PROJECT NAME: COLLEGE MANAGEMENT SYSTEM

TEAM LEADER: R.Monisha Rameshbabu (EBEON0322582330)

TEAM MEMBERS: S.Dharani-EBEON0322575081

R.Revathi-EBEON322570615

K.Ramya-EBEON0322571078

A.Priyadharshini - EBEON0FWL562309

PROJECT STEPS

MODULE NAME :Placement and Laboratory

MODULE DONE BY : S.Dharani-EBEON0322575081

Step 1: Open Spring Initializr http://start.spring.io.

Step 2: Select the Spring Boot version 2.3.0.M1.

Step 3: Provide the Group name. We have provided

Step 4: Provide the Artifact Id. We have provided spring-boot-crud-operation.

Step 5: Add the dependencies Spring Web, Spring Data JPA, and H2 Database.

Step 6: Click on the Generate button. When we click on the Generate button, it wraps the specifications in a Jar file and downloads it to the local system.

Step 7: Extract the Jar file and paste it into the STS workspace.

Step 8: Import the project folder into STS.

File -> Import -> Existing Maven Projects -> Browse -> Select the folder spring-boot-crud-operation -> Finish

It takes some time to import.

Step 9: Create a package with the name **com.placement.campus** in the folder **src/main/java.**

Step 10: Create a model class in the package **com.placement.campus.** We have created a model class with the name**.** In the class, we have done the following:

- Define variable
- Generate Getter and Setter.

Right-click on the file -> Source -> Generate Getters and Setters.

- Mark the class as an **Entity** by using the annotation **@Entity.**
- Mark the class as **Table** name by using the annotation **@Table.**
- Define each variable as **Column** by using the annotation **@Column.**

Step 11: Create a package with the name **com.Placement.campus.controller** in the folder **src/main/java.**

Step 12: Create a Controller class in the package **com.placement.campus.controller.** We have created a controller class with the name **SudentController.** In the BooksController class, we have done the following:

- Mark the class as **RestController** by using the annotation **@RestController.**
- Autowire the **StudentService** class by using the annotation **@Autowired**.
- Define the following methods:
    - **getAllStudent():** It returns a List of all Student.
    - **getStudent():** It returns a student detail that we have specified in the path variable. We have passed id as an argument by using the annotation @PathVariable. The annotation indicates that a method parameter should be bound to a URI template variable.
    - **deleteStudent():** It deletes a specific Student that we have specified in the path variable.
    - **saveStudent():** It saves the student detail. The annotation @RequestBody indicates that a method parameter should be bound to the body of the web request.

- **update():** The method updates a record. We must specify the record in the body, which we want to update. To achieve the same, we have used the annotation @RequestBody.

Step 13: Create a package with the name **com.placement.campus.service** in the folder **src/main/java.**

Step 14: Create a **Service** class. We have created a service class with the name **BooksService** in the package **com.placement.campus.service.**

Step 15: Create a package with the name **com.placement.campus.repository** in the folder **src/main/java.**

Step 16: Create a **Repository** .We have created a repository interface with the name **StudentRepository** in the package **com.placement.campus.repository.** It extends the **Crud Repository** interface.

Now we will configure the datasource **URL, driver class name, username,** and **password,** in the **application.properties** file.

Step 17: Open the **application.properties** file and configure the following properties.

application.properties

Now we will run the application.

Step 18: Open **SpringBootCrudOperationApplication.java** file and run it as Java Application.

SpringBootCrudOperationApplication.java

Step 19: Click on the **Student** table and then click on the **Run** button. The table shows the data that we have inserted in the body.

MODULE NAME : Online Payment and Chatbot

MODULE DONE BY : R.Monisha Rameshbabu (EBEON0322582330)

ONLINE PAYMENT:

Step 1: Create a Project from **Spring Initializr** and also add the following dependencies,

- ✓ Spring Web.
- ✓ Spring Boot DevTools.
- ✓ Thymleaf

Step 2: After extracting the project, import the project in your IDE such as Eclipse.

Step 3: Get Paytm Properties such as **Merchant ID**, **Merchant Key**, etc.

- ✓ Login or Sign up to your Paytm account using this url: https://dashboard.paytm.com/
- ✓ Go to the **Developer setting** on the left-hand side of the site.
- ✓ Choose **API Keys.**
- ✓ Select **Generate Test API** Details.
- ✓ You will get **Test Merchant ID** and **Test Merchant Key.**

Step 4: Configure the **application.properties** file by using these details from DEVELOPER SETTINGS from your Paytm account.

Step 5: Create a **POJO CLASS, Controller Class**

Step 6: When the API gets called then it will return **home.html**, where we will write the HTML to display the form

Step 7: Validating the checksum and finally displaying the data on the **report.html** file. For validating the checksum we defined a method **validateCheckSum()**

STEP 8: PaytmChecksum to ensure that API requests and responses shared between your application and Paytm over network have not been tampered with.

