

**VALIDATION OF USER CREDENTIALS IN SOCIAL NETWORK BY USING
DJANGO BACKEND AUTHENTICATION**

NUR IZZATI SHOLEHAH BINTI AZLAN

Bachelor of Computer Science (Computer Network Security) with Honours

Faculty of Information and Computing

Universiti Sultan Zainal Abidin , Terenggannu, Malaysia

May 2019

DECLARATION

I hereby declare that this report is based on my original work except for quotations and citations, which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at University Sultan Zainal Abidin or other institutions.

Name: NUR IZZATI SHOLEHAH BINTI AZLAN

Date:

CONFIRMATION

This is to confirm that:

The research conducted and the writing of this report was under my supervision.

Name: DR. AHMAD FAISAL AMRI BIN ABIDIN @ BHARUN

Date:

DEDICATION

Praise to Allah, the Most Gracious and the Most Merciful. Alhamdulillah, for blessing me and giving me the opportunity to undergo and complete this final year project entitled Validation of user credentials in social network by using Django backend authentication.

Foremost, my greatest appreciation to my loving family especially my beloved mother, PM Dr Zarina binti Mohamad and father, Azlan bin Jahan that always giving me fully support as well encouragement throughout this project.

Here, I would like to take this opportunity to express my heartiest appreciation and gratitude to my versatile supervisor, Mr. Ahmad Faisal Amri Bin Abidin @ Bharun for the encouragement, guidance, critics, motivation and also support. Without his continuing support, this project would not have been the same as presented here. Thank you.

In addition, I would like to thank the panels during my final year project, Dr. Wan Nor Syuhadah Binti Wan Nik and Dr. Nor Aida Binti Mahiddin for all supportive words, guidance, advice as well as suggestion in order to make my project a better one. Thank you.

To Faculty of Informatics and Computing, and all lecturers in this faculty, I also would like to thank you for the chance to expose and explore as a degree student with this project. Last but not least, I would take this chance to thank you to my fellow friends for everything.

ABSTRACT

Validating user credentials are important in social network to increase security because it contains the privacy of user's information. There are many techniques to validate user credentials and, in this project, the technique of backend Authentication is utilised. In addition, this project is leveraged Django framework due to it comes with a user authentication system which consist of backend authentication. Django is a server-side web framework that encourages rapid development of secure and maintainable websites. Django is written in Python. Django also include a robust user authentication and permission system that has been built with security in mind.

ABSTRAK

Pengesahan kelayakan pengguna penting di laman sosial seperti Facebook dan Twitter untuk meningkatkan keselamatan kerana kebanyakan laman sosial menyimpan maklumat pengguna yang sensitif . Terdapat banyak teknik untuk mengesahkan kelayakan pengguna dan dalam projek ini, teknik '*backend authentication*' digunakan. Di samping itu, projek ini memanfaatkan Django sepenuhnya kerana rangka kerja Django ini menyediakan system pengesahan pengguna yang terdiri daripada '*backend authentication*' . Django adalah rangka yang memetingkan keselamatan. dan ditulis dalam Python. Django juga termasuk dalam system pengesahan yang teguh dan sistem kebenaran yang sentiasa menjadikan keselamatan sebagai teratas.

CONTENTS

	PAGE
DECLARATION	i
CONFIRMATION	ii
DEDICATION	iii
ABSTRACT	iv
ABSTRAK	v
CONTENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATION	xiii

CHAPTER I

INTRODUCTION

1.1	Project Background	1
1.2	Problem Statement	3
1.3	Objectives	3
1.4	Scopes of Project	4
1.5	Expected Result	4
1.6	Report structure	5

CHAPTER II	LITERATURE REVIEW	
2.1	Introduction	6
2.2	Backend Authentication	7
2.3	Backend Authentication in social networks	8
2.4	Django and Django Rest Framework	9
2.5	Summary	13
CHAPTER III	METHODOLOGY	
3.1	Introduction	14
3.2	Analysis study of Process Model	15
3.3	Data Model	18
3.4	Proof of Concept	19
	3.4.1 Install Python and Django	19
	3.4.2 Configure Django	21
	3.4.3 Configure Django Rest Framework	23
3.5	Summary	25
CHAPTER IV	IMPLEMENTATION AND RESULTS	
4.1	Introduction	26
4.2	Project Implementation	27

	4.2.1 Setting up virtual environment	27
	4.2.2 Download framework	28
	4.2.3 Create API and configurations	29
	4.2.4 Permissions	35
	4.2.5 Implementing the Token	39
	4.2.6 Connect framework to phpMyAdmin	42
	4.2.7 Matching hashing	45
	4.2.8 Try CSRF attack on website	49
4.3	Summary	50
CHAPTER V	CONCLUSION	
5.1	Introduction	51
5.2	Project Contribution	52
5.3	Project constraints and limitations	52
5.4	Future Works	53
5.5	Summary	54
REFERENCES		55
APPENDIX		59

