

COMP4920 Senior Design Project II, Spring 2021
Advisor: Mutlu BEYAZIT

DRONOMATRIX: CYBER DRONE
Product Manual

Revision Final
23.05.2021

By
Simge Binnaz ÖZDEMİR,
Bora GÜZEL,
Ege Erberk USLU

Revision History

Revision	Date	Explanation
1.0	02.04.2021	Initial Product Manual
2.0	14.05.2021	Section 2.2 is updated with adding new software development tools. Section 3 and 4 are updated with adding new test on project.
Final	23.05.2021	Final Version of Product Manual

Table of Figure

Figure 1: Simulation part test.....	12
Figure 2:Platforms-2.....	12
Figure 3:Platforms-1.....	12
Figure 4:Pickup object.....	13
Figure 5:"API/User/Delete/{UserID}" URL Test.	14
Figure 6:"API/User/Add" URL Test.....	14
Figure 7:"API/User/Add" URL Test-2	15
Figure 8:"API/User/Change-Password "URL Test	15
Figure 9:"API/User/Change-Password "URL Test-2.....	16
Figure 10:API/User/Change-Password "URL Test-3.....	16
Figure 11:API/User/Change-Password "URL Test-4.....	17
Figure 12:"API/User/Forget-Password" URL Test	17
Figure 13: API/User/logs" URL Test	18
Figure 14: API/bug/all" URL Test	19
Figure 15 :API/bug/all" URL Test 2	19
Figure 16: API/user/personal/update/{UserID}" URL Test.	20
Figure 17 :API/user/personal/update/{UserID}" URL Test 2.	20
Figure 18: API/user/personal/update/{UserID}" URL Test 3	21
Figure 19: API/user/personal/update/{UserID}" URL Test 4	21
Figure 20: API/bug/upload" URL Test.....	22
Figure 21: API/bug/upload" URL Test 2.....	22
Figure 22: API/bug/delete/{bugID}" URL Test.	23
Figure 23: API/bug/delete/{bugID}" URL Test 2.	23
Figure 24: API/bug/logs/" URL Test.....	24
Figure 25: API/bug/logs/" URL Test 2.....	24
Figure 26: API/config/user/" URL Test.	25
Figure 27 :API/config/user/" URL Test 2.	25
Figure 28: API/config/updateuser" URL Test	26
Figure 29 :API/config/updateuser" URL Test 2.	26
Figure 30 :API/config/updateuser" URL Test 3	27
Figure 31: API/record/upload" URL Test.....	27
Figure 32: API/record/upload" URL Test 2	28
Figure 33:API/record/download/{filename}" URL Test	28
Figure 34 :API/record/delete/{filename}" URL Test.....	29
Figure 35: API/record/delete/{filename}" URL Test 2	29
Figure 36: API/record/delete/{filename}" URL Test 3	30
Figure 37: API/record/delete/{filename}" URL Test 4.....	30
Figure 38: API/record/logs" URL Test.	31
Figure 39: API/record/thumbnails/{userID}" URL Test.	31
Figure 40: API/record/thumbnails/{userID}" URL Test-2	32
Figure 41: API/record/thumbnails/{userID}" URL Test-3.....	32
Figure 42:Login Test.....	33
Figure 43:Login Test-2	33
Figure 44:Admin/User Navigatio Bar Test.....	34
Figure 45:Add user test.	34

Figure 46:Add user test-2.....	35
Figure 47:Add user test mail.....	35
Figure 48:Getting All User & More Detailed Test.....	35
Figure 49:User Details Updating Test.....	36
Figure 50:Changing Password Test.....	36
Figure 51:Forget Password Test.....	37
Figure 52:Forget password test-2	37
Figure 53: Forget password test mail.....	37
Figure 54:Online User Test-1.....	38
Figure 55:Online User Test-2.....	38
Figure 56:User Update Personal Information Test.	38
Figure 57:Record Download and Watch Test.....	39

Table of Contents

Revision History	2
Table of Contents	5
1. Introduction	7
2. Cyber Drone Software Subsystem Implementation	7
2.1. Source Code and Executable Organization	7
• Unity part	7
• Web Client part	8
• Web Server part	8
• Online User Server part	9
2.2. Software Development Tools	9
2.3. Hardware and System Software Platform	10
3. Cyber Drone Software Subsystem Testing	11
3.1. Unity System Testing	11
• Login Page Test	11
• Menu & Settings Part Test	11
• Simulation Part Tests	12
3.2 Web Server System Testing	13
• WITH POSTMAN	13
○ User Controller API Test	13
○ “API/User/Delete/{UserID}” URL Test	13
○ “API/User/Add” URL Test	14
○ “API/User/Change-Password “URL Test	15
○ “API/User/Update/{UserID}” URL Test	17
○ “API/User/Forget-Password” URL Test	17
○ “API/User/logs” URL Test	17
○ “API/bug/all” URL Test	18
○ “API/user/personal/update/{UserID}” URL Test	19
○ “API/bug/upload” URL Test	21
○ “API/bug/delete/{bugID}” URL Test	22
○ “API/bug/logs/” URL Test	23
○ “API/config/user/” URL Test	24
○ “API/config/updateuser” URL Test	25
○ “API/record/upload” URL Test	27
○ “API/record/download/{filename}” URL Test	28

○ “API/record/delete/{filename}” URL Test	29
○ “API/record/logs” URL Test	30
○ “API/record/thumbnails/{userID}” URL Test	31
3.3 Web Client System Testing	33
• Login Test	33
• Admin/User Navigation Bar Test	33
• Add User Test	34
• Getting All User & More Detailed Test	35
• User Details Updating Test	36
• Changing Password Test	36
• Forget Password Test	37
• Online User Test	38
• User Update Personal Information Test	38
• Record Download and Watch Test	39
4. Cyber Drone Installation, Configuration and Operation	39
• User:	39
• Admin:	40
References	42

1. Introduction

Cyber Drone desktop application is a Unity 3D based simulation project. In this project, the use of drones has been tried to be implemented on a simulation. The purpose of this product manual is to document the implementation, testing, installation and operation of this Cyber Drone desktop application as a software product.

Cyber Drone desktop application is implemented and tested as it is described in Cyber Drone desktop application-Design Specification Document, Revision 2.0, satisfying the requirements in Cyber Drone desktop application application-Requirements Specification Document, Revision 2.0.

Implementation, testing and operation details are given in the following sections of this document.

2. Cyber Drone Software Subsystem Implementation

2.1. Source Code and Executable Organization

Cyber drone desktop application is Unity 3D based simulation. The simulation environment is implemented on Unity Game engine. The transactions regarding users and administrators of the project are structured on the web. Explanations about the Unity part and the web part are below.

- **Unity part**

For this part there will be two files. One for the developers, one for the users.

- **User**

Users will download the builder version of the program. In the file there will be two folders, two applications and one application extension. One of the applications will be the “CyberDrone.exe”. By using this application, the user can start the simulation. The other application is “UnityCrashHandler64.exe”. This application is created by Unity while building the project. The purpose of this application is when the game crashes it informs the user and sends a report back to the developer. One of the folders will be “MonoBleedingEdge” and the other one is CyberDrone_Data. The MonoBleedingEdge folder contains the required C# and MonoDevelop libraries to run the application. The other CyberDrone_Data folder contains all the needed data and external files (DLL's etc.). Lastly the application extension is UnityPlayer.dll. It is a Dynamic Link Library file which is a type of file containing code and data that can be used by multiple programs at the same time. Records folder stores the record that user' replays.

- **Developer**

Developers will download the project file of the program. Inside of this file there will be seven folders, four Visual C# Project files, one COLLABIGNORE file and one Visual Studio Solution. The C# Project files are for the C# editors for the coding part. The COLLABIGNORE file is a special file created by Unity which for using the Unity Collaborate while developing. Six of the seven files are classic Unity folders.” Library”, ” Logs”, ” obj”, ” Packages”, ” ProjectSettings” and ” Temp”. The Library folder includes all the necessary libraries and data for the simulation. For instance, ScriptAssemblies, TerrainTools, UIElements and so on. The Logs folder includes package update records of the project. The ProjectSettings file contains settings data of the project(VFXManager,QualitySettings,InputManager, etc.). The Packages file contains json data of Unity's dependencies. The Temp file contains backup scenes of the project if there is any. Except these six folders, there is also an Asset folder. This folder is mainly created by the developer itself. It contains all the assets, scripts, fonts, scenes, textures ...Developers organize this folder in their own style. For the CyberDrone

project, in the Asset folder there are five main folders. “Animation”, “AssetStore”, “Fonts”, “Images”, “Scenes”, “Scripts” and “Audio”. Animation folder contains every animation and animator that the developer created. AssetStore folder contains every asset that used in the project. Fonts folder contains each font that is used in the project. Images folder contains each image that is used in the project. The Scenes folder contains every scene that was created and used in the project by the developer. Finally, the Scripts folder contains every script that was created and used in the project by the developer. On the other hand, in files there are two different files called logininfo.xml and settings.json. logininfo.xml stores the login information of last logged in user. This file is necessary for autologin. In addition, settings.json stores user settings that effects the simulation’s hotkeys, sounds and resolution of program. Records folder stores the record that user’ replays.

- **Web Client part**

- **Developer**

Developers will download the project file of the program. Inside of this file there will be five folders, index.html and package.json file. package json file contains dependencies. Developers are able to add new libraries by using that file. On the other hand, index.html is the main file of web client system. Most important folder is src folder. This folder is responsible for all web operations in vuejs. In the src folder there are five folders and two files. Assets and image file is for storing icons and images that used in web client. Component folder is for creating sub web pages. In this folder, there are many vue files which is for rendering html and inserting vue JavaScript codes. Router folder has index.js file. In this file all routing operations are done. In store folder, there is file called store.js. This file is created for vuex. Vuex is a state management library for Vue.js applications. App.vue file is the main vue js file for all pages. Initially, this file is created and served. Other vue pages that is in component folder are created by this App.vue file. Main.js is another file for importing libraries that installed before into vue project. This file effects all files that contain a vue instance.

- **Web Server part**

- **Developer**

Developers will download the project file of the program. Inside of this file there will be nine folders, appsettings.json program.cs, startup.cs. appsettings.json includes application settings attributes such as javascript web token (jwt) information and database connection string. On the other hand, startup.cs is created for configuring project. In this script, developers are able to insert new technologies, controller, jwt configuration, connect databases, models, using authorization and authentication. Etc. Program.cs is created for starting the server. First folder is properties folder. This folder contains launchSettings.json file. This file is responsible for changing api url or development URL and ports. Second folder is controller folder. This folder contains all controllers that used in the web server. Controllers are responsible for managing url and getting information back that desired. Another folder is model’s folder. Models’ folder is responsible for creating models as same with database model. Last file is the security file. Security folder includes a Crypto.cs file. This file is responsible for all encryption and decryption and hashing operations. Records folder stores the record that user’ replays. Deleted record’s file stores all records that has been deleted from system. This is created for storing records for logs. On the other hand, FFMEG folder handles the mp4 files and is responsible for extracting thumbnails from these mp4 videos. It contains ffmpeg program. Thumbnails folder contains the jpeg file that are extracted from

uploaded videos. At least, profileImages folder stores the jpeg files that are for user profiles.

- **Online User Server part**

- **Developer**

Developers will download the project file of the program. Inside of this file there will be two folders. First folder is Classes folder which contains two different classes which are information and player javascript files. Node modules folder contains the libraries of that used in node.js. On the other hand, most important file is index.js. This file is for starting the server. Package.json is for showing and adding the new modules to project.

2.2. Software Development Tools

Cyber Drone desktop application is prepared by more than one software development tool.

Complete specification of software tools, such as

- **GitHub** is used for version control systems. Systems of project are divided into three git repositories that mentioned as follows; **Frontend of Web, Backend of Web and Unity, Unity**
- **Unity** is a cross-platform game engine developed by Unity Technologies. This engine is used for development of simulation. (v2019.4.23f1) [1]
- **MSSQL** (Microsoft SQL Server) is used as a relational database for storing user's data. (v2019)
- **Microsoft SQL Server Management Studio 18 (SSMS 18)** is an IDE that provides a graphical interface for connecting and working with MS SQL server. [2]
- **Unity Teams** is used for collaboration, cloud storage builds and version controls with group members. This tool is used in purpose of generating code simultaneously.
- **Postman** is a popular API client that makes it easy for developers to create, share, test and document APIs. In this project, this tool is used for testing and building new API's. (v8.0) [3]
- **Visual Studio C#** is an Integrated Development Environment (IDE) tool from Microsoft. Unity's Visual Studio integration allows you to create and maintain Visual Studio project files automatically. In addition, this tool is used for developing the backend part of the web and unity subsystems, creating APIs for multi-platform clients such as desktop and web applications. [4]
- **Visual Studio Code** has built-in support for the Vue.js building blocks of HTML, CSS, and JavaScript. [5] In this project this code editor is used for developing the frontend part of the web-client subsystem.
- **Vue JavaScript** is a well-known JavaScript framework for building single page applications on the web. For developing the frontend part of the project and adding functionalities on the web page, vue js is used. (v2.6.11)

- **Vue Bootstrap** is a CSS library based on Bootstrap that is created for implementing vue js codes easier. The purpose of using Vue Bootstrap in this project is developing better looking websites and increasing code usability while using vue js. (**v2.15.0**)
- **Node-Forge** is a library for cryptology which is built for JavaScript. Purpose of usage in this project is implementing RSA encryption and decryption. It is used on the client side of the web. (**v0.10.0**)
- **Socket.IO** is a JavaScript library for realtime web applications. It enables real time, bi-directional communication between web clients and servers. It is used on simulation subsystem and web subsystem. [6]
- **FFMPEG** FFmpeg is a free and open-source software project consisting of a large suite of libraries and programs for handling video, audio, and other multimedia files and streams. [7]
- **Photon** consists of a server and multiple client SDKs for major platforms. Photon Unity Network (PUN) is our is our take on a Unity specific, high-level solution: Matchmaking, easy to use callbacks, components to synchronize GameObjects, Remote Procedure Calls (RPCs) and similar features provide a great start. Beyond that is a solid, extensive API for more advanced control. [8]

used during our project implementation and testing, that is, our source code production, testing, configuration, etc.

2.3. Hardware and System Software Platform

Unity Game Engine was used while developing this Cyber Drone desktop application. For this reason, the hardware and software platforms requirements listed below are determined to use Unity Game Engine. Computer hardware, network, system software belonging to the Cyber Drone project as follow:

In the Unity Documentation, Unity specifies the performance requirements as follows. [9]

- Windows
 - **Operating system version:** Windows 7 (SP1+) and Windows 10
 - **CPU:** x86, x64 architecture with SSE2 instruction set support.
 - **Graphics API:** DX10, DX11, DX12 capable.
 - **Additional Requirements:** Hardware vendor officially supported drivers.
- Linux
 - **Operating system version:** Ubuntu 16.04 and Ubuntu 18.04
 - **CPU:** x64 architecture with SSE2 instruction set support.
 - **Graphics API:** OpenGL 3.2+, Vulkan capable.
 - **Additional Requirements:** Gnome desktop environment running on top of X11 windowing system. Other configuration and user environment as provided stock with the supported distribution (such as Kernel or Compositor). Nvidia and AMD GPUs using Nvidia official proprietary graphics driver or AMD Mesa graphics driver.
- macOS
 - **Operating system version:** Sierra 10.12+

- **CPU:** x64 architecture with SSE2.
- **Graphics API:** Metal capable Intel and AMD GPUs
- **Additional Requirements:** Apple officially supported drivers.

According to our survey, performance requirement test analysis needs various hardware. In the light of this result the project will be tested on the computers of the group members. Computer with the lowest hardware level in the group which has

- Computer hardware
 - **Processor:** Intel Core i5-7200U
 - **RAM:** 8GB
 - **Graphics Card:** NVIDIA GEFORCE 940 mx 1GB
- System software
 - **Operating system version:** Windows 10 x64 bit
- Network
 - **Download Speed:** 92.4 mbps.

On a computer with these features' suites work well for Web server.

3. Cyber Drone Software Subsystem Testing

This section will be detailed in three part which are Unity system, web server and web client part testing.

3.1. Unity System Testing

In unity subsystem, for the login, menu and settings parts were tested, then the test results showed us all methods worked correctly. Tests were implemented by manually.

- **Login Page Test**
 These test combinations of email and passwords were tested in all cases. Cases can be examined as follows: Invalid email and password, empty password, empty email, valid email and password and empty password empty email. As a result, for all cases excluding valid email and password, server sent invalid credentials error. For valid email and password, server sent success message then user logged in unity subsystem successfully. In unity user logged in time, token and email are stored in xml file. When user logged in these properties are updated. Test showed us, xml file was updated successfully. After user quits from application, when user wants to log in again, from xml file auto login on fields are checked by system then token information is send to server. Another test implemented to this function. And results show us this function works successfully. Until the now tested are implemented for login functions.
- **Menu & Settings Part Test**
 In menu parts, there are several functions, exit button is tested for exiting from simulation, and it passed the test. Play button is tested for loading simulation and it passed the test. Another function is changing user. When user clicked on change user, from xml file token and email values were deleted from that file. Then unity subsystem loads login scene. These functions are passed the test. Thus, settings button loads the settings menu. On the other hand, user configs are got from server and implied to simulation settings. This functionality is tested by music volume and user key. System changes music volume when configs are received and user hotkeys for game is set to user's hotkeys. It is understood by playing game with these hotkeys. When settings menu is opened, user configs are loaded correctly, this function is tested by comparing these values with database records. Another function is

saved user settings, this function is tested by changing user's settings from unity subsystem and comparing changing values with database records. This test is passed.

- **Simulation Part Tests**

For the simulation parts, there are several functions. Each one of them tested manually. Every collider tested one by one. For the drone camera script different position values were tested. Also, different field of view values were observed, and the most optimal version was chosen. These tests were made for both TPV (Third person view) and FPV (First person view).