STEP 9: But before Running make sure that you have all the dependencies as mentioned in **the pom.xml** file.

CHATBOT AI:

PREREOUISITES:

Reference AIML Implementation

- ✓ To get started, we shall use an already working reference application.
- ✓ There is one such java based implementation called **program-ab** hosted on google-code repository.
- ✓ Download the program-ab latest distribution from maven code repository.

STEP1: Create eclipse project

STEP 2: Import AIML library

After we have created the maven project to start the development, let us choose packaging as *jar* and maven coordinate as your choice and import to eclipse.Now create a folder lib in the base folder and copy the Ab.jar from the *program-ab distribution* to this folder.

STEP3: Add AIML to Classpath

- ▶ To add AIML to classpath, add Ab.jar to deployment assembly in eclipse. Alternatively, you can install this jar into your local maven repository and then use it.
- ▶ To install locally, place the jar in any folder and provide that location in the *systemPath* tag. Now, add below *AIML maven dependency* to pom.xml. Now build the maven project by command mvn clean install.

STEP 4: Conclusion

- In this **AIML chatbot,** we have learned to create a **simple command-line based chatbot program** with **program-ab** reference application.

MODULE NAME: Faculty

MODULE DONE BY: K.Ramya-EBEON0322571078

Step 1: Open Spring Initializr http://start.spring.io.

Step 2: Select the Spring Boot version 2.3.0.M1.

Step 3: Provide the Group name. We have provided

Step 4: Provide the Artifact Id. We have provided spring-boot-crud-operation.

Step 5: Add the dependencies Spring Web, Spring Data JPA, and H2 Database.

Step 6: Click on the Generate button. When we click on the Generate button, it wraps the specifications in a Jar file and downloads it to the local system.

Step 7: Extract the Jar file and paste it into the STS workspace.

Step 8: Import the project folder into STS.

File -> Import -> Existing Maven Projects -> Browse -> Select the folder spring-boot-crud-operation -> Finish

It takes some time to import.

Step 9: Create a package with the name in the folder **src/main/java.**

Step 10: Create a model class in the package**.** We have created a model class with the name**.** In the class, we have done the following:

- Define variable
- Generate Getter and Setter.

Right-click on the file -> Source -> Generate Getters and Setters.

- Mark the class as an **Entity** by using the annotation **@Entity.**
- Mark the class as **Table** name by using the annotation **@Table.**
- Define each variable as **Column** by using the annotation **@Column.**

Step 11: Create a package with the name in the folder **src/main/java.**

Step 12: Create a Controller class in the package **.** We have created a controller class with the name **.** In the class, we have done the following:

- Mark the class as **RestController** by using the annotation **@RestController.**
- Autowire the class by using the annotation **@Autowired**.
- Define the following methods:
    - **getAllFacultyt():** It returns a List of all faculty.
    - **getFaculty():** It returns a faculty detail that we have specified in the path variable. We have passed id as an argument by using the annotation @PathVariable. The annotation indicates that a method parameter should be bound to a URI template variable.
    - **deleteFaculty():** It deletes a specific faculty that we have specified in the path variable.
    - **saveFaculty():** It saves the faculty detail. The annotation @RequestBody indicates that a method parameter should be bound to the body of the web request.
    - **update():** The method updates a record. We must specify the record in the body, which we want to update. To achieve the same, we have used the annotation @RequestBody.

Step 13: Create a package with the name in the folder **src/main/java.**

Step 14: Create a **Service** class. We have created a service class with the name in the package **.**

Step 15: Create a package with the name in the folder **src/main/java.**

Step 16: Create a **Repository** interface. We have created a repository interface with the name in the package **.** It extends the **Crud Repository** interface.

Now we will configure the datasource **URL, driver class name, username,** and **password,** in the **application.properties** file.

Step 17: Open the **application.properties** file and configure the following properties.

application.properties

Now we will run the application.

Step 18: Open **SpringBootCrudOperationApplication.java** file and run it as Java Application.

SpringBootCrudOperationApplication.java

Step 19: Click on the **faculty** table and then click on the **Run** button. The table shows the data that we have inserted in the body.

MODULE NAME: Admission

MODULE DONE BY: R.Revathi-EBEON322570615

Step 1: Open Spring Initializr http://start.spring.io.

Step 2: Select the Spring Boot version 2.3.0.M1.

Step 3: Provide the Group name. We have provided

Step 4: Provide the Artifact Id. We have provided spring-boot-crud-operation.

Step 5: Add the dependencies Spring Web, Spring Data JPA, and H2 Database.

Step 6: Click on the Generate button. When we click on the Generate button, it wraps the specifications in a Jar file and downloads it to the local system.

Step 7: Extract the Jar file and paste it into the STS workspace.

Step 8: Import the project folder into STS.

File -> Import -> Existing Maven Projects -> Browse -> Select the folder spring-boot-crud-operation -> Finish

It takes some time to import.

