

Full Name	Sachin Rangnath Gawade
Batch	Dec 2020
Project Title	E-commerce Website for Sporty Shoes
Project Submission Date	29-April – 2022
Git Hub Link	<a href="https://github.com/saching3234/simply_learn_sport_shoes_project">https://github.com/saching3234/simply_learn_sport_shoes_project</a>

Source Code :-

### **Com.to.resources:- (Package)**

#### **1. AdminLoginResource.java**

```
package com.to.resources;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.to.Constants;
import com.to.entities.AdminLogin;
import com.to.entities.User;
import com.to.services.AdminLoginService;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

@RestController
@RequestMapping("/api/admin")
public class AdminLoginResource {

    @Autowired
    AdminLoginService adminLoginService;

    @PostMapping("/login")
    public ResponseEntity<Map<String, String>> loginAdmin(@RequestBody Map<String, Object>
adminMap) {
        String adminId = (String) adminMap.get("adminId");
        String password = (String) adminMap.get("password");
        // checking the admin id and password is correct or not
        AdminLogin adminLogin = adminLoginService.adminLogin(adminId, password);
```

```

        return new ResponseEntity<>(generateJWTToken(adminLogin), HttpStatus.OK);
    }

    @PostMapping("/changeAdminPass")
    public Map<String, String> changeAdminPass(@RequestBody Map<String, Object> adminMap) {
        String adminId = (String) adminMap.get("adminId");
        String password = (String) adminMap.get("password");
        String newPass = (String) adminMap.get("newPass");

        // checking the admin id and password is correct or not
        adminLoginService.adminPasswordChange(adminId, password, newPass);

        Map<String, String> map = new HashMap<>();
        map.put("Message", "Password Changed Successfully Login with new Password " + adminId);
        return map;
    }

    // methodn for generating the token

    private Map<String, String> generateJWTToken(AdminLogin adminLogin) {
        // getting the current time in milisecods
        long timestamp = System.currentTimeMillis();
        String token = Jwts.builder().signWith(SignatureAlgorithm.HS256,
Constants.API_SECRET_KEY_STRING)
        .setIssuedAt(new Date(timestamp)).setExpiration(new Date(timestamp +
Constants.TOKEN_VALIDITY))
        .claim("adminId", adminLogin.getAdminId()).compact();

        Map<String, String> map = new HashMap<>();
        map.put("Succes", "Login Successfull Admin Use the given token");
        map.put("token", token);
        return map;
    }
}

```

## **2. AdminCategoryResource.java**

```

package com.to.resources;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.to.entities.Category;
import com.to.entities.Product;
import com.to.services.CategoryService;
import com.to.services.ProductServices;

@RestController
@RequestMapping("/api/admin/category")
public class AdminCategoryResource {

    @Autowired
    CategoryService categoryService;

    // controller for getting the all category details
    @GetMapping("")
    public ResponseEntity<List<Category>> getAllCategories(HttpServletRequest request) {
        // getting the admin id
        String adminId = (String) request.getAttribute("adminId");

        List<Category> categories = categoryService.getAllCategories();
        System.out.println(categories);
        return new ResponseEntity<>(categories, HttpStatus.OK);
    }

    // controller for getting the specific category details
    @GetMapping("/{catId}")
    public ResponseEntity<Category> getCategoryById(HttpServletRequest request,
    @PathVariable("catId") Integer catId) {

        // getting the admin Id
        String adminId = (String) request.getAttribute("adminId");

        // searching the category
        Category category = categoryService.getCategoryById(catId);
        System.out.println("get category by id called" + category + " " + catId);
    }
}

```

```

        return new ResponseEntity<>(category, HttpStatus.OK);
    }

    // controller for deleting the specific category details

    @DeleteMapping("/delete/{catId}")
    public ResponseEntity<Map<String, String>> deleteCategoryById(HttpServletRequest request,
        @PathVariable("catId") Integer catId) {

        // getting the admin Id
        String adminId = (String) request.getAttribute("adminId");

        // deleting the category
        categoryService.deleteCategory(catId);
        Map<String, String> map = new HashMap<>();
        map.put("Success", "Category Id : " + catId + " Deleted Successfully");
        return new ResponseEntity<>(map, HttpStatus.OK);
    }

    // controller for adding the new category details
    @PostMapping("")
    public ResponseEntity<Map<String, String>> addCategory(HttpServletRequest request,
        @RequestBody Map<String, Object> catMap) {

        // getting the admin id from token provided
        String adminId = (String) request.getAttribute("adminId");

        System.out.println("Control is here");
        // adding category details into the bean
        Category category = new Category();
        category.setCatName((String) catMap.get("cname"));

        // calling the service method to save the data
        categoryService.createCategory(category.getCatName());

        System.out.println("Control is here after product");

        Map<String, String> map = new HashMap<>();
        map.put("msg", "Category Details added");
        return new ResponseEntity<>(map, HttpStatus.OK);
    }

    // controller for updating existing category details
    @PatchMapping("")
    public ResponseEntity<Map<String, String>> updateCategory(HttpServletRequest request,
        @RequestBody Map<String, Object> catMap) {

```

```

        // getting the admin id from token provided
        String adminId = (String) request.getAttribute("adminId");

        System.out.println("Control is here");
        // adding category details into the bean
        Category category = new Category();
        category.setCatId(Integer.parseInt((String) catMap.get("cid")));
        category.setCatName((String) catMap.get("cname"));

        // calling the service method to save the data
        categoryService.updateCategory(category.getCatId(), category.getCatName());

        Map<String, String> map = new HashMap<>();
        map.put("msg", "Category Details updates Successfully");
        return new ResponseEntity<>(map, HttpStatus.OK);
    }
}

```

### **3. AdminProductResource.java**

```

package com.to.resources;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.to.entities.Product;
import com.to.entities.UserPurchase;
import com.to.services.ProductServices;

@RestController
@RequestMapping("/api/admin/product")
public class AdminProductResource {
    @Autowired
    ProductServices productServices;

    // controller for getting the all product details

```

```

@GetMapping("")
public ResponseEntity<List<Product>> getAllProducts(HttpServletRequest request) {
    String adminId = (String) request.getAttribute("adminId");
    List<Product> products = productService.getAllProductDetails();
    System.out.println(products);
    return new ResponseEntity<>(products, HttpStatus.OK);
}

// controller for getting the specific product details
@GetMapping("/{pId}")
public ResponseEntity<Product> getProductById(HttpServletRequest request, @PathVariable("pId")
Integer pId) {
    // getting the admin Id
    String adminId = (String) request.getAttribute("adminId");
    // searching the product
    Product products = productService.getProductById(pId);
    System.out.println("get product by id called" + products + " " + pId);
    return new ResponseEntity<>(products, HttpStatus.OK);
}

// controller for deleting the specific product details
@GetMapping("/delete/{pId}")
public ResponseEntity<Map<String, String>> deleteProductById(HttpServletRequest request,
    @PathVariable("pId") Integer pId) {
    // getting the admin Id
    String adminId = (String) request.getAttribute("adminId");
    // deleting the product
    productService.deleteProduct(pId);
    Map<String, String> map = new HashMap<>();
    map.put("Success", "Product Id : " + pId + " Deleted Successfully");
    return new ResponseEntity<>(map, HttpStatus.OK);
}

// controller for adding the new product details
@PostMapping("")
public ResponseEntity<Map<String, String>> addProduct(HttpServletRequest request,
    @RequestBody Map<String, Object> productMap) {
    // getting the admin add from token provide
    String adminId = (String) request.getAttribute("adminId");
    System.out.println("Control is here");
    // adding product details into the bean
    Product prodcut = new Product();
    prodcut.setPname((String) productMap.get("pname"));
    prodcut.setPdescription((String) productMap.get("pdescription"));
    prodcut.setPrice(Integer.parseInt(productMap.get("price").toString()));
    prodcut.setGender((String) productMap.get("gender"));
    prodcut.setCid(Integer.parseInt(productMap.get("cid").toString()));
    // calling the service method to save the data

```

```

        productService.createNewProduct(prodcut);
        System.out.println("Control is here aftefr product");
        Map<String, String> map = new HashMap<>();
        map.put("msg", "Product Details added");
        return new ResponseEntity<>(map, HttpStatus.OK);

    }

}

```

#### **4. AdminReportsResource:-**

```

package com.to.resources;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.to.entities.Product;
import com.to.entities.User;
import com.to.services.AdminReportService;
import com.to.services.ProductServices;

@RestController
@RequestMapping("/api/admin/report")
public class AdminReportsResource {
    @Autowired
    AdminReportService adminReportService;

    // controller for getting the all product details
    @GetMapping("/loggedUsers")
    public ResponseEntity<List<User>> getAllUsers(HttpServletRequest request) {
        //getting the admin od from token
        String adminId = (String) request.getAttribute("adminId");

        List<User> users = adminReportService.getAllLoggedUsersDetails();
        System.out.println(users);
        return new ResponseEntity<>(users, HttpStatus.OK);
    }
}

```

#### **5. UserResource.java :-**

```

package com.to.resources;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.to.Constants;
import com.to.entities.User;
import com.to.services.UserService;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
@RestController
@RequestMapping("/api/users")
public class UserResource {
    @Autowired
    UserService userService;

    @PostMapping("/register")
    public ResponseEntity<Map<String, String>> registerUser(@RequestBody
Map<String, Object> userMap) {
        String firstName = (String) userMap.get("fname");
        String lastName = (String) userMap.get("lname");
        String email = (String) userMap.get("email");
        String password = (String) userMap.get("password");
        User user = userService.registerUser(firstName, lastName, email,
password);

        return new ResponseEntity<>(generateJWTToken(user),
HttpStatus.OK);
    }

    @PostMapping("/loginuser")
    public ResponseEntity<Map<String, String>> loginUser(@RequestBody
Map<String, Object> userMap) {
        String emailString = (String) userMap.get("email");
        String passwordString = (String) userMap.get("password");
        // checking the user id and password is correct or not
        User user = userService.validateUser(emailString,
passwordString);

```



```

        return new ResponseEntity<>(generateJWTToken(user),
HttpStatus.OK);
    }

    // methodn for generating the token

    private Map<String, String> generateJWTToken(User user) {
        // getting the current time in milisecods
        long timestamp = System.currentTimeMillis();
        String token = Jwts.builder().signWith(SignatureAlgorithm.HS256,
Constants.API_SECRET_KEY_STRING)
            .setIssuedAt(new Date(timestamp)).setExpiration(new
Date(timestamp + Constants.TOKEN_VALIDITY))
            .claim("userId", user.getUserId()).claim("email",
user.getEmail()).claim("firstName", user.getFname())
            .claim("lastName", user.getLname()).compact();

        Map<String, String> map = new HashMap<>();
        map.put("token", token);
        return map;
    }
}

```

## **6. UserPurchaseResource.java**

```

package com.to.resources;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.to.entities.Product;
import com.to.entities.UserPurchase;
import com.to.services.UserPurchaseService;

