List<Book> getAllBooks() {
      return bookService.getAllBooks();
   }

   @GetMapping("/get/{bookID}")
   public Book getBook(@PathVariable String bookID) {
      return bookService.getBook(bookID);
   }

   @PostMapping("/createBook")
   public Book createBook(@RequestBody Book book) {
      return bookService.create(book);
   }

   @DeleteMapping("/deleteBook/{bookId}")
   public String deleteBook(@PathVariable String bookId) {
      bookService.delete(bookId);
      return "Employee Deleted";
   }

   @PutMapping("/updateBook/{bookId}")
   public Book updateBook(@RequestBody Book book, @PathVariable String bookId) {
      return bookService.update(book, bookId);
   }
```

```
    @DeleteMapping("/deleteAll")
    public String deleteBooks() {
        bookService.deleteAll();
        return "All Emplyees data deleted";
    }
}
```

6.src-main-java-com-example-*BookCrud*-model:

Book.java

```
package com.example.BookCrud.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class Book {
    @Id
    private String id;
    @Column
    private String author;
    @Column
    private String publisher;
        public Book()
        {

        }
    public Book(String string, String string2, String string3) {
        id=string;
            author=string2;
            publisher=string3;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
```

```
        this.publisher = publisher;
    }
}
```

7.src-main-java-com-example-*BookCrud*-repository:

BookRepository.java

```
package com.example.BookCrud.repository;

import com.example.BookCrud.model.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface BookRepository extends JpaRepository<Book, String> {
}
```

8. src-main-java-com-example-*BookCrud*-service:

BookService.java

```
package com.example.BookCrud.service;

import com.example.BookCrud.model.Book;
import com.example.BookCrud.repository.BookRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class BookService {

    private final BookRepository bookRepository;

    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    public Book getBook(String bookID) {
        return bookRepository.findById(bookID).orElse(null);
    }


    public Book create(Book book) {
        return bookRepository.save(book);
    }

    public void delete(String bookId) {
        bookRepository.deleteById(bookId);
    }

    public Book update(Book book, String bookId) {
        Book book1 = bookRepository.findById(bookId).get();
        book1.setAuthor(book.getAuthor());
```

```
        book1.setPublisher(book.getPublisher());
        bookRepository.save(book1);
        return book1;
    }

    public void deleteAll() {
        bookRepository.deleteAll();
    }
}
```

9. Save all and open terminal from original BookCrud folder:

Run:

```
./mvnw spring-boot:run
```

//note the port number

10. Output (postman)
http://localhost:8080/demo/add?name=bbb&email=bbb@aa.edu
GET:

http://localhost:*8080*/getAllBooks
http://localhost:8080/get/1 (will work only after entering value 1 in db)

DELETE:

http://localhost:8080/deleteBook/1

POST:

http://localhost:8080/createBook

Select the Body
Select raw and type JSON
Insert the data. We have inserted the following data in the Body:

```
{
    "id": "1",
    "author": "V. Rajaraaman",
    "publisher": "PHI Learning"
}
{
    "id": "2",
    "author": "Ivan Bayrossn",
    "publisher": "McGraw-Hill"
}
{
    "id": "3",
    "author": "Dinesh Rajput",
    "publisher": "Perlego"
}
```

PUT (update):

http://localhost:8080/updateBook/2

Select the Body
Select raw and type JSON
Insert the data. We have inserted the following data in the Body:

```
{
    "id": "2",
    "author": "Ivan Bayrossn",
    "publisher": "Westland Publications"
```

```
}
```

2. Kubernetes

-Login to DockerHub
-Open localhost phpmyadmin database and keep

Commands:

```
#give app name as : appname followed by last three digits of your usn
# for eg. 1ms99cs001 -> appname001
# give port name : digit 9 follwed by last three digits of your usn
# for eg 1ms99cs001 -> 9001

sudo su
docker build -t nodeapp .
docker tag nodeapp reshmaram/nodeapp
docker login
docker push reshmaram/nodeapp
kubectl create deployment nodeapp --image=reshmaram/nodeapp
kubectl get deployment nodeapp
kubectl get pods | grep '^nodeapp'
kubectl expose deployment nodeapp --type=LoadBalancer --port=8080
kubectl get service nodeapp
# to run open browser
http://172.1.14.168:<node_port>/index.html

kubectl delete service nodeapp
kubectl delete deployment nodeapp
kubectl delete --all pods
#kubectl expose deployment/nodeapp --type="NodePort" --port 8080
```

1. index.html

```
<html>
<body>
<form action="process_get" method="get">
First Name: <input type="text" name="first_name">  <br>
Last Name: <input type="text" name="last_name">
<input type="submit" value="Submit">
</form>
<a href="form.html">Google</a>
<a href="/">welcome</a>
</body>
</html>
```

2. get_ex1.js

```
var express = require('express');
var app = express();

app.get('/index.html', function (req, res) {
   res.sendFile( __dirname + "/" + "index.html" );

})
app.get('/process_get', function (req, res) {
response = {
       first_name:req.query.first_name,
```

```
      last_name:req.query.last_name
   };
   console.log(response);
            console.log("Sent data are (GET): first name :"+req.query.first_name+" and last name
:"+req.query.last_name);
   //res.end(JSON.stringify(response));
   res.end("Sent data are (GET): first name :"+req.query.first_name+" and last name :"+req.query.last_name);
})
var server = app.listen(8080, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

3. Dockerfile:

```
FROM node:10

RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app

WORKDIR /home/node/app

COPY package*.json ./

USER node

RUN npm install

COPY --chown=node:node . .

EXPOSE 8080

CMD [ "node", "get_ex1.js" ]
```

4. .dockerignore

```
node_modules
npm-debug.log
Dockerfile
.dockerignore
```

5. package.json

```
{
  "name": "nodejs-image-demo",
  "version": "1.0.0",
  "description": "nodejs image demo",
  "author": "HKS",
  "license": "RIT",
  "main": "app.js",
  "scripts":{
            "start":"node get_ex1.js"
  },
  "dependencies": {
    "express": "^4.16.4"
  }
}
```