UserFeedbackDocumentaion

About: As a part of developing an ecommerce web application, a REST resource is needed to capture user feedback. Feedback data will be received from third-party apps and websites. The data will be sent to the REST API which will collect feedback from various sources.

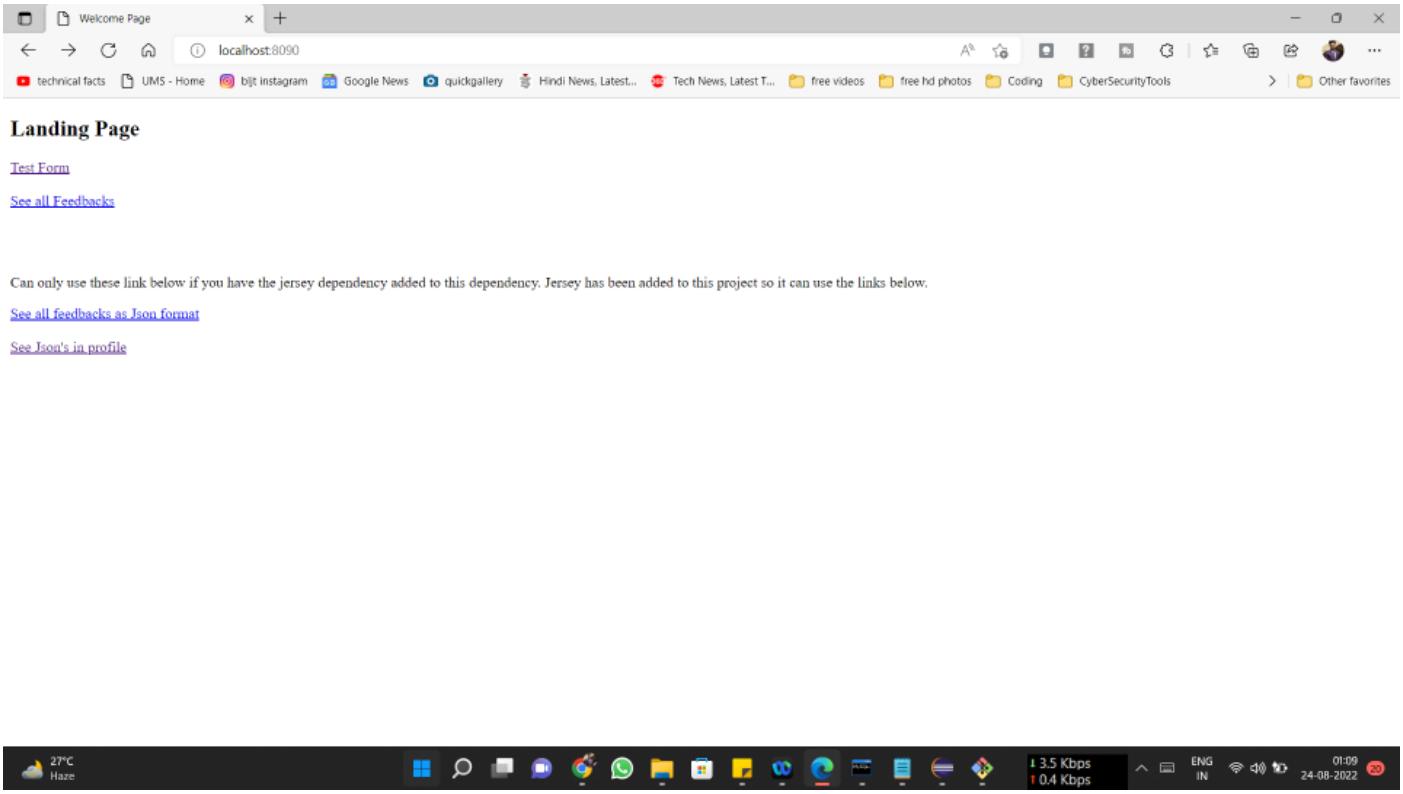
Installation Guide:

