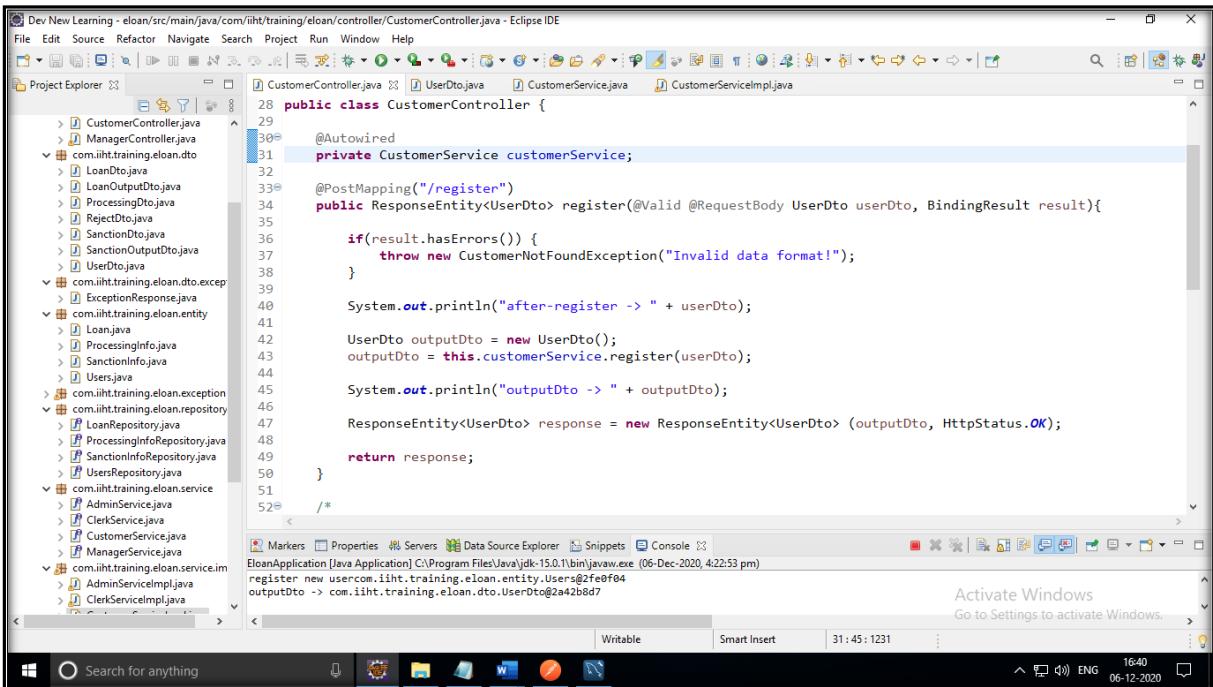


Jeevan Gajawada SBA2 Project Explanation

1) CustomerController:

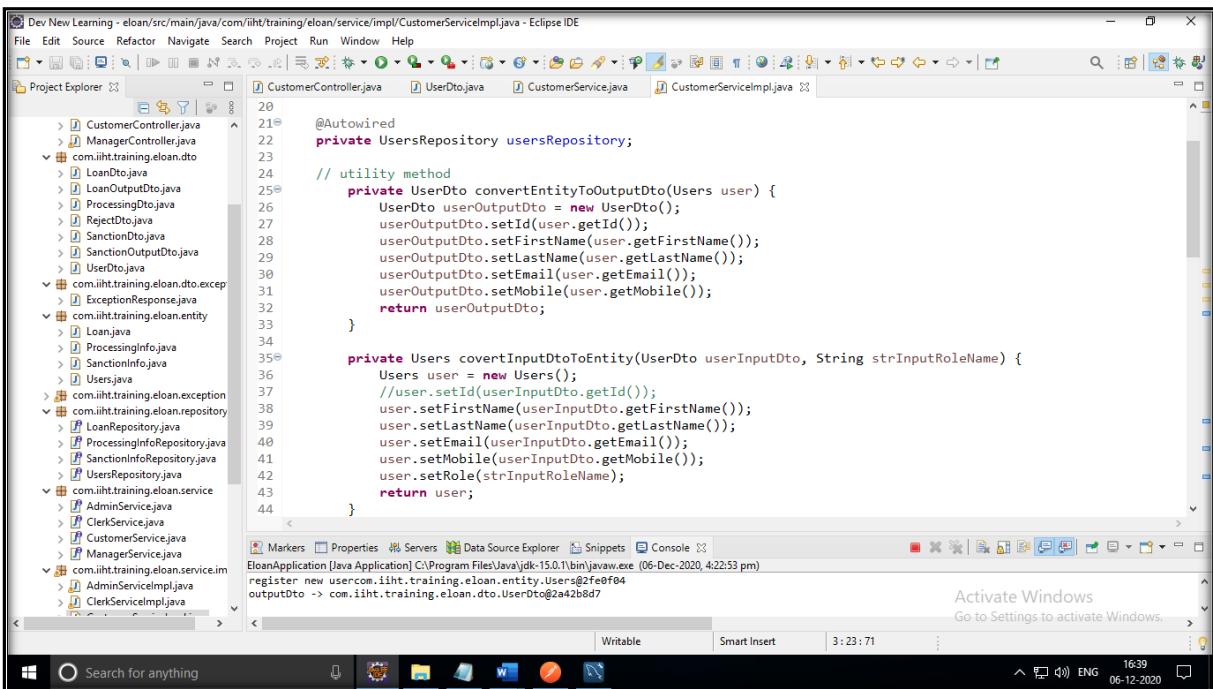
Let's start with 'CustomerController' module it has the below 4 URL's and below is the code and output execution for the same.

Register functionality code and output execution



The screenshot shows the Eclipse IDE interface with the 'CustomerController.java' file open in the editor. The code implements a controller class with a register method that uses a POST mapping to handle user registration. The code includes validation logic, service layer interaction, and response handling. The Eclipse interface includes a Project Explorer, a code editor, and a Console tab showing the output of a run command.

```
public class CustomerController {  
    @Autowired  
    private CustomerService customerService;  
  
    @PostMapping("/register")  
    public ResponseEntity<UserDto> register(@Valid @RequestBody UserDto userDto, BindingResult result){  
        if(result.hasErrors()) {  
            throw new CustomerNotFoundException("Invalid data format!");  
        }  
        System.out.println("after-register -> " + userDto);  
        UserDto outputDto = new UserDto();  
        outputDto = this.customerService.register(userDto);  
        System.out.println("outputDto -> " + outputDto);  
        ResponseEntity<UserDto> response = new ResponseEntity<UserDto>(outputDto, HttpStatus.OK);  
        return response;  
    }  
}
```



The screenshot shows the Eclipse IDE interface with the 'CustomerServiceImpl.java' file open in the editor. This implementation class contains utility methods for converting between entity and DTO objects. It uses autowiring to inject the 'UsersRepository'. The code demonstrates how to map a User entity to a UserOutputDto and vice versa. The Eclipse interface includes a Project Explorer, a code editor, and a Console tab showing the output of a run command.

```
@Autowired  
private UsersRepository usersRepository;  
  
// utility method  
private UserDto convertEntityToOutputDto(Users user) {  
    UserDto userOutputDto = new UserDto();  
    userOutputDto.setId(user.getId());  
    userOutputDto.setFirstName(user.getFirstName());  
    userOutputDto.setLastName(user.getLastName());  
    userOutputDto.setEmail(user.getEmail());  
    userOutputDto.setMobile(user.getMobile());  
    return userOutputDto;  
}  
  
private Users convertInputDtoToEntity(UserDto userInputDto, String strInputRoleName) {  
    Users user = new Users();  
    //user.setId(userInputDto.getId());  
    user.setFirstName(userInputDto.getFirstName());  
    user.setLastName(userInputDto.getLastName());  
    user.setEmail(userInputDto.getEmail());  
    user.setMobile(userInputDto.getMobile());  
    user.setRole(strInputRoleName);  
    return user;  
}
```

```

56     * @Override public UserDto register(UserDto userDto) { // TODO Auto-generated
57     * method stub return null; }
58     */
59
60     @Override
61     public UserDto register(UserDto userDto) {
62         // convert dto into entity
63         Users registeruser = this.covertInputDtoToEntity(userDto, "Customer");
64         // save entity in DB : returns the copy of newly added record back
65
66         System.out.println("register user" + registeruser);
67
68         Users newregisteruser = this.usersRepository.save(registeruser);
69         // convert entity into output dto
70
71         System.out.println("register new user" + newregisteruser);
72
73         UserDto userOutputDto = this.convertEntityToOutputDto(newregisteruser);
74         return userOutputDto;
75     }
76
77     @Override
78     public LoanOutputDto applyLoan(Long customerId, LoanDto loanDto) {
79         // TODO Auto-generated method stub
80         return null;
81     }

```

Markers Properties Servers Data Source Explorer Snippets Console

ElloanApplication [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (06-Dec-2020, 4:22:53 pm)

register new user com.iith.training.eloan.entity.Users@fe0f04
outputDto -> com.iith.training.eloan.dto.UserDto@2a42bd7

Activate Windows Go to Settings to activate Windows.

Postman execution and inserted 3 values into the DB

POST http://localhost:9090/eloon-app/customer/register

Params Authorization Headers (8) Body Body (1) Pre-request Script Tests Settings Cookies Code

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

```

1 {
2   "email": "jeevan@email.com",
3   "firstName": "jeevan2",
4   "lastName": "Gajawada",
5   "mobile": "7799111012",
6   "role": "Customer"
7 }

```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 81 ms Size: 267 B Save Response

Activate Windows Go to Settings to activate Windows.

