BookCrudApplocation.java

```java
package com.example.BookCrud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BookCrudApplication {

    public static void main(String[] args) {
        SpringApplication.run(BookCrudApplication.class, args);
    }

}
```

model----->Book.java

```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;
    }
}
```

repository---->BookRepository.java

```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> {
}
```

services--->BookService.java

```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;
```

```java
@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();
    }
}
```

controller---->BookController.java

```java
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";
    }
}
```

Application.properties
```
spring.jpa.hibernate.ddl-auto = update
spring.datasource.url= jdbc:mysql://localhost:3306/bootdb
spring.datasource.username= root
#TODO: Change this
spring.datasource.password=
spring.datasource.driver-class-name= com.mysql.cj.jdbc.Driver
```

StudentCrudApplication

StudentApplication.java
```
package com.example.Student;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class StudentApplication {

    public static void main(String[] args) {
        SpringApplication.run(StudentApplication.class, args);
    }

}
```

model--->User.java
```
package com.example.Student.model;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Column;

@Entity
@Table
public class User{
    @Id
    private Integer id;
    @Column
    private String name;
```

```java
    @Column
    private String email;

    public User(){

    }
    public User(Integer x,String string,String string2){
        id=x;
        name=string;
        email=string2;
    }

    public Integer getId(){
        return id;
    }

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

    public String getName(){
        return name;
    }

    public void setName(String name){
        this.name=name;
    }

    public String getEmail(){
        return email;
    }

    public void setEmail(String email){
        this.email=email;
    }
}
```

repository--->UserRepository.java

```java
package com.example.Student.repository;
import com.example.Student.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User,Integer> {
}
```

service ---->UserService.java

```
package com.example.Student.service;
import com.example.Student.model.User;
import com.example.Student.repository.UserRepository;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class UserService {
    private final UserRepository userRepository;
    public UserService(UserRepository userRepository){
        this.userRepository=userRepository;
    }

    public List<User> getAllUsers(){
        return userRepository.findAll();
    }

    public User getUser(Integer studentId){
        return userRepository.findById(studentId).orElse(null);

    }
    public User create(User user){
        return userRepository.save(user);
    }

    public void delete(Integer studentId){
        userRepository.deleteById(studentId);
    }
    public void deleteAll(){
        userRepository.deleteAll();

    }

    public User update(User user,Integer studentId){
        User user1=userRepository.findById(studentId).get();
        user1.setName(user.getName());
        user1.setEmail(user.getEmail());
        return userRepository.save(user1);
    }
}
```

controller---->UserController.java

```java
package com.example.Student.controller;
import com.example.Student.model.User;
import com.example.Student.service.UserService;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
public class UserController {
    private final UserService userService;

    public UserController(UserService userService){
        this.userService=userService;
    }

    @GetMapping("/getAllUsers")
    public List<User> getAllUsers(){
        return userService.getAllUsers();
    }

    @GetMapping("/get/{studentId}")
    public User getUser(@PathVariable Integer studentId){
        return userService.getUser(studentId);
    }

    @PostMapping("/createUser")
    public User createUser(@RequestBody User user){
        return userService.create(user);
    }

    @PutMapping("/updateUser/{studentId}")
    public User updateUser(@RequestBody User user,@PathVariable Integer
studentId){
        return userService.update(user,studentId);
    }

    @DeleteMapping("/deleteUser/{studentId}")
    public String deleteUser(@PathVariable Integer studentId){
        userService.delete(studentId);
        return "Student Deleted";
    }
    @DeleteMapping("/deleteAll")
    public String deleteAll(){
        userService.deleteAll();
        return "All Users Deleted";
    }
}
```

Application.properties

```
spring.jpa.hibernate.ddl-auto = update
spring.datasource.url= jdbc:mysql://localhost:3306/students
spring.datasource.username= root
#TODO: Change this
spring.datasource.password=
spring.datasource.driver-class-name= com.mysql.cj.jdbc.Driver
```

KUBERNETES

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>
```

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)

})
```

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"
  }
}
```

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" ]
```

```
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

docker build -t nodeapp .
docker tag nodeapp cserit/nodeapp
docker login
docker push cserit/nodeapp
kubectl create deployment nodeapp --image=cserit/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
```