Max Speeds	
Max Forward Speed	10
Max Sideway Speed	5
Front & Side Movement	
Forward Movement Force	500
Side Movement Force	300
Up & Down Movement	
Up Movement Force	450
Down Movement Force	-200
Tilt Values	
Wanted Forward Tilt	20
Wanted Side Tilt	20
Tilt Movement Speed	0.1
No Tilt Movement Speed	0.3
Rotation Speed	
Rotation Amount	2.5
Drone Slowdown Speed	
Slowdown Time	0.95
Drone Sound Amplifier	
Drone Movement Sound Cr	1

For the drone movement part, different speeds, movement forces, tilt values and rotation speed were tested manually to get the most realistic movement possible. Especially forward and side tilt values tested cautiously. Because in the third person view tilting values effects directly to control the drone. If there is not a good balance between tilting values, it can cause dizziness to the user. You can see what I explained above in the **figure 1**.

Figure 1: Simulation part test.

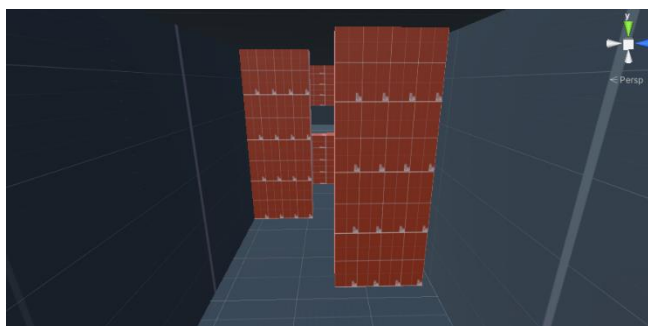


Figure 3: Platforms-1.

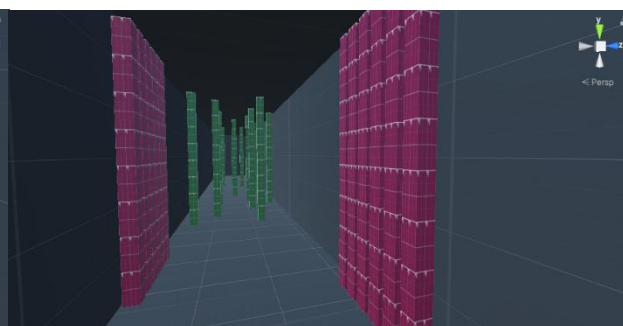


Figure 2: Platforms-2.

For the platforms as you can see in **figure 2 and 3**, the position values were tested to prevent them to stuck inside the walls and floors. The moving platforms especially tested. Because they are moving with their special scripts. These platforms tested manually and observed if they get stuck inside of the walls or floors. Different moving speed values were tested to see if it is fair to interact with. As you can see at the figure above these platforms needs to move in some specific moving speeds, cause in some cases it gets really frustrating to pass them.

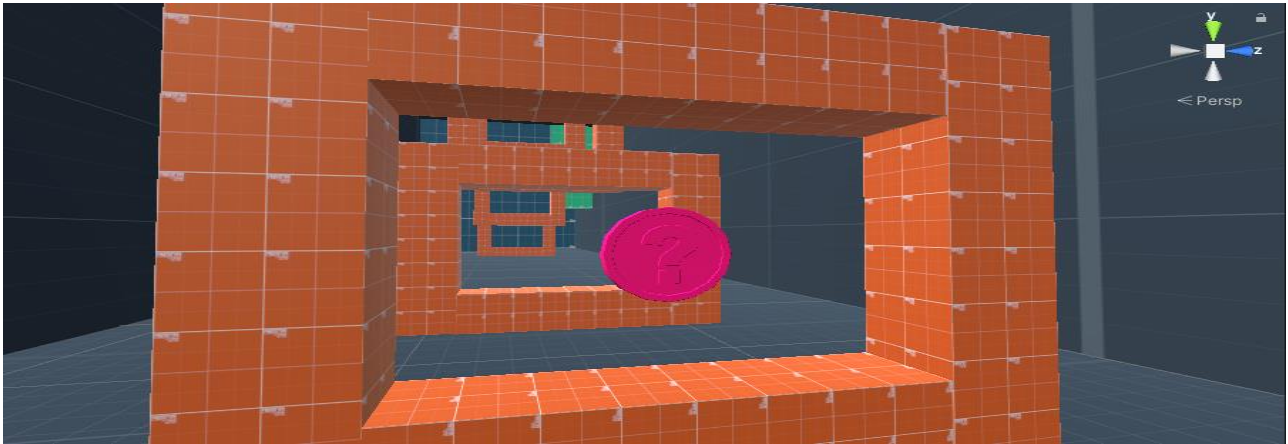


Figure 4:Pickup object.

For the pickup objects like in **figure 4**, trigger tests were made. It tested manually with the “Drone” game object if the trigger of the object working properly. When drone collides with pick-up objects collider it triggers its specific function and proper operation of this function is vital for picking up those objects.

3.2 Web Server System Testing

Web server parts are tested by using Postman Software. Web server services by using APIs. Postman software is suitable for API testing. API tests are done by manually. On the other hand, these APIs are tested by manually from client side of web with using user interface. The request URLs on the web server have been tested one by one.

- **WITH POSTMAN**

- **User Controller API Test**

This controller test was done by using postman software. Firstly, “api/user/squad “ url is tested. In postman, software developers insert one admin token and one user token one by one because this URL serves only admins. Results shows that when authorization token is admin, server returns all users as a response, but in case that token is belongs to normal user, server response 403 forbidden error. It means that the function is passed from that test.

- **“API/User/Delete/{UserID}” URL Test**

In postman, software developers insert one admin token and one user token one by one because this URL serves only admins. Results shows that when authorization token is admin, server continues its functionality, then developers check the valid user id and invalid user id. If user id does not belong to user that saved in system, server must return an error message otherwise server must return a success message. In postman developers inserted one valid user id and one invalid user id one by one. Both of scenarios shows us proper results. It means that function is passed from test. But in case that token is belongs to normal user, server response 403 forbidden error. It means that function is passed from that test below **figure 5** shows us how developers test API’s.

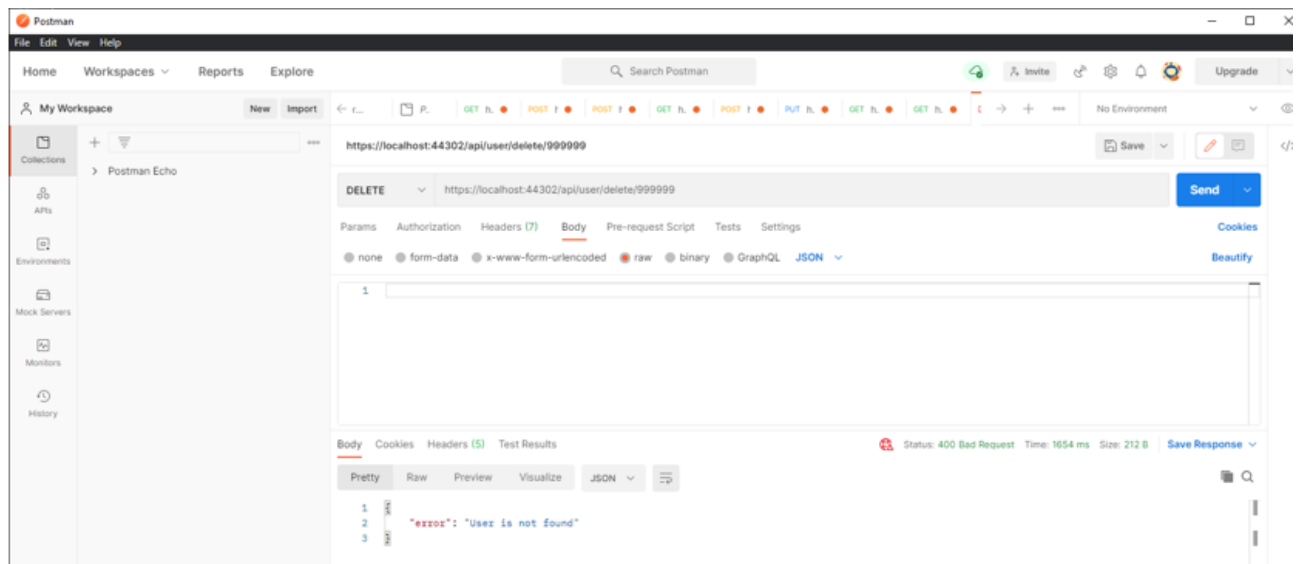


Figure 5: "API/User/Delete/{UserID}" URL Test.

○ "API/User/Add" URL Test

In postman, software developers insert one admin token and one user token one by one because this URL serves only admins. Results shows that when authorization token is admin, server continues its functionality, then in body of request json object must include firstname, lastname, username and email fields on the other hand these values must not be empty string. As you can see in **figure 6** that below, server could not serve due to empty strings.

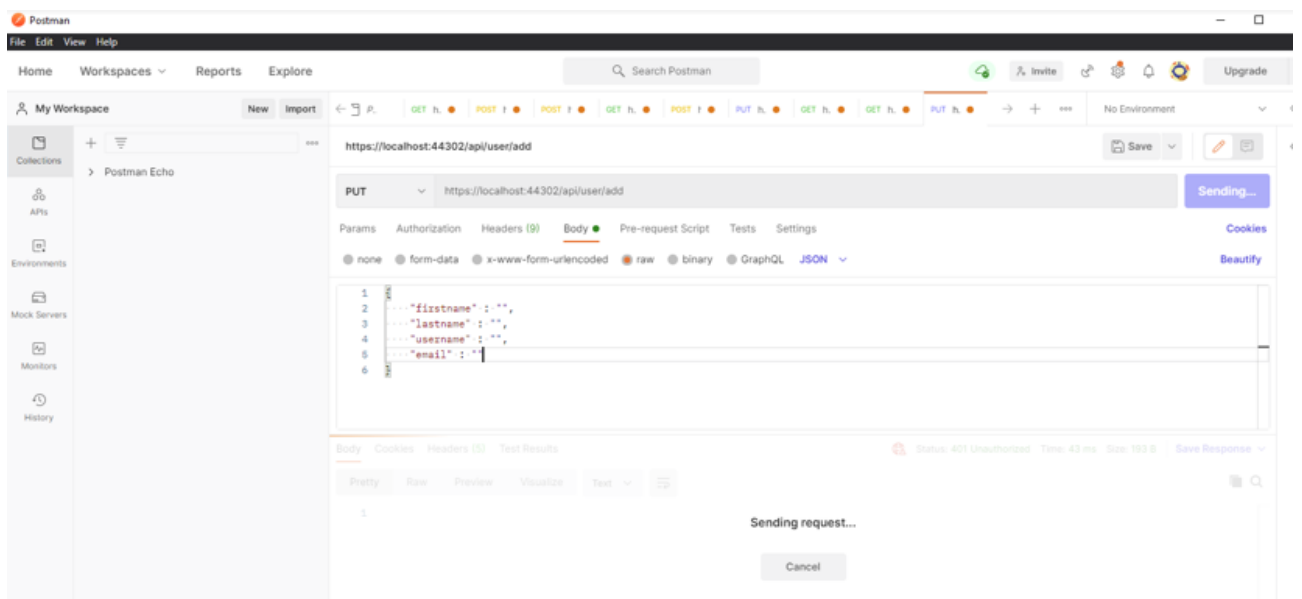


Figure 6: "API/User/Add" URL Test

It means that this does not work correctly. Developers understood the problem and fixed it by adding the code that controls the object information that must be include and attributes inside for empty string. Developers tested API again then developers reached the proper result. **Figure 7** that below shows us to success of API. In that case, server responses an error message.

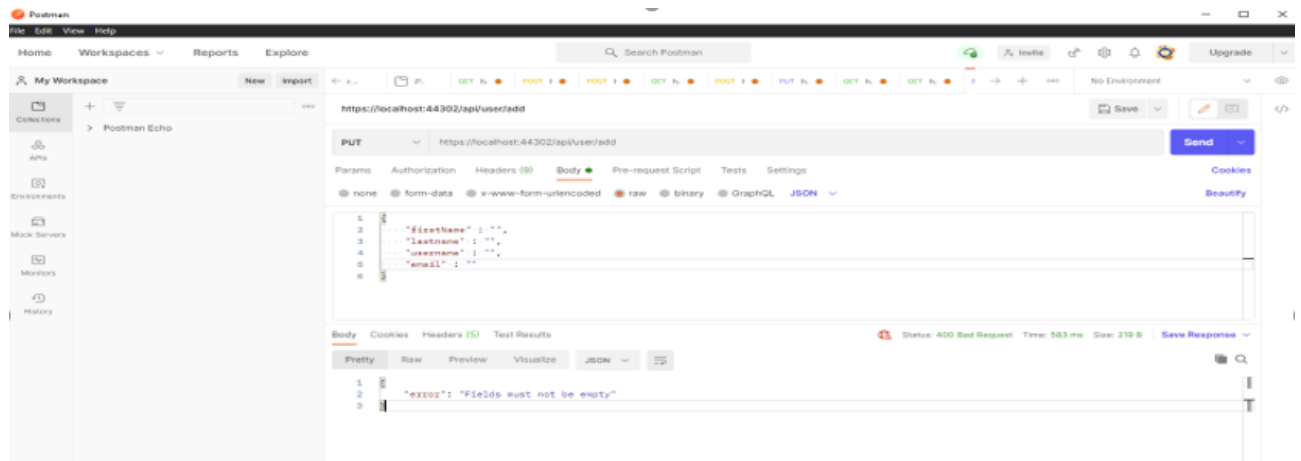


Figure 7: “API/User/Add” URL Test-2

But in case that token is belongs to normal user, server response 403 forbidden error. It means that function is passed from that test this figure shows us how developers test API’s.

○ “API/User/Change-Password “URL Test

In postman, this API serves all user’s that be authenticated, developers tested this functionality by using postman. Developers added authentication token and did not add authentication token. When token was not sent to server, server responded unauthorized 401 error. Otherwise, function continued the service. Request of body must include oldPassword and newPassword values in json object form. When developers tested this API, result shows us an exception which is cause of empty string. **Figure 8** that shown below explains it.

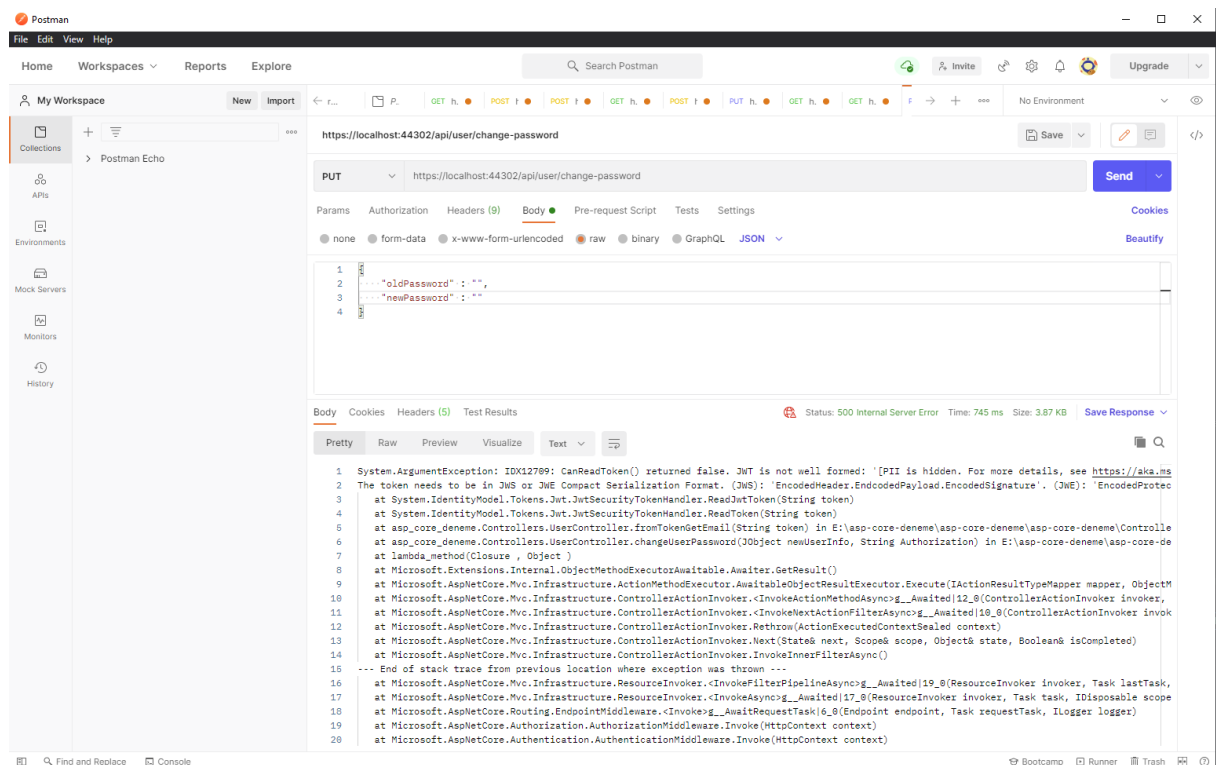


Figure 8: "API/User/Change-Password “URL Test

It means that function could not passed from test. Developers understood the problem and fixed it by adding the code that controls the object information that must be include and attributes inside empty string like in **figure 9**.