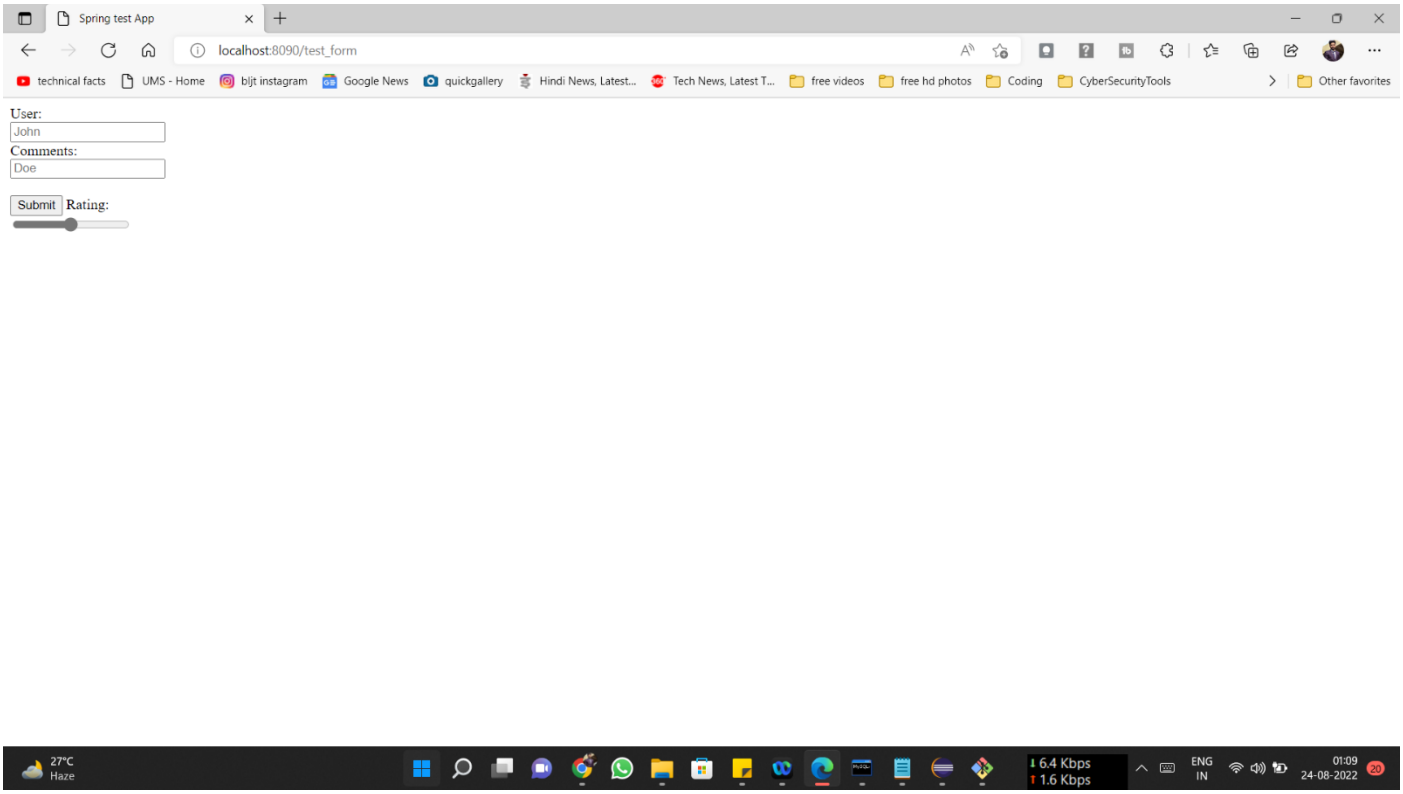
1. GitHub link: https://github.com/baljeet-singh97/JAVA-Projects/tree/main/Phase%203/Feedback