```

```

@RestController
@RequestMapping("/api/userPurchase")
public class UserPurchaseResource {

    @Autowired
    UserPurchaseService userPurchaseService;

    @GetMapping("")
    public ResponseEntity<List<UserPurchase>> getAllPurchase(HttpServletRequest request) {
        int userId = (Integer) request.getAttribute("userId");
        List<UserPurchase> userPurchases = userPurchaseService.fetchAllPurchases(userId);
        System.out.println(userPurchases);
        return new ResponseEntity<>(userPurchases, HttpStatus.OK);
    }

    @GetMapping("/products/{catId}")
    public ResponseEntity<List<Product>> getAllProductCategorywise(HttpServletRequest request,
        @PathVariable("catId") Integer catId) {
        int userId = (Integer) request.getAttribute("userId");
        List<Product> products = userPurchaseService.getAllProductByCategory(catId);
        System.out.println(products);
        return new ResponseEntity<>(products, HttpStatus.OK);
    }

    // controller for adding the purchase details
    @PostMapping("")
    public ResponseEntity<Map<String, String>> addUserPurchase(HttpServletRequest request,
        @RequestBody Map<String, Object> userMap) {
        Integer userId = (Integer) request.getAttribute("userId");

        System.out.println("Control is here");

        UserPurchase userPurchase = new UserPurchase();
        userPurchase.setProduct_id(Integer.parseInt(userMap.get("product_id").toString()));
        userPurchase.setPdate((String) userMap.get("pdate"));
        userPurchase.setCat_id(Integer.parseInt(userMap.get("cat_id").toString()));
        userPurchase.setQuantity(Integer.parseInt(userMap.get("quantity").toString()));
        userPurchase.setPrice(Integer.parseInt(userMap.get("price").toString()));
        userPurchase.setTotal_price(Integer.parseInt(userMap.get("total_price").toString()));
        userPurchase.setUser_id(userId);

        System.out.println("Control is here before add purchase");
        userPurchaseService.addPurchase(userPurchase);
        System.out.println("Control is here aftefr add purchase");
        Map<String, String> map = new HashMap<>();
        map.put("msg", "Purchase Details added" + userPurchase);

        return new ResponseEntity<>(map, HttpStatus.OK);
    }
}

```

```

    }

    // method for getting the purchase details by id
    @GetMapping("/{purId}")
    public ResponseEntity<UserPurchase> getUserPurchaseById(HttpServletRequest request,
        @PathVariable("purId") Integer purId) {
        // getting the logged user id
        int userId = (Integer) request.getAttribute("userId");
        // calling the fetch method
        UserPurchase userPurchase = userPurchaseService.fetchUserPurchaseById(userId, purId);
        return new ResponseEntity<>(userPurchase, HttpStatus.OK);
    }
}

```

---

### **Com.to.filters:- (Package)**

#### **1. AdminAuthFilter.java**

```

package com.to.filters;
import java.io.IOException;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.http.HttpStatus;
import org.springframework.web.filter.GenericFilterBean;
import com.to.Constants;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
public class AdminAuthFilter extends GenericFilterBean {

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain
filterChain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) servletRequest;
        HttpServletResponse httpResponse = (HttpServletResponse) servletResponse;

        // getting the bearer string
        String authHeader = httpRequest.getHeader("Authorization");

        // checking the header is null or not

```

```

        System.out.println("Auth header string " + authHeader);
        if (authHeader != null) {
            // split the string into the array
            String[] authHeaderArr = authHeader.split("Bearer ");

            if (authHeaderArr.length > 1 && authHeaderArr[1] != null) {
                String token = authHeaderArr[1];

                try {

                    Claims claims =
Jwts.parser().setSigningKey(Constants.API_SECRET_KEY_STRING).parseClaimsJws(token)
                        .getBody();
                    httpRequest.setAttribute("adminId", claims.get("adminId").toString());

                    System.out.println("Admin Filter : "+claims.get("adminId").toString());

                } catch (Exception e) {
                    httpResponse.sendError(HttpStatus.FORBIDDEN.value(),
"Invalid/Expired Admin token reason :"+e.getMessage());
                    return;
                }
            }
            else {
                httpResponse.sendError(HttpStatus.FORBIDDEN.value(), "Authorization
token must be Bearer[token] in admin token");
                return;
            }
        }
        else {
            httpResponse.sendError(HttpStatus.FORBIDDEN.value(), "Authorization token must be
provided for admin");
            return;
        }

        //If all is set then continue the process
        filterChain.doFilter(servletRequest, servletResponse);
    }
}

```

## 2. AuthFilter.java

```

package com.to.filters;
import java.io.IOException;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

```

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.http.HttpStatus;
import org.springframework.web.filter.GenericFilterBean;
import com.to.Constants;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;

public class AuthFilter extends GenericFilterBean {

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain
filterChain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) servletRequest;
        HttpServletResponse httpResponse = (HttpServletResponse) servletResponse;

        // getting the bearer string
        String authHeader = httpRequest.getHeader("Authorization");

        // checking the header is null or not
        System.out.println("Auth header string " + authHeader);
        if (authHeader != null) {
            // split the string into the array
            String[] authHeaderArr = authHeader.split("Bearer ");

            if (authHeaderArr.length > 1 && authHeaderArr[1] != null) {
                String token = authHeaderArr[1];

                try {
                    System.out.println("filter Control here");
                    Claims claims =
Jwts.parser().setSigningKey(Constants.API_SECRET_KEY_STRING).parseClaimsJws(token)
                        .getBody();
                    // parsing the userId into Integer and setting to http request object
                    httpRequest.setAttribute("userId",
Integer.parseInt(claims.get("userId").toString()));

                } catch (Exception e) {
                    httpResponse.sendError(HttpStatus.FORBIDDEN.value(),
"Invalid/Expired token" + e.getMessage());
                    return;
                }
            } else {
                httpResponse.sendError(HttpStatus.FORBIDDEN.value(), "Authorization
token must be Bearer[token]");
                return;
            }
        }
    }
}

```

```

        }
    } else {
        httpResponse.sendError(HttpStatus.FORBIDDEN.value(), "Authorization token must be
provided");
        return;
    }

    // If all is set then continue the process

    filterChain.doFilter(servletRequest, servletResponse);

}
}

```

---

### **Com.to.exceptions:- (package)**

#### **1. EtAuthException.java**

```

package com.to.exceptions;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.UNAUTHORIZED)
public class EtAuthException extends RuntimeException{

    public EtAuthException(String message) {
        super(message);
    }
}

```

#### **2. EtBadRequestException.java**

```

package com.to.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.BAD_REQUEST)
public class EtBadRequestException extends RuntimeException{

```

```
        public EtBadRequestException(String message) {
            super(message);
        }
    }
}
```

### **3. EtResourceNotFoundException.java**

```
package com.to.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND)
public class EtResourceNotFoundException extends RuntimeException{

    public EtResourceNotFoundException(String message) {
        super(message);
    }
}
```

---

### **Com.to.repositories:- (package)**

#### **1. AdminLoginRepository.java**

```
package com.to.repositories;

import com.to.entities.AdminLogin;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

public interface AdminLoginRepository {

    AdminLogin findById(String adminId, String password) throws EtResourceNotFoundException;

    void update(String adminId, String password, String newPass) throws EtBadRequestException;
}
```

#### **2. AdminLoginRepositoryImpl.java:-**

```
package com.to.repositories;

import org.apache.logging.log4j.message.Message;
```

```

import org.mindrot.jbcrypt.BCrypt;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.RowMapperResultSetExtractor;
import org.springframework.stereotype.Repository;

import com.to.entities.AdminLogin;
import com.to.entities.User;
import com.to.exceptions.EtAuthException;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

@Repository
public class AdminLoginRepositoryImpl implements AdminLoginRepository {

    private static final String SQL_FIND_BY_ADMIN_ID_PASS = "select * from admin_login WHERE adminId=? and adminPass=?";
    private static final String SQL_FIND_BY_ADMIN_ID = "select * from admin_login WHERE adminId=?";
    private static final String SQL_UPDATE_PASS = "update admin_login set adminPass=? where adminId=?";

    @Autowired
    JdbcTemplate jdbcTemplate;

    @Override
    public AdminLogin findById(String adminId, String password) throws EtResourceNotFoundException {

        try {
            // getting the user record from db
            AdminLogin adminLogin = jdbcTemplate.queryForObject(SQL_FIND_BY_ADMIN_ID,
new Object[] { adminId },
                                adminRowMapper);

            // checking the password
            if (!password.equals(adminLogin.getPassword()))
                throw new EtAuthException("Invalid AdminId/Pasword");

            return adminLogin;

        } catch (Exception e) {
            throw new EtAuthException("Invalid AdminId/Pasword");
        }

    }

    @Override
    public void update(String adminId, String password, String newPass) throws EtBadRequestException {

```



```

        try {
            // getting the user record from db
            AdminLogin adminLogin = jdbcTemplate.queryForObject(SQL_FIND_BY_ADMIN_ID,
new Object[] { adminId },
                        adminRowMapper);
            adminLogin.setNewPass(newPass);
            System.out.println(adminLogin);
            // checking the password

            if (password.equals(newPass))
                throw new EtAuthException("Enter Old Password and new Pasword should
be different");
            else if (!password.equals(adminLogin.getPassword()))
                throw new EtAuthException("Enter Correct Old Password ");
            else {
                // update the admin password with new password
                jdbcTemplate.update(SQL_UPDATE_PASS, newPass, adminId);
            }

        } catch (EtAuthException e) {
            throw new EtAuthException(e.getMessage());
        }

        catch (Exception e) {
            throw new EtAuthException("Invalid AdminId/Pasword");
        }

    }

    // row mapper
    private RowMapper<AdminLogin> adminRowMapper = ((rs, rowNum) -> {
        return new AdminLogin(rs.getString("adminId"), rs.getString("adminPass"));
    });
}

```

### **3. AdminReports.java**

```

package com.to.repositories;

import java.util.List;
import com.to.entities.User;

public interface AdminReports {

```

```
List<User> getAllLoggedUsers();  
  
}
```

#### **4. AdminReportsImpl.java**

```
package com.to.repositories;  
  
import java.util.List;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.jdbc.core.RowMapper;  
import org.springframework.stereotype.Repository;  
  
import com.to.entities.User;  
  
@Repository  
public class AdminReportsImpl implements AdminReports {  
  
    @Autowired  
    JdbcTemplate jdbcTemplate;  
    private static String SQL_ALL_USERS = "select * from user_login";  
  
    @Override  
    public List<User> getAllLoggedUsers() {  
  
        return jdbcTemplate.query(SQL_ALL_USERS, userRowMapper);  
    }  
  
    // row mapper  
    private RowMapper<User> userRowMapper = ((rs, rowNum) -> {  
        return new User(rs.getInt("userid"), rs.getString("first_name"), rs.getString("last_name"),  
            rs.getString("email"));  
    });  
}
```

#### **5. CategoryRepository.java**

```
package com.to.repositories;  
  
import java.util.List;  
  
import com.to.entities.Category;  
import com.to.entities.Product;
```

```
import com.to.entities.UserPurchase;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;
```