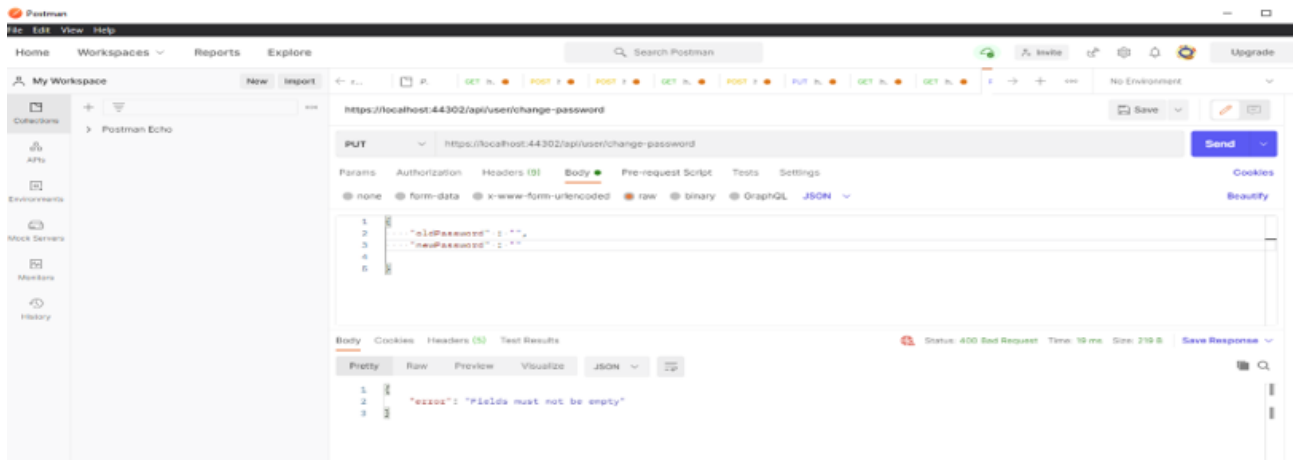


Figure 9:"API/User/Change-Password "URL Test-2

But developers examined another exception called base64 string form. When passwords are not in base64 form, servers throws a exception like in **figure 10**.

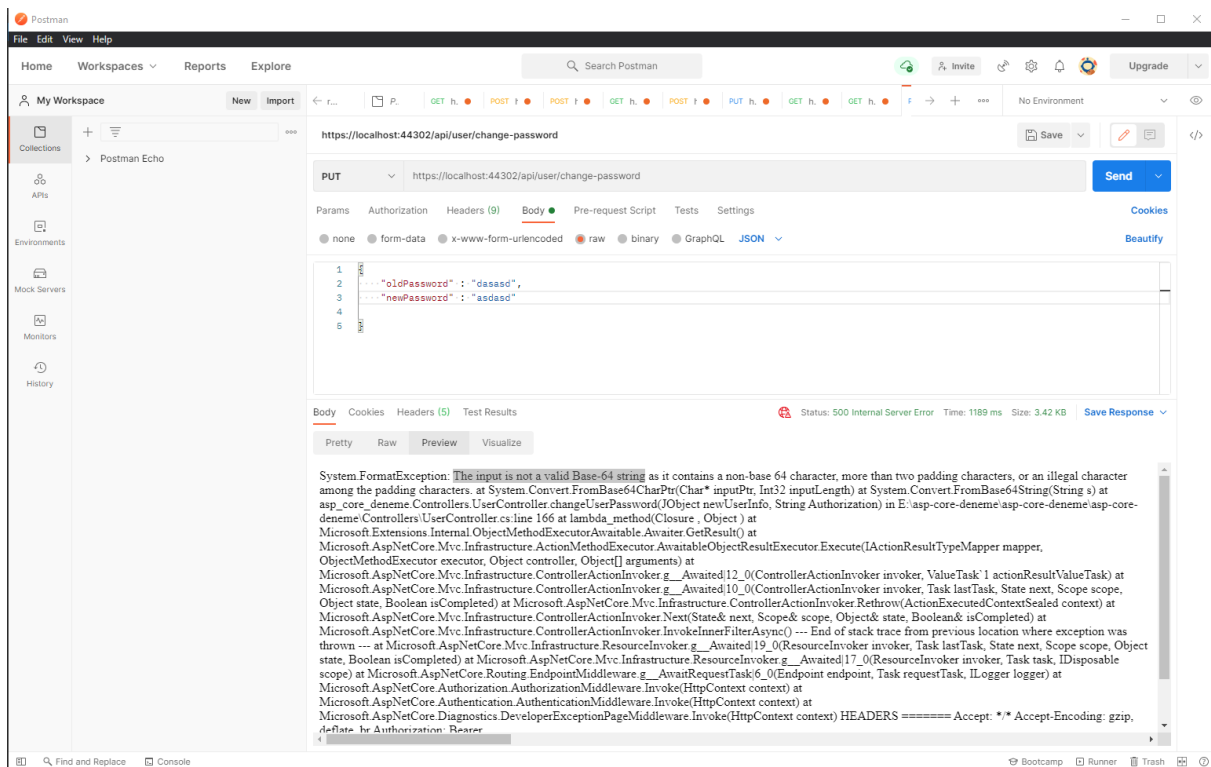


Figure 10:API/User/Change-Password "URL Test-3

Developers understood the problem and fixed it by adding the code that catches exception. **Figure 11** that mentioned below shows that the success of this API.

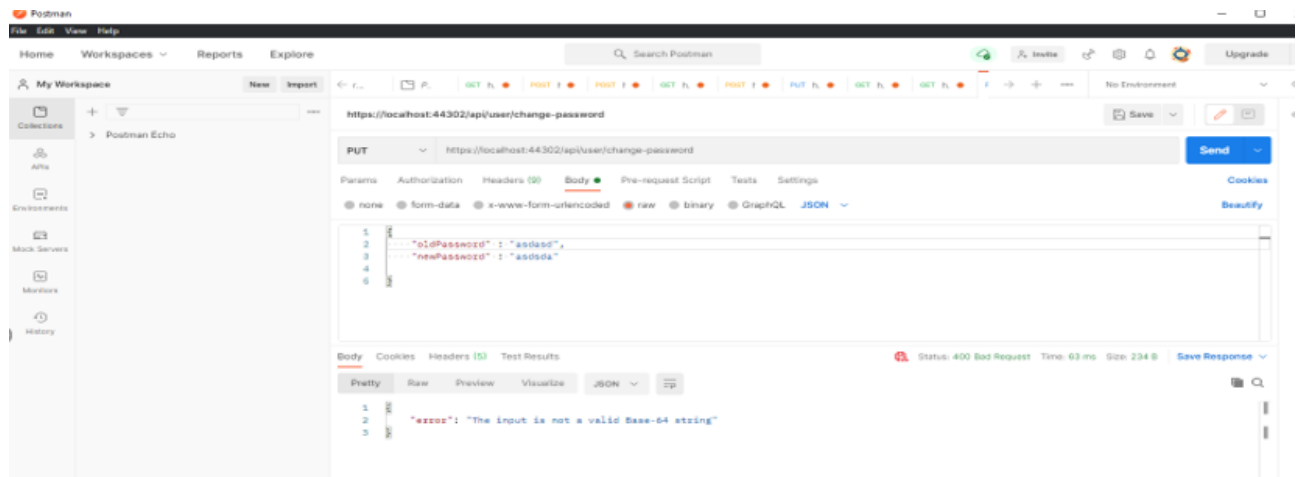


Figure 11:API/User/Change-Password “URL Test-4

○ “API/User/Update/{UserID}” URL Test

In postman, This API serves admin user that be authenticated, developers tested this functionality by using postman. Developers added authentication token and did not add authentication token. When token was not sent to server, server responded unauthorized 401 error. Otherwise, function continued the service. Request of body must include firstName, lastName, username, email, and type information values in json object form. When developers tested this API, developers do not face off any error according to combinations of these information that mentioned above.

○ “API/User/Forget-Password” URL Test

In postman, These API servers everyone. Request of the body must contain email and username fields as json form. The test was sending request of body without these properties and set the properties empty string. In both cases, this function is passed from this test. On the other hand, developers checked the new password is received the email or not. This test was also done successfully as you can see in **figure 12**.

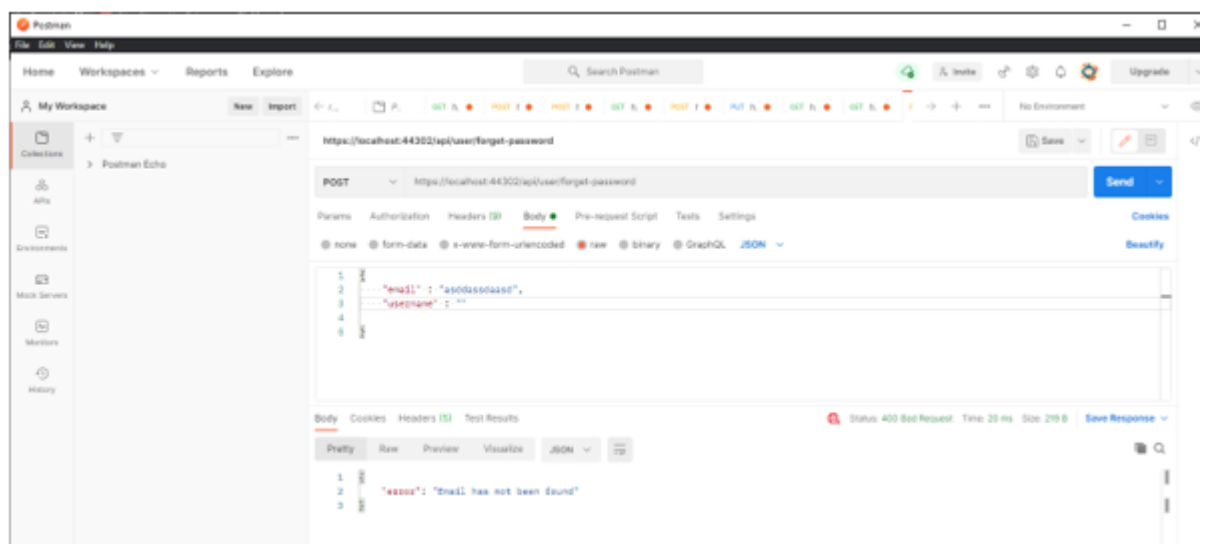


Figure 12:“API/User/Forget-Password” URL Test

○ “API/User/logs” URL Test

This controller test was done by using postman software. “api/user/logs “ url is tested. This api is for only admins. In postman, software developers insert one admin token and

one user token one by one because this URL serves only admins. Results shows that when authorization token is admin, server returns all users as a response, but in case that token is belongs to normal user, server response 403 forbidden error. It means that the function is passed from that test. This test was also done successfully as you can see in **figure 13**.

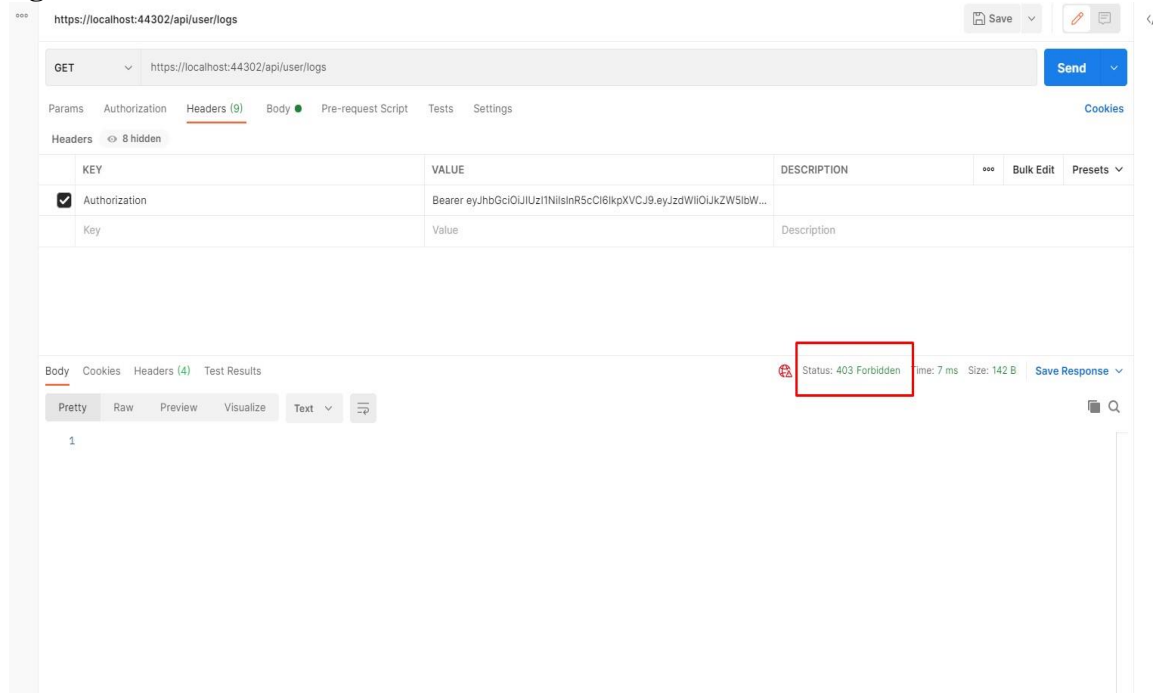


Figure 13: API/User/logs” URL Test

○ “API/bug/all” URL Test

This controller test was done by using postman software. Firstly, “api/bug/all “ url is tested. In postman, software developers insert one admin token and one user token one by one because this URL serves only admins. Results shows that when authorization token is admin, server returns all bugs as a response, but in case that token is belongs to normal user, server response 403 forbidden error. It means that the function is passed from that test. The figure 14 shows the admin user wants to access all bugs by using related API.

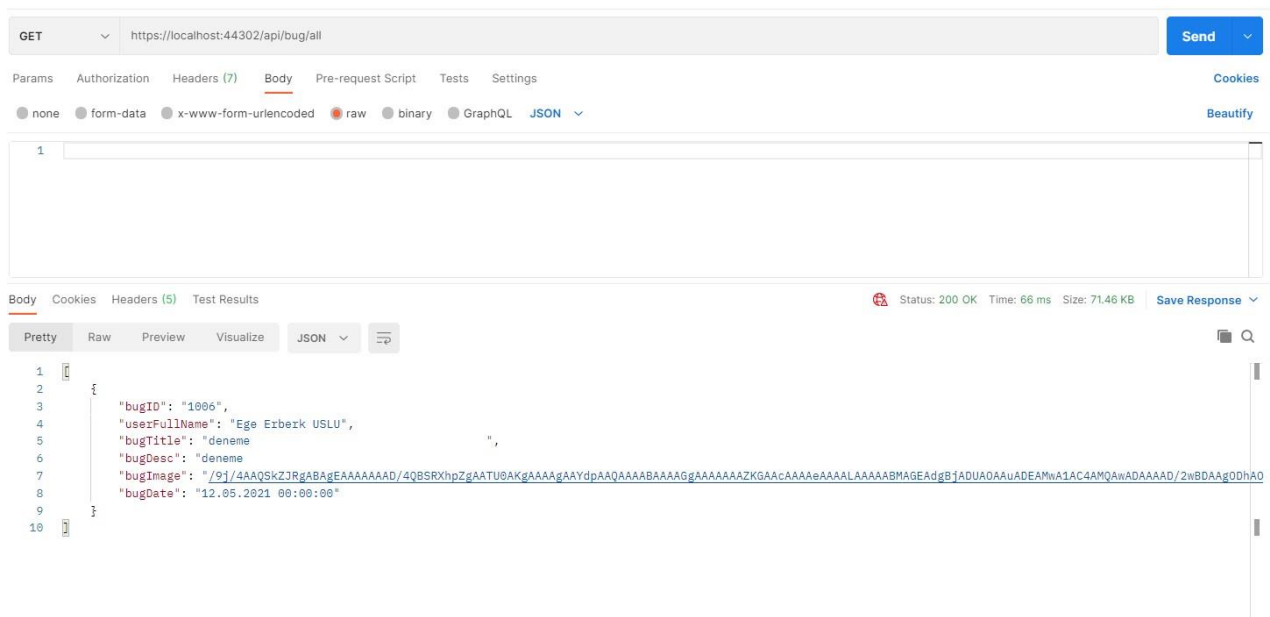


Figure 14: API/bug/all” URL Test

The **figure 15** that below, shows the normal user wants to access all bugs by using related API and it return 403 Forbidden error due to not accessibility for users. This test is passed from our trials.

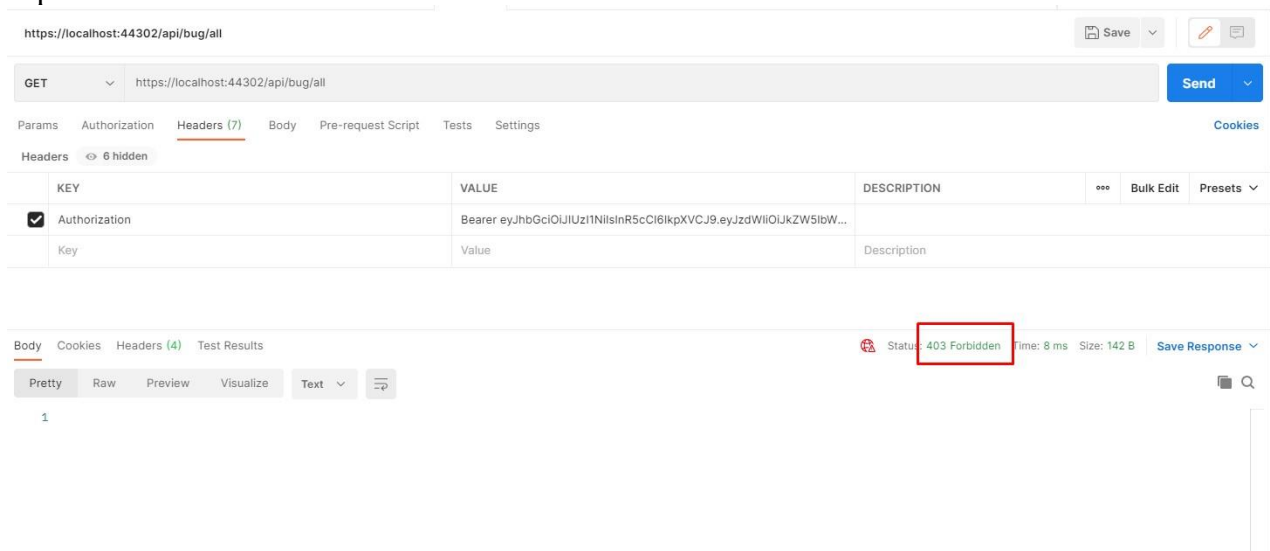


Figure 15 :API/bug/all” URL Test 2

○ “API/user/personal/update/{UserID}” URL Test

This api is for every user that authenticated to system. Request of the body must contain userName, firstName, lastName, email, type, userID and image fields as json form.

