from app.models import \*  
from app import \*  
from flask\_restful import Resource  
from flask\_apispec.views import MethodResource  
from flask\_apispec import marshal\_with, doc, use\_kwargs  
from app.schemas import \*  
from app.services import \*  
  
"""  
[Sign Up API] : Its responsibility is to perform the signup activity for the user.  
"""  
  
  
# Restful way of creating APIs through Flask Restful  
class SignUpAPI(MethodResource, Resource):  
 @doc(description='Sign Up API)', tags=['SignUp API'])  
 @use\_kwargs(SignUpRequest, location='json')  
 @marshal\_with(APIResponse)  
 def post(self, \*\*kwargs):  
 try:  
 create\_user(\*\*kwargs)  
 return APIResponse().dump(dict(message='User is successfully registered')), 200  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f'Not able to register user : {str(err)}')), 400  
  
  
api.add\_resource(SignUpAPI, '/signup')  
docs.register(SignUpAPI)  
  
"""  
[Login API] : Its responsibility is to perform the login activity for the user and   
create session id which will be used for all subsequent operations.  
"""  
  
  
class LoginAPI(MethodResource, Resource):  
 @doc(description='Login API', tags=['Login API'])  
 @use\_kwargs(LoginRequest, location='json')  
 @marshal\_with(APIResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_logged\_in, session\_id = login\_user(\*\*kwargs)  
 if is\_logged\_in:  
 return APIResponse().dump(dict(message='User is successfully logged in')), 200  
 else:  
 return APIResponse().dump(dict(message='User not found')), 404  
  
 except Exception as err:  
 print(str(err))  
 return APIResponse().dump(dict(message=f'Not able to login user : {str(err)}')), 400  
  
  
api.add\_resource(LoginAPI, '/login')  
docs.register(LoginAPI)  
  
"""  
[Logout API] : Its responsibility is to perform the logout activity for the user.  
"""  
  
  
class LogoutAPI(MethodResource, Resource):  
 @doc(description='Logout API', tags=['Logout API'])  
 @use\_kwargs(LogoutRequest, location='json')  
 @marshal\_with(APIResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_logged\_out = logout\_user(kwargs['session\_id'])  
 if is\_logged\_out:  
 return APIResponse().dump(dict(message='User is successfully logged out')), 200  
  
 else:  
 return APIResponse().dump(dict(message='User is not logged123 in')), 401  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f'Not able to logout user : {str(err)}')), 400  
  
  
api.add\_resource(LogoutAPI, '/logout')  
docs.register(LogoutAPI)  
  
"""  
[Add Question API] : Its responsibility is to add question to the question bank.  
Admin has only the rights to perform this activity.  
"""  
  
  
class AddQuestionAPI(MethodResource, Resource):  
 @doc(description='Add Question API', tags=['Question'])  
 @use\_kwargs(AddQuestionRequest, location='json')  
 @marshal\_with(APIResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
  
 is\_admin = check\_if\_admin(user\_id)  
  
 if not is\_admin:  
 return APIResponse().dump(dict(message='Logged in user is not have admin Rights')), 401  
  
 add\_question(\*\*kwargs)  
  
 return APIResponse().dump(dict(message='question is successfully added')), 200  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f' Not able to add question : {str(err)}')), 400  
  
  
api.add\_resource(AddQuestionAPI, '/add.question')  
docs.register(AddQuestionAPI)  
  
"""  
[List Questions API] : Its responsibility is to list all questions present activly in the question bank.  
Here only Admin can access all the questions.  
"""  
  
  
class ListQuestionAPI(MethodResource, Resource):  
 @doc(description='List Question API', tags=['Question'])  
 @use\_kwargs(QuestionsRequest, location='json')  
 @marshal\_with(ListQuestionsResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
 print(is\_active, user\_id)  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
  
 is\_admin = check\_if\_admin(user\_id)  
  
 if not is\_admin:  
 return APIResponse().dump(dict(message='Logged in user is not have admin Rights')), 401  
  
 questions\_list = list\_questions()  
  
 return ListQuestionsResponse().dump(dict(questions=questions\_list)), 200  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f' Not able to List question : {str(err)}')), 400  
  
  
api.add\_resource(ListQuestionAPI, '/list.questions')  
docs.register(ListQuestionAPI)  
  