```
public interface CategoryRepository {

    void create(String categoryName);

    void update(Integer catId, String catName) ;

    void delete(Integer catId) ;

    Category getById(Integer catId);

    List<Category> getAllCategories();

}
```

#### **6. CategoryRepositoryImpl.java**

```
package com.to.repositories;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
import com.to.entities.Category;
import com.to.entities.Product;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

@Repository
public class CategoryRepositoryImpl implements CategoryRepository {
    @Autowired
    JdbcTemplate jdbcTemplate;
    private static String SQL_INSERT = "INSERT INTO category(cname) values(?)";
    private static String SQL_SELECT = "select * from category";
    private static String SQL_SELECT_BY_ID = "select * from category where cid=?";
    private static String SQL_DELETE = "delete from category where cid=?";
    private static String SQL_UPDATE = "update category set cname=? where cid=?";

    @Override
    public void create(String categoryName) {
```

```

        try {
            jdbcTemplate.update(SQL_INSERT, new Object[] { categoryName });
        } catch (Exception e) {
            throw new EtBadRequestException(
                "Something went wrong while creating the category try again" +
e.getMessage());
        }

    }

    @Override
    public void update(Integer catId, String catName) {
        try {
            int i = jdbcTemplate.update(SQL_UPDATE, new Object[] { catName, catId });
            if (i == 0)
                throw new EtResourceNotFoundException(
                    "Sorry Given Category Is not present first add the category
then update");
        } catch (Exception e) {
            throw new EtBadRequestException(
                "Something went wrong while updating the category try again Reason
:" + e.getMessage());
        }

    }

    @Override
    public void delete(Integer catId) {
        try {
            int i = jdbcTemplate.update(SQL_DELETE, new Object[] { catId });
            if (i == 0)
                throw new EtResourceNotFoundException("Sorry The given cat id: " + catId +
" not available ");
        } catch (Exception e) {
            throw new EtResourceNotFoundException(e.getMessage());
        }

    }

    // row mapper
    private RowMapper<Category> catRowMapper = ((rs, rowNum) -> {

        return new Category(rs.getInt("cid"), rs.getString("cname"));
    });

    @Override
    public Category getByld(Integer catId) {

        try {

```

```

        return jdbcTemplate.queryForObject(SQL_SELECT_BY_ID, new Object[] { catid },
catRowMapper);
    } catch (Exception e) {
        throw new EtResourceNotFoundException(" category not available details " +
e.getMessage());
    }

}

@Override
public List<Category> getAllCategories() {
    try {
        return jdbcTemplate.query(SQL_SELECT, catRowMapper);
    } catch (Exception e) {
        throw new EtResourceNotFoundException("Failed to fetch the all category details " +
e.getMessage());
    }
}
}

```

### **7. ProductRepository.java**

```

package com.to.repositories;

import java.util.List;
import com.to.entities.Product;
import com.to.entities.UserPurchase;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

public interface ProductRepository {

    void create(Product product) throws EtBadRequestException;

    void update(Integer userId, Integer purId, UserPurchase userPurchase) throws
EtBadRequestException;

    void delete(Integer pld);

    Product getByld(Integer pid) throws EtResourceNotFoundException;

    List<Product> getAllProduct() throws EtResourceNotFoundException;

}

```

### **8. ProductRepositoryImpl.java**

```

package com.to.repositories;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
import com.to.entities.Product;
import com.to.entities.User;
import com.to.entities.UserPurchase;
import com.to.exceptions.EtAuthException;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

@Repository
public class ProductRepositoryImpl implements ProductRepository {
    private static final String SQL_CREATE = "insert into product(pname,pdescription,price,gender,cid)
values(?,?,?,?,?)";
    private static final String SQL_FIND_ALL = "select * from product";
    private static final String SQL_FIND_BY_ID = "select * from product where pid=?";
    private static final String SQL_DELETE_BY_ID = "delete from product where pid=?";
    @Autowired
    JdbcTemplate jdbcTemplate;

    @Override
    public void create(Product product) throws EtBadRequestException {

        try {
            jdbcTemplate.update(SQL_CREATE, product.getPname(), product.getPdescription(),
product.getPrice(),
                                product.getGender(), product.getCid());

        } catch (Exception e) {
            throw new EtAuthException("Failed to insert into product table try again" +
e.getMessage());
        }

    }

    @Override
    public void update(Integer userId, Integer purId, UserPurchase userPurchase) throws
EtBadRequestException {
        // TODO Auto-generated method stub

    }

    @Override

```

```

        public void delete(Integer pld) {
            try {
                int i = jdbcTemplate.update(SQL_DELETE_BY_ID, new Object[] { pld });
                if (i == 0)
                    throw new EtResourceNotFoundException("Sorry The given product id: " + pld
+ " not available ");

            } catch (Exception e) {
                throw new EtResourceNotFoundException(e.getMessage());
            }

        }

        @Override
        public Product getByld(Integer pid) throws EtResourceNotFoundException {
            try {
                return jdbcTemplate.queryForObject(SQL_FIND_BY_ID, new Object[] { pid },
productRowMapper);

            } catch (Exception e) {
                throw new EtResourceNotFoundException(
                    "Sorry The given product id: " + pid + " not available " +
e.getMessage());
            }

        }

        // method for getting the all product details
        public List<Product> getAllProduct() throws EtResourceNotFoundException {

            try {
                return jdbcTemplate.query(SQL_FIND_ALL, productRowMapper);

            } catch (Exception e) {
                throw new EtResourceNotFoundException("Failed to fetch the all product details " +
e.getMessage());
            }

        }

        // row mapper
        private RowMapper<Product> productRowMapper = ((rs, rowNum) -> {

            return new Product(rs.getInt("pid"), rs.getString("pname"), rs.getString("pdescription"),
rs.getInt("price"),
                rs.getString("gender"), rs.getInt("cid"), rs.getString("imgpath"));

        });

    }

```

### **9. UserProductDisplay.java**

```
package com.to.repositories;

import java.util.List;

import com.to.entities.Product;
import com.to.exceptions.EtResourceNotFoundException;

public interface UserProductDisplay {

    List<Product> getAllProductByCategory(Integer catId);

}
```

### **10. UserProductDisplayImpl.java**

```
package com.to.repositories;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
import com.to.entities.Product;
import com.to.exceptions.EtResourceNotFoundException;

@Repository
public class UserProductDisplayImpl implements UserProductDisplay {

    @Autowired
    JdbcTemplate jdbcTemplate;
    private static String SQL_FIND_ALL_PRODUCT_BY_CAT_ID = "select * from product where cid=?";

    @Override
    public List<Product> getAllProductByCategory(Integer catId) {
        try {
            return jdbcTemplate.query(SQL_FIND_ALL_PRODUCT_BY_CAT_ID, new Object[] { catId
            }, productRowMapper);

        } catch (Exception e) {
            throw new EtResourceNotFoundException("Failed to fetch the all product details " +
            e.getMessage());
        }

    }

    // row mapper
```



```

        private RowMapper<Product> productRowMapper = ((rs, rowNum) -> {

            return new Product(rs.getInt("pid"), rs.getString("pname"), rs.getString("pdescription"),
rs.getInt("price"),
            rs.getString("gender"), rs.getInt("cid"), rs.getString("imgpath"));

        });

    }

```

### **11. UserPurchaseRepository.java**

```

package com.to.repositories;

import java.util.List;

import com.to.entities.UserPurchase;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

public interface UserPurchaseRepository {

    List<UserPurchase> fetchAll(Integer userId) throws EtResourceNotFoundException;

    UserPurchase findByld(Integer userId, Integer purld) throws EtResourceNotFoundException;

    void create(UserPurchase userPurchase) throws EtBadRequestException;

    void update(Integer userId, Integer purld, UserPurchase userPurchase) throws
EtBadRequestException;

}

```

### **12. UserPurchaseRepositoryImpl.java**

```

package com.to.repositories;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.objenesis.instantiator.basic.NewInstanceInstantiator;
import org.springframework.stereotype.Repository;

import com.to.entities.UserPurchase;
import com.to.exceptions.EtAuthException;

```

```

import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

@Repository
public class UserPurchaseRepositoryImpl implements UserPurchaseRepository {
    private static final String SQL_FIND_ALL = "SELECT * FROM user_purchases where user_id=?";
    private static final String SQL_CREATE = "INSERT INTO user_purchases(product_id, pdate, cat_id,
quantity, price, total_price,user_id) VALUES (?, ?, ?, ?, ?, ?, ?)";
    private static final String SQL_FIND_BY_ID = "SELECT * FROM user_purchases where pid=? and
user_id=?";

    @Autowired
    JdbcTemplate jdbcTemplate;

    @Override
    public List<UserPurchase> fetchAll(Integer userId) throws EtResourceNotFoundException {

        try {
            return jdbcTemplate.query(SQL_FIND_ALL, new Object[] { userId },
userPurchaseRowMapper);

        } catch (Exception e) {
            throw new EtResourceNotFoundException(
                "Purchase details not found for user id :"+ userId + " purchased id :");
        }
    }

    @Override
    public UserPurchase findById(Integer userId, Integer purId) throws EtResourceNotFoundException {
        try {
            return jdbcTemplate.queryForObject(SQL_FIND_BY_ID, new Object[] { purId, userId, },
userPurchaseRowMapper);

        } catch (Exception e) {
            throw new EtResourceNotFoundException(
                "Purchase details not found for user id :"+ userId + " purchased id :"+
purId);
        }
    }

    // row mapper
    private RowMapper<UserPurchase> userPurchaseRowMapper = ((rs, rowNum) -> {

        return new UserPurchase(rs.getInt("pid"), rs.getInt("product_id"), rs.getString("pdate"),
rs.getInt("cat_id"),
            rs.getInt("quantity"), rs.getInt("price"), rs.getInt("total_price"),
rs.getInt("user_id"));
    });

```

```

        @Override
        public void create(UserPurchase userPurchase) throws EtBadRequestException {

            try {
                jdbcTemplate.update(SQL_CREATE, userPurchase.getProduct_id(),
userPurchase.getPdate(),
                                userPurchase.getCat_id(), userPurchase.getQuantity(),
userPurchase.getPrice(),
                                userPurchase.getTotal_price(), userPurchase.getUser_id());

            } catch (Exception e) {
                throw new EtAuthException("Invalid details. Failed to insert user purchase details");
            }

        }

        @Override
        public void update(Integer userId, Integer purId, UserPurchase userPurchase) throws
EtBadRequestException {
            // TODO Auto-generated method stub

        }

    }

```

### **13. UserRepository.java**

```
package com.to.repositories;
```

```
import com.to.entities.User;
```

```
import com.to.exceptions.EtAuthException;
```

```
public interface UserRepository {
```

```
    Integer create(String fname, String lname, String email, String password) throws EtAuthException;
```

```
    User findByEmailAndPassword(String email, String password) throws EtAuthException;
```