Request of header must contain Authorization token. Because this API is for authenticated users. Firstly, system checks whether all fields are included or not. If not, server returns error message called “Fields are empty” message. Figure 16 shows this functionality.

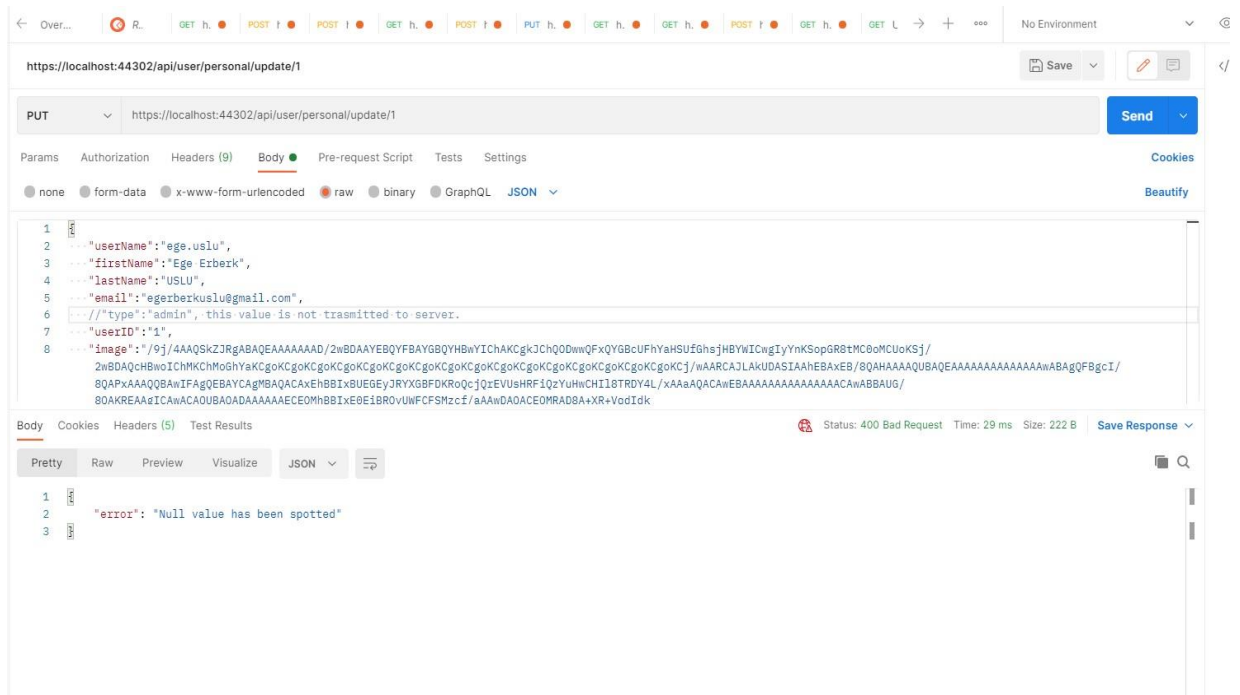


Figure 16: API/user/personal/update/{UserID}” URL Test.

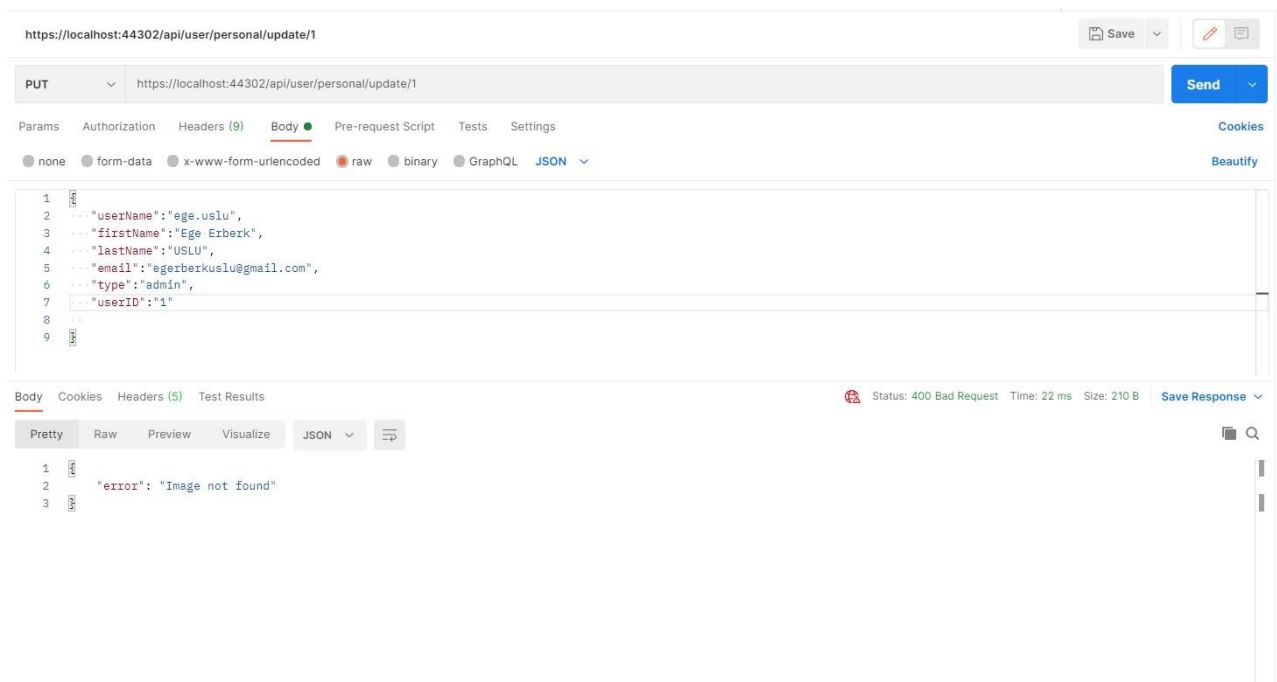


Figure 17 :API/user/personal/update/{UserID}” URL Test 2.

Otherwise, server checks the credentials (Authorization token) are match with the url’s userID. If they are not match, it means clients wants to change another user’s informations. This is forbidden in this api. Figure 17 shows this functionality.

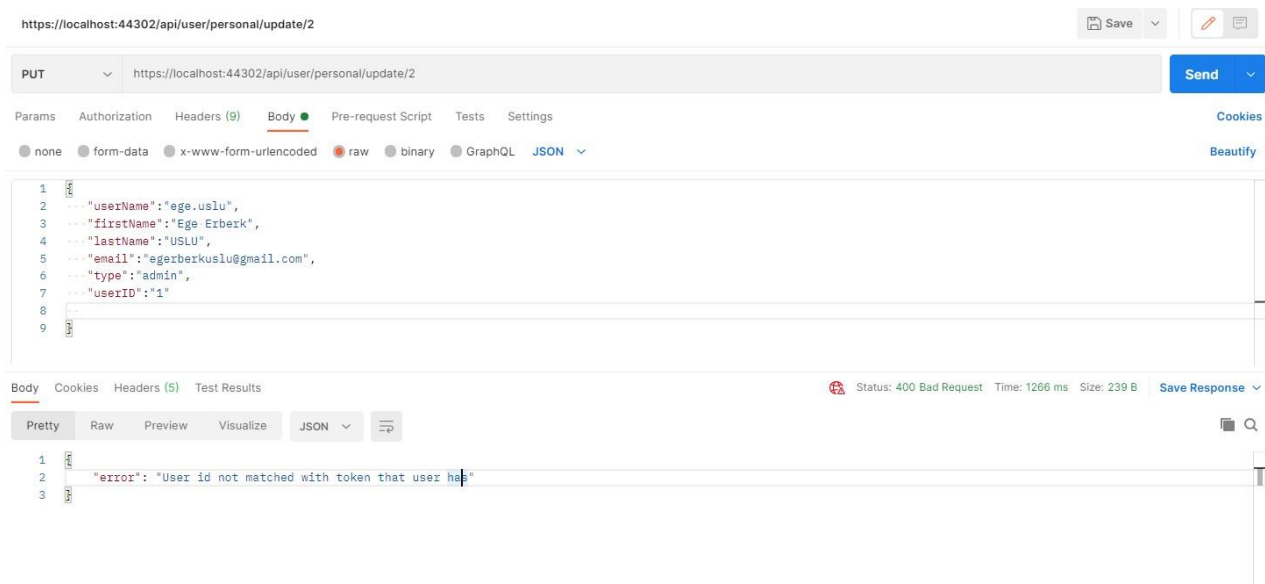


Figure 18: API/user/personal/update/{UserID}" URL Test 3

If everything goes correctly, server's updates related user's information and returns success message to client. **Figure 18** shows this functionality.

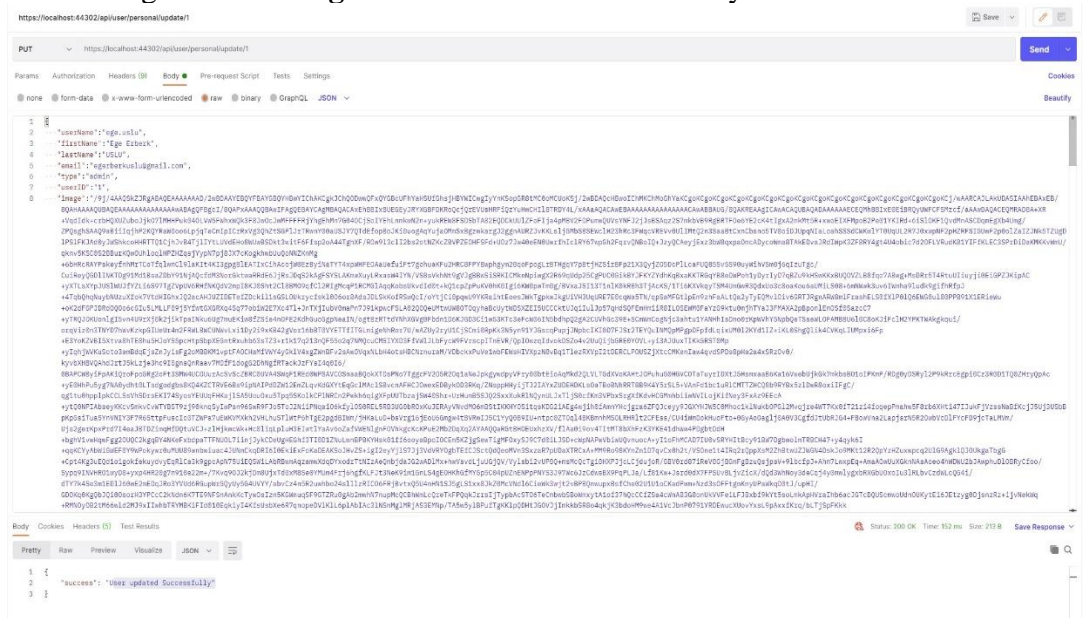


Figure 19: API/user/personal/update/{UserID}" URL Test 4

To sum up, all testes that are applied are passed.

○ “API/bug/upload” URL Test

This api is for every user that authenticated to system. Request of the body must contain title, description and images fields as json form. The test was sending request of body without these properties and set the properties empty string. In both cases, this function is passed from this test. Without using these properties, server sends a bad request error, and it includes the null values has been spotted error message in that. Figure 20 shows the success of uploading bug report.

Authorization. Results shows that when authorization token is admin, server continues its functionality. In postman developers inserted one valid bug id and one invalid bug id one by one. Both of scenarios shows us proper results. It means that function is passed from test. But in case that token is belongs to normal user, server response 403 forbidden error. It means that function is passed from that test below **figure 22** shows us how developers test API's.

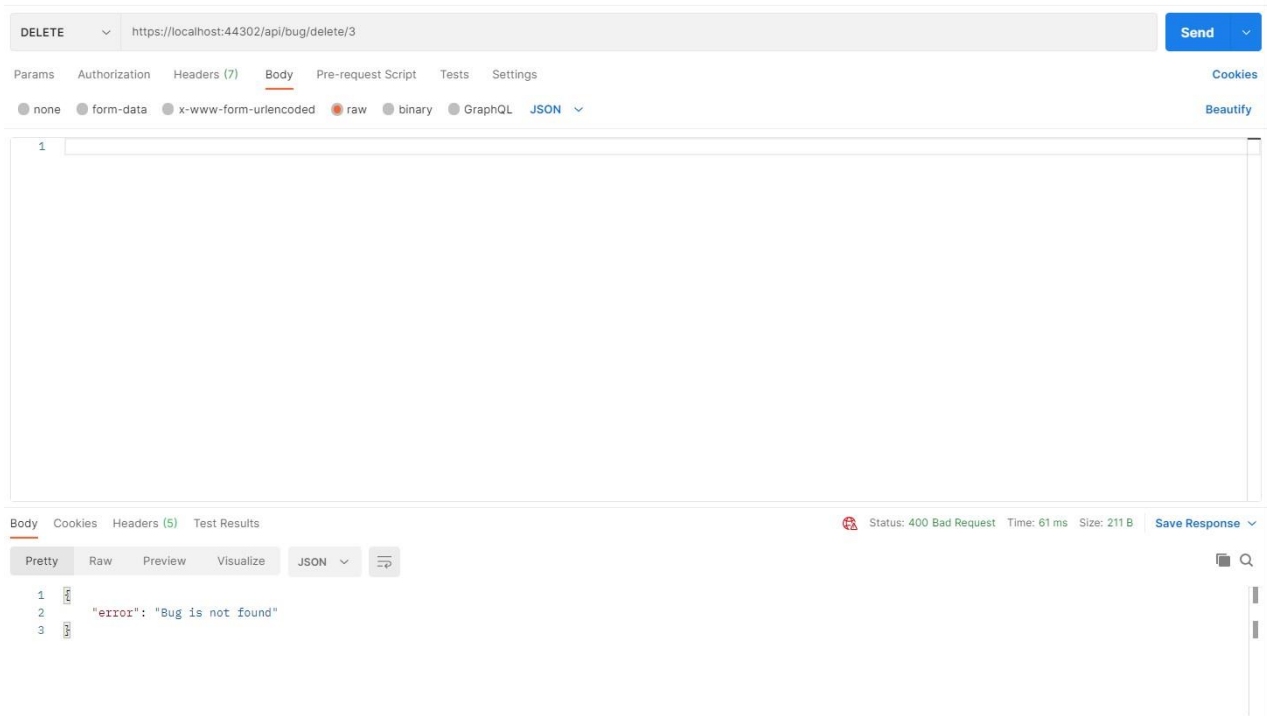


Figure 22: API/bug/delete/{bugID}” URL Test.

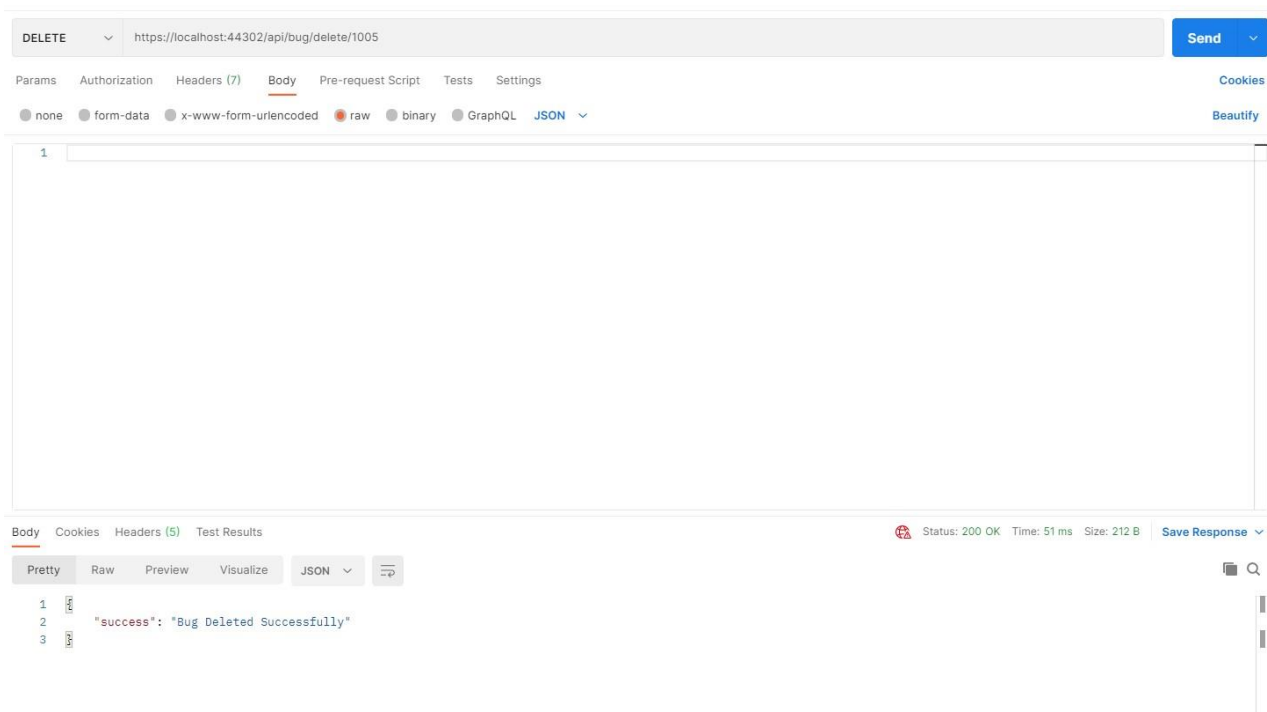


Figure 23: API/bug/delete/{bugID}” URL Test 2.