LIST OF TABLES

TABLE	TITLE	PAGE
2.1	Concern of each Authentication Technique	7

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	Flow user to social network	7
2.2	Example of API	10
2.3	Flow Token Based Authentication	11
3.1	Process model	15
3.2	Data Model	18
3.3	POC Install python, PIP and virtual virtualenv	19
3.4	Setting file in Django project	21
3.5	PyCharm using Django Framework	22
3.6	Command to run the server	22
3.7	Django framework successfully install	22
3.8	Install Django Rest Framework	23
3.9	Optional packages DRF	23
3.10	Create super user	24
3.11	Django Rest Framework	24
4.1	python and pip version	27

4.2	Path of the project	27
4.3	Activate the python virtual environment	28
4.4	Django and Django Rest Framework version	29
4.5	List of INSTALLED_APPS	30
4.6	Models.py code	31
4.7	Models.py code	31
4.8	Serializer.py code	32
4.9	views.py code	33
4.10	views.py code	34
4.11	snippet/urls.py code	34
4.12	create super user	35
4.13	permission is authenticated	35
4.14	urls.py code	36
4.15	Permission code who can edit and delete	36
4.16	wrong username and password	37
4.17	Right credential	38
4.18	user input with correct credential	38
4.19	Add rest_framework.authtoken	39
4.20	Add to rest framework token and session	40

4.21	Testing without token	40
4.22	Token	41
4.23	Testing with Token	42
4.24	List of Django default with db.sqlite3 database	43
4.25	Change database to mysql	44
4.26	Start all port to connect with server	44
4.27	Testing mysql python connector	45
4.28	init.py_ code	45
4.29	Hashing password from database	46
4.30	Login interface to input credentials	46
4.31	Main interface when user can logi	47
4.32	PHP code login from dummy website	47
4.33	PHP code to matching hashing password	48
4.34	PHP code to matching hashing password	48
4.35	PHP code token function	49
4.36	Using RawCap to sniff packets	50
4.37	Packets that got from localhost	50
4.38	Data that sniff from website	50

LIST OF ABBREVIATIONS/TERMS/SYMBOLS

TBA	Token Based Authentication
SBA	Session Based Authentication
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
CSRF	Cross-site Request Forgery
SQL	Structured Query Language
API	Application Programming Interface
POC	Proof of concept
PIP	Package management system
DRF	Django Rest Framework

CHAPTER 1

INTRODUCTION

1.1 Background

Social network is quickly becoming one of the top tools of choice for everyone. Since the advent of social networking sites such as Facebook and Twitter, it's allowed us to connect with peoples around the world. Social networking becomes a medium for people to keep in touch with acquaintances and share their thought, feeling and personal information. However, as much as people connected to the social network, there was also a tremendous opportunity for them to get exposed to danger. A login page is the biggest opportunity for hackers. The login page can provide too much information, however it will be the first target by hackers for their research due to the weak authentication.

Authentication is the process or action of identifying genuine credentials. However, in social network it is important for users to trust established social networks such as Facebook and Twitter to protect their personal data. Authentication was accomplished by the social network that being accessed. The server will validate users by implemented username and password, credentials of the login page will recognize

that the user is authenticated. There are many techniques to validate user credentials and for this project, the technique of backend authentication is utilized.

In addition, this project is leveraged Django framework due to it comes with a user authentication system which consists of backend authentication. Django is a server-side web framework that encourages rapid development of secure and maintainable websites. Django is written in Python. This framework also includes robust user authentication and permission system that has been built with security in mind.

Besides, authentication backends provide extensible systems, when a username and password stored with the user model need to be authenticated against a different service than Django's defaults. It allowed the ability to change the method to check user credentials. Django has a list of "authentication backends" to check for authentication. Django tried authenticating across all of its authentication backends. If the first authentication method fails, Django will try the second one, and so on, until all the backends have been attempted.

Envisaged, in this project the user credentials will secure by using authentication backend, the order of authentication backends, matters so if the same username and password is valid in multiple backends, Django will stop a process in the first try. It will make the credential more secure because it performed additional check in backend authentication

In conclusion, Django is known to many developers as a web framework, but it also used to build a backend, which in this case is an Application programming.

1.2 Problem Statement

Every time a user is authenticated, the server will need to create a record somewhere on our server. This is usually done in memory and when there are many users authenticating, the overhead on our server increases. Username and password were not secure because cookies can be accessed by the server by parsing the Cookie HTTP request header.

Need to hook into another authentication source, source of usernames and password or authentication methods.

1.3 Objective

1. To study validation of user credentials in social network on backend authentication.
2. To configure a Django framework platform for authentication backend server.
3. To integrate token based authentication technique as integral part of Django based authentication backend server.
4. To test Django framework backend authentication in social network.

1.4 Scope

1. Be able to validate the user credentials in social network by using backend authentication.
2. Configuring a Django framework platform for authentication backend server.
3. Modifying token based authentication to be embedded into backend Django.
4. To test functionality of Django based authentication in social network setting.

1.5 Expected Result

1. To improve security of user credentials in social network.
2. Django allow to build applications that share permissions with another.