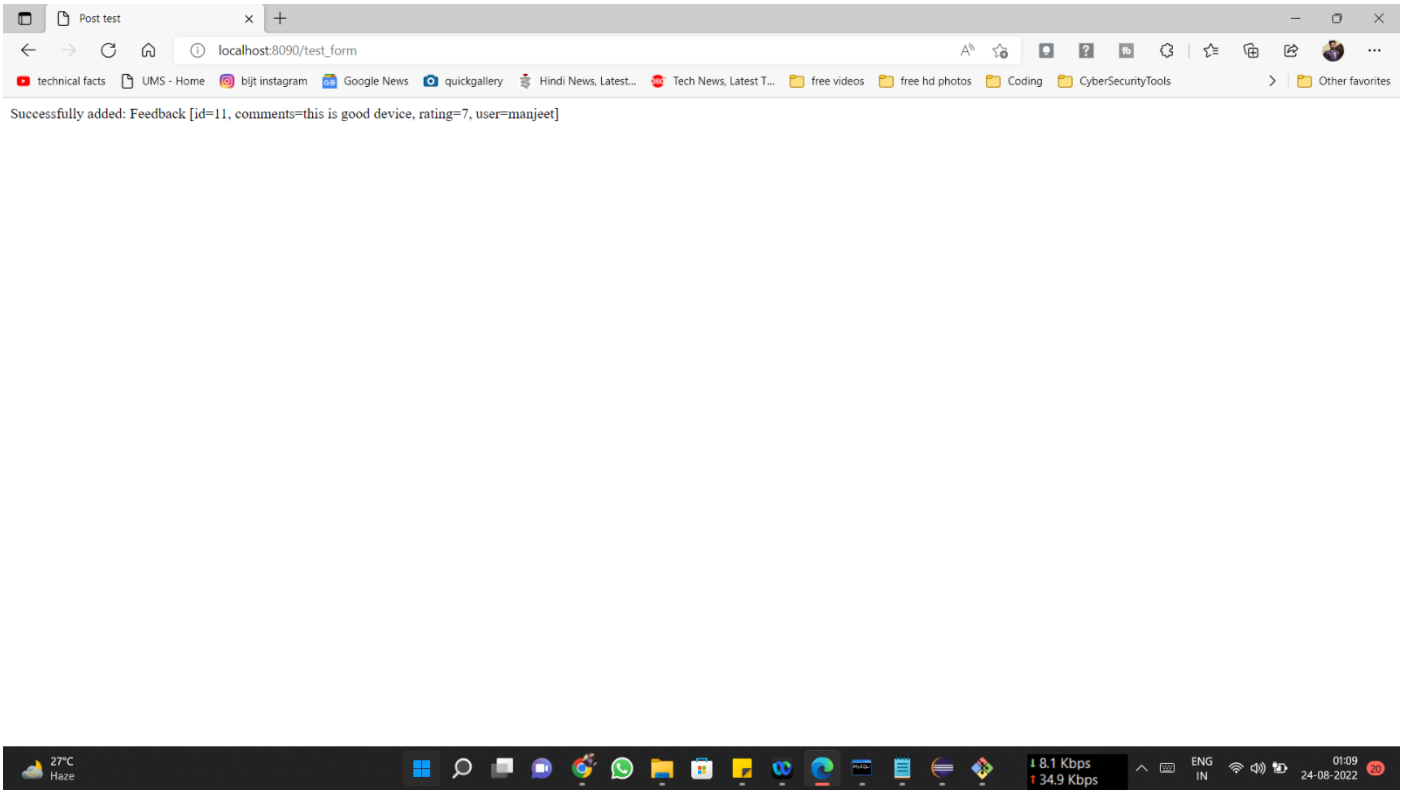
2. Download the entire project as Zip in local system.