"""  
[Create Quiz API] : Its responsibility is to create quiz and only admin can create quiz using this API.  
"""  
  
  
class CreateQuizAPI(MethodResource, Resource):  
 @doc(description='Create Quiz API', tags=['Quiz'])  
 @use\_kwargs(CreateQuizRequest, location='json')  
 @marshal\_with(APIResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
  
 is\_admin = check\_if\_admin(user\_id)  
  
 if not is\_admin:  
 return APIResponse().dump(dict(message='Logged in user is not have admin Rights')), 401  
  
 create\_quiz(\*\*kwargs)  
  
 return APIResponse().dump(dict(message='Quiz has been created successfully')), 401  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f' Not able to List question : {str(err)}')), 400  
  
  
api.add\_resource(CreateQuizAPI, '/create.quiz')  
docs.register(CreateQuizAPI)  
  
"""  
[Assign Quiz API] : Its responsibility is to assign quiz to the user. Only Admin can perform this API call.  
"""  
  
  
class AssignQuizAPI(MethodResource, Resource):  
 @doc(description='Assign Quiz API', tags=['Quiz'])  
 @use\_kwargs(AssignQuizRequest, location='json')  
 @marshal\_with(APIResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
  
 is\_admin = check\_if\_admin(user\_id)  
  
 if not is\_admin:  
 return APIResponse().dump(dict(message='Logged in user is not have admin Rights')), 401  
  
 assign\_quiz(\*\*kwargs)  
  
 return APIResponse().dump(dict(message='Quiz has been assigned successfully')), 401  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f' Not able to Assign Quiz to User : {str(err)}')), 400  
  
  
api.add\_resource(AssignQuizAPI, '/assign.quiz')  
docs.register(AssignQuizAPI)  
  
"""  
[View Quiz API] : Its responsibility is to view the quiz details.  
Only Admin and the assigned users to this quiz can access the quiz details.  
"""  
  
  
class ViewQuizAPI(MethodResource, Resource):  
 @doc(description='View Quiz API', tags=['Quiz'])  
 @use\_kwargs(ViewQuizRequest, location='json')  
 @marshal\_with(ViewQuizResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
 print(is\_active, user\_id)  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
 print('Checking')  
 has\_access = check\_quiz\_access(kwargs['quiz\_id'], user\_id)  
 print(has\_access)  
 if not has\_access:  
 return APIResponse().dump(dict(message='user have no access for teh requested quiz')), 401  
  
 questions = view\_quiz(\*\*kwargs)  
  
 return ViewQuizResponse().dump(dict(questions=questions)), 200  
  
 except Exception as err:  
 print(str(err))  
 return APIResponse().dump(dict(message=f'Error in accessing quiz: {str(err)}')), 400  
  
  
api.add\_resource(ViewQuizAPI, '/view.quiz')  
docs.register(ViewQuizAPI)  
  
"""  
[View Assigned Quiz API] : Its responsibility is to list all the assigned quizzes   
 with there submittion status and achieved scores.  
"""  
  
  
class ViewAssignedQuizAPI(MethodResource, Resource):  
 @doc(description='View Assigned Quiz API', tags=['Quiz'])  
 @use\_kwargs(AssignedQuizRequest, location='json')  
 @marshal\_with(AssignedQuizResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
  
 quiz\_info = get\_assigned\_quiz\_info(user\_id)  
  
 if len(quiz\_info) == 0:  
 return APIResponse().dump(dict(message='No quiz is assigned to the user')), 401  
  
 return AssignedQuizResponse().dump(dict(quiz\_info=quiz\_info)), 200  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f'Error in accessing quiz info: {str(err)}')), 400  
  
  
api.add\_resource(ViewAssignedQuizAPI, '/assigned.quizzes')  
docs.register(ViewAssignedQuizAPI)  
  