○ “API/bug/logs/” URL Test

This controller test was done by using postman software. “api/bug/logs “ url is tested. This api is for only admins. In postman, software developers insert one admin token and one user token one by one because this URL serves only admins. Results

GET https://localhost:44302/api/bog/logs

Headers (7) Body Pre-request Script Tests Settings

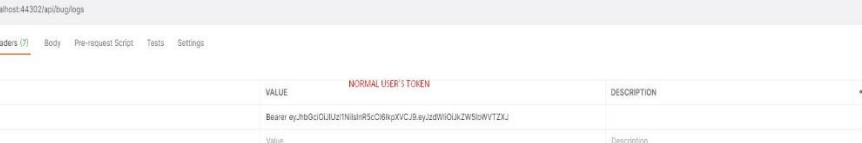
Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIjOiJWQWZlZWY5bmVtZkJ9

Key Value Description Bulk Edit Presets

Key Value Description

Status: 200 OK Time: 1883 ms Size: 732 KB Save Response

```
{
  "logID": "1",
  "operationType": "INSERT",
  "uuid": "1",
  "title": "ORCHestrator",
  "description": "CDKHLIST",
  "image": "schdowcl.jpg",
  "date": "8.04.2023 00:00:00",
  "tagline": "8.04.2023 00:00:00",
  "userFullName": "Egr Kabeok USLU"
},
{
  "logID": "2",
  "operationType": "DELETE",
  "uuid": "1",
  "title": "ORCHestrator",
  "description": "CDKHLIST",
  "image": "schdowcl.jpg",
  "date": "8.04.2023 00:00:00",
  "tagline": "8.04.2023 00:00:00",
  "userFullName": "Egr Kabeok USLU"
}
```



https://localhost:44302/api/bug/logs

GET https://localhost:44302/api/bug/logs

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Headers 6 hidden

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Authorization	Bearer ey_JnQdUjUzTlNmR3c0RlpKVCJzayZdWjY0UkZW50WVZKJw	
Key	Value	Description

Body Cookies Headers (4) Test Results

1

Status: 403 Forbidden

17 ms Size: 142 B Save Response

Pretty Raw Preview Visualize Text

- **“API/config/user/” URL Test**

24

Figure 26 and 27 shows the true functionality of this method.

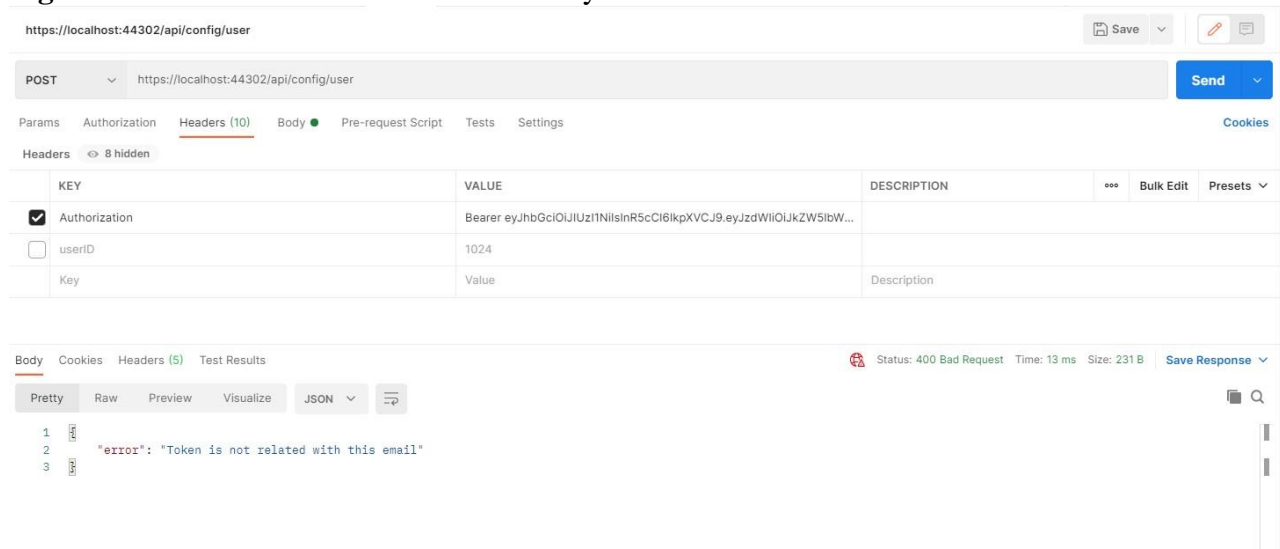


Figure 26: API/config/user/” URL Test.

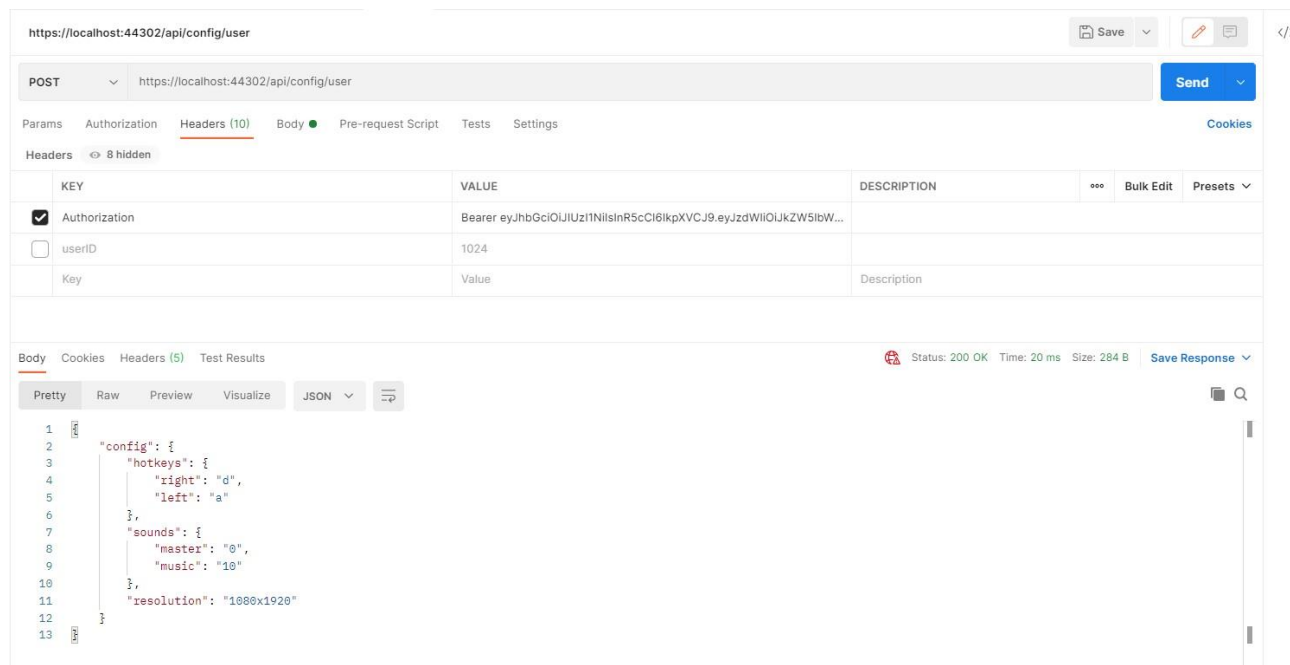


Figure 27 :API/config/user/” URL Test 2.

○ “API/config/updateuser” URL Test

This api is for any user that authenticated to system. Request must contain the email and config value in body of request as json form. Request must contain the authorization token in the header part of request called Authorization. Results shows that when authorization token is valid for the system, server continues its functionality. Server checks the email address of given token and email address of json object. This is applied for user only can change its own configuration settings. Server checks all values that must contains such as email and config fields. If request of body does not contain these values server return a bad request and sends the error message. **Figure 28** shows that this

functionality.

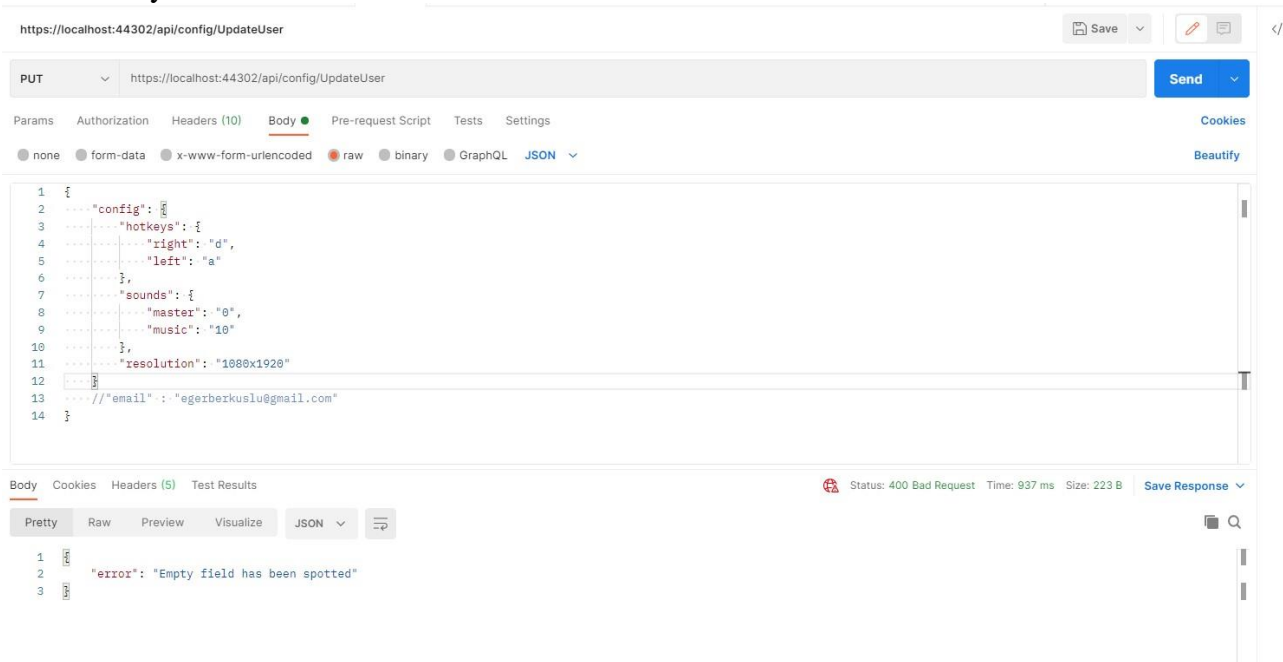


Figure 28: API/config/updateuser” URL Test

Otherwise, if all fields are filled and token and email are matched, server gives the configuration settings in json form. **Figure 29** shows that this functionality.

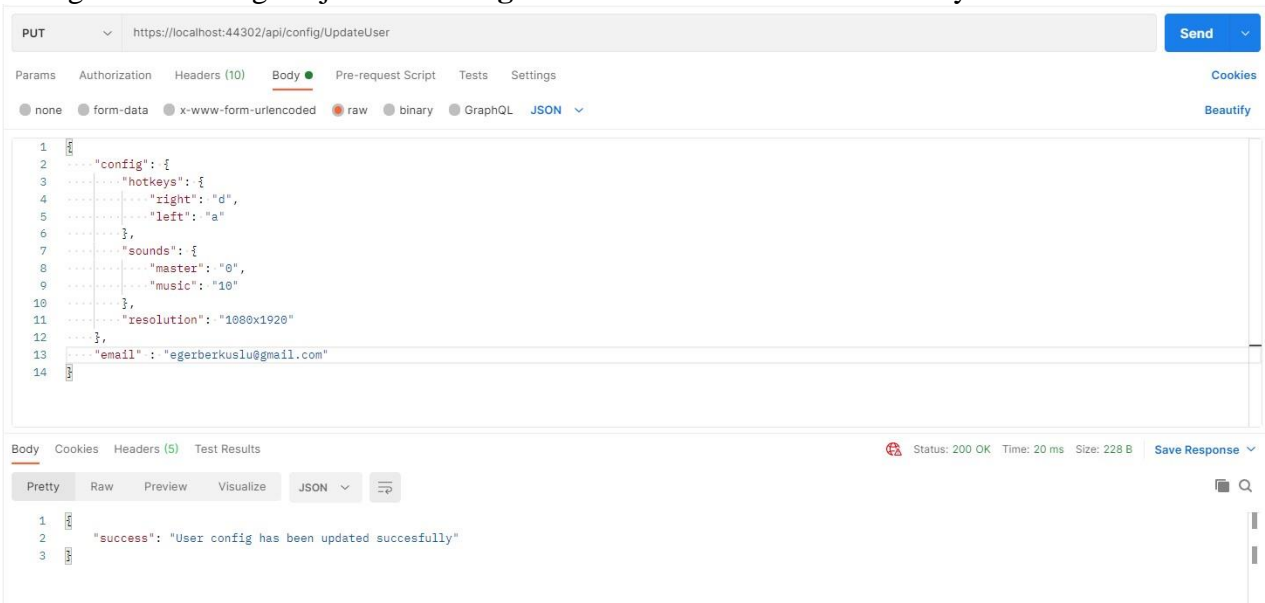


Figure 29 :API/config/updateuser” URL Test 2.

if all fields are filled and token and email are not matched, server gives the error message. **Figure 30** shows that this functionality.

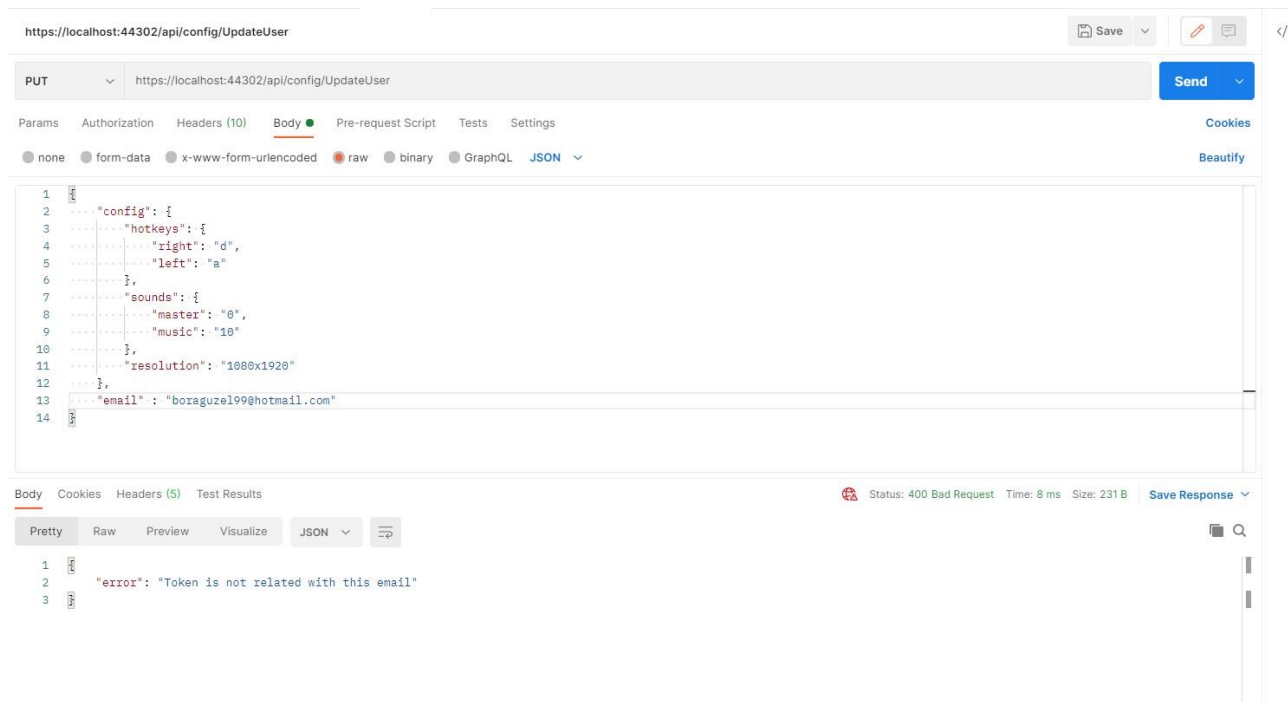


Figure 30 :API/config/updateuser” URL Test 3

To sum up, all testes that are applied are passed from our trials.

○ “API/record/upload” URL Test

This api is for any user that authenticated to system. Request must contain theFile, token and name fields in body of request as form data form. Firstly, token is controlled by the web server. If token field is empty in form-data, system returns token is null error to client. After that, if token is not matched with any token that is stored in system previously, system returns 403 forbidden(unauthorized) error. Then system checks the other fields for validity. One of the fields is empty, system returns bad request error. **Figure 31** shows this functionality works properly.