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Query' tab is active, displaying a query window with the following content:

```

Query 1 x
1 • Use eloanbdb;
2
3 • Select * from users;

```

The results grid shows the following data:

	id	email	first_name	last_name	mobile	role
1	1	jeevan@email.com	Jeevan	Gejwada	7799121010	Customer
2	2	jeevan1@email.com	Jeevan1	Gejwada1	7799121011	Customer
3	3	jeevan2@email.com	Jeevan2	Gejwada2	7799121012	Customer

The 'Output' pane shows the execution history:

- 2 16:24:09 Select * from users LIMIT 0, 1000
- 3 16:30:42 Select * from users LIMIT 0, 1000
- 4 16:31:16 Select * from users LIMIT 0, 1000

The status bar at the bottom right indicates: Duration / Fetch 0.000 sec / 0.000 sec, Go to Settings to activate Windows 0.000 sec / 0.000 sec, and 0.000 sec / 0.000 sec.

Apply Loan functionality code and output execution

The screenshot shows the Eclipse IDE interface with the project 'Dev New Learning - eloan' open. The 'CustomerController.java' file is the active editor, displaying the following code:

```

40     UserDto outputDto = new UserDto();
41     outputDto = this.customerService.register(userDto);
42     ResponseEntity<UserDto> response = new ResponseEntity<UserDto>(outputDto, HttpStatus.OK);
43     return response;
44 }
45
46 @PostMapping("/apply-loan/{customerId}")
47 public ResponseEntity<LoanOutputDto> applyLoan(@PathVariable Long customerId,
48                                                 @RequestBody LoanDto loanDto, BindingResult result){
49     if(result.hasErrors()){
50         throw new CustomerNotFoundException("Invalid data format! Failed to Apply for Loan");
51     }
52
53     LoanOutputDto outputDto = new LoanOutputDto();
54     outputDto = this.customerService.applyLoan(customerId, loanDto);
55     ResponseEntity<LoanOutputDto> response = new ResponseEntity<LoanOutputDto>(outputDto, HttpStatus.OK);
56
57     return response;
58 }
59
60 @GetMapping("/loan-status/{loanAppId}")
61 public ResponseEntity<LoanOutputDto> getStatus(@PathVariable Long loanAppId{
62     return null;
63 }
64

```

The 'Markers' view shows an error: 'ElloanApplication [Java Application] 2020-12-06 18:12:28.952 INFO 10464 --- [nio-9090-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms'. The status bar at the bottom right indicates: Duration / Fetch 0.000 sec / 0.000 sec, Go to Settings to activate Windows 0.000 sec / 0.000 sec, and 0.000 sec / 0.000 sec.

The screenshot shows the Eclipse IDE interface with two files open: `CustomerController.java` and `CustomerServiceImpl.java`. The code implements utility methods for `LoanDto` and `Loan`, including conversion between input and output DTOs and entity models.

```
78 // utility methods for Loan Dto
79     private LoanOutputDto convertEntityToOutputDto(Loan loan) {
80         LoanOutputDto loanOutputDto = new LoanOutputDto();
81         loanOutputDto.setCustomerId(loan.getCustomerId());
82         loanOutputDto.setLoanAppId(loan.getId());
83         loanOutputDto.setStatus("Applied");
84         loanOutputDto.setRemark(loan.getRemark());
85         return loanOutputDto;
86     }
87
88
89     private Loan convertInputDtoToEntityLoan(LoanDto loanInputDto, Long customerId) {
90         Loan loan = new Loan();
91         loan.setCustomerId(customerId);
92         loan.setLoanName(loanInputDto.getLoanName());
93         loan.setLoanAmount(loanInputDto.getLoanAmount());
94         loan.setLoanApplicationDate(loanInputDto.getLoanApplicationDate());
95         loan.setBusinessStructure(loanInputDto.getBusinessStructure());
96         loan.setBillingIndicator(loanInputDto.getBillingIndicator());
97         loan.setTaxIndicator(loanInputDto.getTaxIndicator());
98         loan.setStatus(0);
99         loan.setRemark("All Good"); // Giving comment as as 'All Good' as he is applying for loan newly
100    return loan;
101}
102
```

Registered multiple loans with same and different customer id's

The screenshot shows the Postman application interface. The top navigation bar includes 'File', 'Edit', 'View', 'Help', 'New', 'Import', 'Runner', and a search bar labeled 'Filter'. The main header says 'My Workspace' with an 'Invite' button. On the left, there's a sidebar with 'History' (selected), 'Collections', 'APIs', and a toggle for 'Save Responses' with 'Clear all' below it. Below this is a list of recent API requests:

- POST http://localhost:9090/eloan-app/customer/apply-loan/789124
- POST http://localhost:9090/eloan-app/customer/apply-loan/789124
- POST http://localhost:9090/eloan-app/customer/apply-loan/789124
- POST http://localhost:9090/eloan-app/customer/apply-loan/789123
- POST http://localhost:9090/eloan-app/customer/apply-loan/789123
- POST http://localhost:9090/eloan-app/customer/register
- POST http://localhost:9090/eloan-app/customer/register
- POST http://localhost:9090/eloan-app/customer/register
- POST http://localhost:9060/eloan-app/customer

The main workspace shows an 'Untitled Request' with a 'POST' method to 'http://localhost:9090/eloan-app/customer/apply-loan/789124'. The 'Body' tab is selected, showing a JSON payload:

```
1 "customerId": "789124",
2 "loanName": "Second House Loan",
3 "loanAmount": "20000",
4 "loanApplicationDate": "06-12-2020",
5 "businessStructure": "Individual",
6 "billingIndicator": "Salaried",
7 "taxIndicator": "Tax Payer"
```

The response tab shows a successful status: 'Status: 200 OK'.

DB Table with inserted values

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the schema 'eloin_jeevanganjawa', there are two tables: 'eloanregistrationpage' and 'loan'. The 'loan' table has columns: id, billing_indicator, business_structure, customer_id, loan_amount, loan_application_date, loan_name, remark, status, and tax_indicator. A query window titled 'Query 1' contains the following SQL code:

```

1 • Use eloin_jeevanganjawa;
2
3 • Select * from loan;

```

The Result Grid displays the following data:

	id	billing_indicator	business_structure	customer_id	loan_amount	loan_application_date	loan_name	remark	status	tax_indicator
1	Salaried	Individual	781123	25000	06-12-2020	Home Loan	All Good	0	Tax Payer	
2	Salaried	Individual	789124	25000	06-12-2020	Home Loan	All Good	0	Tax Payer	
3	Salaried	Individual	789124	26000	06-12-2020	Car Loan	All Good	0	Tax Payer	
4	Salaried	Individual	789124	27000	06-12-2020	Second House Loan	All Good	0	Tax Payer	

Loan Applied getStatus code with postman execution

The screenshot shows the Eclipse IDE interface. The left side shows the Project Explorer with several Java packages and files. The right side shows the code editor for 'CustomerController.java'.

```

46     @PostMapping("/apply-loan/{customerId}")
47     public ResponseEntity<LoanOutputDto> applyLoan(@PathVariable long customerId,
48                                                       @RequestBody LoanDto loanDto, BindingResult result){
49         if(result.hasErrors()) {
50             throw new CustomerNotFoundException("Invalid data format! Failed to Apply for Loan");
51         }
52
53         LoanOutputDto outputDto = new LoanOutputDto();
54         outputDto = this.customerService.applyLoan(customerId, loanDto);
55         ResponseEntity<LoanOutputDto> response = new ResponseEntity<LoanOutputDto>(outputDto, HttpStatus.OK);
56
57         return response;
58     }
59
60     @GetMapping("/loan-status/{loanAppId}")
61     public ResponseEntity<LoanOutputDto> getStatus(@PathVariable Long loanAppId){
62
63         LoanOutputDto loanOutputDto = this.customerService.getStatus(loanAppId);
64
65         ResponseEntity<LoanOutputDto> response =
66             new ResponseEntity<LoanOutputDto>(loanOutputDto, HttpStatus.OK);
67
68         return response;
69     }
70

```

The status bar at the bottom indicates the date and time: 2020-12-06 18:30:31.880 INFO 10464 --- [nio-9090-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the project structure for "bootapp-restful-crud".
- Code Editor:** Shows the Java code for `CustomerServiceImpl.java`. The code handles saving new loans, converting entities to output DTOs, and getting loan statuses.
- Console:** Shows log messages from the application startup and database queries.
- Bottom Bar:** Includes the Windows taskbar with icons for File Explorer, Task View, Start, Taskbar settings, and system status.

```
106     // save entity in DB : returns the copy of newly added record back
107     Loan newregisterLoan = this.loanRepository.save(registerloan);
108
109     // convert entity into output dto
110     LoanOutputDto loanOutputDto = this.convertEntityToOutputDtoLoan(newregisterloan);
111     return loanOutputDto;
112 }
113
114 @Override
115 public LoanOutputDto getStatus(Long loanAppId) {
116     // fetch record from DB
117     //Loan loan = this.loanRepository.findById(loanAppId).orElse(null);
118     Loan loan = this.loanRepository.findById(loanAppId).orElse(null);
119     // convert entity into output dto
120     LoanOutputDto loanOutputDto = this.convertEntityToOutputDtoLoan(loan);
121     return loanOutputDto;
122 }
123
124 @Override
125 public List<LoanOutputDto> getStatusAll(Long customerId) {
126     // TODO Auto-generated method stub
127     return null;
128 }
129
130 }
```

The screenshot shows the Postman application interface with the following details:

- Left Sidebar:** History tab is selected, showing a list of recent API calls.
- Request Panel:** An "Untitled Request" is being configured with a GET method to `http://localhost:9090/eloan-app/customer/loan-status/4`.
- Body Tab:** Contains the JSON response body.
- Bottom Status:** Status: 200 OK, Time: 54 ms, Size: 314 B.
- Bottom Bar:** Includes the Windows taskbar with icons for File Explorer, Task View, Start, Taskbar settings, and system status.

```
1
2   "customerId": 789124,
3   "loanAppId": 4,
4   "userDto": null,
5   "loanDto": null,
6   "processingDto": null,
7   "sanctionOutputDto": null,
8   "status": "Applied",
9   "remark": "All Good"
10 }
```