```
    Integer getCountByEmail(String email);
```

```
    User findById(Integer userId);
```

```
}
```

### **14. UserRepositoryImpl.java**

```
package com.to.repositories;
```

```

import java.beans.Statement;
import java.security.interfaces.RSAKey;
import java.sql.PreparedStatement;
import org.mindrot.jbcrypt.BCrypt;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;
import org.springframework.stereotype.Repository;
import com.to.entities.User;
import com.to.exceptions.EtAuthException;
import com.to.services.UserService;
import net.bytebuddy.asm.Advice.Return;

@Repository
public class UserRepositoryImpl implements UserRepository {
    private static final String SQL_CREATE = "INSERT INTO user_login (first_name, last_name, email, password) VALUES (?, ?, ?, ?)";
    private static final String SQL_COUNT_BY_EMAIL = "SELECT COUNT(*) FROM user_login WHERE email=?";
    private static final String SQL_FIND_BY_ID = "SELECT * FROM user_login WHERE userid=?";
    private static final String SQL_FIND_BY_EMAIL = "SELECT * FROM user_login WHERE email=?";
    private static final String SQL_FIND_BY_EMAIL_USER_ID = "SELECT * FROM user_login WHERE email=?";

    @Autowired
    JdbcTemplate jdbcTemplate;

    // method for creating new user
    @Override
    public Integer create(String fname, String lname, String email, String password) throws EtAuthException {
        Integer userId = null;
        // generating the hashed password
        String hashedPassword = BCrypt.hashpw(password, BCrypt.gensalt(10));

        try {
            jdbcTemplate.update(SQL_CREATE, fname, lname, email, hashedPassword);
            User user;
            user = findByEmail_userId(email);
            userId = user.getUserId();
        } catch (Exception e) {
            throw new EtAuthException("Invalid details. Failed to create account");
        }

        return userId;
    }

```

```

    }

    // method for getting the user by email id
    public User findByEmail_userId(String email) {

        return jdbcTemplate.queryForObject(SQL_FIND_BY_EMAIL_USER_ID, new Object[] { email },
userRowMapper);
    }

//method for checking the email id and password is correct or not
@Override
public User findByEmailAndPassword(String email, String password) throws EtAuthException {

    try {
        // getting the user record from db
        User user = jdbcTemplate.queryForObject(SQL_FIND_BY_EMAIL, new Object[] { email
}, userRowMapper);

        // checking the password
        if (!BCrypt.checkpw(password, user.getPassword()))
            throw new EtAuthException("Invalid Email/Pasword");
        return user;

    } catch (Exception e) {
        throw new EtAuthException("Invalid Email/Pasword");
    }
}

// method for checking the email is already used or not
@Override
public Integer getCountByEmail(String email) {

    return jdbcTemplate.queryForObject(SQL_COUNT_BY_EMAIL, new Object[] { email },
Integer.class);
}

// method for finding the user by id
@Override
public User findById(Integer userId) {

    return jdbcTemplate.queryForObject(SQL_FIND_BY_ID, new Object[] { userId },
userRowMapper);
}

// row mapper
private RowMapper<User> userRowMapper = ((rs, rowNum) -> {
    return new User(rs.getInt("userid"), rs.getString("first_name"), rs.getString("last_name"),
rs.getString("email"), rs.getString("password"));
});

```

```
});  
  
}
```

---

**package com.to.services**

**1. AdminLoginService.java**

```
package com.to.services;  
  
import com.to.entities.AdminLogin;  
import com.to.exceptions.EtAuthException;  
  
public interface AdminLoginService {  
  
    AdminLogin adminLogin(String adminId, String password) throws EtAuthException;  
  
    void adminPasswordChange(String adminId, String password, String newPass) throws  
EtAuthException;  
  
}
```

**2. AdminLoginServiceImpl.java**

```
package com.to.services;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
import com.to.entities.AdminLogin;  
import com.to.exceptions.EtAuthException;  
import com.to.repositories.AdminLoginRepository;  
  
@Service  
@Transactional  
public class AdminLoginServiceImpl implements AdminLoginService {  
  
    @Autowired  
    AdminLoginRepository adminLoginRepository;  
  
    @Override  
    public AdminLogin adminLogin(String adminId, String password) throws EtAuthException {
```

```

        return adminLoginRepository.findById(adminId, password);
    }

    @Override
    public void adminPasswordChange(String adminId, String password, String newPass) throws
    EtAuthException {

        adminLoginRepository.update(adminId, password, newPass);
    }
}

```

### **3. AdminReportService.java**

```

package com.to.services;

import java.util.List;

import com.to.entities.User;

public interface AdminReportService {
    List<User> getAllLoggedUsersDetails();
}

```

### **4. AdminReportServiceImpl.java**

```

package com.to.services;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.to.entities.User;
import com.to.repositories.AdminReports;

@Service
@Transactional
public class AdminReportServiceImpl implements AdminReportService {
    @Autowired
    AdminReports adminReports;

    @Override
    public List<User> getAllLoggedUsersDetails() {
        // TODO Auto-generated method stub
    }
}

```

```
        return adminReports.getAllLoggedUsers();
    }
}
```

### **5. CategoryService.java**

```
package com.to.services;
import java.util.List;
import com.to.entities.Category;
import com.to.exceptions.EtBadRequestException;
public interface CategoryService {

    void createCategory(String categoryName);

    void updateCategory(Integer catId, String catName) throws EtBadRequestException;

    void deleteCategory(Integer catId) ;

    Category getCategoryById(Integer catId);

    List<Category> getAllCategories();

}
```

### **6. CategoryServiceImpl.java**

```
package com.to.services;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.to.entities.Category;
import com.to.exceptions.EtBadRequestException;
import com.to.repositories.CategoryRepository;

@Service
@Transactional
public class CategoryServiceImpl implements CategoryService {

    @Autowired
    CategoryRepository categoryRepository;
```



```

@Override
public void createCategory(String categoryName) {
    categoryRepository.create(categoryName);
}

@Override
public void updateCategory(Integer catId, String catName) throws EtBadRequestException {
    categoryRepository.update(catId, catName);
}

@Override
public void deleteCategory(Integer catId) {
    categoryRepository.delete(catId);
}

@Override
public Category getCategoryById(Integer catId) {

    return categoryRepository.getById(catId);
}

@Override
public List<Category> getAllCategories() {

    return categoryRepository.getAllCategories();
}
}

```

## **7. ProductServices.java**

```

package com.to.services;
import java.util.List;
import com.to.entities.Product;
import com.to.entities.UserPurchase;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

public interface ProductServices {
    void createNewProduct(Product product) throws EtBadRequestException;
    void updateProduct(Integer userId, Integer purId, UserPurchase userPurchase) throws
EtBadRequestException;

    void deleteProduct(Integer pId) throws EtResourceNotFoundException;
}

```

```
Product getProductById(Integer pid) throws EtResourceNotFoundException;

List<Product> getAllProductDetails() throws EtResourceNotFoundException;

}
```

## **8. ProductServicesImpl.java**

```
package com.to.services;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.to.entities.Product;
import com.to.entities.UserPurchase;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;
import com.to.repositories.ProductRepository;

@Service
@Transactional
public class ProductServicesImpl implements ProductServices {

    @Autowired
    ProductRepository productRepository;

    @Override
    public void createNewProduct(Product product) throws EtBadRequestException {
        productRepository.create(product);
    }

    @Override
    public void updateProduct(Integer userId, Integer purId, UserPurchase userPurchase) throws
EtBadRequestException {
        // TODO Auto-generated method stub
    }

    @Override
    public void deleteProduct(Integer pld) throws EtResourceNotFoundException {
        productRepository.delete(pld);
    }
}
```

```

    @Override
    public Product getProductById(Integer pid) throws EtResourceNotFoundException {
        // TODO Auto-generated method stub
        return productRepository.getByid(pid);
    }

    @Override
    public List<Product> getAllProductDetails() throws EtResourceNotFoundException {

        return productRepository.getAllProduct();
    }
}

```

### **9. UserPurchaseService.java**

```

package com.to.services;

import java.util.List;

import com.to.entities.Product;
import com.to.entities.UserPurchase;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;

public interface UserPurchaseService {
    List<UserPurchase> fetchAllPurchases(Integer userId);

    UserPurchase fetchUserPurchaseById(Integer userId, Integer purId) throws
EtResourceNotFoundException;

    void addPurchase(UserPurchase userPurchase) throws EtBadRequestException;

    void updatePurchase(Integer userId, Integer purId, UserPurchase userPurchase) throws
EtBadRequestException;

    List<Product> getAllProductByCategory(Integer catId);
}

```

### **10. UserPurchaseServiceImpl.java**

```

package com.to.services;

import java.util.List;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.to.entities.Product;
import com.to.entities.UserPurchase;
import com.to.exceptions.EtBadRequestException;
import com.to.exceptions.EtResourceNotFoundException;
import com.to.repositories.UserProductDisplay;
import com.to.repositories.UserPurchaseRepository;

@Service
@Transactional
public class UserPurchaseServiceImpl implements UserPurchaseService {

    @Autowired
    UserPurchaseRepository userPurchaseRepository;
    @Autowired
    UserProductDisplay userProductDisplay;

    @Override
    public List<UserPurchase> fetchAllPurchases(Integer userId) {
        return userPurchaseRepository.fetchAll(userId);
    }

    @Override
    public UserPurchase fetchUserPurchaseById(Integer userId, Integer purId) throws
EtResourceNotFoundException {

        return userPurchaseRepository.findById(userId, purId);
    }

    // method for saving the purchased product details
    public void addPurchase(UserPurchase userPurchase) throws EtBadRequestException {
        userPurchaseRepository.create(userPurchase);
    }

    @Override
    public void updatePurchase(Integer userId, Integer purId, UserPurchase userPurchase) throws
EtBadRequestException {
        // TODO Auto-generated method stub
    }

    @Override
    public List<Product> getAllProductByCategory(Integer catId) {

```

```
        return userProductDisplay.getAllProductByCategory(catId);
    }
}
```

### **11. UserService.java**

```
package com.to.services;

import com.to.entities.User;
import com.to.exceptions.EtAuthException;

public interface UserService {

    User validateUser(String email, String password) throws EtAuthException;

    User registerUser(String fname, String lname, String email, String password) throws EtAuthException;
}
```

### **12. UserServiceImpl.java**

```
package com.to.services;
import java.util.regex.Pattern;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.to.entities.User;
import com.to.exceptions.EtAuthException;
import com.to.repositories.UserRepository;

@Service
@Transactional
public class UserServiceImpl implements UserService {

    @Autowired
    UserRepository userRepository;

    // method for validating the user name password for logging
    @Override
    public User validateUser(String email, String password) throws EtAuthException {
```

```

        if (email != null)
            email = email.toLowerCase();

        // returnig the user
        return userRepository.findByEmailAndPassword(email, password);

    }

    @Override
    public User registerUser(String fname, String lname, String email, String password) throws
    EtAuthException {

        // pettern for validating email
        Pattern pattern = Pattern.compile("^(.+)?@(.+)$");
        if (email != null)
            email = email.toLowerCase();

        if (!pattern.matcher(email).matches())
            throw new EtAuthException("Invalid email format");