1.6 Project Structure

This thesis consists of five chapters. The first chapter focuses on the project background, problem statements, objectives, scopes of project and. Chapter two is a review of all related study regarding to this project. In chapter three, the methodology will be discussed including the methods and techniques used for the developing this application. In this chapter also will discuss the expected results through the data model and design. In chapter four, will be explained about implementation and testing process that has been run in this project. The last chapter is conclusion and results are presented.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

This chapter discusses about literature reviews related to this project. There was some research analysis that have been done which consists of backend authentication and authentication techniques in order to secure the user credentials over social networks. The ubiquitous adoption of the social web has become an attractive to many people nowadays used social network anytime and anywhere. At the end of this chapter will have summary and the main point highlighted from each of the articles regarding the concern of the technique that been used in this project which is Token based Authentication and framework used to build backend authentication which is Django.

2.2 Backend Authentication

Backend authentication is an important term when talking about authentication. Authentication is accomplished when the user provides a credential. Focus on backend or server-side is referring to everything that user cannot see in the browser. Most backend developers worried about were things like structure, content management and the most important is security. A backend authentication server is an entity that provides an authentication service to an authenticator. The simple way backend authentication is only server-to-server communication where a user is not directly interacted to a database. Backend authentication helps prevent that traffic from being hijacked.

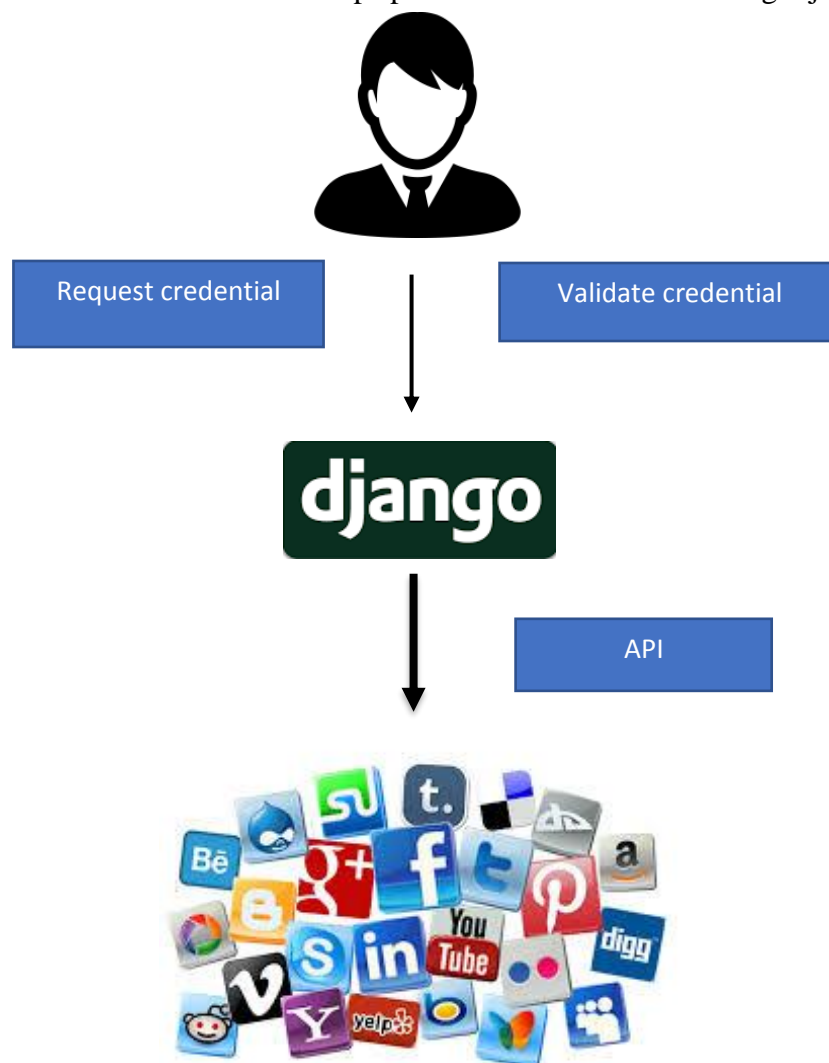


Figure 2.1: User are not directly with social network but into Django backend first.

2.3 Backend Authentication in social networks

Many researches have been conducted into validate user credential in social networks using a variety of techniques and models. This includes techniques such as session-based authentication and photo-based authentication, and summary is shown in Table 1. However, only two models, Session-based authentication and Token-based authentication can take into consideration. TBA are a powerful technique when it comes to backend authenticate user. It also can work with third party using Django Framework. Tokens are the best way to handle authentication for multiple users.

TABLE 1. MODELS OF AUTHENTICATION TO VALIDATE USER CREDENTIALS

Techniques	Remarks	Concern
Server based authentication	Store the user logged in information on the server.	Susceptible to CSRF attacks.
One-time based authentication	Provide mechanism for login to a network using a unique password which can only be used once.	Easily accessible to outsiders if not protected accurately.
Biometrics based authentication	Protect account with two factors.	Need additional device.

Knowledge based authentication	Ask the question from information.	Hard to remember based on personal history.
Photo based authentication	Photo present to the user must include a group of unique people known to identify human subject in photo.	Trust between individuals.
Session based authentication	Server will create a session for the user after the user logs in.	Session is stored in the cookie on the user's browser.
Token based authentication	Signed token that is sent to the server on each request	Best way to handle authentication for multiple users.