Raw Input JSON Data for the first 3 URL's

```
SBA-2 Notes - Notepad
File Edit Format View Help
Service Call URL:|
-----
http://localhost:9090/eloan-app

CustomerController:
-----
http://localhost:9090/eloan-app/customer/register
http://localhost:9090/eloan-app/customer/apply-loan/{customerId}
http://localhost:9090/eloan-app/customer/loan-status/{loanAppId}
/customer/loan-status-all/{customerId}

Example1-register:
-----
{
  "email" : "jeevan@email.com",
  "firstName" : "Jeevan",
  "lastName" : "Gajawada",
  "mobile" : "7799121010",
  "role" : "Customer"
}

Example2-apply-loan:
-----
{
  "customerId" : "789123",
  "loanName" : "Home Loan",
  "loanAmount" : "25000",
  "loanApplicationDate" : "06-12-2020",
  "businessStructure" : "Individual",
  "billingIndicator" : "Salaried",
  "taxIndicator" : "Tax Payer"
}

Example3-getStatus:
-----
http://localhost:9090/eloan-app/customer/loan-status/4

Activate Windows
Go to Settings to activate Windows.
```

Loan Applied getStatusAll code with postman execution

```
Dev New Learning - eloan/src/main/java/com/ihtt/training/eloan/controller/CustomerController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CustomerController.java CustomerServiceImpl.java EmployeeController.java EmployeeServiceImpl.java
69 }
70
71 @GetMapping("/loan-status-all/{customerId}")
72 public ResponseEntity<List<LoanOutputDto>> getStatusAll(@PathVariable Long customerId){
73     // call the service layer method
74     List<LoanOutputDto> loanOutputDtos = this.customerService.getStatusAll(customerId);
75
76     ResponseEntity<List<LoanOutputDto>> response =
77         new ResponseEntity<List<LoanOutputDto>>(loanOutputDtos, HttpStatus.OK);
78
79     return response;
80 }
81
82 @ExceptionHandler(CustomerNotFoundException.class)
83 public ResponseEntity<ExceptionResponse> handler(CustomerNotFoundException ex){
84     ExceptionResponse exception =
85         new ExceptionResponse(ex.getMessage(),
86             System.currentTimeMillis(),
87             HttpStatus.NOT_FOUND.value());
88     ResponseEntity<ExceptionResponse> response =
89         new ResponseEntity<ExceptionResponse>(exception, HttpStatus.NOT_FOUND);
90     return response;
91 }
92 }
93 <
```

The screenshot shows the Eclipse IDE interface with two files open: `CustomerController.java` and `CustomerServiceImpl.java`. The code is related to a loan application system, specifically handling loan status retrieval and processing.

```
CustomerController.java
117
118@ Override
119    public LoanOutputDto getStatus(Long loanAppId) {
120        // fetch record from DB
121        Loan loan = this.loanRepository.findById(loanAppId).orElse(null);
122        // convert entity into output dto
123        LoanOutputDto loanOutputDto = this.convertEntityToOutputDtoLoan(loan);
124        return loanOutputDto;
125    }
126
127@ Override
128    public List<LoanOutputDto> getStatusAll(Long customerId) {
129
130        List<Loan> loans = this.loanRepository.findAll();
131        List<LoanOutputDto> loanOutputDtos =
132            loans.stream()
133                .map(this :: convertEntityToOutputDtoLoan)
134                .collect(Collectors.toList());
135
136        return loanOutputDtos;
137    }
138
139
140}
141
```

```
CustomerServiceImpl.java
117
118@ Override
119    public LoanOutputDto getStatus(Long loanAppId) {
120        // fetch record from DB
121        Loan loan = this.loanRepository.findById(loanAppId).orElse(null);
122        // convert entity into output dto
123        LoanOutputDto loanOutputDto = this.convertEntityToOutputDtoLoan(loan);
124        return loanOutputDto;
125    }
126
127@ Override
128    public List<LoanOutputDto> getStatusAll(Long customerId) {
129
130        List<Loan> loans = this.loanRepository.findAll();
131        List<LoanOutputDto> loanOutputDtos =
132            loans.stream()
133                .map(this :: convertEntityToOutputDtoLoan)
134                .collect(Collectors.toList());
135
136        return loanOutputDtos;
137    }
138
139
140}
141
```

Below the code editor, the Eclipse status bar shows the date and time: 06-12-2020 18:49. The taskbar at the bottom of the screen also displays the date and time: 06-12-2020 18:49.

The screenshot shows the Postman application interface. A collection named "Untitled Request" is displayed, containing several API requests (GET and POST) to the local host port 9090. One specific request is highlighted: a GET request to `http://localhost:9090/eloan-app/customer/loan-status-all/789124`. The response is shown in a JSONpretty tab, displaying the following data:

```
1 [
2   {
3     "customerId": 781123,
4     "loanAppId": 1,
5     "userDto": null,
6     "loanDto": null,
7     "processingDto": null,
8     "sanctionOutputDto": null,
9     "status": "Applied",
10    "remark": "All Good"
11  },
12  {
13    "customerId": 789124,
14    "loanAppId": 2,
15    ...
16  }
]
```

Postman's status bar at the bottom indicates the date and time: 06-12-2020 18:50. The taskbar at the bottom of the screen also displays the date and time: 06-12-2020 18:50.

Postman

History Collections APIs

Untitled Request

GET http://localhost:9090/eloan-app/customer/loan-status-all/789124

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

KEY	VALUE	DESCRIPTION
customer_id	789124	
loan_app_id	3	
user_dto	null	
loan_dto	null	
processing_dto	null	
sancion_output_dto	null	
status	"Applied"	
remark	"All Good"	

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 215 ms Size: 769 B Save Response

Activate Windows Go to Settings to activate Windows.

Find and Replace Console

Build Browse

Search for anything

Windows Taskbar: Search for anything, Start button, File Explorer, Task View, Taskbar icons, Language: ENG, Date: 06-12-2020, Time: 18:50

The Postman data result matches with the DB

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

eloan_jeevangajawada

- Tables
 - eloanregistrationpage
 - loaninformation
 - loan
- Views
- Stored Procedures
- Functions

eloanb

- Tables
 - loan

Administration Schemas

Information

Schema: eloin_jeevangajawada

Query 1

```

1 • Use eloanb;
2
3 • Select * from loan where customer_id = '789124';
  
```

Result Grid

	id	billing_indicator	business_structure	customer_id	loan_amount	loan_application_date	loan_name	remark	status	tax_indicator
1	2	Salaried	Individual	789124	25000	06-12-2020	Home Loan	All Good	0	Tax Payer
2	3	Salaried	Individual	789124	26000	06-12-2020	Car Loan	All Good	0	Tax Payer
3	4	Salaried	Individual	789124	27000	06-12-2020	Second House Loan	All Good	0	Tax Payer

Activate Windows Go to Settings to activate Windows.

Object Info Session loan 13 X

Search for anything

Windows Taskbar: Search for anything, Start button, File Explorer, Task View, Taskbar icons, Language: ENG, Date: 06-12-2020, Time: 18:50

Console Success output for the CustomerController

The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the log for the 'CustomerServiceImpl.java' file. The log includes Spring Boot startup messages, database configuration, and specific log entries for the 'CustomerController'. One notable entry is 'Hibernate: select loan0_.id as id1_0..., loan0_.loan_amount as loan...'. The Eclipse interface includes toolbars, menus, and a status bar at the bottom.

```

Dev New Learning - elan/src/main/java/com/ihi/training/elan/service/impl/CustomerServiceImpl.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Markers Properties Servers Data Source Explorer Snippets Console
EloanApplication [Java Application]
{spring.web.resources.chain.cache=false, spring.web.resources.cache.period=0}

:: Spring Boot :: (v2.4.0)

2020-12-06 18:46:10.083 INFO 10464 --- [ restartedMain] c.ihi.training.elan.EloanApplication : Starting EloanApplication using Java 15.0.1 on wave2wf273 with PID 10464 (C:\Users\Hikari\IdeaProjects\eloa...
2020-12-06 18:46:10.083 INFO 10464 --- [ restartedMain] c.ihi.training.elan.EloanApplication : No active profile set, falling back to default profiles: default
2020-12-06 18:46:10.339 INFO 10464 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFERRED mode.
2020-12-06 18:46:10.398 INFO 10464 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 59 ms. Found 4 JPA repository interfaces.
2020-12-06 18:46:10.541 INFO 10464 --- [ restartedMain] o.s.b.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 9090 (http)
2020-12-06 18:46:10.542 INFO 10464 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-12-06 18:46:10.543 INFO 10464 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.39]
2020-12-06 18:46:10.565 INFO 10464 --- [ restartedMain] o.a.c.c.C.[localhost]./[eloa-app] : Initializing Spring embedded WebApplicationContext
2020-12-06 18:46:10.565 INFO 10464 --- [ restartedMain] w.s.c.WebServerApplicationContext : Root WebApplicationContext: initialization completed in 479 ms
2020-12-06 18:46:10.598 INFO 10464 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-12-06 18:46:10.618 INFO 10464 --- [ task-1] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2020-12-06 18:46:10.631 INFO 10464 --- [ task-1] com.zaxxer.hikaricp.HikariDataSource : HikariPool-25 - Starting...
2020-12-06 18:46:10.645 INFO 10464 --- [ task-1] com.zaxxer.hikaricp.HikariDataSource : HikariPool-25 - Start completed.
2020-12-06 18:46:10.646 INFO 10464 --- [ task-1] org.hibernate.dialect.Dialect : HHH000408: Using dialect: org.hibernate.dialect.MySQL5Dialect
2020-12-06 18:46:10.722 WARN 10464 --- [ restartedMain] o.h.e.t.j.p.JtaPlatformInitiator : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be slow in this scenario.
2020-12-06 18:46:10.759 INFO 10464 --- [ restartedMain] o.a.c.c.C.[localhost]./[eloa-app] : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be slow in this scenario.
2020-12-06 18:46:10.769 INFO 10464 --- [ restartedMain] o.a.c.c.C.[localhost]./[eloa-app] : HHH000400: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformImpl]
2020-12-06 18:46:10.794 INFO 10464 --- [ task-2] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2020-12-06 18:46:10.852 INFO 10464 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : LiveReload server is running on port 35729
2020-12-06 18:46:10.853 INFO 10464 --- [ restartedMain] DeferredRepositoryInitializationListener : Tomcat started on port(s): 9090 (http) with context path '/eloa-app'
2020-12-06 18:46:10.853 INFO 10464 --- [ restartedMain] DeferredRepositoryInitializationListener : Triggering deferred initialization of Spring Data repositories...
2020-12-06 18:46:10.939 INFO 10464 --- [ restartedMain] DeferredRepositoryInitializationListener : Spring Data repositories initialized!
2020-12-06 18:46:10.942 INFO 10464 --- [ restartedMain] c.ihi.training.elan.EloanApplication : Started EloanApplication in 0.882 seconds (JVM running for 3051.687)
2020-12-06 18:46:10.943 INFO 10464 --- [ restartedMain] [nio-9090-exec-2] o.a.c.c.C.[localhost]./[eloa-app] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-12-06 18:47:05.596 INFO 10464 --- [nio-9090-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2020-12-06 18:47:05.597 INFO 10464 --- [nio-9090-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
Hibernate: select loan0_.id as id1_0..., loan0_.loan_amount as loan...
Go to Settings to activate Windows