Step 9: Create a package with the name in the folder **src/main/java.**

Step 10: Create a model class in the package**.** We have created a model class with the name**.** In the class, we have done the following:

- Define variable
- Generate Getter and Setter.

  Right-click on the file -> Source -> Generate Getters and Setters.

- Mark the class as an **Entity** by using the annotation **@Entity.**
- Mark the class as **Table** name by using the annotation **@Table.**
- Define each variable as **Column** by using the annotation **@Column.**

Step 11: Create a package with the name in the folder **src/main/java.**

Step 12: Create a Controller class in the package **.** We have created a controller class with the name **.** In the class, we have done the following:

- Mark the class as **RestController** by using the annotation **@RestController.**
- Autowire the class by using the annotation **@Autowired**.
- Define the following methods:
  - **getAllStudent():** It returns a List of all Student.
  - **getStudent():** It returns a student detail that we have specified in the path variable. We have passed id as an argument by using the annotation @PathVariable. The annotation indicates that a method parameter should be bound to a URI template variable.
  - **deleteStudent():** It deletes a specific Student that we have specified in the path variable.

- **saveStudent():** It saves the student detail. The annotation @RequestBody indicates that a method parameter should be bound to the body of the web request.
- **update():** The method updates a record. We must specify the record in the body, which we want to update. To achieve the same, we have used the annotation @RequestBody.

Step 13: Create a package with the name in the folder **src/main/java.**

Step 14: Create a **Service** class. We have created a service class with the name in the package **.**

Step 15: Create a package with the name in the folder **src/main/java.**

Step 16: Create a **Repository** interface. We have created a repository interface with the name in the package **.** It extends the **Crud Repository** interface.

Now we will configure the datasource **URL, driver class name, username,** and **password,** in the **application.properties** file.

Step 17: Open the **application.properties** file and configure the following properties.

application.properties

Now we will run the application.

Step 18: Open **SpringBootCrudOperationApplication.java** file and run it as Java Application.

SpringBootCrudOperationApplication.java

Step 19: Click on the **Student** table and then click on the **Run** button. The table shows the data that we have inserted in the body.

MODULE NAME: Library and Course

MODULE DONE BY: A.Priyadharshini - EBEON0FWL562309

Step 1: Open Spring Initializr http://start.spring.io.

Step 2: Select the Spring Boot version 2.3.0.M1.

Step 3: Provide the Group name. We have provided

Step 4: Provide the Artifact Id. We have provided spring-boot-crud-operation.

Step 5: Add the dependencies Spring Web, Spring Data JPA, and H2 Database.

Step 6: Click on the Generate button. When we click on the Generate button, it wraps the specifications in a Jar file and downloads it to the local system.

Step 7: Extract the Jar file and paste it into the STS workspace.

Step 8: Import the project folder into STS.

File -> Import -> Existing Maven Projects -> Browse -> Select the folder spring-boot-crud-operation -> Finish

It takes some time to import.

Step 9: Create a package with the name in the folder **src/main/java.**

Step 10: Create a model class in the package**.** We have created a model class with the name**.** In the class, we have done the following:

- Define variable
- Generate Getter and Setter.

Right-click on the file -> Source -> Generate Getters and Setters.

- Mark the class as an **Entity** by using the annotation **@Entity.**
- Mark the class as **Table** name by using the annotation **@Table.**
- Define each variable as **Column** by using the annotation **@Column.**

Step 11: Create a package with the name in the folder **src/main/java.**

Step 12: Create a Controller class in the package **.** We have created a controller class with the name **.** In the class, we have done the following:

- Mark the class as **RestController** by using the annotation **@RestController.**
- Autowire the class by using the annotation **@Autowired**.
- Define the following methods:
    - **getAllStudent():** It returns a List of all Student.
    - **getStudent():** It returns a student detail that we have specified in the path variable. We have passed id as an argument by using the annotation @PathVariable. The annotation indicates that a method parameter should be bound to a URI template variable.
    - **deleteStudent():** It deletes a specific Student that we have specified in the path variable.
    - **saveStudent():** It saves the student detail. The annotation @RequestBody indicates that a method parameter should be bound to the body of the web request.
    - **update():** The method updates a record. We must specify the record in the body, which we want to update. To achieve the same, we have used the annotation @RequestBody.

Step 13: Create a package with the name in the folder **src/main/java.**

Step 14: Create a **Service** class. We have created a service class with the name in the package **.**

Step 15: Create a package with the name in the folder **src/main/java.**

Step 16: Create a **Repository** interface. We have created a repository interface with the name in the package **.** It extends the **Crud Repository** interface.

Now we will configure the datasource **URL, driver class name, username,** and **password,** in the **application.properties** file.

Step 17: Open the **application.properties** file and configure the following properties.

application.properties

Now we will run the application.

Step 18: Open **SpringBootCrudOperationApplication.java** file and run it as Java Application.

SpringBootCrudOperationApplication.java

Step 19: Click on the **Student** table and then click on the **Run** button. The table shows the data that we have inserted in the body.