**Отчет о выполнении лабораторной работы №3**

Якухин Иван, 6303-010302D

*Идея: приложение с карточками для запоминания английских слов*

1. **Реализация механизма аутентификации**

Для аутентификации пользователей были реализованы API endpoint’ы для регистрации и входа в систему. JWT токены были добавлены из бибилиотеки django-rest-framework-simple-jwt.

class UserAuth(views.APIView):  
 def post(self, request, action, \*args, \*\*kwargs):

...

elif action == "login":  
 username = request.data.get('username')  
 password = request.data.get('password')  
 user = authenticate(username=username, password=password)  
 if user is None:  
 return Response({'error': 'Invalid credentials. Try again!'}, status=401)  
  
 refresh = RefreshToken.for\_user(user)  
 response = Response()  
 response.set\_cookie(  
 key='refresh\_token',  
 value=str(refresh),  
 secure=False,  
 httponly=True,  
 samesite='Lax'  
 )  
 response.set\_cookie(  
 key='access\_token',  
 value=str(refresh.access\_token),  
 secure=False,  
 httponly=True,  
 samesite='Lax'  
 )  
 response.data = {  
 'message': 'Login successful'  
 }  
 return response  
 elif action == "register":  
 if get\_user\_model().objects.filter(username = request.data.get('username')).exists():  
 return Response("User already exists!", 400)  
 user = get\_user\_model().objects.create(  
 username = request.data.get('username'),  
 email = request.data.get('email'),  
 first\_name = request.data.get('first\_name'),  
 last\_name = request.data.get('last\_name')  
 )  
 user.set\_password(request.data.get('password'))  
 user.save()  
 login\_res = self.post(request, action='login')  
 return login\_res

...

1. **Настройка middleware для защиты API**

Для защиты определенных эндпоинтов было использовано разрешение (permission) isAuthenticated из библиотеки django-rest-framework. Также для проверки валидности токенов и их обновления было написано middleware.

from rest\_framework\_simplejwt.exceptions import InvalidToken, TokenError  
from rest\_framework\_simplejwt.tokens import RefreshToken  
from rest\_framework\_simplejwt.authentication import JWTAuthentication  
  
class RefreshAccessTokenMiddleware:  
 def \_\_init\_\_(self, get\_response):  
 self.get\_response = get\_response  
  
 def \_\_call\_\_(self, request):  
 access\_token = request.COOKIES.get('access\_token')  
 if access\_token:  
 try:  
 request.META['HTTP\_AUTHORIZATION'] = f'Bearer {access\_token}'  
 jwt\_auth = JWTAuthentication()  
 jwt\_auth.get\_validated\_token(access\_token)  
 except (InvalidToken, TokenError):  
 refresh\_token = request.COOKIES.get('refresh\_token')  
  
 if refresh\_token:  
 try:  
 new\_access\_token = RefreshToken(refresh\_token).access\_token  
 request.META['HTTP\_AUTHORIZATION'] = f'Bearer {new\_access\_token}'  
 response = self.get\_response(request)  
 response.set\_cookie(  
 key='access\_token',  
 value=str(new\_access\_token),  
 httponly=True,  
 samesite='Lax'  
 )  
 response.status\_code = 200  
 return response  
 except (InvalidToken, TokenError):  
 response.data = {'error': 'Invalid refresh token. You need to login again!'}  
 response.status\_code = 401  
 response.delete\_cookie('refresh\_token')  
 response.delete\_cookie('access\_token')  
 return response  
 response = self.get\_response(request)  
 return response

1. **Реализация управления пользователями**

Для управления пользователями были реализованы эндпоинты смены пароля, выхода из аккаунта, изменения пользовательских данных.

class UserAuth(views.APIView):  
 def post(self, request, action, \*args, \*\*kwargs):  
 if action == "change-password":  
 if not request.user.is\_authenticated:  
 return Response("Not allowed to do this. You need to login first.", 401)  
 user = request.user  
 serializer = ChangePasswordSerializer(data=request.data, context={'user': user})  
  
 if serializer.is\_valid():  
 new\_password = serializer.validated\_data['new\_password']  
 user.set\_password(new\_password)  
 user.save()  
  
 return Response({"message": "Password changed successfully."}, status=200)  
  
 return Response(serializer.errors, status=400)  
 elif action == "logout":  
 if not request.user.is\_authenticated:  
 return Response("Not allowed to do this. You need to login first.", 401)  
 refresh\_token = request.COOKIES.get('refresh\_token')  
 if refresh\_token:  
 token = RefreshToken(refresh\_token)  
 token.blacklist()  
  
 response = Response({'message': 'Logout successful'})  
 response.delete\_cookie('refresh\_token')  
 response.delete\_cookie('access\_token')  
 return response  
 ...  
 elif action == 'edit':  
 if get\_user\_model().objects.filter(username = request.data.get('username')).exists():  
 return Response("User already exists!", 400)  
 user = request.user  
 user.username=request.data.get('username')  
 user.email=request.data.get('email')  
 user.first\_name=request.data.get('first\_name')  
 user.last\_name=request.data.get('last\_name')  
 user.save()  
 return Response("Profile editing complited successfully!",200)  
 else:  
 return Response(f"Bad Request. We dont have {action}/ page...", status = 400)

1. **Тестирование API**

Для тестирования API была использована программа Postman. В ней были созданы шаблоны запросов ко всем эндпоинтам. В процессе написания API и впоследствии в программе были отлажены все API запросы.