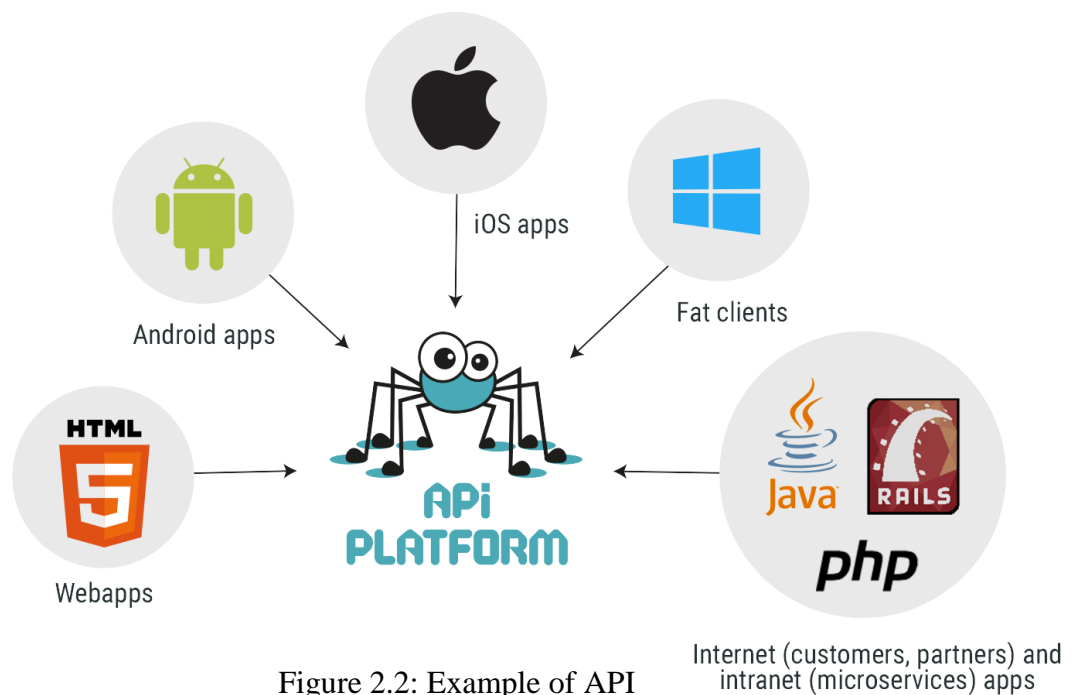
2.4 Django and Django Rest Framework

As this project leveraged Django framework because it comes with a user authentication system which consists of backend authentication. Django can take care as much as a problem of web development. Django framework using Python programming language due to its easiness to learn, clarity, versatile and faster to write. Python also comes with a package of debugging tools to handle issues.

Focus on Django, it is about high-quality code and transparent writing provides. Django is a good platform because it has been designed in a way to help writing coding

as fast as possible. Django framework making it is effective and efficient to become an ideal solution for having a primary focus on deadlines. Next, it works in a way that includes extras to help with user authentication. Django frameworks are ensured to not commit any mistakes related to security. Most of the front-page common mistakes in SQL injection, cross-site request forgery, clickjacking, and cross-site scripting. It is important for social networks to be careful of attackers because hackers are always craving for private information. To get the effectively secure username and password, the user authentication system that Django provides is the main key.

When using social networks, not only one people log in. Django framework can meet the heaviest demand. Therefore, the busiest sites use this medium to quickly meet the traffic demands. It is also versatile, the content management and all aspects are very efficiently managing by the use of Django.



When it comes to backend authentication, Django also provides Django rest framework to build web Application programming interface which is a software intermediary that allows two application to talk to each other. This project can connect into social networks using an API. Authentication section of the Django rest framework can work with a lot of techniques but the best one is Token based authentication.

Token based authentication is stateless, there is no record or track of interaction. Most of the application remember or keep track of what we are doing, using token TBA there is no record not just only about information but all configuration setting means there is no user's information on a server.