```

Raw Data use for the CustomerController

The screenshot shows a Microsoft Notepad window titled 'SBA-2 Notes - Notepad'. It contains several examples of raw data used for API requests to the 'CustomerController'. These examples include 'register' data, 'apply-loan' data, and URLs for 'getStatus' and 'getStatusAll' methods. The Notepad window has a standard Windows title bar and status bar.

```

SBA-2 Notes - Notepad
File Edit Format View Help
Service Call URL:
-----
http://localhost:9090/eloan-app

CustomerController:
-----
http://localhost:9090/eloan-app/customer/register
http://localhost:9090/eloan-app/customer/apply-loan/{customerId}
http://localhost:9090/eloan-app/customer/loan-status/{loanAppId}
http://localhost:9090/eloan-app/customer/loan-status-all/{customerId}

Example1-register:
-----
{
  "email" : "jeevan@email.com",
  "firstName" : "Jeevan",
  "lastName" : "Gajawada",
  "mobile" : "7799121010",
  "role" : "Customer"
}

Example2-apply-loan:
-----
{
  "customerId" : "789123",
  "loanName" : "Home Loan",
  "loanAmount" : "25000",
  "loanApplicationDate" : "06-12-2020",
  "businessStructure" : "Individual",
  "billingIndicator" : "Salaried",
  "taxIndicator" : "Tax Payer"
}

Example3-getStatus and Example4-getStatusAll:
-----
http://localhost:9090/eloan-app/customer/loan-status/4
http://localhost:9090/eloan-app/customer/loan-status-all/789124

```

2) AdminController:

register-clerk code and Postman execution

The screenshot shows the Eclipse IDE interface with the AdminController.java file open in the editor. The code implements a REST controller for managing users. It includes methods for registering a clerk and a manager. The controller uses an autowired AdminService to perform the actual registration. The service layer is responsible for saving the user entity into the database.

```
1 package com.iiht.training.eloan.controller;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/admin")
7 public class AdminController {
8
9     @Autowired
10    private AdminService adminService;
11
12    @PostMapping("/register-clerk")
13    public ResponseEntity<UserDto> registerClerk(@Valid @RequestBody UserDto userDto, BindingResult result) {
14        if(result.hasErrors()) {
15            throw new ClerkNotFoundException("Invalid data format! Cannot create Clerk");
16        }
17
18        UserDto outputDto = new UserDto();
19        outputDto = this.adminService.registerClerk(userDto);
20        ResponseEntity<UserDto> response = new ResponseEntity<UserDto>(outputDto, HttpStatus.OK);
21        return response;
22    }
23
24    @PostMapping("/register-manager")
25 }
```

The screenshot shows the Eclipse IDE interface with the AdminServiceImpl.java file open in the editor. This implementation converts input DTOs into domain entities before saving them to the database. It also handles the conversion of registered users back into output DTOs for the controller.

```
30    private Users convertInputDtoToEntity(UserDto userinputDto, String strInputRoleName) {
31        Users user = new Users();
32        user.setId(userinputDto.getId());
33        user.setFirstName(userinputDto.getFirstName());
34        user.setLastName(userinputDto.getLastName());
35        user.setEmail(userinputDto.getEmail());
36        user.setMobile(userinputDto.getMobile());
37        user.setRole(strInputRoleName);
38        return user;
39    }
40
41    @Override
42    public UserDto registerClerk(UserDto userDto) {
43        // convert dto into entity
44        Users registeruser = this.convertInputDtoToEntity(userDto, "Clerk");
45        // save entity in DB : returns the copy of newly added record back
46
47        Users newregisteruser = this.usersRepository.save(registeruser);
48        // convert entity into output dto
49
50        UserDto userOutputDto = this.convertEntityToOutputDto(newregisteruser);
51        return userOutputDto;
52    }
53
54    @Override
55 }
```

Postman execution is successful and the same is reflected in the DB

The screenshot shows the Postman application interface. In the center, there is a "POST" request to "http://localhost:9090/eloan-app/admin/register-clerk". The "Body" tab is selected, displaying the following JSON payload:

```
1 {
2   "email" : "clerk3@gmail.com",
3   "firstName" : "Clerk3",
4   "lastName" : "Lastname3",
5   "mobile" : "7799111113",
6   "role" : "Clerk"
7 }
```

Below the body, the response status is shown as "Status: 200 OK" with a response size of "265 B". At the bottom right of the main window, there is an "Activate Windows" message: "Go to Settings to activate Windows".

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database schema, including the "eloan_jeevangajawada" database and its tables like "loan" and "users".

In the center, a query editor window titled "Query 1" contains the following SQL code:

```
1 • Use eloandb;
2 • Select * from users where role = 'Customer';
```

Below the query editor is a "Result Grid" table showing the results of the query:

	id	email	first_name	last_name	mobile	role
1	1	jeevan@email.com	Jeevan	Gajawada	7799121010	Customer
2	2	jeevan1@email.com	Jeevan1	Gajawada1	7799121011	Customer
3	3	jeevan2@email.com	Jeevan2	Gajawada2	7799121012	Customer
*	NULL	NULL	NULL	NULL	NULL	NULL

At the bottom of the interface, the "Output" pane shows the execution history:

#	Time	Action	Message	Duration / Fetch
34	19:58:40	Select * from users LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
35	19:59:57	Select * from users where role = 'Customer' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
36	20:00:19	Use eloandb	0 row(s) affected	0.000 sec
37	20:00:19	Select * from users where role = 'Customer' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

register-manager code and Postman execution

Eclipse IDE - Dev New Learning - eloan/src/main/java/com/iith/training/eloan/controller/AdminController.java

```

37     UserDto outputDto = new UserDto();
38     outputDto = this.adminService.registerClerk(userDto);
39     ResponseEntity<UserDto> response = new ResponseEntity<UserDto>(outputDto, HttpStatus.OK);
40     return response;
41 }
42
43 @PostMapping("/register-manager")
44 public ResponseEntity<UserDto> registerManager(@Valid @RequestBody UserDto userDto, BindingResult result){
45
46     if(result.hasErrors()){
47         throw new ManagerNotFoundException("Invalid data format! Cannot create Manager");
48     }
49
50     UserDto outputDto = new UserDto();
51     outputDto = this.adminService.registerManager(userDto);
52     ResponseEntity<UserDto> response = new ResponseEntity<UserDto>(outputDto, HttpStatus.OK);
53     return response;
54 }
55
56 @GetMapping("/all-clerks")
57 public ResponseEntity<List<UserDto>> getAllClerks(){
58     return null;
59 }
60
61 @GetMapping("/all-managers")

```

Markers Properties Servers Data Source Explorer Snippets Console

Hibernate: insert into users (email, first_name, last_name, mobile, role) values (?, ?, ?, ?, ?)
Hibernate: insert into users (email, first_name, last_name, mobile, role) values (?, ?, ?, ?, ?)

Activate Windows Go to Settings to activate Windows.

Writable Smart Insert 25:31:1009

File Edit Source Refactor Navigate Search Project Run Window Help

Eclipse IDE - Dev New Learning - eloan/src/main/java/com/iith/training/eloan/service/impl/AdminServiceImpl.java

```

48     // convert entity into output dto
49
50     UserDto userOutputDto = this.convertEntityToOutputDto(newregisteruser);
51     return userOutputDto;
52 }
53
54 @Override
55 public UserDto registerManager(UserDto userDto) {
56     // convert dto into entity
57     Users registeruser = this.covertInputDtoToEntity(userDto, "Manager");
58     // save entity in DB : returns the copy of newly added record back
59
60     Users newregisteruser = this.usersRepository.save(registeruser);
61     // convert entity into output dto
62
63     UserDto userOutputDto = this.convertEntityToOutputDto(newregisteruser);
64     return userOutputDto;
65 }
66
67 @Override
68 public List<UserDto> getAllClerks() {
69     // TODO Auto-generated method stub
70     return null;
71 }
72

```

Markers Properties Servers Data Source Explorer Snippets Console

Hibernate: insert into users (email, first_name, last_name, mobile, role) values (?, ?, ?, ?, ?)
Hibernate: insert into users (email, first_name, last_name, mobile, role) values (?, ?, ?, ?, ?)

Activate Windows Go to Settings to activate Windows.

Writable Smart Insert 41:14:1351

File Edit Source Refactor Navigate Search Project Run Window Help

Postman execution is successful and the same is reflected in the DB

The screenshot displays two windows: Postman and MySQL Workbench.