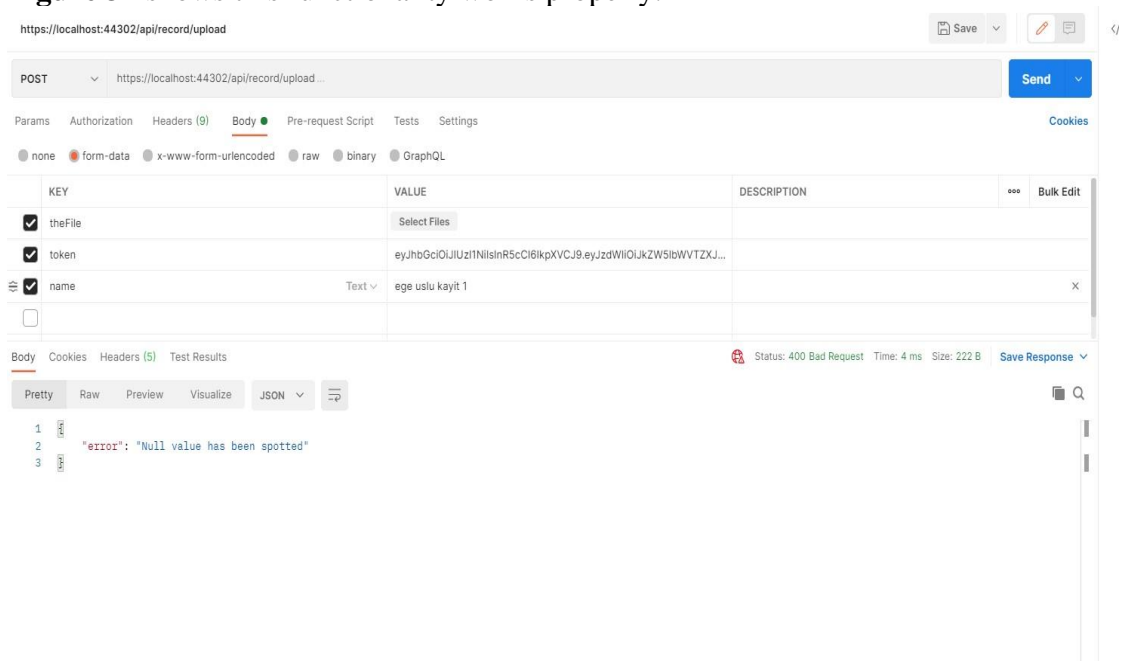


Figure 31: API/record/upload” URL Test

At the end, if all fields are valid and not empty, system returns success message to client. **Figure 32** shows this functionality works properly.

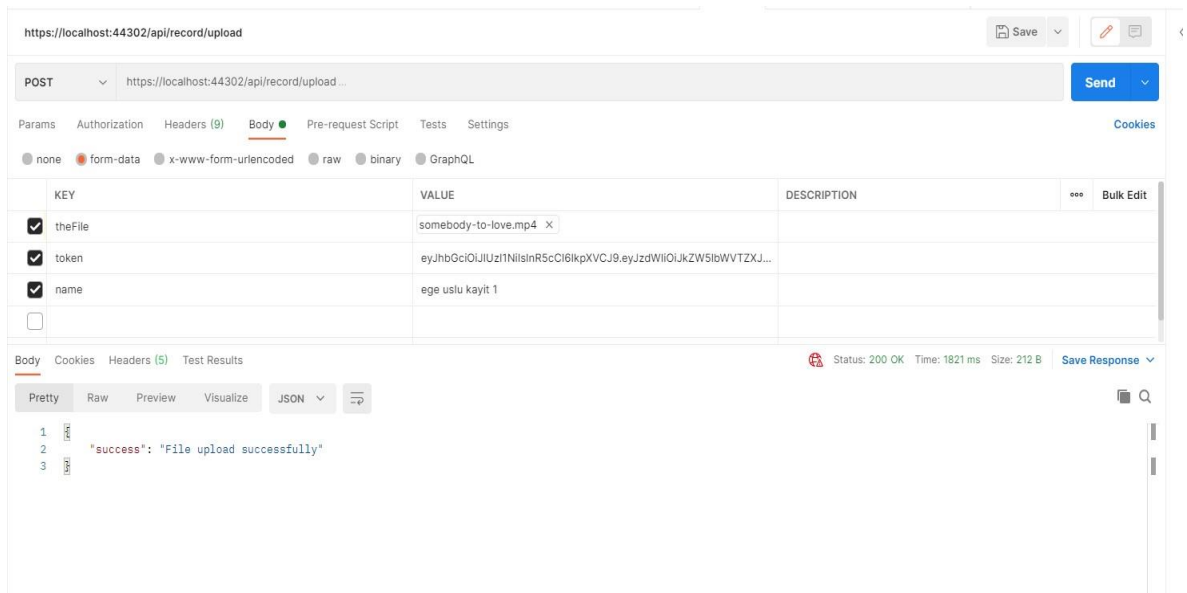


Figure 32: “API/record/upload” URL Test 2

○ “API/record/download/{filename}” URL Test

This api is for any user that authenticated to system. Request must contain Authorization token and userID in the header part of request. Web server checks the token’s validity firstly, if token is not stored in the system, system return bad request error. Otherwise , system checks the userID field if normal user makes a request. If user is admin user, system does not check userid. System returns error message if userID field is not filled. Otherwise, system goes its functionality. If userid is not matched with the token or requested file is not related to user that makes a request, server returns an error message.

Figure 33 shows this functionality works properly.

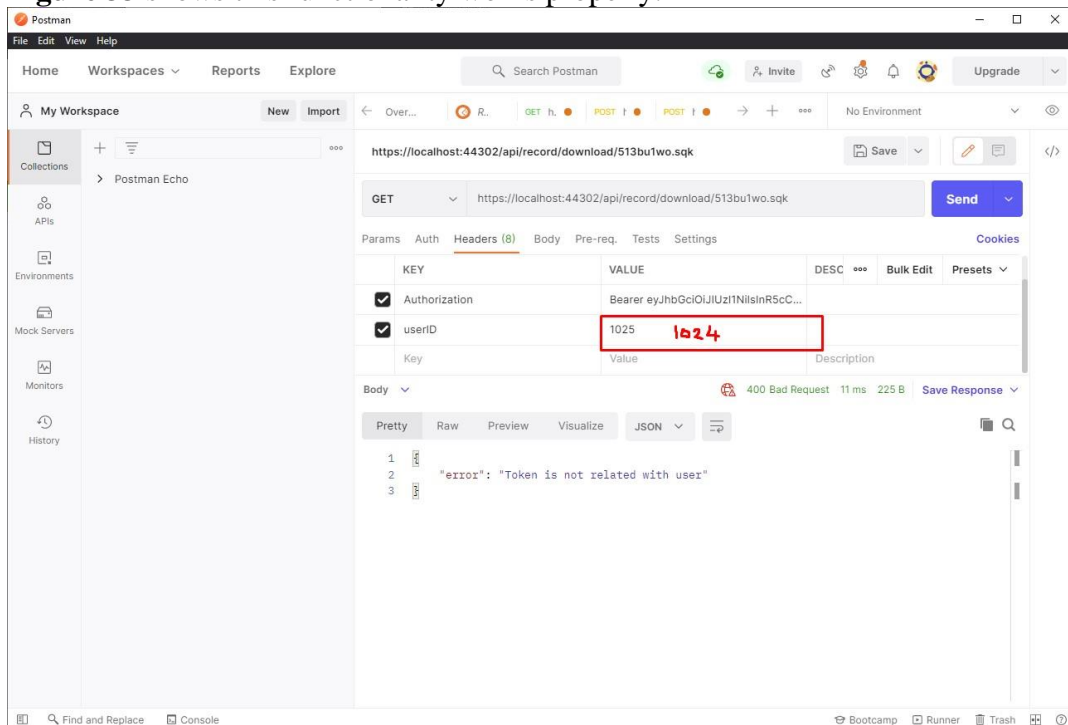


Figure 33: “API/record/download/{filename}” URL Test .

If system does not catch any error, related file is transmitted to client. To sum up, all testes are passed from our trials.

- **“API/record/delete/{filename}” URL Test**

This api is for any user that authenticated to system. Request must contain Authorization token and userID in the header part of request. Web server checks the token’s validity firstly, if token is not stored in the system, system return bad request error. Otherwise, system checks the userID field if normal user makes a request. If user is admin user, system does not check userid. System returns error message if userID field is not filled. Otherwise, system goes its functionality. If userid is not matched with the token or requested file is not related to user that makes a request, server returns an error message. Figure (BIR ALTTAKI) shows normal user gives wrong id according to files’s userid. Then server returns error message.

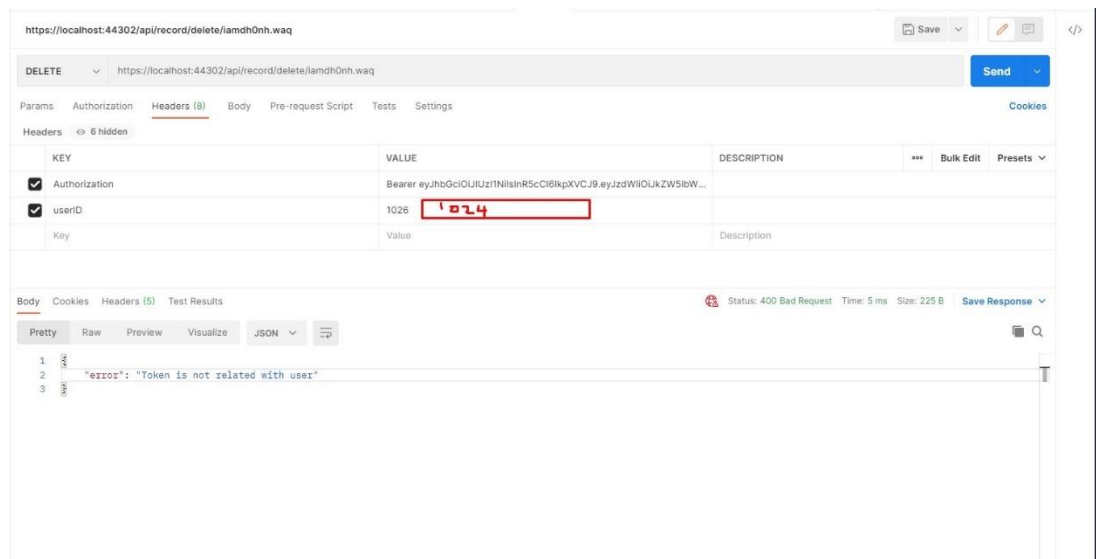


Figure 34 :API/record/delete/{filename}” URL Test.

Figure 35 shows normal user gives true id according to files’s userid. Then server returns success message.

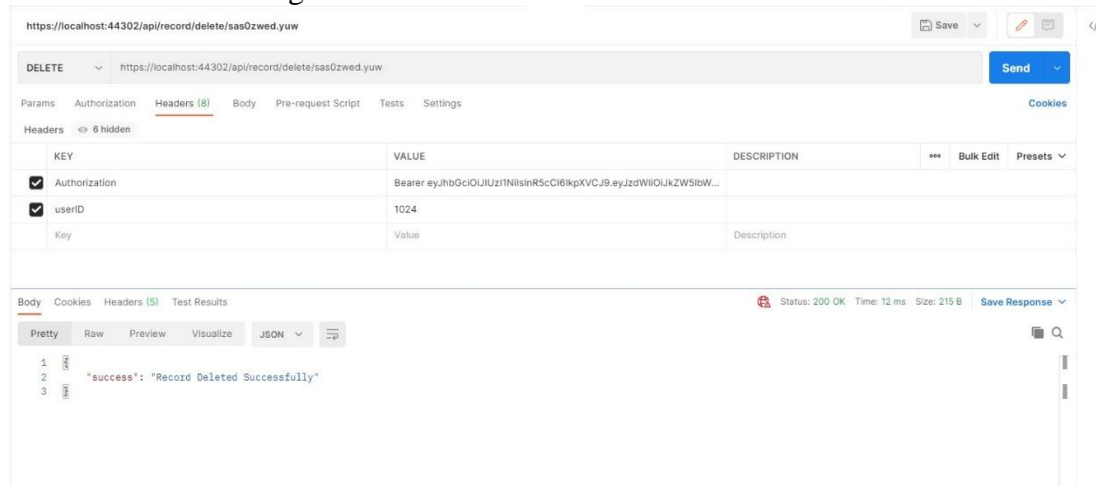


Figure 35: API/record/delete/{filename}” URL Test 2

If admin makes a request to api, system does not check the user id fields so admin is able to delete any file that are in the system. **Figure 36** shows admin user making a

request to delete record.

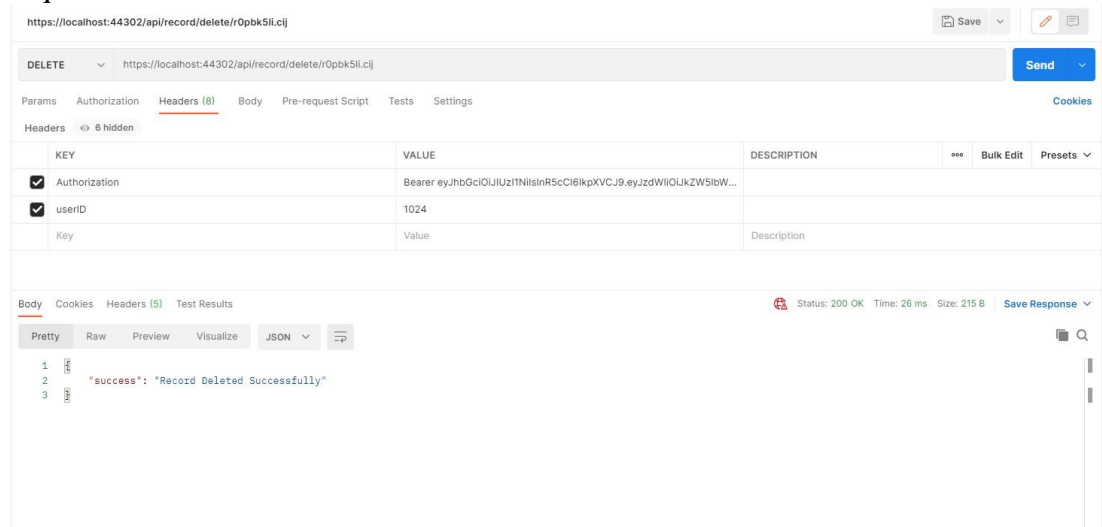


Figure 36: API/record/delete/{filename}” URL Test 3

Another exception is file not found. If any user wants to delete file does not exist in server, web server returns an error message called “File not found” error. **Figure 37** shows this functionality works properly.

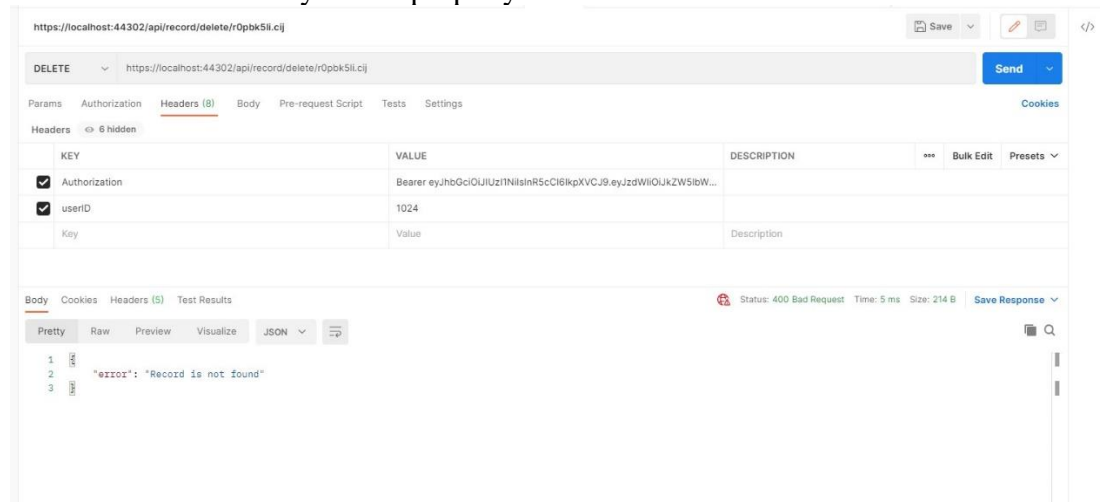


Figure 37: API/record/delete/{filename}” URL Test 4

To sum up, all testes that applied are passed from our trials.

○ “API/record/logs” URL Test

This controller test was done by using postman software. “api/record/logs “ url is tested. This api is for only admins. In postman, software developers insert one admin token and one user token one by one because this URL serves only admins. Results shows that when authorization token is admin, server returns all users as a response, but in case that token is belongs to normal user, server response 403 forbidden error. It means that the function is passed from that test. This test was also done successfully as you can see in **figure 38**.

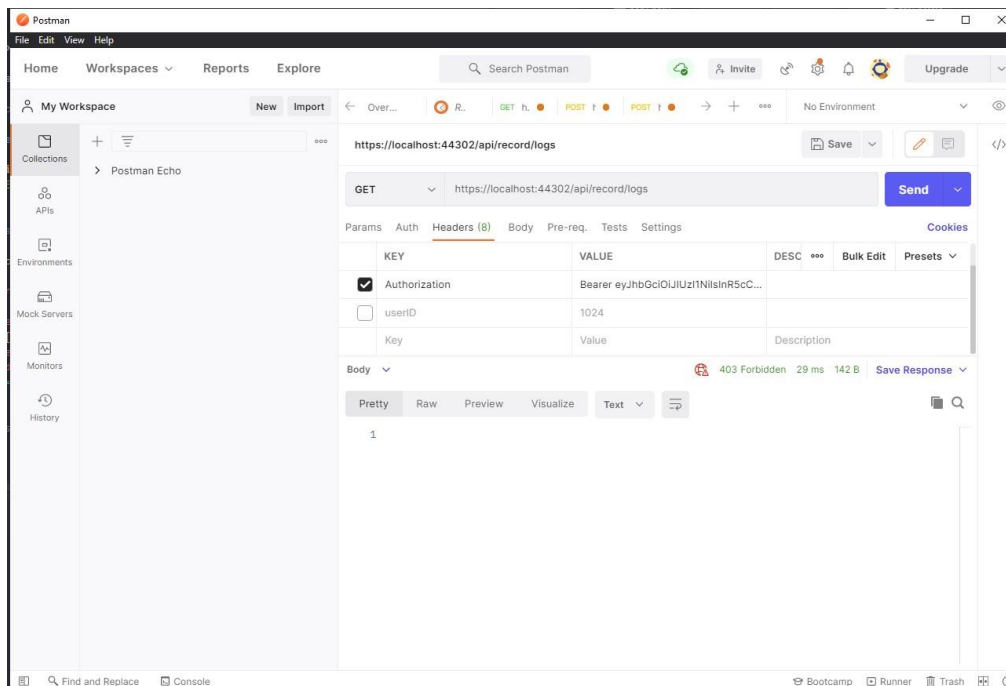


Figure 38: API/record/logs” URL Test.