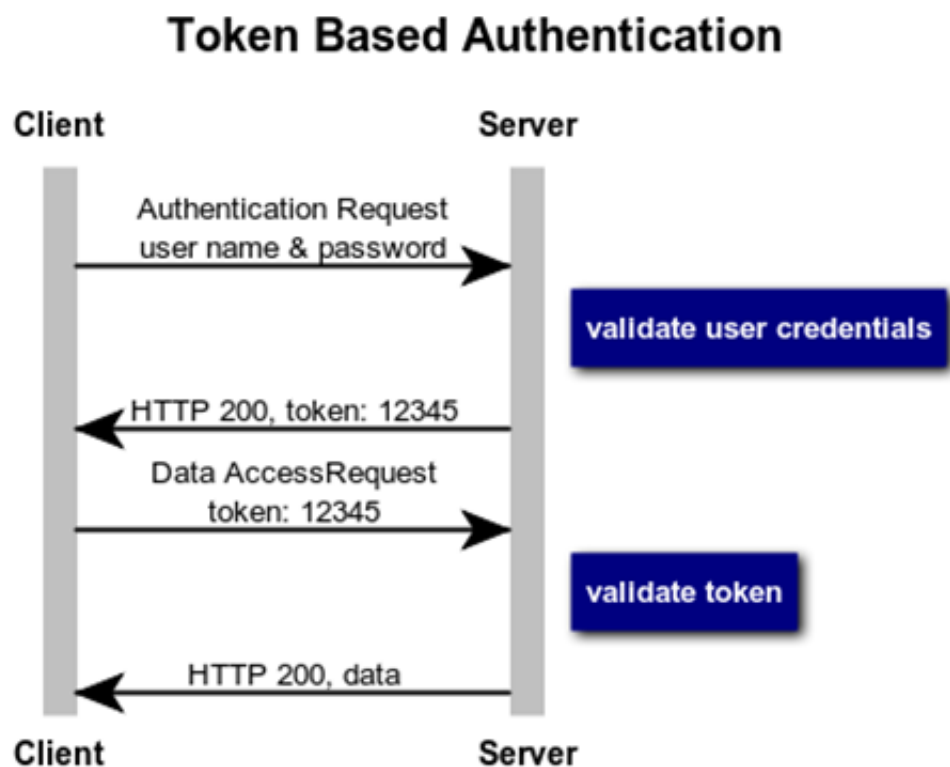


Figure 2.3 Flow of Token based authentication

There is no issue with scaling because a token is stored on the client side. It is completely stateless since there is no state or session information anywhere our load balancers are able to pass a user along to any of our servers. TBA did not have session affinity when the user logged in to the same server, the client required the server to keep sending the same session and cause heavy traffic.

Token based authentication is a perfect way to secure a website's login process. TBA is typically used on the web because it allows users to stay logged in to social networks without the use of cookies. In addition, token is more secure because they can be used to replace a user's actual credentials. A token that been sent on every request is a way that helps to prevent CSRF attacks because no cookie being sent to the server. Meanwhile, for the implementation, a token is stored within a cookie on the client side, the cookie is just a storage mechanism substitute of authentication one. After the prescribed time (can set amount of time) the token will be expired and required the user to login once again. It is best to stay secure.

For this project, the important part is token allow us to build an application that share permissions with others. Validate user credentials in social networks using Django, this project connecting social network as a third-party application with Django.

2.5 Summary

The various sources of research related to backend authentication and some of techniques just can be used for front end authentication. The various approach and techniques used helps in generate a better of project research in future. Literature review is done to ensure no identical is done.

CHAPTER III

METHODOLOGY

3.1 Introduction

This chapter reports the approach or model development and application of a comprehensive framework taken in the development of system, application or implementation of the study. This chapter contains methods, technique or approach that will use during the design and implementation of the project. The selection of the most suitable as suitable methodology for methodology is chaotic enough because the project might not complete on the right schedule or the project might completely fail because the developer might be lost guidance in order to complete the project development. All the phases that involved during this project will detailed.