Postman: The "My Workspace" tab is selected. An "Untitled Request" is shown with a POST method to `http://localhost:9090/eloan-app/admin/register-manager`. The "Body" tab shows the following JSON payload:

```
1 {
2   "email": "Manager3@gmail.com",
3   "firstName": "Manager3",
4   "lastName": "MgrLastname3",
5   "mobile": "7711234566",
6   "role": "Manager"
7 }
```

The response status is 200 OK, time: 38 ms, size: 272 B. The response body is:

```
1 {
2   "id": 9,
3   "firstName": "Manager3",
4   "lastName": "MgrLastname3",
5   "email": "Manager3@gmail.com",
6   "mobile": "7711234566"
7 }
```

MySQL Workbench: The "Local instance MySQL80" connection is selected. The "Query 1" tab contains the following SQL query:

```
1 • Use eloandb;
2 • Select * from users where role = 'Manager';
```

The result grid shows the following data:

	id	email	first_name	last_name	mobile	role
▶	7	Manager1@gmail.com	Manager1	MgrLastname1	7711234567	Manager
▶	8	Manager2@gmail.com	Manager2	MgrLastname2	7711234588	Manager
▶	9	Manager3@gmail.com	Manager3	MgrLastname3	7711234566	Manager
*	HULL	HULL	HULL	HULL	HULL	HULL

The "Action Output" section shows the following log entries:

#	Time	Action	Message	Duration / Fetch
38	20:04:20	Use eloandb	0 row(s) affected	0.000 sec
39	20:04:20	Select * from users where role = 'Manager' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
40	20:05:21	Use eloandb	0 row(s) affected	0.000 sec
41	20:05:21	Select * from users where role = 'Manager' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

all-clerks and Managers code and postman execution

The screenshot shows the Eclipse IDE interface with the AdminController.java file open in the editor. The code implements two endpoints: getAllClerks and getAllManagers. Both methods use the @GetMapping annotation with paths /all-clerks and /all-managers respectively. They both call the getAllClerks and getAllManagers methods from the AdminService interface.

```
53     ResponseEntity<UserDto> response = new ResponseEntity<UserDto>(outputDto, HttpStatus.OK);
54     return response;
55   }
56
57   @GetMapping("/all-clerks")
58   public ResponseEntity<List<UserDto>> getAllClerks()
59   {
60     List<UserDto> userOutputDtos = this.adminService.getAllClerks();
61
62     ResponseEntity<List<UserDto>> response =
63       new ResponseEntity<List<UserDto>>(userOutputDtos, HttpStatus.OK);
64
65     return response;
66   }
67
68   @GetMapping("/all-managers")
69   public ResponseEntity<List<UserDto>> getAllManagers()
70   {
71     List<UserDto> userOutputDtos = this.adminService.getAllManagers();
72
73     ResponseEntity<List<UserDto>> response =
74       new ResponseEntity<List<UserDto>>(userOutputDtos, HttpStatus.OK);
75
76     return response;
77   }
```

The status bar at the bottom indicates the application was started at 21:37 on 06-12-2020.

The screenshot shows the Eclipse IDE interface with the AdminServiceImpl.java file open in the editor. This implementation provides the logic for the getAllClerks and getAllManagers methods. It uses the UserRepository to find users by role ('Clerk' or 'Manager') and then maps them to UserOutputDtos using a lambda expression.

```
67
68   @Override
69   public List<UserDto> getAllClerks() {
70
71     List<Users> users = this.usersRepository.findByUserRole("Clerk");
72     List<UserDto> userOutputDtos =
73       users.stream()
74         .map(this :: convertEntityToOutputDto)
75         .collect(Collectors.toList());
76
77     return userOutputDtos;
78   }
79
80   @Override
81   public List<UserDto> getAllManagers() {
82
83     List<Users> users = this.usersRepository.findByUserRole("Manager");
84     List<UserDto> userOutputDtos =
85       users.stream()
86         .map(this :: convertEntityToOutputDto)
87         .collect(Collectors.toList());
88
89     return userOutputDtos;
90   }
91
```

The status bar at the bottom indicates the application was started at 21:37 on 06-12-2020.

Postman execution for all-clerks and all-managers

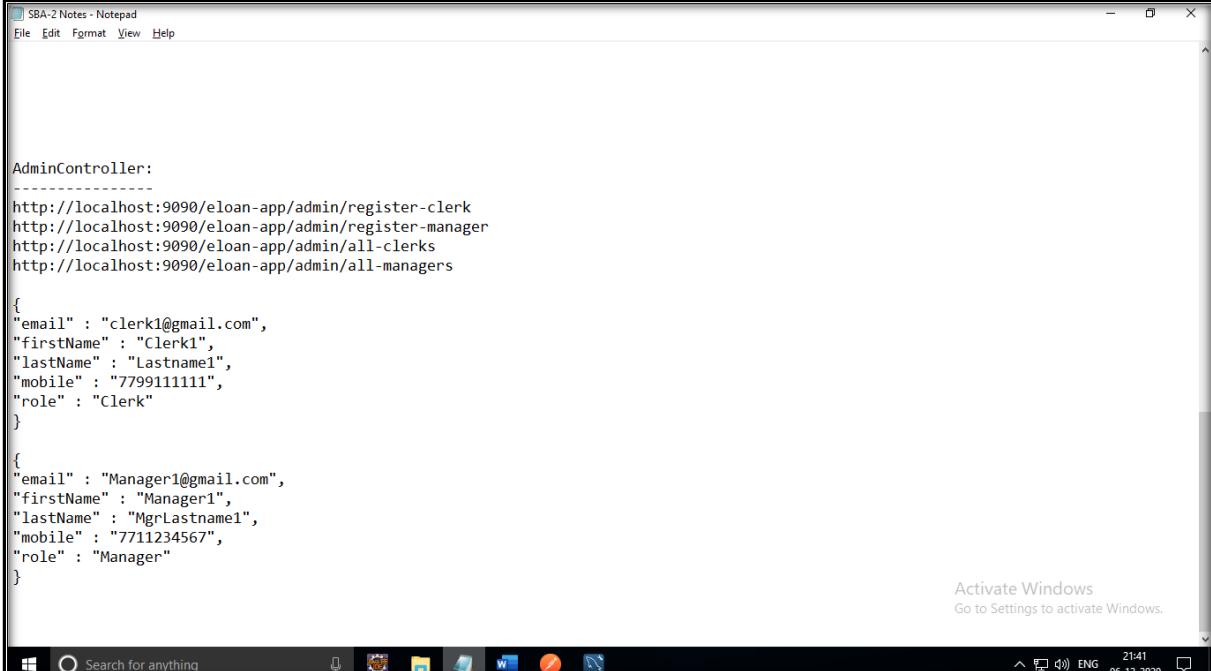
The screenshot shows the Postman interface with an 'Untitled Request' tab selected. The request method is 'GET' and the URL is 'http://localhost:9090/eloan-app/admin/all-clerks'. The response status is 200 OK, time is 19 ms, and size is 472 B. The response body is a JSON array containing two objects:

```
[{"id": 4, "firstName": "Clerk1", "lastName": "Lastname1", "email": "clerk1@gmail.com", "mobile": "7799111111"}, {"id": 5, "firstName": "Clerk2", "lastName": "Lastname2", "email": "clerk2@gmail.com", "mobile": "7799111110"}]
```

The screenshot shows the Postman interface with an 'Untitled Request' tab selected. The request method is 'GET' and the URL is 'http://localhost:9090/eloan-app/admin/all-managers'. The response status is 200 OK, time is 20 ms, and size is 492 B. The response body is a JSON array containing three objects:

```
[{"id": 7, "firstName": "Manager1", "lastName": "MgrLastname1", "email": "Manager1@gmail.com", "mobile": "7711234567"}, {"id": 8, "firstName": "Manager2", "lastName": "MgrLastname2", "email": "Manager2@gmail.com", "mobile": "7711234588"}, {"id": 9, "firstName": "Manager3", "lastName": "MgrLastname3", "email": "Manager3@gmail.com", "mobile": "7711234569"}]
```

Raw Data used for execution and DB snapshot

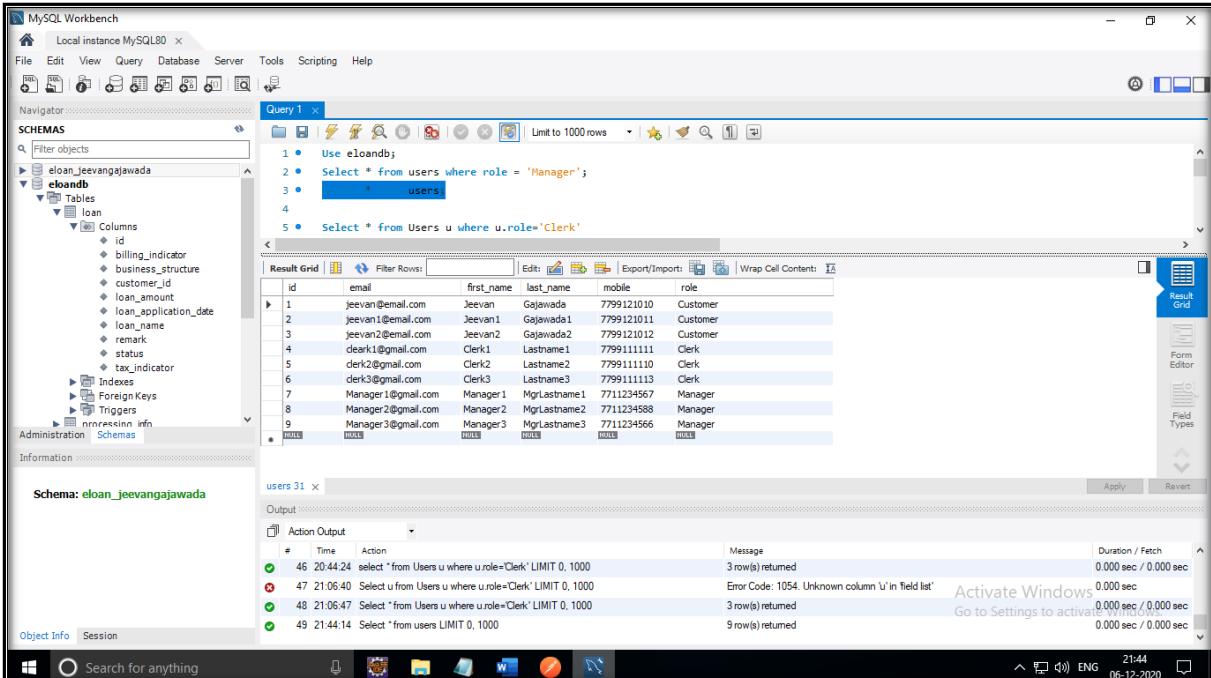


AdminController:

http://localhost:9090/eloan-app/admin/register-clerk
http://localhost:9090/eloan-app/admin/register-manager
http://localhost:9090/eloan-app/admin/all-clerks
http://localhost:9090/eloan-app/admin/all-managers

```
{  
    "email" : "clerk1@gmail.com",  
    "firstName" : "Clerk1",  
    "lastName" : "Lastname1",  
    "mobile" : "7799111111",  
    "role" : "Clerk"  
}  
  
{  
    "email" : "Manager1@gmail.com",  
    "firstName" : "Manager1",  
    "lastName" : "MgrLastname1",  
    "mobile" : "7711234567",  
    "role" : "Manager"  
}
```

Activate Windows
Go to Settings to activate Windows.