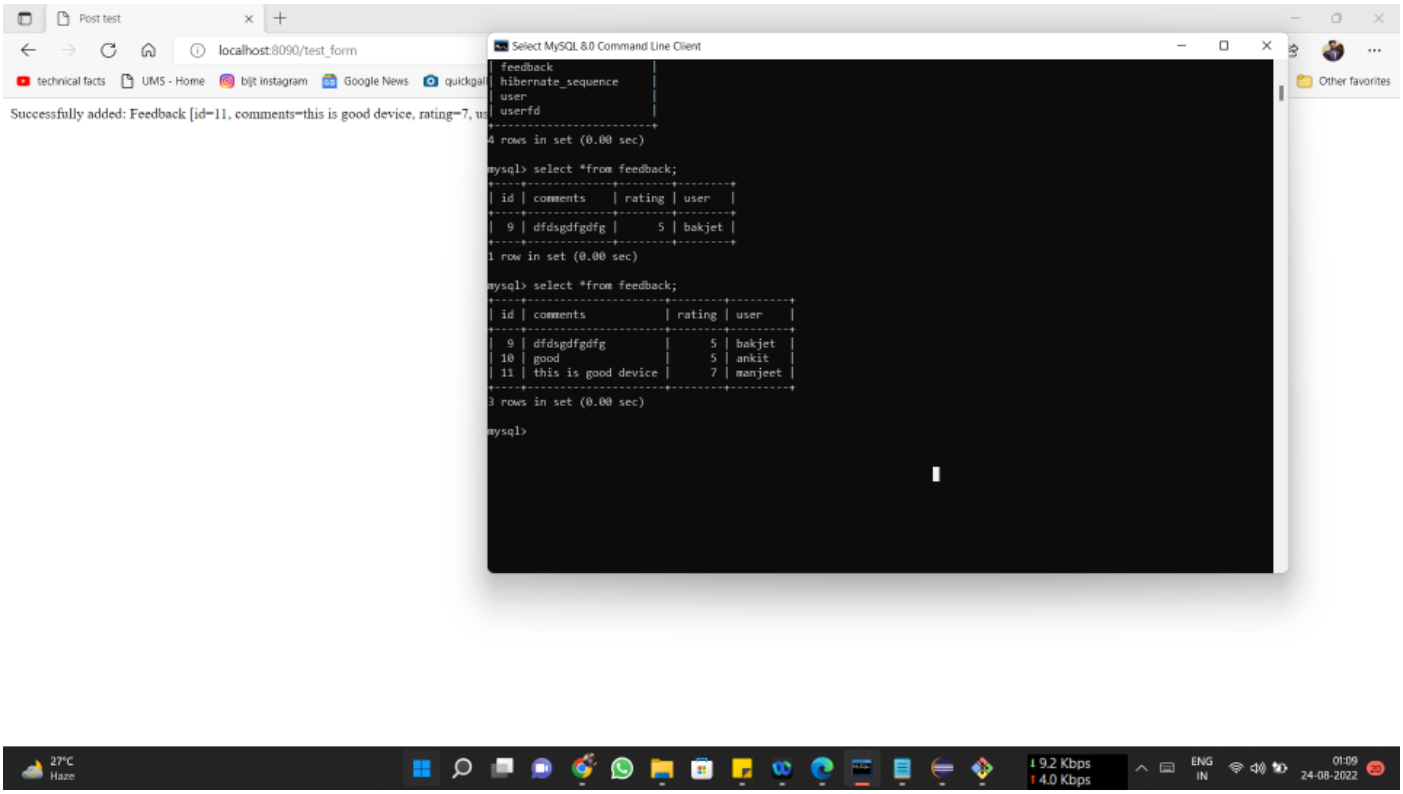
3. import the project in Eclipse IDE

How to use:




Spring test App
localhost:8090/test_form
technical facts
UMS - Home
bljt instagram
Google News
quickgallery
Hindi News, Latest...
Tech News, Latest T...
free videos
free hd photos
Coding
CyberSecurityTools
Other favorites
User:
John
Comments:
Doe
Submit
Rating:
27°C
Haze
6.4 Kbps
1.6 Kbps
ENG
IN
01:09
24-08-2022


Post test
localhost:8090/test_form
technical facts
UMS - Home
bljt instagram
Google News
quickgallery
Hindi News, Latest...
Tech News, Latest T...
free videos
free hd photos
Coding
CyberSecurityTools
Other favorites
Successfully added: Feedback [id=11, comments=this is good device, rating=7, user=manjeet]
27°C
Haze
8.1 Kbps
34.9 Kbps
ENG
IN
01:09
24-08-2022



Code Description

TestFormController.java

The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. The @Autowired annotation can be used to autowire bean on the setter method just like @Required annotation, constructor, a property or methods with arbitrary names and/or multiple arguments.

Autowiring the the FeedbackService class here

```
@Autowired
        FeedbackService feedbackService;
```

Getmapping is a Spring notation and is widely used in mapping HTTP GET requests onto some specific handler methods. Returning to the testformjsp page to get all the data from user.

```
@GetMapping("/test_form")
        public String showTestForm(ModelMap model) {
                model.addAttribute("test", new Feedback());
```

```
		return "testformjsp";
	}
```

After user enter all the details using post method data will be redirected to here, adding data into the database and returning the post.jsp page.

```
@PostMapping("/test_form")
	public String submitTestForm(@ModelAttribute("testUser") Feedback fb, ModelMap m) {
		feedbackService.addNewFeedback(fb);
		m.addAttribute("test", fb);
			return "post";
	}
```

Feedback.java

Defined a feedback.java , here defined all the valid fields and getter setters and to string method so that we can save the data into database as an object.

```
public class Feedback {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	@Column(name="id")
	@NotNull
	private Integer id;

	@Column(name="comments")
	private String comments;

	@Column(name="rating")
	@NotNull
	private int rating;

	@Column(name="user")
	private String user;
```

```
	public Feedback() {
		super();
	}

	public Feedback(String comments, Integer rating, String user) {
		this.comments = comments;
		this.rating = rating;
		this.user = user;
	}

	/*
	 * Needed the setters and getters to be able to add name and comments otherwise
	 * they are nulls when entering the SQL DB
	 */
	public String getComments() {
		return comments;
	}

	public void setComments(String comments) {
		this.comments = comments;
	}
```

Feedbackrepository.java

Spring Repository annotation is a specialization of @Component annotation, so Spring Repository classes are autodetected by spring framework through classpath scanning. Spring Repository is very close to DAO pattern where DAO classes are responsible for providing CRUD operations on database tables.

```
@Repository
public interface FeedbackRepository extends CrudRepository<Feedback, Integer> {

	public Feedback findByUser(String feedback);
}
```

Feedbackservice.java

Spring @Service annotation is used with classes that provide some business functionalities. Spring context will autodetect these classes when annotation-based configuration and classpath scanning is used.

The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. The @Autowired annotation can be used to autowire bean on the setter method just like @Required annotation, constructor, a property or methods with arbitrary names and/or multiple arguments.

Autowiring the the Feedbackrepository class here

```
@Autowired
	FeedbackRepository feedbackRepo;
```

Using this method to get the all the data from the database and show it to the user

```
public Iterable<Feedback> GetAllFeedback() {
		return feedbackRepo.findAll();
	}
```

Using this method to save the feedback to the database request will be coming from controller

```
public Feedback addNewFeedback(Feedback fb) {
		return feedbackRepo.save(fb);
	}
```

Application.properties

Configured the database settings in here.

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/userfeedback
spring.datasource.username=root
spring.datasource.password=978213
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL55Dialect
server.port=8090
```

```
logging.level.org.springframework.web: DEBUG
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```