○ “API/record/thumbnails/{userID}” URL Test

this api is for any user that authenticated to system. Request must contain Authorization token in the header part of request. Web server checks the token’s validity firstly, if token is not stored in the system, system return bad request error. Otherwise, system checks the userID field if normal user makes a request. If user is admin user, system does not check userid. System returns error message if userID field is not filled. Otherwise, system goes its functionality. If userid is not matched with the token and user is normal user, server returns a error message. Figure 39 shows normal user gives wrong id according to request URL. Then server returns error message.

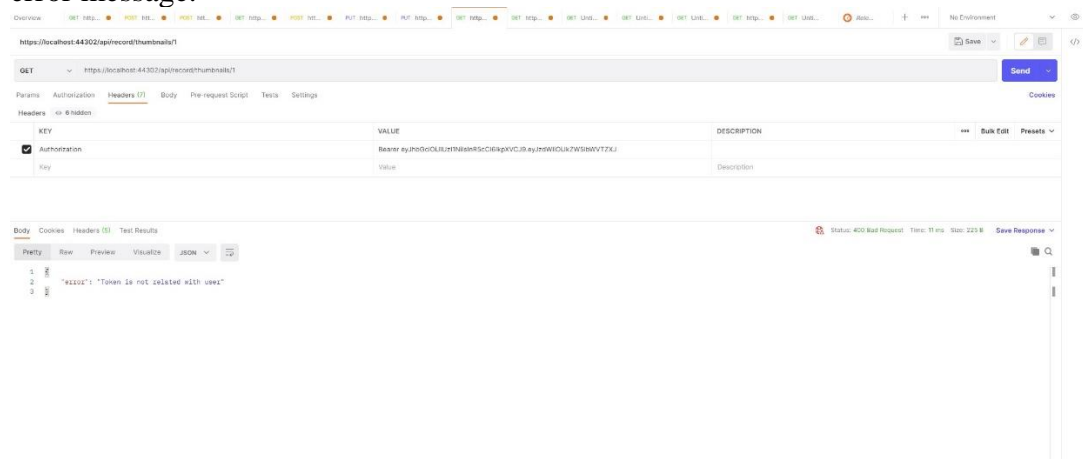


Figure 39: API/record/thumbnails/{userID}” URL Test.

If token’s id and requested user’s id (in URL) is matched and user is normal user, systems return the related json in response message. **Figure 40** shows normal user gives

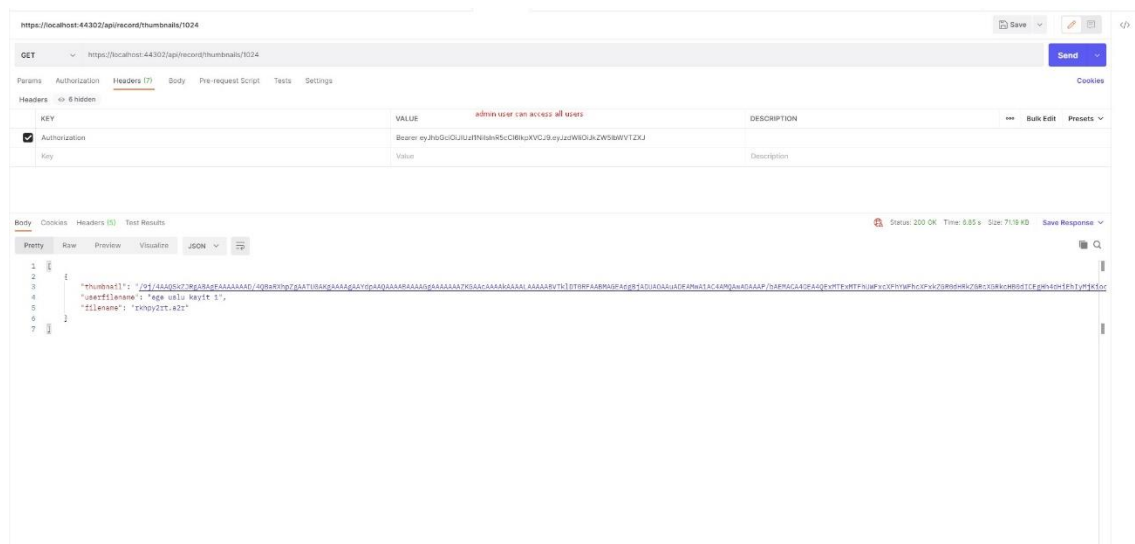
The screenshot displays a REST client interface with the following details:

- URL:** `https://localhost:44302/api/reorder/thumbnails/1024`
- Method:** GET
- Headers:**
 - Authorization:** Bearer ey.jhb9cDUJzstNtkon-K5cCGgKVCjBeyJzdBWOLuZ9SbMYTZUj
- Body:**

```

{
  "thumbnail": "/7f/4AM2bZ3hg8hgAAAAAAAAA/4QaRxp2gMTb8AgAAAAgATdpAAgMAABMAAAQAAAAAAAAAAZGAAcAAAAAAAAAALMAAAZVbT3T8FAMgMAAggZJQa0AAuA2cAAALC4APQAAuABAAAP/BALWMAc4ZEAQ2AArTEAArTfThMFxCGfYhMFbYFAZSRQhRz2SRzSRbuc89dC1Sgm4oh16h3rMfJzS",
  "overwrite": "qgv vsls heyit 1",
  "filename": "xohpyt11.s2f"
}

```
- Status:** 200 OK, Time: 52 ms, Size: 7159 KB



3.3 Web Client System Testing

Web client system tested by manually.

- **Login Test**

Developers checked login page working properly or not. Email input and password input must not be empty. On the other hand, email field must contain @ symbol. These were tested and functions were passed from test in **figure 42**.

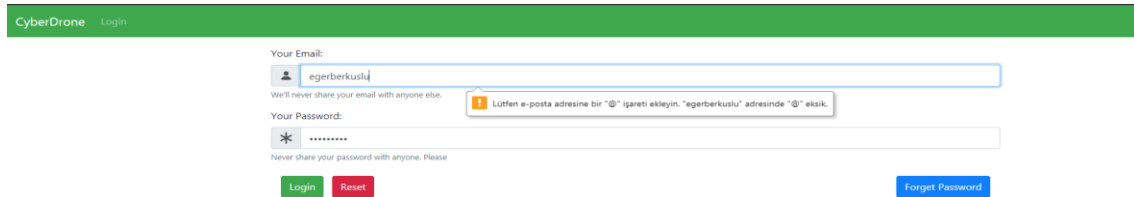
The screenshot shows the CyberDrone login interface. At the top, there's a green header with 'CyberDrone' and a 'Login' link. Below the header, the 'Your Email:' field contains 'egerberkuslu'. A small message below the email field says 'We'll never share your email with anyone else.' and a red warning box contains the text 'Lütfen e-posta adresine bir "@" işareti ekleyin. "egerberkuslu" adresinde "@" eksik.' The 'Your Password:' field is masked with asterisks. Below the password field, there's a note 'Never share your password with anyone. Please'. At the bottom, there are three buttons: 'Login' (green), 'Reset' (red), and 'Forgot Password' (blue).

Figure 42:Login Test

After that developers tried wrong login information. Then system was passed from test. Server response an error message and web clients show an error message to user like in **figure 43**.

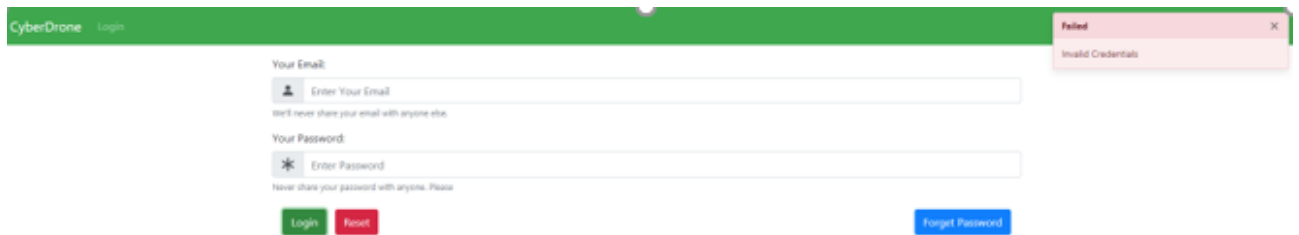
The screenshot shows the CyberDrone login interface after an unsuccessful login attempt. The 'Your Email:' field is empty and labeled 'Enter Your Email'. The 'Your Password:' field is empty and labeled 'Enter Password'. A red error message box in the top right corner says 'Failed' and 'Invalid Credentials'. The 'Login' (green) and 'Reset' (red) buttons are at the bottom left, and the 'Forgot Password' (blue) button is at the bottom right.

Figure 43:Login Test-2

- **Admin/User Navigation Bar Test**

After entering the web client subsystem, only admin must see the add user field and users' field in navigation bar. Developers tested this function. System was passed from that test. Another test is if user is normal user, this user must not be able to see add user or user's page. This function is tested in routing/navigation guard part. This test was passed. These **figure 44** that shown in below explains the correctness of this function.

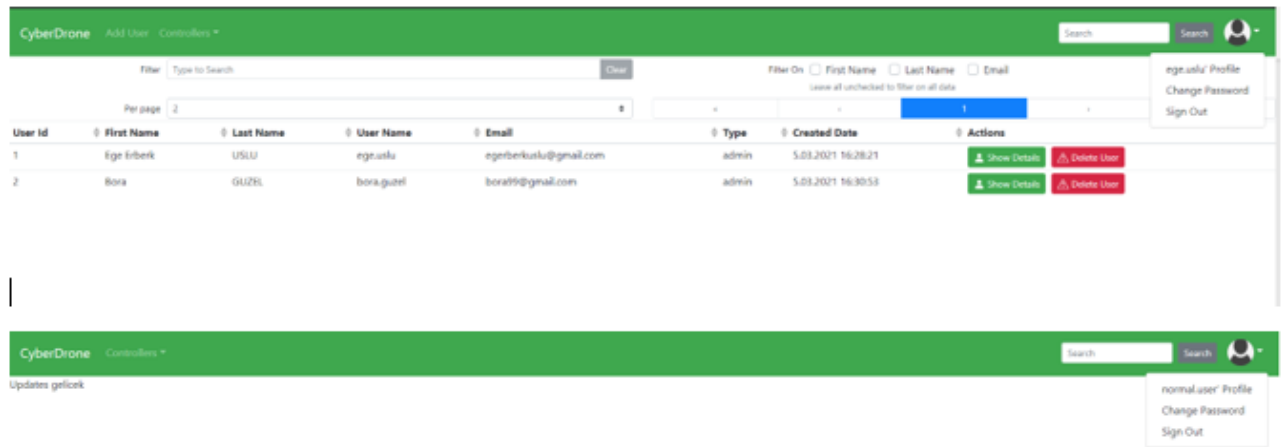


Figure 44:Admin/User Navigatio Bar Test.

- **Add User Test**

Another test was applied to adding user function. When web client user wants to add user firstname, lastname, email and username fields must be field. The system should show an error message when any of them are even empty. **Figure 45** that shown below, shows the function is passed from that test.

Figure 45:Add user test.

The system must check emails registration status cause of emails are unique. This functionality was tested, and results shows us to this function was passed from that test. Another test is sending new password to newcomer user. This test passed from test. **Figures 46, 47** that mentioned below shows us that. Also, when admin enters these fields properly system shows success message.

CyberDrone Add User Controllers

Failed
Email has already registered

First Name: Ege

Last Name: USLU

Email: egerberkuslu@gmail.com

UserName: ege.uslu

User password generated randomly and automatically. Passwords are send to user's email.

Register Reset

Figure 46: Add user test-2

Your Password has been set [Gelen Kutusu](#)

cyberdronedronomatrix@gmail.com
Alici: ben

Your password is

wgoo3348TW

Figure 47: Add user test mail.

- **Getting All User & More Detailed Test**

Another test is showing users and their details. When users page is loaded, this page must show all users and details. **Figure 48** that shown below, this function is passed from that test.

CyberDrone Add User Controllers

Filter: Type to Search

Filter On: First Name Last Name Email

Leave all unchecked to filter on all data

User Id	First Name	Last Name	User Name	Email	Created Date	Actions
1	Ege Erberk	USLU	ege.uslu	egerberkuslu@gmail.com	5.03.2021 16:28:21	Show Details Delete User
2	Bora	GUZEL	bora.guzel	bora.guzel@gmail.com	5.03.2021 16:30:53	Show Details Delete User
17	Simge	Ozdemir	simge.binnaz	simge.binnaz@gmail.com	17.03.2021 20:21:20	Show Details Delete User
18	Bora	Guzel	bora.guzel	bora.guzel@gmail.com	17.03.2021 20:24:31	Show Details Delete User
1014	normal user	normal user	normal.user	perver@gmail.com	25.03.2021 21:37:52	Show Details Delete User
1015	memhmet	mehmet	emu.uslu	emu.uslu@gmail.com	30.03.2021 16:32:05	Show Details Delete User

Ege Erberk USLU's Detailed Info

First Name: Ege Erberk

Last Name: USLU

User Name: ege.uslu

User Email: egerberkuslu@gmail.com

Role Name: admin

Update Close

Figure 48: Getting All User & More Detailed Test

- **User Details Updating Test**

Another test was applied to user detail's updating test. When admin wants to change any user's record, web system must check all fields must be not empty and if admin wants to change any user's email, that email must not be registered in system. This test was applied to this functionality. This functionality was passed from test and **figure 49** shows that also when admin enters these fields properly system shows success message.

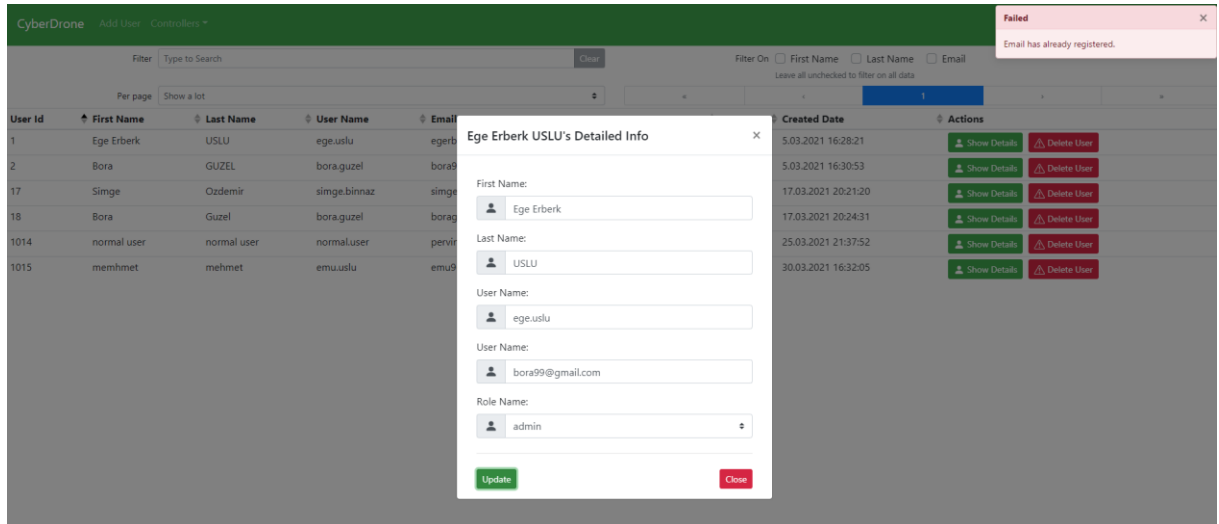


Figure 49:User Details Updating Test

- **Changing Password Test**

New test is changing password test, for sending new password, new passwords must be equal. Web client checks this functionality firstly, after that it sends old and new password to server and server subsystem checks credentials. New test is applied to test that functionality. This functionality passed from that test. **Figure 50** that shown below shows us the success of this function.