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

SCHEMAS

eloan_jeevangajawada

Tables

loan

Columns

id
billing_indicator
business_structure
customer_id
loan_amount
loan_application_date
loan_name
remark
status
tax_indicator

Indexes
Foreign Keys
Triggers

Information Schema

Query 1

```
1 • Use eloan_db;  
2 • Select * from users where role = 'Manager';  
3 •     * users;  
4 •  
5 • Select * from Users u where u.role='Clerk'
```

Result Grid

id	email	first_name	last_name	mobile	role
1	jeevan@email.com	Jeevan	Gajavada	7799121010	Customer
2	jeevan1@email.com	Jeevan1	Gajavada1	7799121011	Customer
3	jeevan2@email.com	Jeevan2	Gajavada2	7799121012	Customer
4	deark1@gmail.com	Clerk1	Lastname1	7799111111	Clerk
5	clerk2@gmail.com	Clerk2	Lastname2	7799111110	Clerk
6	clerk3@gmail.com	Clerk3	Lastname3	7799111113	Clerk
7	Manager1@gmail.com	Manager1	MgrLastname1	7711234567	Manager
8	Manager2@gmail.com	Manager2	MgrLastname2	7711234588	Manager
9	Manager3@gmail.com	Manager3	MgrLastname3	7711234566	Manager

users 31

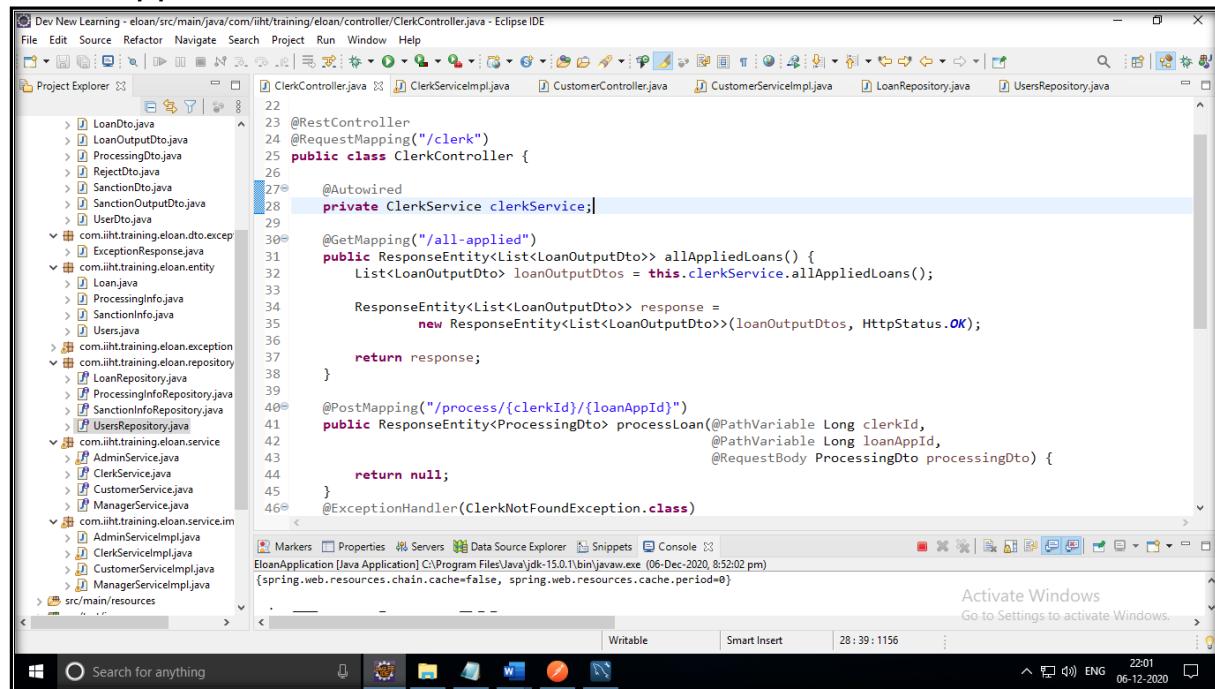
Action Output

#	Time	Action	Message	Duration / Fetch
46	20:44:24	select * from Users u where u.role='Clerk' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
47	21:06:40	Select u from Users u where u.role='Clerk' LIMIT 0, 1000	Error Code: 1054. Unknown column 'u' in field list"	0.000 sec
48	21:06:47	Select * from Users u where u.role='Clerk' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
49	21:44:14	Select * from users LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec

Activate Windows
Go to Settings to activate Windows.

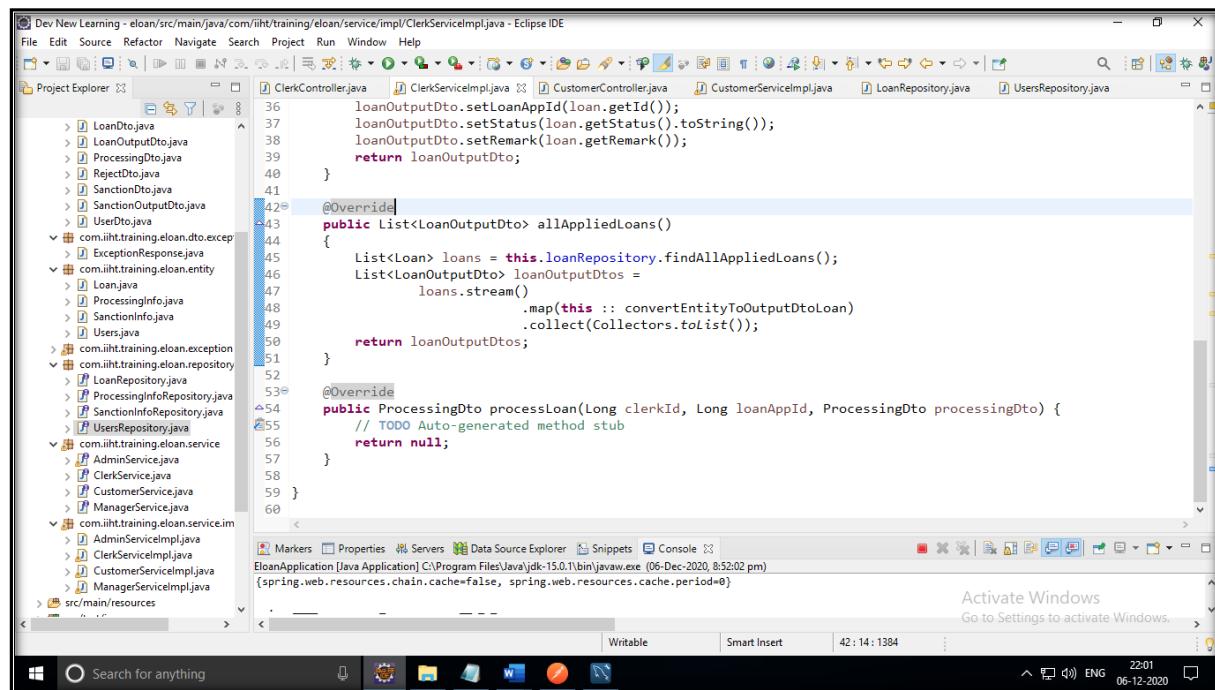
3. ClerkController and 4. ManagerController

findAllApplied loans code and execution



The screenshot shows the Eclipse IDE interface with the file `ClerkController.java` open in the editor. The code implements a `@RestController` with two methods: `allAppliedLoans()` and `processLoan()`. It uses `@Autowired` to inject the `ClerkService`. The `allAppliedLoans()` method returns a list of `LoanOutputDto` objects. The `processLoan()` method takes `clerkId`, `loanAppId`, and `processingDto` as parameters and returns a `ProcessingDto`. An exception handler for `ClerkNotFoundException` is also defined.

```
22
23 @RestController
24 @RequestMapping("/clerk")
25 public class ClerkController {
26
27     @Autowired
28     private ClerkService clerkService;
29
30     @GetMapping("/all-applied")
31     public ResponseEntity<List<LoanOutputDto>> allAppliedLoans() {
32         List<LoanOutputDto> loanOutputDtos = this.clerkService.allAppliedLoans();
33
34         ResponseEntity<List<LoanOutputDto>> response =
35             new ResponseEntity<List<LoanOutputDto>>(loanOutputDtos, HttpStatus.OK);
36
37         return response;
38     }
39
40     @PostMapping("/process/{clerkId}/{loanAppId}")
41     public ResponseEntity<ProcessingDto> processLoan(@PathVariable Long clerkId,
42             @PathVariable Long loanAppId,
43             @RequestBody ProcessingDto processingDto) {
44
45         return null;
46     }
47     @ExceptionHandler(ClerkNotFoundException.class)
48 }
```



