|  |  |
| --- | --- |
| **Full Name** | **Vajid Ali** |
| **Batch** | **Apr 2022** |
| **Project Title** | **E-commerce Website for Sporty Shoes** |
| **Project Submission Date** | **14-Aug – 2022** |
| **Git Hub Link** | **https://github.com/vajid-ali/simply\_learn\_sport\_shoes\_project** |

|  |
| --- |
| 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 tocken  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 tocken 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 aftefr 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 tocken 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 = productServices.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 = productServices.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  productServices.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 tocken 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  productServices.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 tocken  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 tocken  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 tocken 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 tocken 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);  }  }   1. **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);    }    }   1. **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 getById(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 pId);  Product getById(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 pId) {  try {  int i = jdbcTemplate.update(SQL\_DELETE\_BY\_ID, new Object[] { pId });  if (i == 0)  throw new EtResourceNotFoundException("Sorry The given produt id: " + pId + " not available ");  } catch (Exception e) {  throw new EtResourceNotFoundException(e.getMessage());  }  }  @Override  public Product getById(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 produt 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 findById(Integer userId, Integer purId) throws EtResourceNotFoundException;  void create(UserPurchase userPurchase) throws EtBadRequestException;  void update(Integer userId, Integer purId, 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;  }   1. **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);  }  }   1. **AdminReportService.java**   package com.to.services;  import java.util.List;  import com.to.entities.User;  public interface AdminReportService {  List<User> getAllLoggedUsersDetails();  }   1. **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();  }  }   1. **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();      }   1. **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();  }  }   1. **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;  }   1. **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 pId) throws EtResourceNotFoundException {  productRepository.delete(pId);  }  @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();  }  }   1. **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);  }   1. **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);  }  }   1. **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;  }   1. **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 tocken  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 + "]";  }  }  **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;  }  }   1. **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 settetrs  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;  }   1. **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);  }  // registerning 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;  }  // registerning 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>  <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 tocken 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-**      1. **Admin Password Change:-**     **3.All Categories List:**    **4.Category Fetching By Id:-**    **5.Adding new Category:-**    **6.Deleting the Category By Id:**    **7.Getting the All Shoes Details:-**    **8.Adding new product:-**    **9.Getting the product details by Id:-**    **10.Deleting the product by Id:**    **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:-** |

**THANK YOU**