3.2 Framework

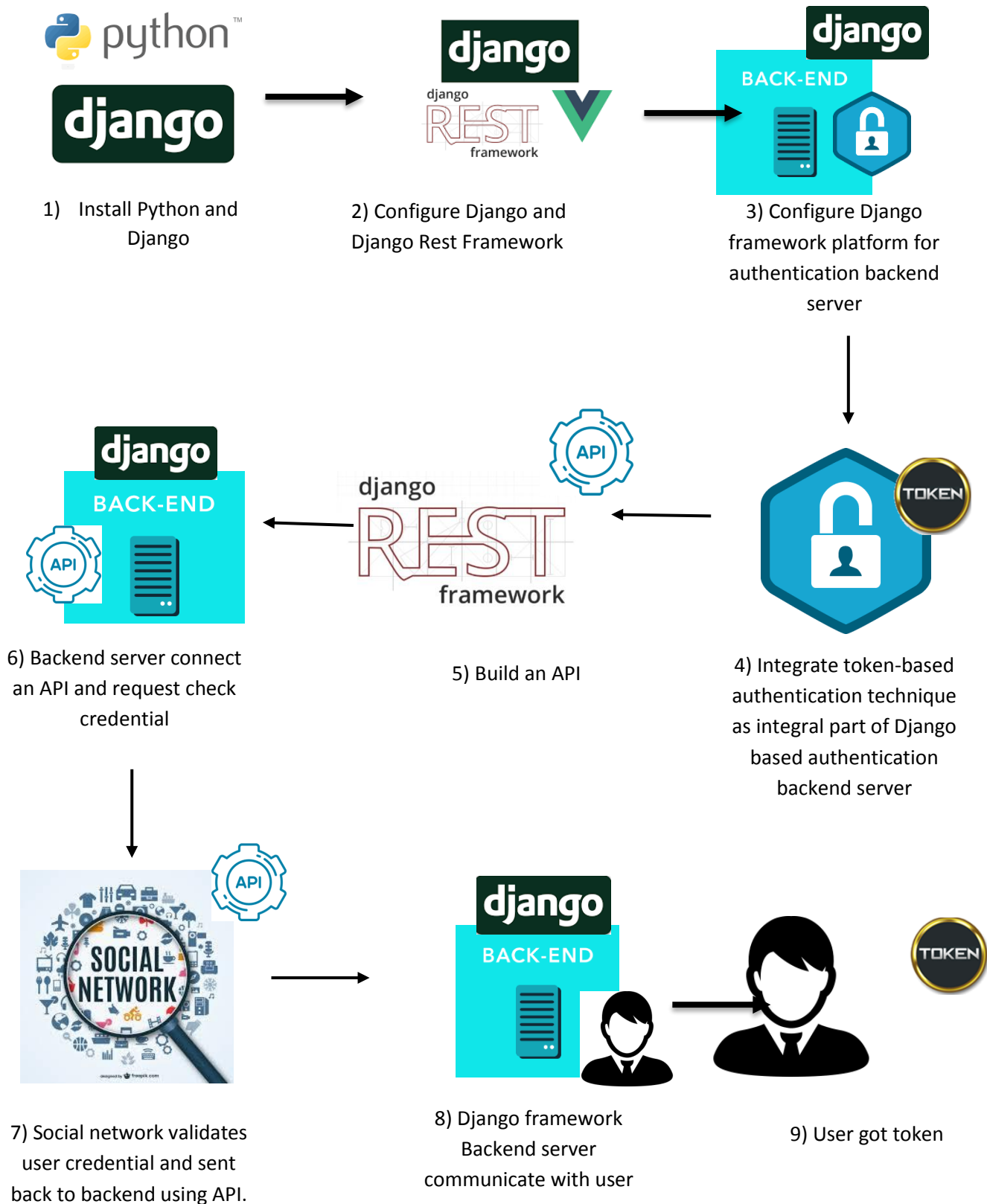


Figure 3.1: The process model validation of user credentials in social network using Django backend authentication: Token based authentication

Figure 4.0 shows an overall process model of validation of user credentials in a social network by using Django backend Authentication. The first step is to install python and Django framework. After installing Django framework install Django Rest framework and configure Django Framework platform for authenticating backend server. Authentication in Django is good enough for most cases but may need not met by the out-of-the-box defaults. These project cases are secure authentication in a social network, authentication backend provides an extensible system when username and password stored with the user model, and need to be authenticated against a different that Django's default.

After installing and configure setting all the application and framework. The next step is making the front-page using Django for user login their username and password. This project using Django as a front page because of it ridiculously fast. Django was designed to help developers take applications from concept to completion as fast as possible. The growing number of people interested in social network needs flexible scale. This project is leverage Django be able to come with the scalable project.

The next flow project the validation of user credentials in social network by using Django backend authentication is configured Django framework platform for authentication backend server. By using `django.contrib.auth.authenticate()`, Django try all the authentication backend one by one on the list of backend authentication. Django tries to attempts until all backends have been attempted. All the list of authentications backends is specified in the `AUTHENTICATION_BACKENDS`. By default, `AUTHENTICATION_BACKEND` is set to:

`'DJANGO.CONTRIB.AUTH.BACKENDS.MODELBACKEND`

The order of `AUTHENTICATION_BACKENDS` matters, so if the same username and password are valid in multiple backends, Django will stop processing at the first positive match. When the user has authenticated, Django will store which backend was using for authenticated before in user's session. Django will re-use the same backend list when the user needed to be authenticated. If a backend raises a `PermissionDenied` exception, authentication will immediately fail.

Next, integrated token-based authentication technique as an integral part of Django backend authentication backend server. However, to get more extraordinary by using token based authentication, this project will integrate token-based authentication technique as an integral part of Django based authentication backend server. TBA is stateless, it's not storing any information about a user on the server or in the session. Since no information is stored, how does the server verify the token against its database of authorized users is, each token is signed with an encrypted key code or digital signature that only the server will know. The server verifies a token by running the algorithm that created the signature. If the two matches, the server knows that the token is valid.

The basic how the token works. User request access with username and password. Applications validate credentials and application provide a signed token to the client. Client stored that token and sends it along every request. Lastly, the server verifies token and responds with data. As well as, build an API using Django rest framework. Then, a backend server connects with API and check credential. Social network validates user credentials and sent back to the backend using an API. Django framework backend server communicates with a user and user got the token.