        // getting the email count
        Integer count = userRepository.getCountByEmail(email);
        if (count > 0)
            throw new EtAuthException("Email Already in use");

        // creating the user and getting the user id back
        Integer userId = userRepository.create(fname, lname, email, password);

        // returning the user detail wth jwt token
        return userRepository.findById(userId);

    }

}

```

---

```
package com.to.entities
```

### **1. AdminLogin.java**

```

package com.to.entities;

public class AdminLogin {
    private String adminId;
    private String password;
    private String newPass;

```

```

// constructor
public AdminLogin(String adminId, String password, String newPass) {
    super();
    this.adminId = adminId;
    this.password = password;
    this.newPass = newPass;
}

public AdminLogin(String adminId, String password) {
    super();
    this.adminId = adminId;
    this.password = password;
}

// getter and setters
public String getAdminId() {
    return adminId;
}

public void setAdminId(String adminId) {
    this.adminId = adminId;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getNewPass() {
    return newPass;
}

public void setNewPass(String newPass) {
    this.newPass = newPass;
}

@Override
public String toString() {
    return "AdminLogin [adminId=" + adminId + ", password=" + password + ", newPass=" +
newPass + "]\n";
}
}

```

## **2. Category.java**

```

package com.to.entities;

public class Category {
    private Integer catId;
    private String catName;

    //constructor using fields
    public Category(Integer catId, String catName) {
        super();
        this.catId = catId;
        this.catName = catName;
    }

    public Category() {
        // TODO Auto-generated constructor stub
    }

    // getter and setters
    public Integer getCatId() {
        return catId;
    }

    public void setCatId(Integer catId) {
        this.catId = catId;
    }

    public String getCatName() {
        return catName;
    }

    public void setCatName(String catName) {
        this.catName = catName;
    }

    @Override
    public String toString() {
        return "Category [catId=" + catId + ", catName=" + catName + "]";
    }
}

```

### **3. Product.java**

```

package com.to.entities;

public class Product {

```



```
private Integer pid;
private String pname;
private String pdescription;
private Integer price;
private String gender;
private Integer cid;
private String imagePath;

// constructor using fields
public Product(Integer pid, String pname, String pdescription, Integer price, String gender, Integer cid,
               String imagePath) {
    super();
    this.pid = pid;
    this.pname = pname;
    this.pdescription = pdescription;
    this.price = price;
    this.gender = gender;
    this.cid = cid;
    this.imagePath = imagePath;
}

public Product() {
    // TODO Auto-generated constructor stub
}

public Integer getPid() {
    return pid;
}

public void setPid(Integer pid) {
    this.pid = pid;
}

public String getPname() {
    return pname;
}

public void setPname(String pname) {
    this.pname = pname;
}

public String getPdescription() {
    return pdescription;
}

public void setPdescription(String pdescription) {
    this.pdescription = pdescription;
}
```

```

    public Integer getPrice() {
        return price;
    }

    public void setPrice(Integer price) {
        this.price = price;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public Integer getCid() {
        return cid;
    }

    public void setCid(Integer cid) {
        this.cid = cid;
    }

    public String getImagePath() {
        return imagePath;
    }

    public void setImagePath(String imagePath) {
        this.imagePath = imagePath;
    }

    @Override
    public String toString() {
        return "Product [pid=" + pid + ", pname=" + pname + ", pdescription=" + pdescription + ",
price=" + price
        + ", gender=" + gender + ", cid=" + cid + ", imagePath=" + imagePath + "]";
    }
}

```

#### **4. User.java**

```
package com.to.entities;
```

```
public class User {
    private Integer userId;
    private String fname;
    private String lname;
    private String email;
    private String password;

    // constructor
    public User(Integer userId, String fname, String lname, String email, String password) {
        super();
        this.userId = userId;
        this.fname = fname;
        this.lname = lname;
        this.email = email;
        this.password = password;
    }

    public User(Integer userId, String fname, String lname, String email) {
        super();
        this.userId = userId;
        this.fname = fname;
        this.lname = lname;
        this.email = email;
    }

    @Override
    public String toString() {
        return "User [userId=" + userId + ", fname=" + fname + ", lname=" + lname + ", email=" + email
+ ", password="
        + password + "];"
    }

    // getters and setters
    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
    }

    public String getFname() {
        return fname;
    }

    public void setFname(String fname) {
        this.fname = fname;
    }
}
```

```

    }

    public String getLname() {
        return lname;
    }

    public void setLname(String lname) {
        this.lname = lname;
    }

    public String getEmail() {
        return email;
    }

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

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}

```

#### 4. UserPurchase.java

```

package com.to.entities;
public class UserPurchase {
    private Integer pid;
    private Integer product_id;
    private String pdate;
    private Integer cat_id;
    private Integer quantity;
    private Integer price;
    private Integer total_price;
    private Integer user_id;

    // constructor

    public UserPurchase() {
    }
}

```

```
public UserPurchase(Integer pid, Integer product_id, String pdate, Integer cat_id, Integer quantity, Integer price,
```

```
Integer total_price, Integer user_id) {  
    super();  
    this.pid = pid;  
    this.product_id = product_id;  
    this.pdate = pdate;  
    this.cat_id = cat_id;  
    this.quantity = quantity;  
    this.price = price;  
    this.total_price = total_price;  
    this.user_id = user_id;  
}
```

```
public UserPurchase(Integer product_id, String pdate, Integer cat_id, Integer quantity, Integer price, Integer total_price, Integer user_id) {
```

```
    super();  
    // this.pid = pid;  
    this.product_id = product_id;  
    this.pdate = pdate;  
    this.cat_id = cat_id;  
    this.quantity = quantity;  
    this.price = price;  
    this.total_price = total_price;  
    this.user_id = user_id;  
}
```

```
// getter and setters
```

```
public Integer getPid() {  
    return pid;  
}
```

```
public void setPid(Integer pid) {  
    this.pid = pid;  
}
```

```
public Integer getProduct_id() {  
    return product_id;  
}
```

```
public void setProduct_id(Integer product_id) {  
    this.product_id = product_id;  
}
```

```
public String getPdate() {  
    return pdate;  
}
```

```
public void setPdate(String pdate) {
    this.pdate = pdate;
}

public Integer getCat_id() {
    return cat_id;
}

public void setCat_id(Integer cat_id) {
    this.cat_id = cat_id;
}

public Integer getQuantity() {
    return quantity;
}

public void setQuantity(Integer quantity) {
    this.quantity = quantity;
}

public Integer getPrice() {
    return price;
}

public void setPrice(Integer price) {
    this.price = price;
}

public Integer getTotal_price() {
    return total_price;
}

public void setTotal_price(Integer total_price) {
    this.total_price = total_price;
}

public Integer getUser_id() {
    return user_id;
}

public void setUser_id(Integer user_id) {
    this.user_id = user_id;
}

// to string method
@Override
public String toString() {
```

```

        return "UserPurchase [pid=" + pid + ", product_id=" + product_id + ", pdate=" + pdate + ",
cat_id=" + cat_id
        + ", quantity=" + quantity + ", price=" + price + ", total_price=" + total_price +
", user_id="
        + user_id + "];
    }
}

```

---

```
package com.to;
```

### 1. Constants.java

```

package com.to;

//class for declaring the constants
public class Constants {

    public static final String API_SECRET_KEY_STRING = "sportshoes";

    public static final long TOKEN_VALIDITY = 2 * 60 * 60 * 1000;

}

```

### 2. SimplyLearnSportShoesProjectApplication.java

```

package com.to;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import com.to.filters.AdminAuthFilter;
import com.to.filters.AuthFilter;
import com.to.services.UserService;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@SpringBootApplication
@EnableSwagger2
public class SimplyLearnSportShoesProjectApplication implements CommandLineRunner {

    private Logger logger = LoggerFactory.getLogger(this.getClass());
    @Autowired

```

```

UserService userService;

public static void main(String[] args) {

    SpringApplication.run(SimplyLearnSportShoesProjectApplication.class, args);
}

// registering a user filter bean
@Bean
public FilterRegistrationBean<AuthFilter> filterRegistrationBean() {
    FilterRegistrationBean<AuthFilter> registrationBean = new FilterRegistrationBean<>();
    AuthFilter authFilter = new AuthFilter();
    registrationBean.setFilter(authFilter);
    // set the url for scanning the request
    registrationBean.addUrlPatterns("/api/userPurchase/*");
    return registrationBean;
}

// registering a admin filter bean
@Bean
public FilterRegistrationBean<AdminAuthFilter> adminfilterRegistrationBean() {
    FilterRegistrationBean<AdminAuthFilter> registrationBean = new FilterRegistrationBean<>();
    AdminAuthFilter adminAuthFilter = new AdminAuthFilter();
    registrationBean.setFilter(adminAuthFilter);
    // set the url for scanning the request
    registrationBean.addUrlPatterns("/api/admin/category/*", "/api/admin/product/*",
"/api/admin/report/*");
    return registrationBean;
}

@Override
public void run(String... args) throws Exception {

}
}

```

---

### **Application.properties**

#### **#mysql properties**

```

spring.datasource.url=jdbc:mysql://localhost:3306/sporty_shoes_ecom_website
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

```

#### **#swagger**