The screenshot shows the Eclipse IDE interface with the file `ClerkServiceImpl.java` open in the editor. The code implements the `ClerkService` interface. It overrides the `allAppliedLoans()` method to return a list of `LoanOutputDto` objects by mapping the `Loan` entities from the `loanRepository`. It also overrides the `processLoan()` method to return a `ProcessingDto` stub.

```
36     loanOutputDto.setLoanAppId(loan.getId());
37     loanOutputDto.setStatus(loan.getStatus().toString());
38     loanOutputDto.setRemark(loan.getRemark());
39
40     return loanOutputDto;
41
42     @Override
43     public List<LoanOutputDto> allAppliedLoans()
44     {
45         List<Loan> loans = this.loanRepository.findAllAppliedLoans();
46         List<LoanOutputDto> loanOutputDtos =
47             loans.stream()
48                 .map(this :: convertEntityToOutputDtoLoan)
49                 .collect(Collectors.toList());
50
51         return loanOutputDtos;
52     }
53
54     @Override
55     public ProcessingDto processLoan(Long clerkId, Long loanAppId, ProcessingDto processingDto) {
56         // TODO Auto-generated method stub
57         return null;
58     }
59 }
```

Postman

File Edit View Help

+ New Import Runner ↻ My Workspace ⓘ Invite Upgrade

History Collections APIs

Save Responses Clear all

POST http://... POST http://... GET http://... GET http://... POST http://... POST http://... GET http://... No Environment

Untitled Request

GET http://localhost:9090/eloan-app/clerk/all-applied

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

7   "processingDto": null,
8   "sanctionOutputDto": null,
9   "status": "0",
10  "remark": "All Good"
11 },
12 {
13   "customerId": 789124,
14   "loanAppId": 2,
15   "userDto": null,
16   "loanDto": null,
17   "processingDto": null,
18   "sanctionOutputDto": null,
19   "status": "0",
20   "remark": "All Good"
21 },
22 {
23   "customerId": 789124,
24   "loanAppId": 3,

```

Status: 200 OK Time: 59 ms Size: 600 B Save Response

Activate Windows Go to Settings to activate Windows.

Bootcamp Build Browse

Search for anything

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS eloan_jeevangajawada eloandb

Tables loan

Columns id, billing_indicator, business_structure, customer_id, loan_amount, loan_application_date, loan_name, remark, status, tax_indicator

Indexes

Foreign Keys

Triggers

processing info

Administration Schemas

Information

Schema: eloan_jeevangajawada

Query 1

```

1 • Use eloan_jeevangajawada;
2 • Select * from users where role = 'Manager';
3 • Select * from loan;
4
5 • Select * from Users u where u.role='Clerk'
6
7
8 8:00:00 | Loan u u -0
9
10

```

Result Grid

id	billing_indicator	business_structure	customer_id	loan_amount	loan_application_date	loan_name	remark	status	tax_indicator
1	Salaried	Individual	781123	25000	06-12-2020	Home Loan	All Good	0	Tax Payer
2	Salaried	Individual	789124	25000	06-12-2020	Home Loan	All Good	0	Tax Payer
3	Salaried	Individual	789124	26000	06-12-2020	Car Loan	All Good	0	Tax Payer
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Loan 35

Action Output

#	Time	Action	Message	Duration / Fetch
50	21:56:41	Select * from loan 0..1000	4 row(s) returned	0.000 sec / 0.000 sec
51	21:57:39	Select * from Loan u where u.status=0 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
52	21:57:59	Select * from Loan u where u.status=0 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
53	22:02:33	Select * from Loan u where u.status=0 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

Activate Windows Go to Settings to activate Windows.

Result Grid Form Editor

Object Info Session

Search for anything

Manager AllProcessed loans

The screenshot shows two Eclipse IDE windows side-by-side.

Top Window (ManagerController.java):

```
1 package com.iith.training.eloan.controller;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/manager")
7 public class ManagerController {
8
9     @Autowired
10    private ManagerService managerService;
11
12    @GetMapping("/all-processed")
13    public ResponseEntity<List<LoanOutputDto>> allProcessedLoans(){
14        List<LoanOutputDto> loanOutputDtos = this.managerService.allProcessedLoans();
15
16        ResponseEntity<List<LoanOutputDto>> response =
17            new ResponseEntity<List<LoanOutputDto>>(loanOutputDtos, HttpStatus.OK);
18
19        return response;
20    }
21
22    @PostMapping("/reject-loan/{managerId}/{loanAppId}")
23    public ResponseEntity<RejectDto> rejectLoan(@PathVariable Long managerId,
24                                                @PathVariable Long loanAppId,
25                                                @RequestBody RejectDto rejectDto){
26
27    }
28}
```

Bottom Window (ManagerServiceImpl.java):

```
34
35    private LoanOutputDto convertEntityToOutputDto(Loan loan) {
36        LoanOutputDto loanOutputDto = new LoanOutputDto();
37        loanOutputDto.setCustomerId(loan.getCustomerId());
38        loanOutputDto.setLoanAppId(loan.getId());
39        loanOutputDto.setStatus(loan.getStatus().toString());
40        loanOutputDto.setRemark(loan.getRemark());
41
42        return loanOutputDto;
43    }
44
45    @Override
46    public List<LoanOutputDto> allProcessedLoans()
47    {
48        List<Loan> loans = this.loanRepository.findAllLoansByStatus(1);
49        List<LoanOutputDto> loanOutputDtos =
50            loans.stream()
51                .map(this :: convertEntityToOutputDto)
52                .collect(Collectors.toList());
53
54        return loanOutputDtos;
55    }
56
57    @Override
58    public RejectDto rejectLoan(Long managerId, Long loanAppId, RejectDto rejectDto) {
59        // TODO Auto-generated method stub
60        return null;
61    }
62}
```

Postman

File Edit View Help

+ New Import Runner

My Workspace Invite

POST http://... POST http://... GET http://... GET http://... POST http://... POST http://... GET http://... No Environment

History Collections APIs

Save Responses Clear all

Untitled Request

GET http://localhost:9090/eloan-app/manager/all-processed

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "customerId": 789124,
3   "loanAppId": 4,
4   "userOto": null,
5   "loanDto": null,
6   "processingDto": null,
7   "sanctionOutputDto": null,
8   "status": "1",
9   "remark": "All Good"
10 }
11 }
12 }
```

Status: 200 OK Time: 17 ms Size: 310 B Save Response

Activate Windows Go to Settings to activate Windows.

Bootcamp Build Browse

Search for anything

Windows Search for anything

22:19 ENG 06-12-2020

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS eloan_jeevangajawada

eloandb Tables loan Columns id, billing_indicator, business_structure, customer_id, loan_amount, loan_application_date, loan_name, remark, status, tax_indicator

Indexes Foreign Keys Triggers processing_info

Administration Schemas

Information Schema: eloan_jeevangajawada

Query 1

```
1 • Use eloandb;
2 • Select * from users where role = 'Manager';
3 • Select * from loan;
4
5 • Select * from Users u where u.role='Clerk'
6
7
8 • Select * from Loan where loan.status = 1
9
10
```

Result Grid Filter Rows Export/Import Wrap Cell Content

#	id	billing_indicator	business_structure	customer_id	loan_amount	loan_application_date	loan_name	remark	status	tax_indicator
4	4	Salaried	Individual	789124	27000	06-12-2020	Second House Loan	All Good	1	Tax Payer
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Loan 37

Action Output

#	Time	Action	Message	Duration / Fetch
52	21:57:59	Select * from Loan u where u.status=0 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
53	22:02:33	Select * from Loan u where u.status=0 LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
54	22:07:03	Select * from Loan LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
55	22:20:01	Select * from Loan where loan.status = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Activate Windows Go to Settings to activate Windows.