3.3 Flowchart

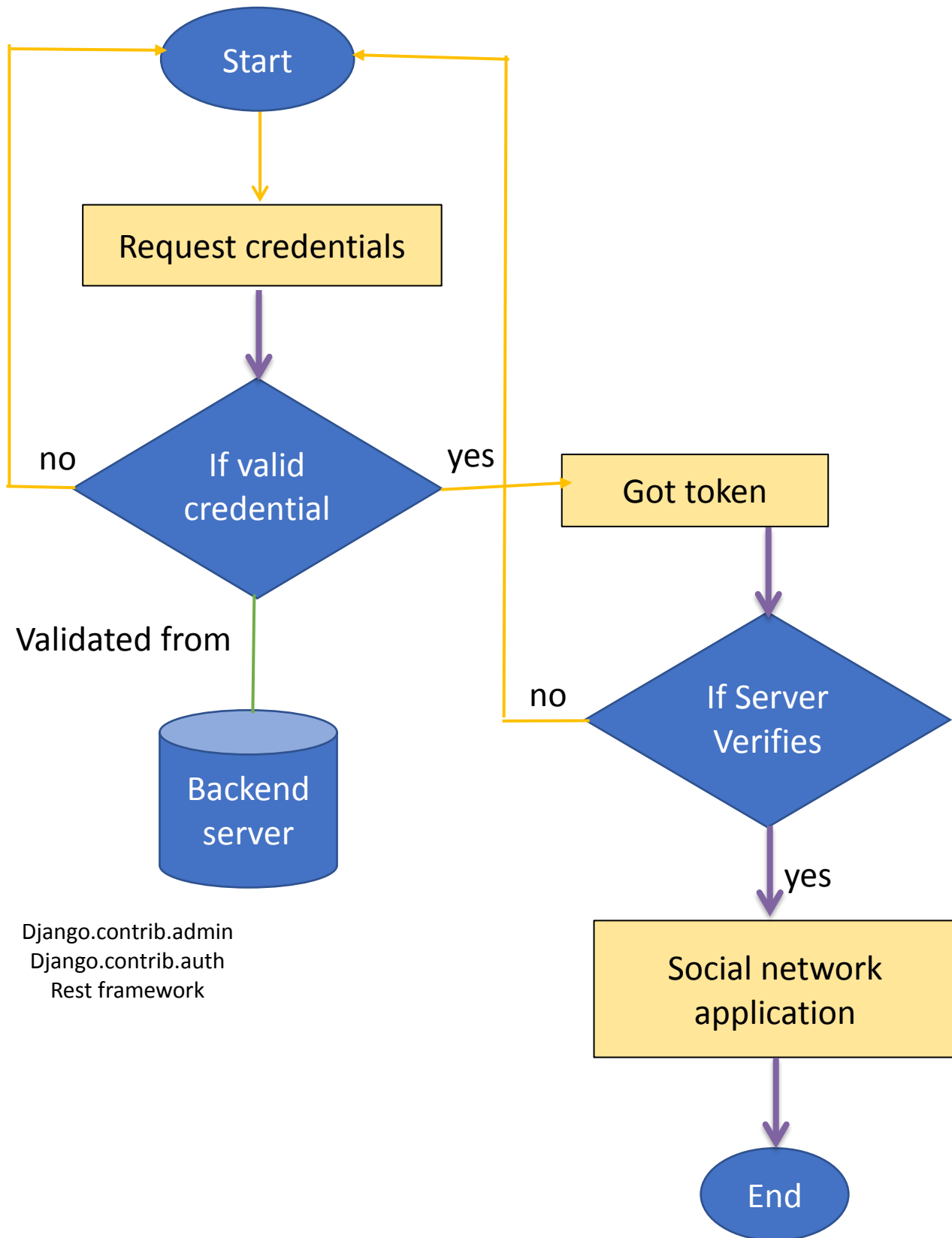


Figure 3.2: Data model validation of user credentials in social network by using Django backend authentication

Figure 5.0 shows the data model in the flowchart. This project started with requests access with username and password and server will check the valid credential or not. If no user has to login again. The next process after backend check with the social network the credential user was valid, the application provided a signed token to the client. Client stores that token and send it along with every request and if server verifies token and it will respond with data.

3.4 Proof of concept

3.4.1 Install Python and Django

Python is an open source development project and has an active supporting community of contributors. This allows python users to share and collaborate effectively. Benefiting from the solution others have already created to common problems, as well as potentially contributing their own solutions to the common pool. It is recommended by the Django software Foundation to use Python 3 because everything is updated.

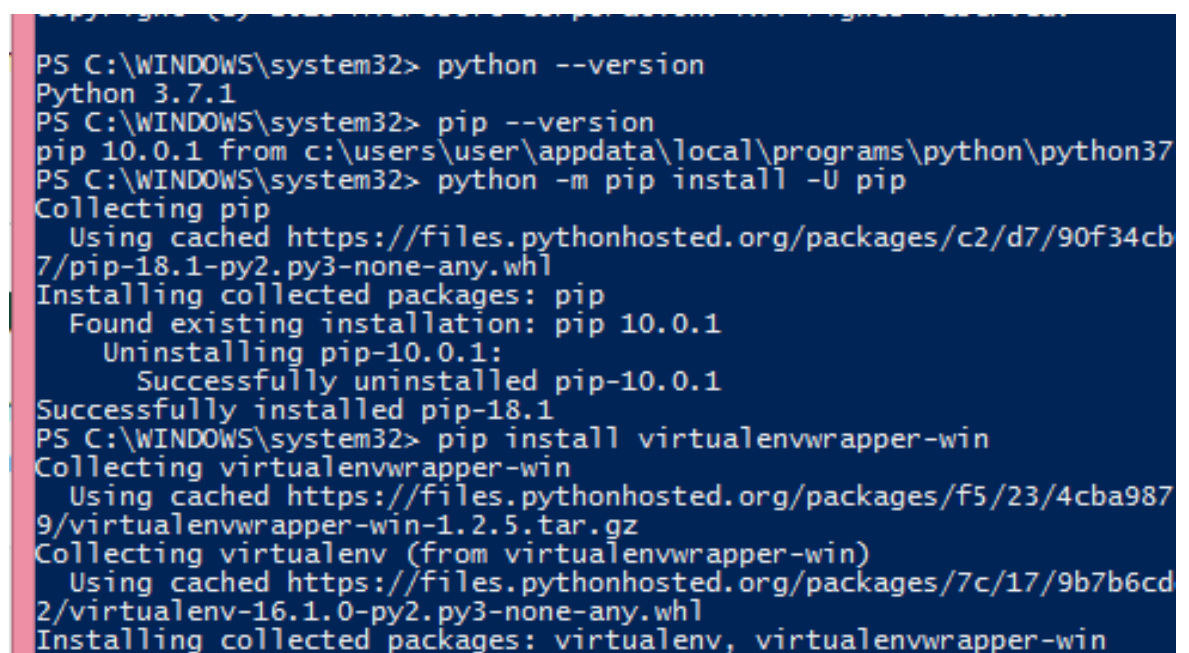
A screenshot of a Windows command prompt window with a dark blue background and white text. The prompt shows the installation of Python, PIP, and virtualenv. The commands and their outputs are as follows:
PS C:\WINDOWS\system32> python --version
Python 3.7.1
PS C:\WINDOWS\system32> pip --version
pip 10.0.1 from c:\users\user\appdata\local\programs\python\python37
PS C:\WINDOWS\system32> python -m pip install -U pip
Collecting pip
Using cached https://files.pythonhosted.org/packages/c2/d7/90f34cb7/pip-18.1-py2.py3-none-any.whl
Installing collected packages: pip
Found existing installation: pip 10.0.1
Uninstalling pip-10.0.1:
Successfully uninstalled pip-10.0.1
Successfully installed pip-18.1
PS C:\WINDOWS\system32> pip install virtualenvwrapper-win
Collecting virtualenvwrapper-win
Using cached https://files.pythonhosted.org/packages/f5/23/4cba9879/virtualenvwrapper-win-1.2.5.tar.gz
Collecting virtualenv (from virtualenvwrapper-win)
Using cached https://files.pythonhosted.org/packages/7c/17/9b7b6cd2/virtualenv-16.1.0-py2.py3-none-any.whl
Installing collected packages: virtualenv, virtualenvwrapper-win