```

spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER

```



## pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.7</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.sachinjava</groupId>
  <artifactId>simply_learn_sport_shoes_project</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>simply_learn_sport_shoes_project</name>
  <description>simply learn e-commerce portal sportyshoes.com</description>
  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- dependency for password encryption and decryption -->
    <!-- https://mvnrepository.com/artifact/org.mindrot/jbcrypt -->
    <dependency>
      <groupId>org.mindrot</groupId>
      <artifactId>jbcrypt</artifactId>
      <version>0.4</version>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
</project>
```

```

        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <!-- swagger dependancies -->
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-boot-starter</artifactId>
        <version>3.0.0</version>
    </dependency>
    <!-- jwt token dependancies -->
    <!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
        <version>0.9.1</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
        <version>3.0.0</version>
    </dependency>
    <!-- swagger dependancies end here -->

    <!-- API, java.xml.bind module -->
    <dependency>
        <groupId>jakarta.xml.bind</groupId>
        <artifactId>jakarta.xml.bind-api</artifactId>
        <version>2.3.2</version>
    </dependency>

    <!-- Runtime, com.sun.xml.bind module -->
    <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
        <version>2.3.2</version>
    </dependency>

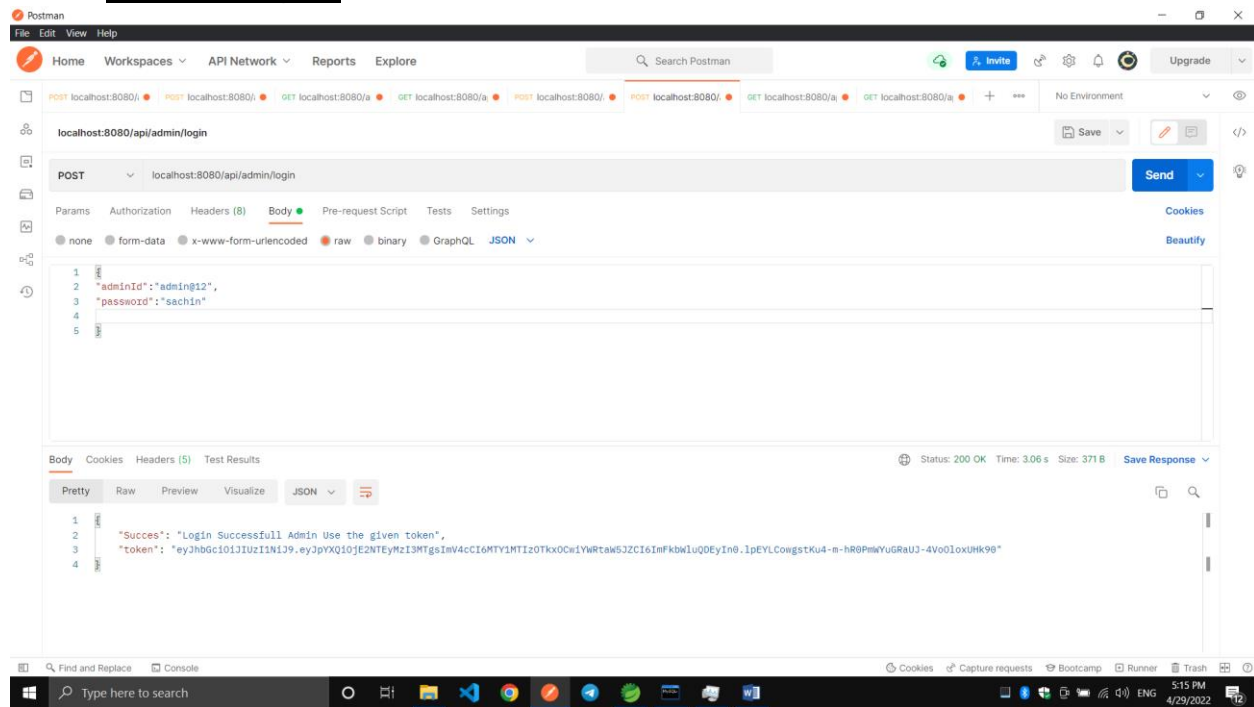
```

```
    </dependencies>

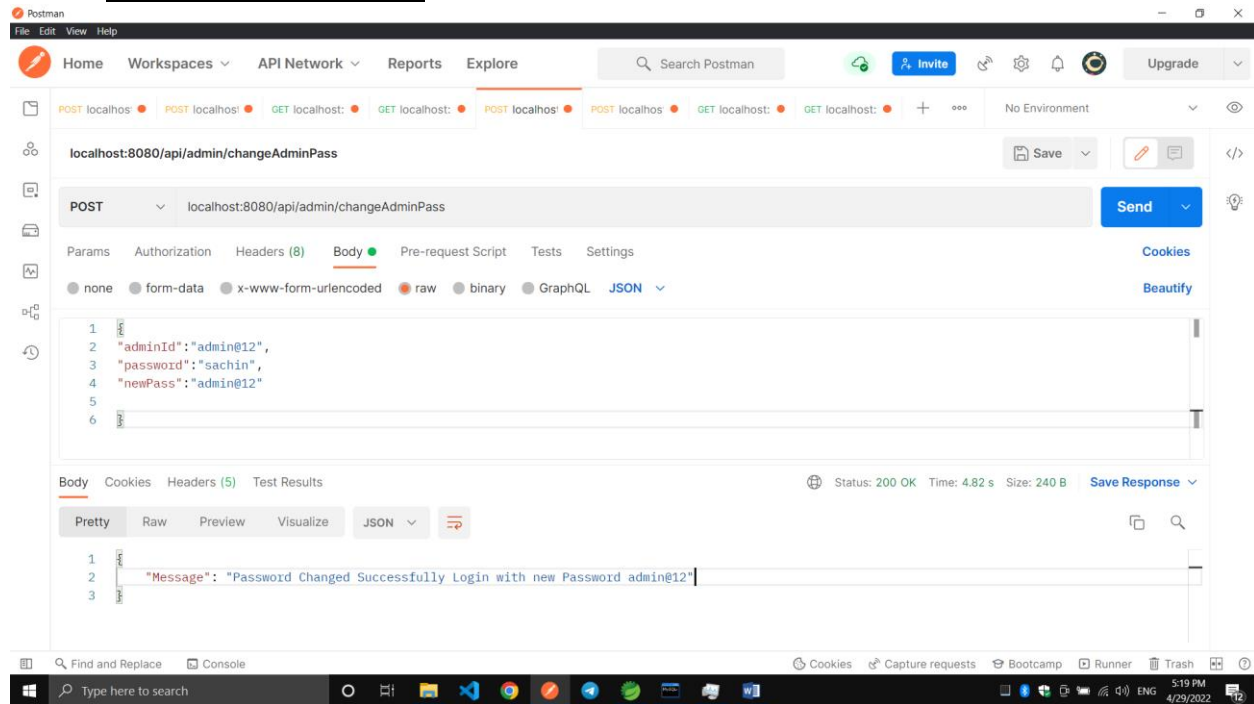
    <build>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
      </plugins>
    </build>
  </project>
```

## Screen Shots:-

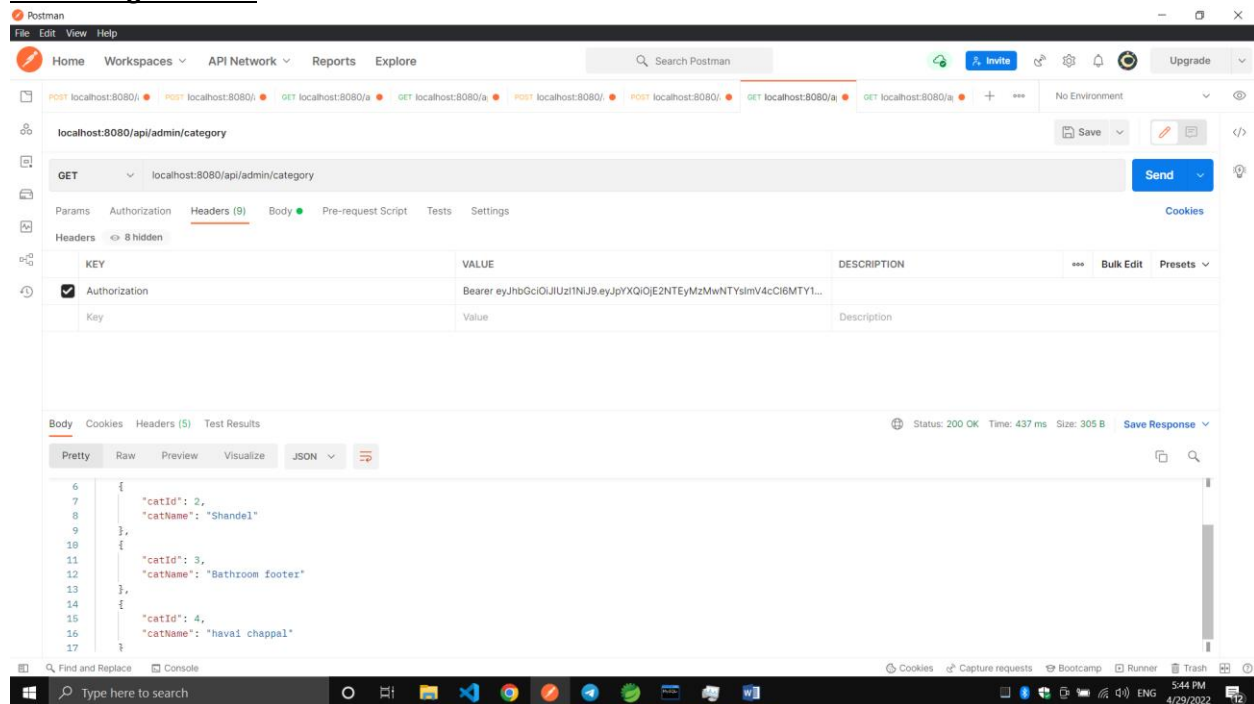
### 1. Admin Login:-



### 2. Admin Password Change:-



### 3.All Categories List:



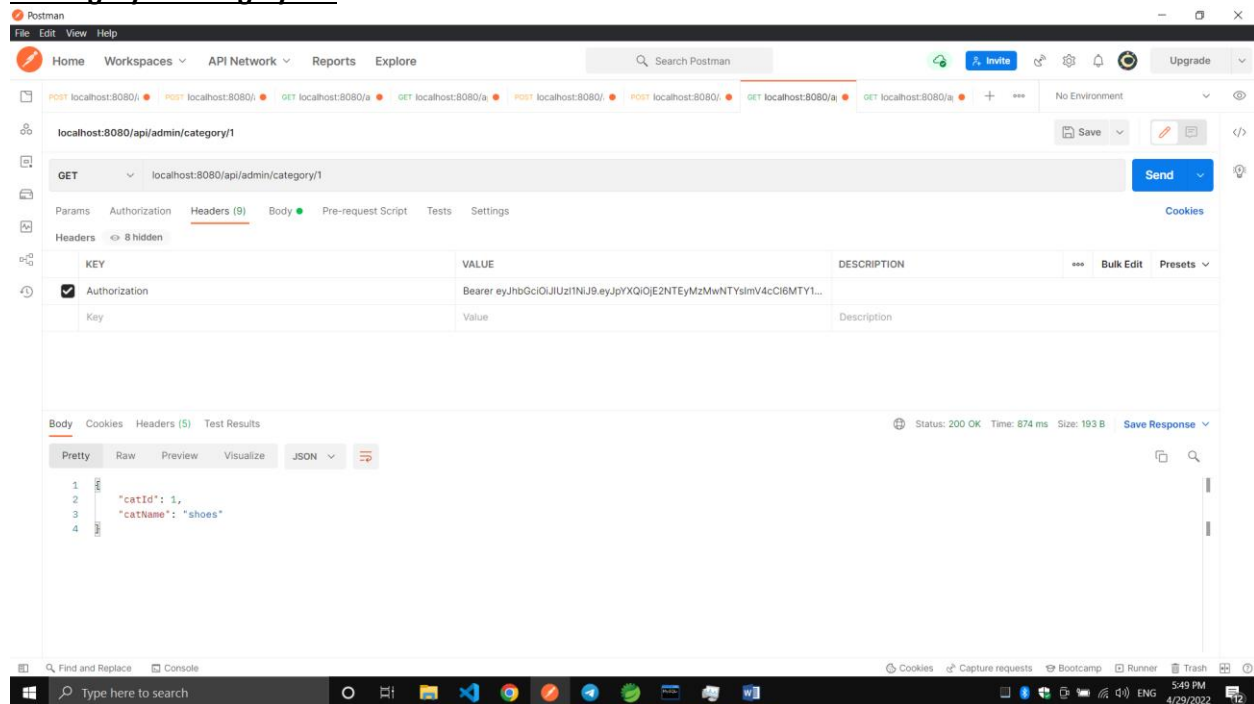
Postman interface showing a GET request to `localhost:8080/api/admin/category`. The response is a JSON array of 4 categories.

