package com.example.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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.example.entity.Account;

import com.example.entity.Message;

import com.example.service.AccountService;

import com.example.service.MessageService;

import java.util.List;

/\*\*

 \* TODO: You will need to write your own endpoints and handlers for your controller using Spring. The endpoints you will need can be

 \* found in readme.md as well as the test cases. You be required to use the @GET/POST/PUT/DELETE/etc Mapping annotations

 \* where applicable as well as the @ResponseBody and @PathVariable annotations. You should

 \* refer to prior mini-project labs and lecture materials for guidance on how a controller may be built.

 \*/

@RestController

@RequestMapping()

public class SocialMediaController {

    @Autowired

    private AccountService accountService;

    @Autowired

    private MessageService messageService;

    @PostMapping("/register")

    public ResponseEntity<?> register(@Valid @RequestBody Account account) {

        if (accountService.findByUsername(account.getUsername()) != null) {

            return ResponseEntity.status(HttpStatus.CONFLICT).body("Username already exists");

        }

        try {

            Account createdAccount = accountService.createAccount(account);

            return ResponseEntity.status(HttpStatus.OK).body(createdAccount);

        } catch (Exception e) {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body("Invalid data provided");

        }

    }

    @PostMapping("/login")

    public ResponseEntity<?> login(@RequestBody Account account) {

        Account existingAccount = accountService.findByUsername(account.getUsername());

        if (existingAccount != null && existingAccount.getPassword().equals(account.getPassword())) {

            return ResponseEntity.status(HttpStatus.OK).body(existingAccount);

        } else {

            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid username or password");

        }

    }

    @PostMapping("/messages")

    public ResponseEntity<?> createMessage(@RequestBody Message message) {

        if (message.getMessageText() == null || message.getMessageText().isBlank() || message.getMessageText().length() > 255) {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);

        }

        if (!messageService.isUserExist(message.getPostedBy())) {

            return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);  // Invalid user

        }

        Message savedMessage = messageService.createMessage(message);

        return ResponseEntity.status(HttpStatus.OK).body(savedMessage);

    }

    @GetMapping("/messages")

    public ResponseEntity<List<Message>> getAllMessages() {

        List<Message> messages = messageService.getAllMessages();

        return ResponseEntity.ok(messages);

    }

    @GetMapping("/messages/{messageId}")

    public ResponseEntity<Message> getMessageById(@PathVariable Integer messageId) {

        Message message = messageService.getMessageById(messageId);

        if (message != null) {

            return ResponseEntity.ok(message);

        } else {

            return ResponseEntity.ok(null);

        }

    }

    @DeleteMapping("/messages/{messageId}")

    public ResponseEntity<Object> deleteMessageById(@PathVariable Integer messageId) {

        int rowsDeleted = messageService.deleteMessageById(messageId);

        if (rowsDeleted > 0) {

            return ResponseEntity.ok(rowsDeleted);

        } else {

            return ResponseEntity.ok().build();

        }

    }

    @PatchMapping("/messages/{messageId}")

    public ResponseEntity<Object> updateMessage(@PathVariable Integer messageId, @RequestBody Message newMessage) {

        if (newMessage.getMessageText() == null || newMessage.getMessageText().isBlank() || newMessage.getMessageText().length() > 255) {

            return ResponseEntity.badRequest().body("Invalid messageText. It must not be blank and should be less than 255 characters.");

        }

        int rowsUpdated = messageService.updateMessageText(messageId, newMessage.getMessageText());

        if (rowsUpdated > 0) {

            return ResponseEntity.ok(rowsUpdated);

        } else {

            return ResponseEntity.badRequest().body("Message not found with ID: " + messageId);

        }

    }

    @GetMapping("/accounts/{accountId}/messages")

    public ResponseEntity<List<Message>> getMessagesByUser(@PathVariable Integer accountId) {

        List<Message> messages = messageService.getMessagesByUser(accountId);

        return ResponseEntity.ok(messages);

    }

}