"""  
[View All Quiz API] : Its responsibility is to list all the created quizzes. Admin can only list all quizzes.  
"""  
  
  
class ViewAllQuizAPI(MethodResource, Resource):  
 @doc(description='View All Quiz API', tags=['Quiz'])  
 @use\_kwargs(ViewAllQuizRequest, location='json')  
 @marshal\_with(ViewAllQuizResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
  
 is\_admin = check\_if\_admin(user\_id)  
  
 if not is\_admin:  
 return APIResponse().dump(dict(message='Logged in user is not have admin Rights')), 401  
  
 quiz\_info = get\_all\_quiz\_info(user\_id)  
  
 if len(quiz\_info) == 0:  
 return APIResponse().dump(dict(message='No quiz is available to view')), 401  
  
 return ViewAllQuizResponse().dump(dict(quiz\_info=quiz\_info)), 200  
  
 except Exception as err:  
 return APIResponse().dump(dict(message=f'Error in accessing quiz info: {str(err)}')), 400  
  
  
api.add\_resource(ViewAllQuizAPI, '/all.quizzes')  
docs.register(ViewAllQuizAPI)  
  
"""  
[Attempt Quiz API] : Its responsibility is to perform quiz attempt activity by   
 the user and the score will be shown as a result of the submitted attempt.  
"""  
  
  
class AttemptQuizAPI(MethodResource, Resource):  
 @doc(description='Attempt Quiz API', tags=['Quiz'])  
 @use\_kwargs(AttemptQuizRequest, location='json')  
 @marshal\_with(AttemptQuizResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
  
 if not is\_active:  
 return APIResponse().dump(dict(message='User is not logged in')), 404  
  
 has\_access = check\_quiz\_access(kwargs['quiz\_id'], user\_id)  
  
 if not has\_access:  
 return APIResponse().dump(dict(message='user have no access for the requested quiz')), 401  
  
 score\_achieved = attempt\_quiz(user\_id, kwargs['quiz\_id'], kwargs['responses'],  
 kwargs['question\_id']) # missed code  
  
 return AttemptQuizResponse().dump(dict(score\_achieved=score\_achieved)), 200  
  
 except Exception as err:  
 print(str(err))  
 return APIResponse().dump(dict(message=f'Error in attempting the given quiz: {str(err)}')), 400  
  
  
api.add\_resource(AttemptQuizAPI, '/attempt.quiz')  
docs.register(AttemptQuizAPI)  
  
"""  
[Quiz Results API] : Its responsibility is to provide the quiz results in which the users   
 having the scores sorted in descending order are displayed,   
 also the ones who have not attempted are also shown.  
 Admin has only access to this functionality.  
"""  
  
  
class QuizResultAPI(MethodResource, Resource):  
 @doc(description='Quiz Result API', tags=['Quiz'])  
 @use\_kwargs(QuizResultRequest, location=('json'))  
 @marshal\_with(QuizResultResponse)  
 def post(self, \*\*kwargs):  
 try:  
 is\_active, user\_id = check\_if\_session\_is\_active(kwargs['session\_id'])  
  
 if not is\_active:  
 return APIResponse().dump(dict(message='user is not logged in.')), 404  
  
 is\_admin = check\_if\_admin(user\_id)  
  
 if not is\_admin:  
 return APIResponse().dump(dict(message="user is not admin")), 401  
  
 results = quiz\_results(kwargs['quiz\_id'])  
  
 if len(results) == 0:  
 return APIResponse().dump(dict(message="No result for the quiz is available")), 405  
  
 return QuizResultResponse().dump(dict(results=results)), 200  
  
 except Exception as err:  
 print(str(err))  
 return APIResponse().dump(dict(message=f' Not able to add question : {str(err)}')), 400  
  
  
api.add\_resource(QuizResultAPI, '/quiz.results')  
docs.register(QuizResultAPI)