KEY	VALUE	DESCRIPTION
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlZ2NTEyMzWmNTY1MzV4cC8iMTY1...	
Key	Value	Description

Body (JSON):

```
6 {
7   "catId": 2,
8   "catName": "Shandel"
9 },
10 {
11   "catId": 3,
12   "catName": "Bathroom footer"
13 },
14 {
15   "catId": 4,
16   "catName": "havai chappal"
17 }
```

### 4.Category Fetching By Id:-



Postman interface showing a GET request to `localhost:8080/api/admin/category/1`. The response is a JSON object for category 1.

Body (JSON):

```
1 {
2   "catId": 1,
3   "catName": "shoes"
4 }
```

## 5. Adding new Category:-

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and navigation tabs (Home, Workspaces, API Network, Reports, Explore). A search bar and utility buttons (Invite, Settings, Notifications, Upgrade) are also present. The main workspace displays a collection of requests for 'localhost:8080'. The selected request is a POST to 'localhost:8080/api/admin/category/'. The 'Body' tab is active, showing a JSON payload: 

```
{  "cname": "Athletic Shoe"}
```

. The 'Send' button is visible. Below the request, the 'Test Results' section shows a status of '200 OK', a time of '849 ms', and a size of '196 B'. The 'Body' tab of the response is active, displaying a JSON message: 

```
{  "msg": "Category Details added"}
```

. The bottom status bar shows the system clock as 5:53 PM on 4/29/2022.

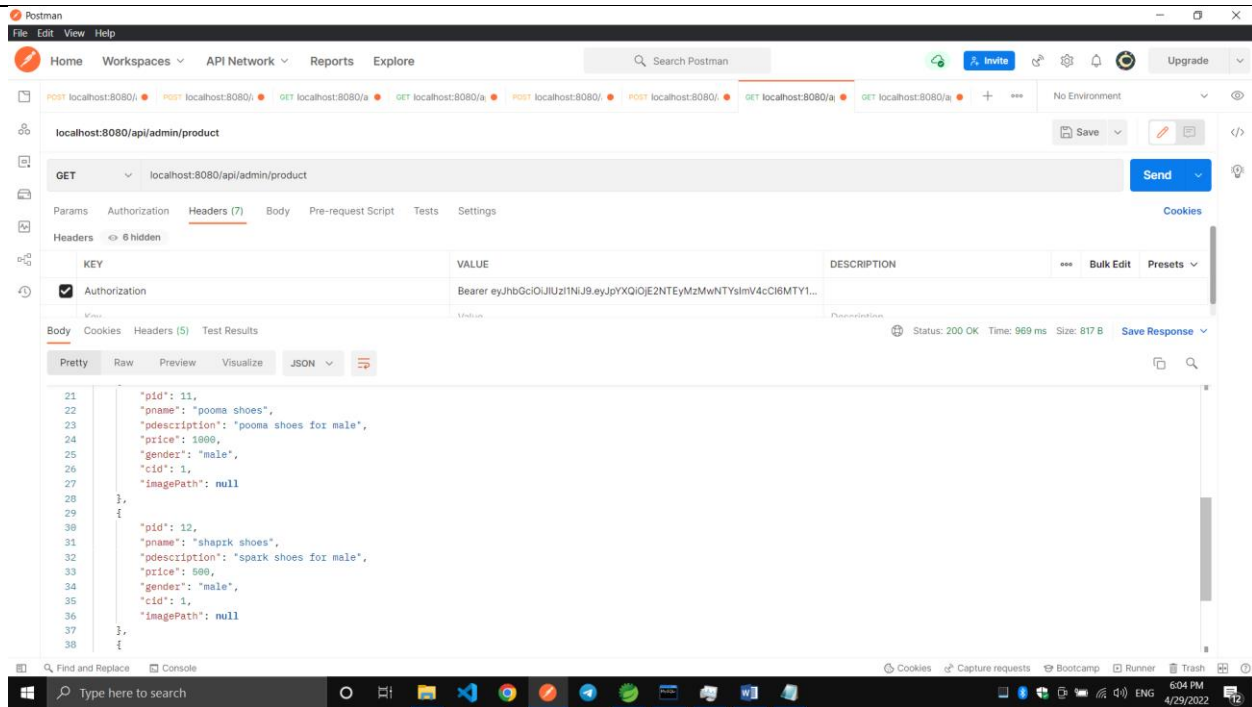
## 6. Deleting the Category By Id:

The screenshot shows the Postman application interface. The top bar is identical to the previous screenshot. The main workspace displays a collection of requests. The selected request is a DELETE to 'localhost:8080/api/admin/category/delete/4'. The 'Body' tab is active, showing an empty body. The 'Send' button is visible. Below the request, the 'Test Results' section shows a status of '200 OK', a time of '307 ms', and a size of '214 B'. The 'Body' tab of the response is active, displaying a JSON message: 

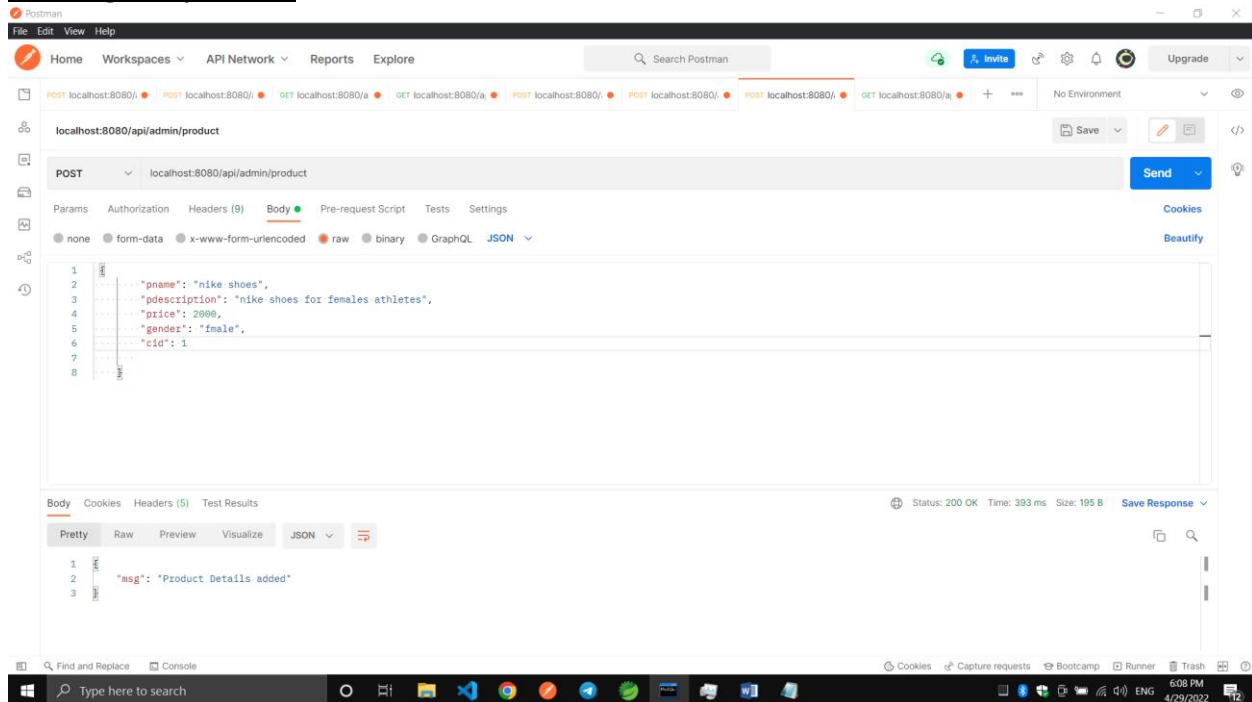
```
{  "Success": "Category Id : 4 Deleted Successfully"}
```

. The bottom status bar shows the system clock as 5:55 PM on 4/29/2022.

## 7. Getting the All Shoes Details:-



## 8. Adding new product:-



## 9. Getting the product details by Id:-

The screenshot shows the Postman interface with a GET request to `localhost:8080/api/admin/product/9`. The request has an Authorization header with a Bearer token. The response is a JSON object with the following details:

```
{
  "pid": 9,
  "pname": "nike shoes",
  "pdescription": "nike shoes for males",
  "price": 2000,
  "gender": "male",
  "cid": 1,
  "imagePath": null
}
```

Status: 200 OK, Time: 34 ms, Size: 286 B

## 10. Deleting the product by Id:

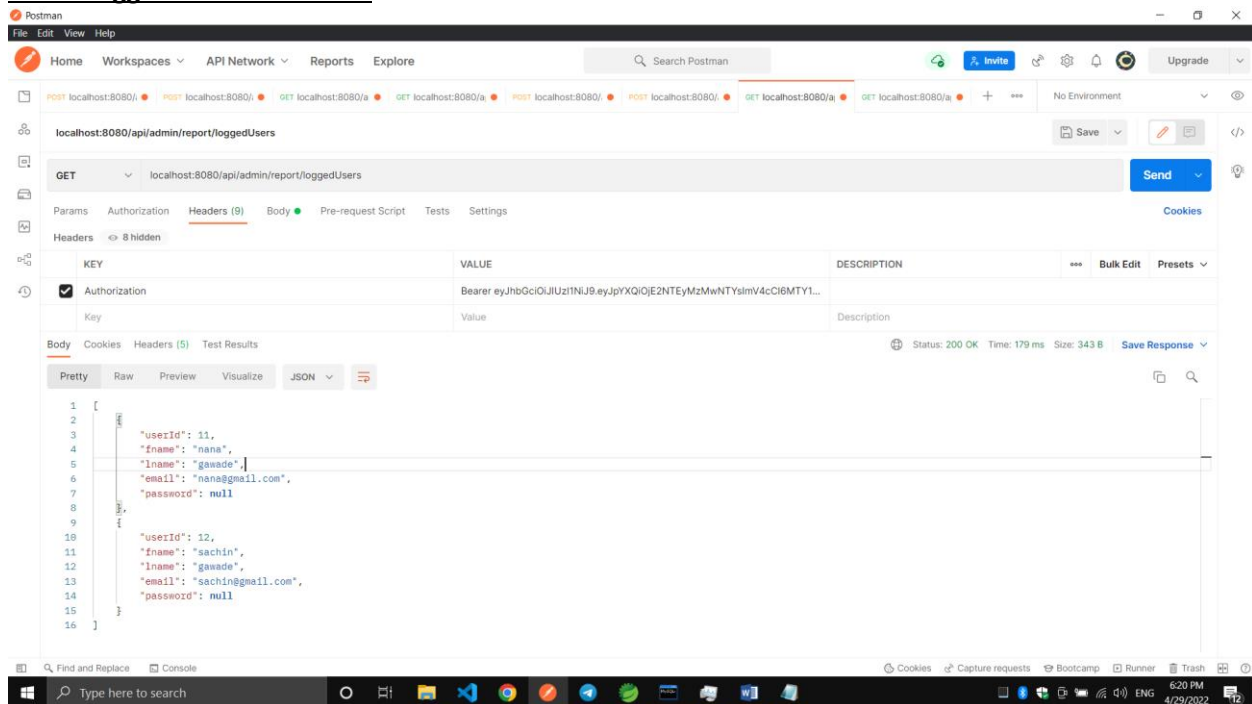
The screenshot shows the Postman interface with a DELETE request to `localhost:8080/api/admin/product/delete/9`. The request has an Authorization header with a Bearer token. The response is a JSON object with a success message:

```
{
  "Success": "Product Id : 9 Deleted Successfully"
}
```

