package com.loanCycle.controller;

import com.loanCycle.model.Group;

import com.loanCycle.model.User;

import com.loanCycle.service.GroupService;

import com.loanCycle.service.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/users")

@RequiredArgsConstructor

@CrossOrigin

public class UserController {

private final UserService userService;

private final GroupService groupService;

// ✅ Get all users (admin feature)

@GetMapping

public List<User> getAllUsers() {

return userService.getAllUsers();

}

// ✅ Get user by ID

@GetMapping("/{id}")

public User getUserById(@PathVariable Long id) {

return userService.getAllUsers().stream()

.filter(u -> u.getUserId().equals(id))

.findFirst()

.orElse(null);

}

// ✅ Assign user to a group

@PutMapping("/{userId}/assign-group/{groupId}")

public User assignUserToGroup(@PathVariable Long userId, @PathVariable Long groupId) {

User user = userService.getAllUsers().stream()

.filter(u -> u.getUserId().equals(userId))

.findFirst().orElseThrow();

Group group = groupService.getAllGroups().stream()

.filter(g -> g.getGroupId().equals(groupId))

.findFirst().orElseThrow();

user.setGroup(group);

return userService.register(user); // reuse register() to update

}

}

package com.loanCycle.controller;

import com.loanCycle.model.Loan;

import com.loanCycle.model.User;

import com.loanCycle.service.LoanService;

import com.loanCycle.service.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/loans")

@RequiredArgsConstructor

@CrossOrigin

public class LoanController {

private final LoanService loanService;

private final UserService userService;

@PostMapping

public Loan applyLoan(@RequestBody Loan loan) {

return loanService.applyLoan(loan);

}

@GetMapping("/user/{userId}")

public List<Loan> getUserLoans(@PathVariable Long userId) {

User user = userService.getAllUsers().stream()

.filter(u -> u.getUserId().equals(userId))

.findFirst().orElseThrow();

return loanService.getLoansByMember(user);

}

@PutMapping("/{loanId}/status")

public Loan updateStatus(@PathVariable Long loanId, @RequestParam String status) {

return loanService.updateLoanStatus(loanId, status);

}

@GetMapping

public List<Loan> getAllLoans() {

return loanService.getAllLoans();

}

}

package com.loanCycle.controller;

import com.loanCycle.model.Group;

import com.loanCycle.service.GroupService;

import lombok.RequiredArgsConstructor;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/groups")

@RequiredArgsConstructor

@CrossOrigin

public class GroupController {

private final GroupService groupService;

@PostMapping

public Group createGroup(@RequestBody Group group) {

return groupService.createGroup(group);

}

@GetMapping

public List<Group> getAllGroups() {

return groupService.getAllGroups();

}

@DeleteMapping("/{id}")

public void deleteGroup(@PathVariable Long id) {

groupService.deleteGroup(id);

}

}

package com.loanCycle.controller;

import com.loanCycle.model.User;

import com.loanCycle.service.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.web.bind.annotation.\*;

import java.util.Optional;

@RestController

@RequestMapping("/api/auth")

@RequiredArgsConstructor

@CrossOrigin

public class AuthController {

private final UserService userService;

@PostMapping("/register")

public User register(@RequestBody User user) {

return userService.register(user);

}

@PostMapping("/login")

public User login(@RequestBody User loginData) {

Optional<User> user = userService.login(loginData.getEmail(), loginData.getPassword());

return user.orElse(null); // return null if login fails

}

}

package com.loanCycle.service;

import com.loanCycle.model.Loan;

import com.loanCycle.model.User;

import com.loanCycle.repository.LoanRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.time.LocalDate;

import java.util.List;

@Service

@RequiredArgsConstructor

public class LoanService {

private final LoanRepository loanRepository;

public Loan applyLoan(Loan loan) {

loan.setApplicationDate(LocalDate.now());

loan.setStatus("pending");

return loanRepository.save(loan);

}

public List<Loan> getLoansByMember(User member) {

return loanRepository.findByMember(member);

}

public List<Loan> getAllLoans() {

return loanRepository.findAll();

}

public Loan updateLoanStatus(Long loanId, String status) {

Loan loan = loanRepository.findById(loanId).orElseThrow();

loan.setStatus(status);

if ("approved".equals(status)) {

loan.setApprovedDate(LocalDate.now());

}

return loanRepository.save(loan);

}

}

package com.loanCycle.service;

import com.loanCycle.model.Group;

import com.loanCycle.repository.GroupRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

@RequiredArgsConstructor

public class GroupService {

private final GroupRepository groupRepository;

public Group createGroup(Group group) {

return groupRepository.save(group);

}

public List<Group> getAllGroups() {

return groupRepository.findAll();

}

public void deleteGroup(Long id) {

groupRepository.deleteById(id);

}

}

package com.loanCycle.service;

import com.loanCycle.model.User;

import com.loanCycle.repository.UserRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

@RequiredArgsConstructor

public class UserService {

private final UserRepository userRepository;

public User register(User user) {

return userRepository.save(user);

}

public Optional<User> login(String email, String password) {

return userRepository.findByEmail(email)

.filter(u -> u.getPassword().equals(password));

}

public List<User> getAllUsers() {

return userRepository.findAll();

}

}

package com.loanCycle.repository;

import com.loanCycle.model.Loan;

import com.loanCycle.model.User;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface LoanRepository extends JpaRepository<Loan, Long> {

List<Loan> findByMember(User member);

}

package com.loanCycle.repository;

import com.loanCycle.model.Group;

import org.springframework.data.jpa.repository.JpaRepository;

public interface GroupRepository extends JpaRepository<Group, Long> {

}

package com.loanCycle.repository;

import com.loanCycle.model.User;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {

Optional<User> findByEmail(String email);

}

package com.loanCycle.model;

import jakarta.persistence.\*;

import lombok.\*;

import java.time.LocalDate;

import java.time.LocalDateTime;

@Entity

@Table(name = "loans")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Loan {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long loanId;

@ManyToOne

@JoinColumn(name = "member\_id")

private User member;

@ManyToOne

@JoinColumn(name = "group\_id")

private Group group;

private Double amount;

private String status = "pending"; // pending, approved, rejected

private LocalDate applicationDate = LocalDate.now();

private LocalDate approvedDate;

}

package com.loanCycle.model;

import jakarta.persistence.\*;

import lombok.\*;

import java.time.LocalDateTime;

import java.util.List;

@Entity

@Table(name = "groups")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Group {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long groupId;

private String groupName;

private String description;

private LocalDateTime createdAt = LocalDateTime.now();

@OneToMany(mappedBy = "group")

private List<User> members;

@OneToMany(mappedBy = "group")

private List<Loan> loans;

}

package com.loanCycle.model;

import jakarta.persistence.\*;

import lombok.\*;

import java.time.LocalDateTime;

import java.util.List;

@Entity

@Table(name = "users")

@Data

@NoArgsConstructor

@AllArgsConstructor

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long userId;

private String fullName;

private String address;

private String email;

private String password; // Plain text for now

private String role; // "member" or "admin"

private LocalDateTime createdAt = LocalDateTime.now();

@ManyToOne

@JoinColumn(name = "group\_id")

private Group group;

@OneToMany(mappedBy = "member")

private List<Loan> loans;

}