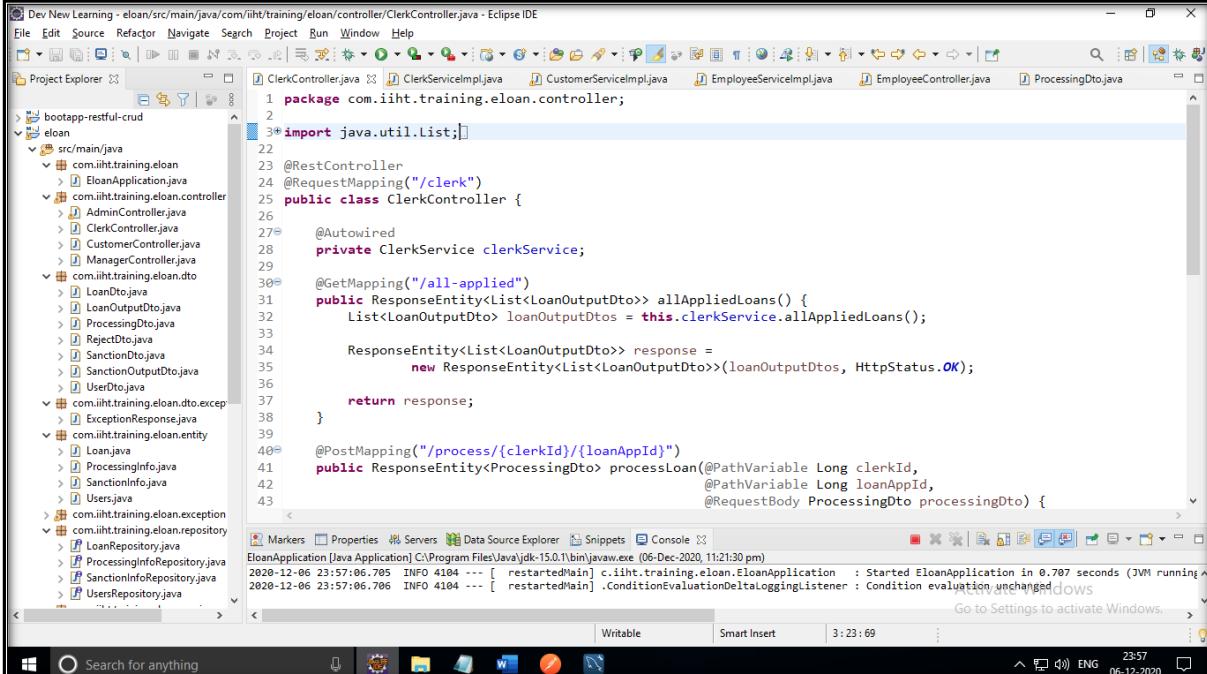
Result Grid Form Editor

Search for anything

Windows Search for anything

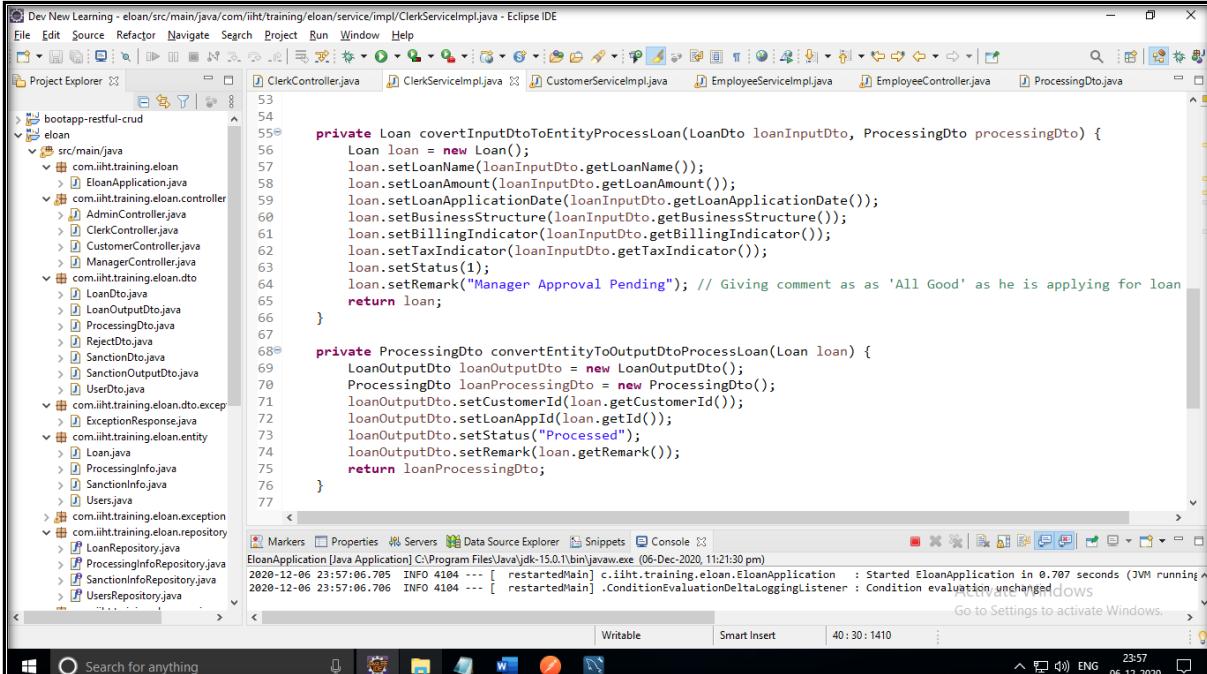
22:20 ENG 06-12-2020

Now added the code for Sanction and Processing to the Output



The screenshot shows the Eclipse IDE interface with the project 'eloan' selected in the Project Explorer. The 'ClerkController.java' file is open in the editor. The code implements a REST controller for managing loans. It includes methods for getting all applied loans and processing a specific loan application.

```
package com.iiht.training.eloan.controller;
import java.util.List;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.beans.factory.annotation.Autowired;
import com.iiht.training.eloan.ClerkService;
import com.iiht.training.eloan.dto.LoanOutputDto;
import com.iiht.training.eloan.exception.ExceptionResponse;
import com.iiht.training.eloan.entity.Loan;
import com.iiht.training.eloan.repository.LoanRepository;
import com.iiht.training.eloan.service.ProcessingService;
import com.iiht.training.eloan.util.ProcessingDto;
import com.iiht.training.eloan.util.ResponseEntity;
import com.iiht.training.eloan.util.ResponseEntity<List<LoanOutputDto>>;
import com.iiht.training.eloan.util.ResponseEntity<List<LoanOutputDto>> response =
    new ResponseEntity<List<LoanOutputDto>>(loanOutputDtos, HttpStatus.OK);
return response;
}
@PostMapping("/process/{clerkId}/{loanAppId}")
public ResponseEntity<ProcessingDto> processLoan(@PathVariable Long clerkId,
    @PathVariable Long loanAppId,
    @RequestBody ProcessingDto processingDto) {
}
```



The screenshot shows the Eclipse IDE interface with the project 'eloan' selected in the Project Explorer. The 'ClerkServiceImpl.java' file is open in the editor. This service class contains methods for converting input DTOs to entity objects and vice versa, as well as for processing loans.

```
private Loan covertInputDtoToEntityProcessLoan(LoanDto loanInputDto, ProcessingDto processingDto) {
    Loan loan = new Loan();
    loan.setLoanName(loanInputDto.getLoanName());
    loan.setLoanAmount(loanInputDto.getLoanAmount());
    loan.setLoanApplicationDate(loanInputDto.getLoanApplicationDate());
    loan.setBusinessStructure(loanInputDto.getBusinessStructure());
    loan.setBillingIndicator(loanInputDto.getBillingIndicator());
    loan.setTaxIndicator(loanInputDto.getTaxIndicator());
    loan.setStatus(1);
    loan.setRemark("Manager Approval Pending"); // Giving comment as as 'All Good' as he is applying for loan
    return loan;
}
private ProcessingDto convertEntityToOutputDtoProcessLoan(Loan loan) {
    LoanOutputDto loanOutputDto = new LoanOutputDto();
    ProcessingDto loanProcessingDto = new ProcessingDto();
    loanOutputDto.setCustomerId(loan.getCustomerId());
    loanOutputDto.setLoanAppId(loan.getId());
    loanOutputDto.setStatus("Processed");
    loanOutputDto.setRemark(loan.getRemark());
    return loanProcessingDto;
}
```

Dev New Learning - elan/src/main/java/com/ihtt/training/elan/service/impl/ClerkServiceImpl.java - Eclipse IDE

```

1     loanOutputDto.setCustomerId(loan.getCustomerId());
2     loanOutputDto.setLoanAppId(loan.getAppId());
3     loanOutputDto.setStatus("Processed");
4     loanOutputDto.setRemark(loan.getRemark());
5     return loanProcessingDto;
6 }
7
8 @Override
9 public ProcessingDto processLoan(Long clerkId, Long loanAppId, ProcessingDto processingDto) {
10
11     // convert dto into entity
12     LoanDto loanDto = new LoanDto();
13     Loan loan = this.convertInputDtoToEntityProcessLoan(loanDto,processingDto);
14     // assign the id to update
15     loan.setId(loanAppId);
16     // repository method to update
17     Loan updatedLoan = this.loanRepository.save(loan);
18     // convert entity into output dto
19     ProcessingDto processingOutputDto = this.convertEntityToOutputDtoProcessLoan(updatedLoan);
20     return processingOutputDto;
21 }
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

```

Markers Properties Servers Data Source Explorer Snippets Console

ElanApplication [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (06-Dec-2020, 11:21:30 pm)

2020-12-06 23:57:06.705 INFO 4104 --- [restartedMain] c.ihtt.training.elan.ElanApplication : Started ElanApplication in 0.707 seconds (JVM running since 2020-12-06 23:57:06.706 INFO 4104 --- [restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged)

Go to Settings to activate Windows.

Dev New Learning Writable Smart Insert 40:30:1410

Windows Search for anything ENG 23:57 06-12-2020

After execution 2 records moved to Manager for Processing

Postman

+ New Import Runner

My Workspace

Untitled Request

GET http://localhost:9090/eloan-app/manager/all-processed

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

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 58 ms Size: 455 B Save Response

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "customerId": 789124,
4     "loanAppId": 4,
5     "userDto": null,
6     "loanDto": null,
7     "processingDto": null,
8     "sanctionOutputDto": null,
9     "status": "1",
10    "remark": "All Good"
11  },
12  {
13    "customerId": 789124,
14    "loanAppId": 5,
15    "userDto": null,
16    "loanDto": null,
17    "processingDto": null,
18    "sanctionOutputDto": null,

```

Activate Windows Go to Settings to activate Windows.

Find and Replace Console

Windows Search for anything ENG 23:58 06-12-2020

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: eloan_jeevangajawada

SCHEMAS: eloan_jeevangajawada

Tables: eloanb

Columns: loan

Result Grid:

	id	billing_indicator	business_structure	customer_id	loan_amount	loan_application_date	loan_name	remark	status	tax_indicator
4	Salaried	Individual	789124	27000	06-12-2020	Second House Loan	All Good	1	Tax Payer	
5	Salaried	Individual	789124	27000	06-12-2020		All Good	1	Tax Payer	

Action Output:

#	Time	Action	Message	Duration / Fetch
55	22:20:01	Select * from Loan where loan.status = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
56	23:22:21	Select * from Users LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
57	23:24:06	Select * from loan LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
58	23:59:23	Select * from loan where status =1 LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

The below implementations are pending. Because of lack of time, it is pending but other than the complete implementation of project is completed 100%

```

public ResponseEntity<List<LoanOutputDto>> allProcessedLoans() {
    List<LoanOutputDto> loanOutputDtos = this.managerService.allProcessedLoans();

    ResponseEntity<List<LoanOutputDto>> response =
        new ResponseEntity<List<LoanOutputDto>>(loanOutputDtos, HttpStatus.OK);

    return response;
}

@PostMapping("/reject-loan/{managerId}/{loanAppId}")
public ResponseEntity<RejectDto> rejectLoan(@PathVariable Long managerId,
                                             @PathVariable Long loanAppId,
                                             @RequestBody RejectDto rejectDto){
    return null;
}

@PostMapping("/sanction-loan/{managerId}/{loanAppId}")
public ResponseEntity<SanctionOutputDto> sanctionLoan(@PathVariable Long managerId,
                                                       @PathVariable Long loanAppId,
                                                       @RequestBody SanctionDto sanctionDto){
    return null;
}

@ExceptionHandler(ManagerNotFoundException.class)
public ResponseEntity<ErrorResponse> handler(ManagerNotFoundException ex){
}

```