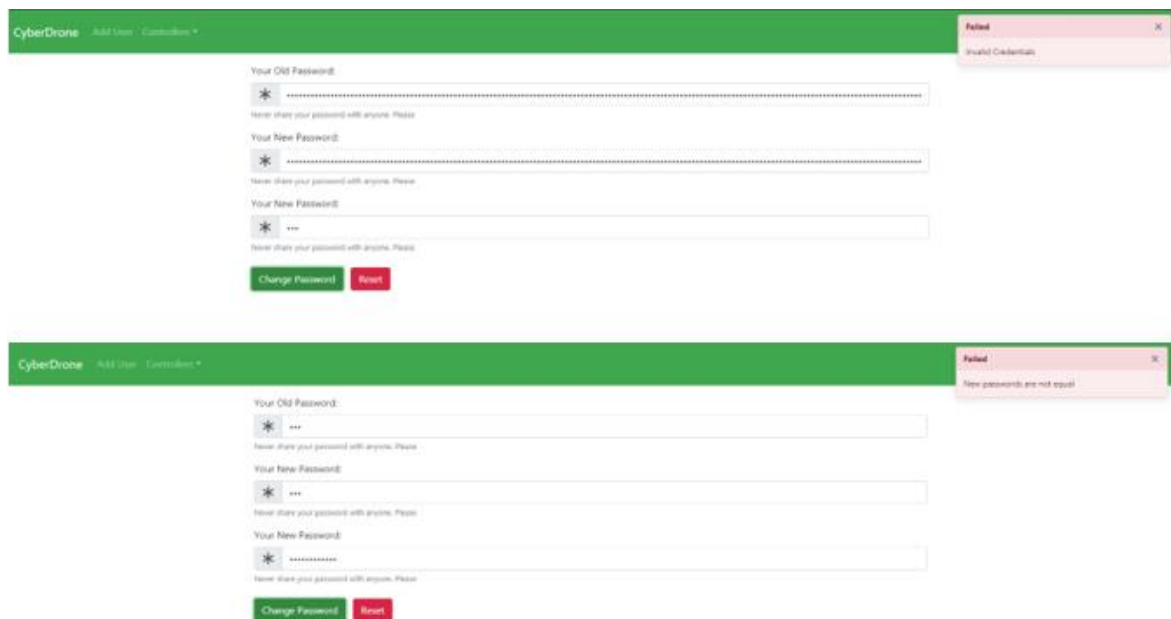
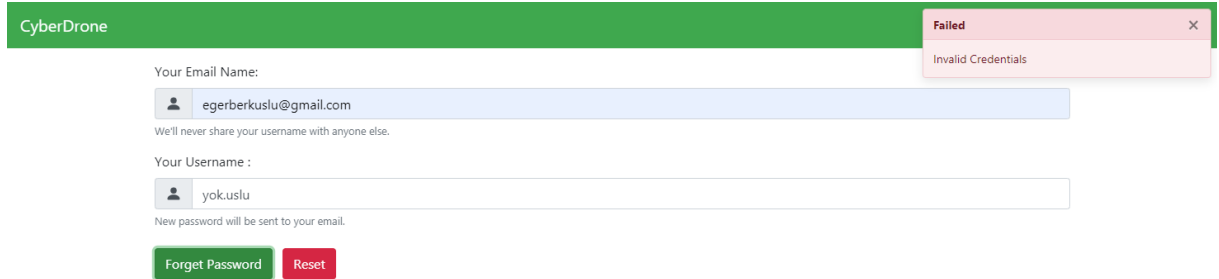


Figure 50:Changing Password Test

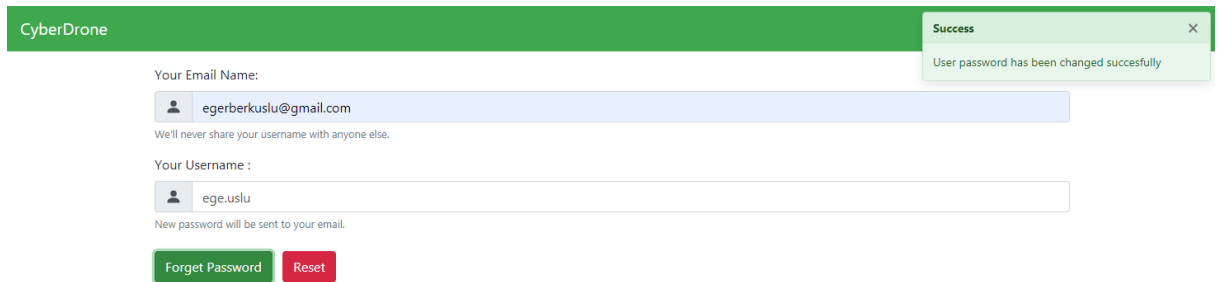
- **Forget Password Test**

New test is forgetting password test, for sending new password, email and username fields must not be empty. Web client checks this functionality firstly, after that it sends email and username to server and server subsystem checks credentials. New test is applied to test that functionality and receiving new password by email. This functionality passed from that test. **Figures 51,52,53** that shown below shows us the success of this function.



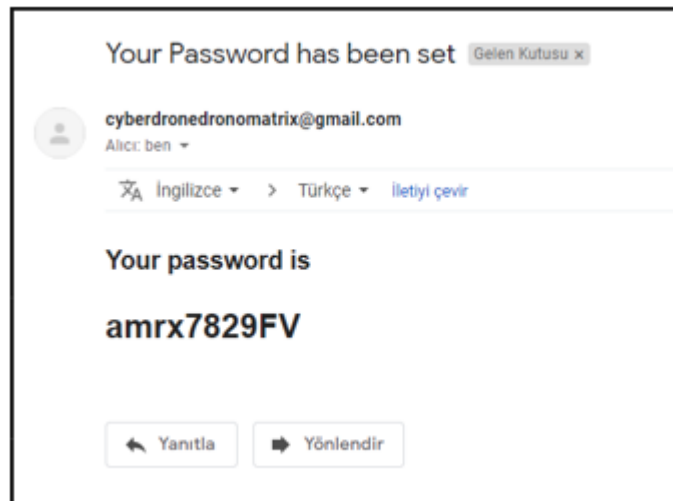
The screenshot shows the CyberDrone web interface. At the top, there is a green header with the text "CyberDrone". On the right side, there is a red notification box with the text "Failed" and "Invalid Credentials". Below the header, there are two input fields. The first field is labeled "Your Email Name:" and contains the text "egerberkuslu@gmail.com". Below this field, there is a small text that says "We'll never share your username with anyone else." The second field is labeled "Your Username :" and contains the text "yok.uslu". Below this field, there is a small text that says "New password will be sent to your email." At the bottom, there are two buttons: "Forget Password" (green) and "Reset" (red).

Figure 51:Forget Password Test



The screenshot shows the CyberDrone web interface. At the top, there is a green header with the text "CyberDrone". On the right side, there is a green notification box with the text "Success" and "User password has been changed succesfully". Below the header, there are two input fields. The first field is labeled "Your Email Name:" and contains the text "egerberkuslu@gmail.com". Below this field, there is a small text that says "We'll never share your username with anyone else." The second field is labeled "Your Username :" and contains the text "ege.uslu". Below this field, there is a small text that says "New password will be sent to your email." At the bottom, there are two buttons: "Forget Password" (green) and "Reset" (red).

Figure 52:Forget password test-2



The screenshot shows an email received from CyberDrone. The email header says "Your Password has been set" with a "Gelen Kutusu x" button. The email body contains the following text: "cyberdronedronomatrix@gmail.com", "Alıcı: ben", "İngilizce > Türkçe İletiyi çevir", "Your password is", "amrx7829FV", and two buttons: "Yanıtla" and "Yönlendir".

Figure 53: Forget password test mail.

- **Online User Test**

New test is online user test. This is applied for showing online users that in system properly. This functionality passed from that test. **Figure 54** that in below shows that two different users are in the system and all users are able to see each other.

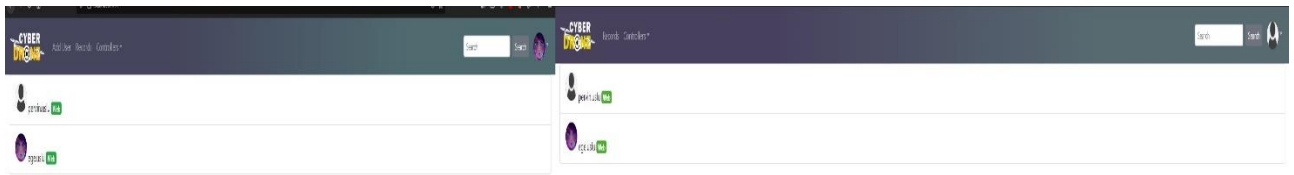


Figure 54:Online User Test-1

The other test is one user is logging out from system. User screen that in system must not show the user that is logged out from system. It means that, logged out user must not be seen in the user's screen that in system. **Figure 55** that in below shows that this functionality.

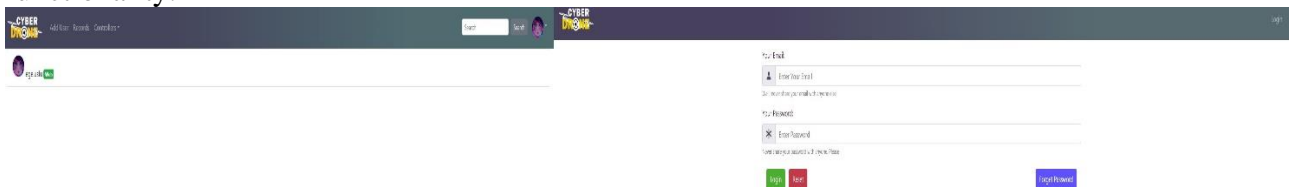


Figure 55:Online User Test-2.

These tests are passed from our examinations.

- **User Update Personal Information Test**

New test is Update Personal Information test. This is applied for updating personal user's information. This functionality passed from that test by giving empty fields to user's information. If field one field is empty, servers give an error message. **Figure 56** that in below that this functionality is passed from test.

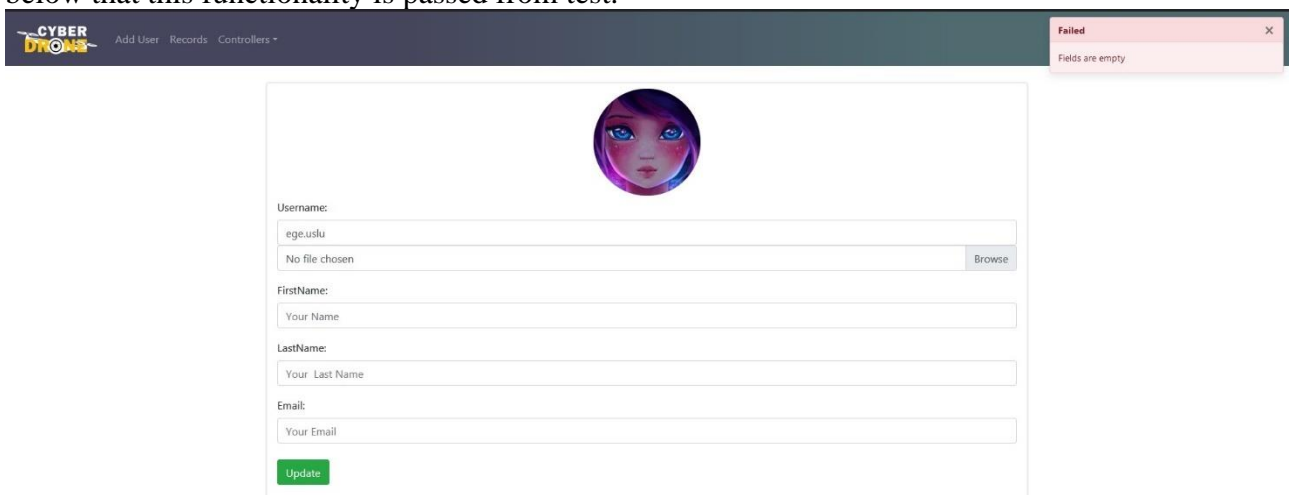


Figure 56:User Update Personal Information Test.

- **Record Download and Watch Test**

New test is Record Download and Watch test. This is applied for download and watching user's records. **Figure 57** that in below that this functionality is passed from test.

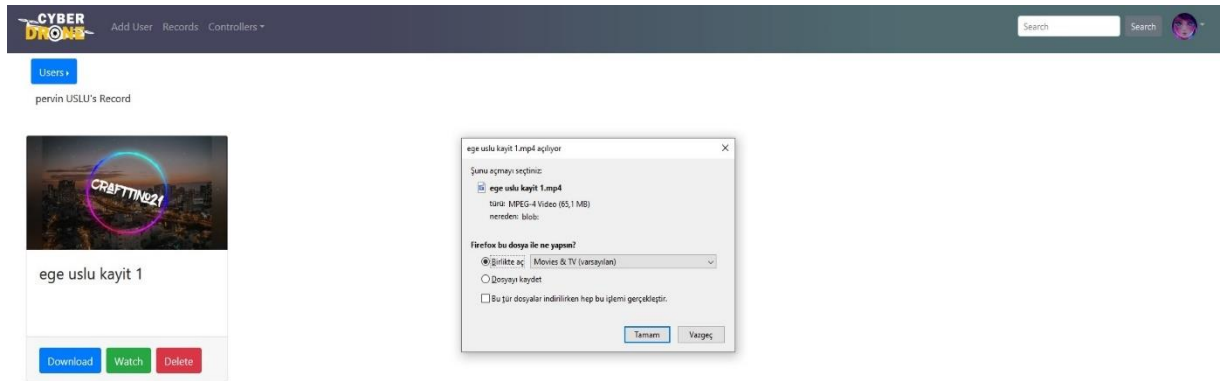


Figure 57:Record Download and Watch Test.

4. Cyber Drone Installation, Configuration and Operation

This section must describe how one installs, configures, and operates your system. **All download links for this project is in the links.txt file.** The operations here cover the requirements described in section 2.2 and 2.3 above. This section will be explained in two part which are for user and admin:

- **User:**

Before using Single/Multiplayer Mode application which CyberDrone.exe and Web Client, you must install and run the web back end, web front end , active user ,database servers.

- **Unity Program-Single/Multiplayer Mode**

- Download the files from given link which is in the links.txt
- Install the Setup.Screen.Capturer.Recorder.v0.12.11.exe that exist in ScreenRecorder folder for screen recording.
- Install CyberDrone.exe.
- Start the CyberDrone.exe
- Enter email and password for login.

- **Web Client**

- Open any web browser such as google chrome and Mozilla Firefox.
- Enter CyberDrone web site to URL. (localhost:8081)
- Press enters.
- Click login field that is in navigation bar.
- Enter your email in email field.
- Enter your password in password field.
- Click login button.

- **Admin:**

- **Web Front-end Part**

- Open terminal.
 - Install Node.js.
 - Download the files from given link which is in the links.txt
 - Write and Enter “cd CyberDroneClientServer”.
 - Write and Enter “npm install” for once.
 - Write “npm run dev” to start the front-end part of the project.
(This command is for running server Runs in localhost:8081)
 - Web Front-end part was installed.

- **Opening Codes in Code Editor**

- Install Visual Studio Code
 - Open Visual Studio Code
 - Click File
 - Click Open Folder
 - Select “CyberDroneClientServer”

- **Web Back-end Server Part**

- Open terminal
 - Download the files from given link which is in the links.txt
 - Write and Enter “cd CyberDroneWebServer”
 - Write “npm install” for once.
 - Write “dotnet run --project CyberDroneWebServer”(This command is for running server Runs in localhost:44302)
 - Web Back-end Server Part was installed and run.

- **Opening Codes in Code Editor**

- Install Visual Studio Code
 - Open Visual Studio Code
 - Click File
 - Click Open Folder
 - Select “CyberDroneWebServer”

- **Database Part**

- Install MSSQL Express Server 2019
 - Install MS SQL Server Management Studio v18.8
 - Open MS SQL server Management Studio
 - Download the files from given link which is in the links.txt
 - Import existing database script file called “CyberDrone” that was provided by system developers.
 - Change the path in line 9 and line 11 in the given SQL script according to where your SQL server is located. (Example: C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\CyberDrone.mdf)
 - Run the given “CyberDrone.sql” file to create database in MS SQL Server Management Studio.
 - First user credentials are email: cyberdronedronomatrix@gmail.com
Password: 123qwe
 - Database was installed.

- **Active User Server Part**

- Download the files from given link which is in the links.txt
- Open terminal
- Write and Enter “cd CyberDroneOnlineUserServer”
- Write and Enter “npm install” for once
- Write and Enter “node index.js” (This command is for running server)
- Active Server was installed and run.

- **Opening Codes in Code Editor**

- Install Visual Studio Code
- Open Visual Studio Code
- Click File
- Click Open Folder
- Select “CyberDroneOnlineUserServer”

- **Simulation Part**

- Download the files from given link which is in the links.txt
- Extract all zip files as a folder.
- There will be four different folder which are called CyberDroneSimulation-DateTime-001, CyberDroneSimulation-DateTime-002, CyberDroneSimulation-DateTime-003, CyberDroneSimulation-DateTime-004.
- Move the files inside the “Assets” and “Library” files in CyberDroneSimulation-DateTime-002/CyberDroneSimulation into the “Assets” and “Library” files inside the CyberDroneSimulation-DateTime-001/CyberDroneSimulation.
- Repeat the same procedure that mentioned above for CyberDroneSimulation-DateTime-003 and CyberDroneSimulation-DateTime-004.
- Install Unity
- Open Unity Hub
- Click “Installs”
- Click “Add” button
- Choose Recommended Release Version
- Click “Next” button
- Check the “Microsoft Visual Studio Community 2019” box
- Click “Done” button.
- Wait for the installment process.
- Click “Projects”
- Click “Add” button
- Choose CyberDrone file
- Choose CyberDrone at the Project list

References

- [1] "Unity Game Engine," Wikipedia, 2021 March 23. [Online]. Available: [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)). [Accessed 30 March 2021].
- [2] Guru99, "SQL Server Management Studio: MS SSMS Download & Install," [Online]. Available: <https://www.guru99.com/sql-server-management-studio.html>. [Accessed 30 March 2021].
- [3] BlazeMeter, "How to use Postman for API Testing Automation," BlazeMeter, [Online]. Available: <https://www.blazemeter.com/blog/how-use-postman-manage-and-execute-your-apis>. [Accessed 30 March 2021].
- [4] Unity, "Visual Studio C# integration," Unity, [Online]. Available: <https://docs.unity3d.com/Manual/VisualStudioIntegration.html>. [Accessed 30 March 2021].
- [5] VisualStudio, "Using Vue in Visual Studio Code," VisualStudio Code, [Online]. Available: <https://code.visualstudio.com/docs/nodejs/vuejs-tutorial>. [Accessed 30 March 2021].
- [6] Wikipedia, "Socket.IO," Wikipedia, 12 March 2021. [Online]. Available: <https://en.wikipedia.org/wiki/Socket.IO>. [Accessed 14 May 2021].
- [7] Wikipedia, "FFmpeg," Wikipedia, 13 May 2021. [Online]. Available: <https://en.wikipedia.org/wiki/FFmpeg>. [Accessed 14 May 2021].
- [8] P. U. Networking, "Photon Unity Networking 2," Photon Unity Networking 2, [Online]. Available: <https://doc-api.photonengine.com/en/PUN/v2/index.html#:~:text=Photon%20consists%20of%20a%20server,features%20provide%20a%20great%20start..> [Accessed 14 May 2021].
- [9] "System requirements for Unity 2019.4," Unity, [Online]. Available: <https://docs.unity3d.com/2019.4/Documentation/Manual/system-requirements.html>. [Accessed 30 March 2021].