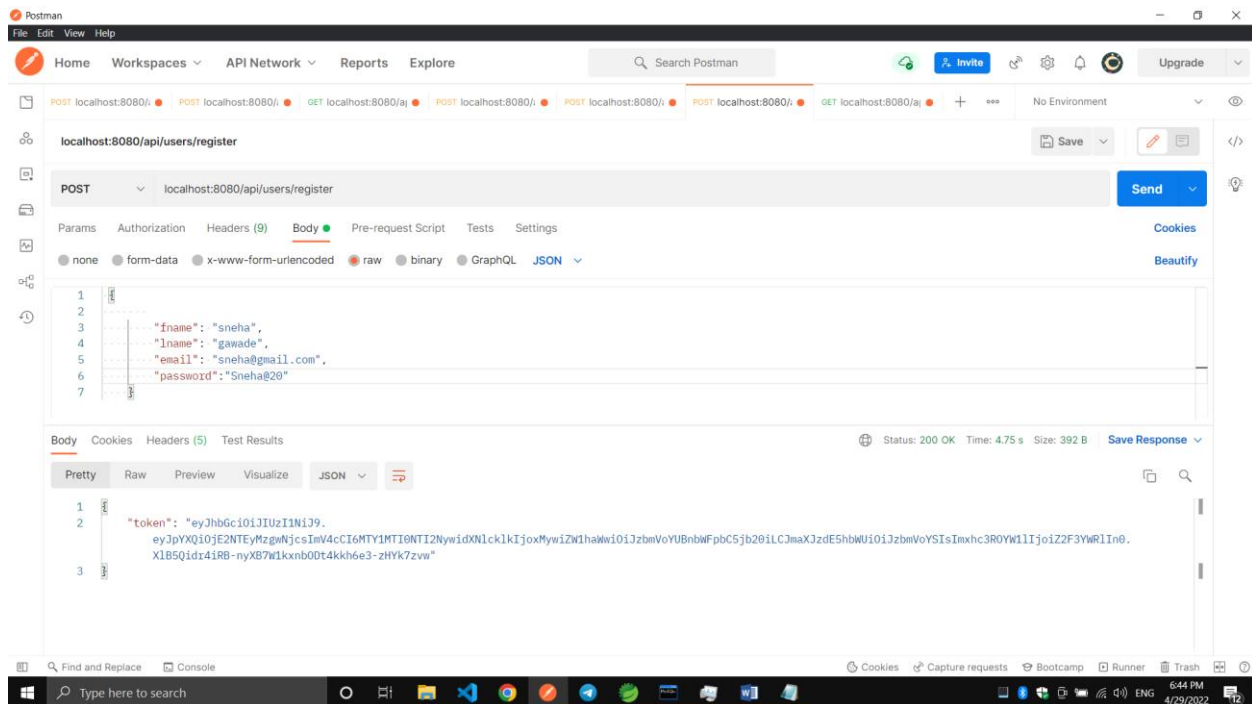
Status: 200 OK, Time: 2:11 s, Size: 213 B



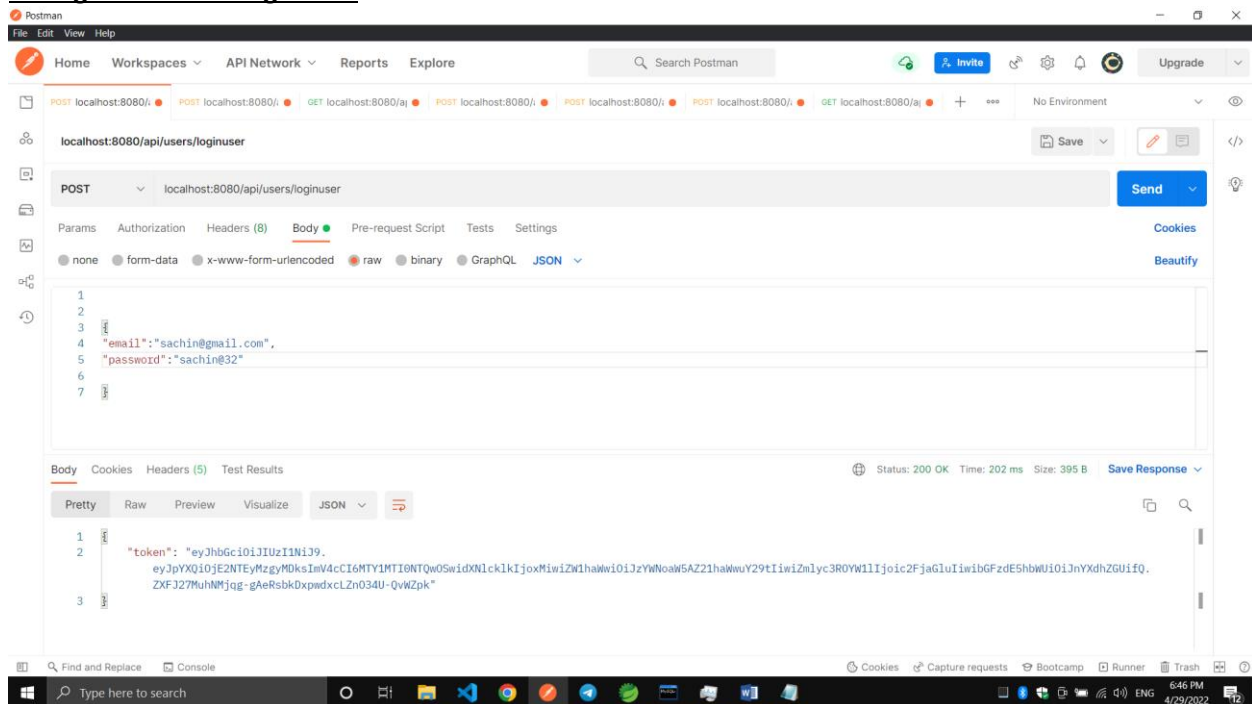
### 11.All Logged In User Details:-



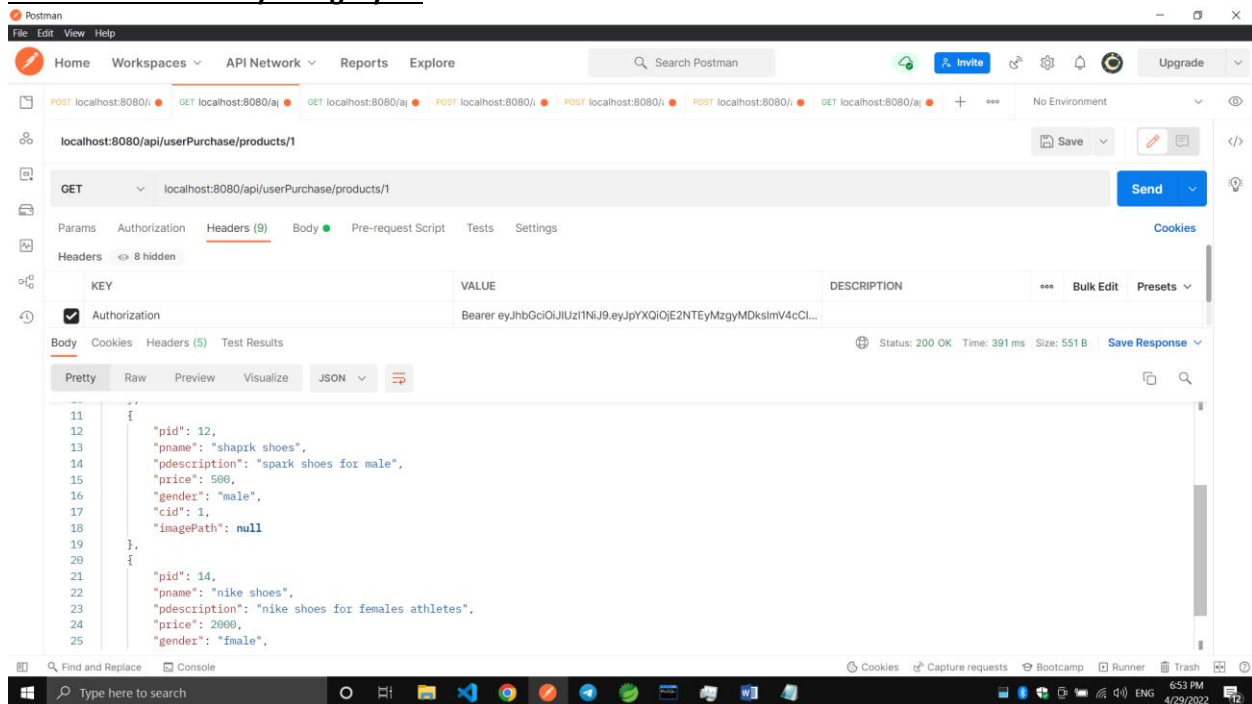
### 12.New User Registration :-



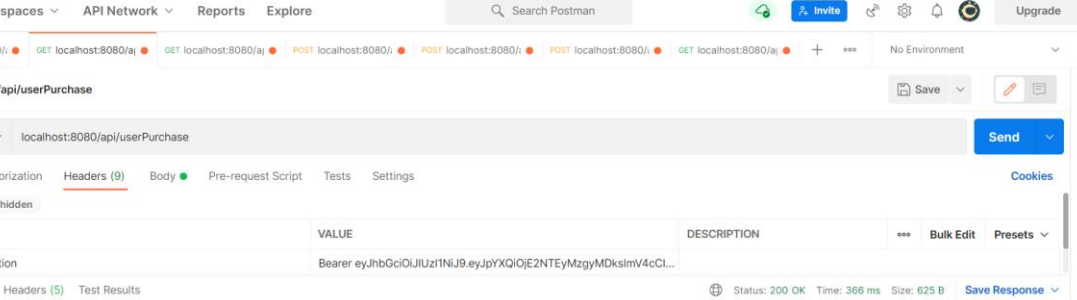
### 13.Login With Existing User:-



### 14:Product Details by Category Id:



## 15. Purchased Product Details of Logged User:-



The screenshot shows the Postman interface with a GET request to `localhost:8080/api/userPurchase`. The response is a JSON array containing two objects. The first object represents a purchased product with the following details:

- `pid`: 4
- `product_id`: 7
- `pdate`: "20/04/2022"
- `cat_id`: 1
- `quantity`: 2
- `price`: 2000
- `total_price`: 4000
- `user_id`: 12

The second object in the array is partially visible and contains:

- `pid`: 5
- `product_id`: 7
- `pdate`: "20/04/2022"
- `cat_id`: 1

# THANK YOU