Figure 3.3: POC Install python, PIP and virtual virtualenv

Pip (package management system) preferred to be installed before install python because it manages software package written in python. Furthermore, virtualenv is a virtual environment where you can install software and python packages in a contained development space, which isolates the installed software and packages from the rest of your machine's global environment. This convenient isolation prevents conflicting packages or software from interacting with each other. Virtualenv is a third-party alternative (and predecessor) to venv. It allows virtual environments to be used on versions of python prior to 3.4, which either don't provide venv at all or aren't able to automatically install pip into created environments.

This project will use windows operating system. Django framework can be installed in windows by using pip within virtual environment. To install Django framework can execute the download Django command use Windows PowerShell. PowerShell is the framework developed by Microsoft for administration tasks such as configuration management and automation of repetitive jobs. This is an important step that to start the project.

Software Requirements of this project are:

- i. Django
- ii. Django rest framework
- iii. Windows PowerShell
- iv. Microsoft office
- v. Microsoft office power point
- vi. Windows operating system
- vii. Social network

Hardware requirements of this project are:

- i. Laptop (Lenovo, 4.00 GB RAM, Intel Core i-55000 CPU @ 2.40GHz 2.40 GHz, 66- bit operating system, 64-based processor
- ii. Mouse
- iii. Printer

3.2.2 Configure Django

A Django settings file constains all the configuration of all Django installation.

The basics settings file is just Python module-level variables.

```
ALLOWED_HOSTS = ['www.example.com']  
DEBUG = False  
DEFAULT_FROM_EMAIL = 'webmaster@example.com'
```

Figure 3.4: Setting file in Django project

If DEBUG to FALSE, it needs a properly set the ALLOWED_HOSTS setting. This is because the setting file is Python module. It doesn't allow Python syntax error.

Django framework can design the setting you're using. It can do this by using environment variable DJANGO_SETTINGS_MODULE. This value of DJANGO_SETTINGS_MODULE should be in python path syntax. When using Django -admin in windows. It can be set the environment variable once.

Django security setting file contains sensitive information such as personal data or password. For every attempt, it should limit access to the file by change file permission.

After installing and use PyCharm for configure the Django. Use command `PYTHON MANAGE.PY RUNSERVER` the file that create.

Example project: myclub_site

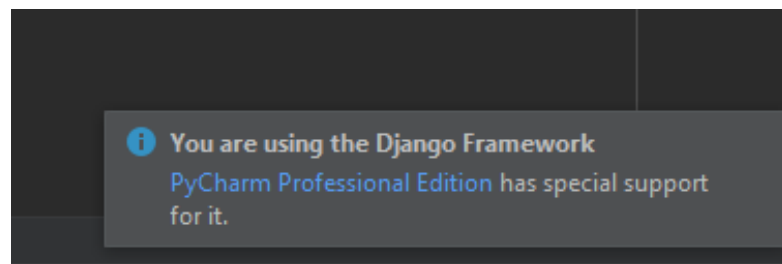


Figure 3.5: PyCharm using Django Framework

```
PS C:\users\user\desktop\myclub_site> python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work proper
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 10, 2018 - 02:06:26
Django version 2.1.4, using settings 'myclub_site.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 3.6 Command to run the server

3.2.3 Configure Django rest framework

This project also needs Django rest framework, it's a flexible toolkit for building Web APIs. Rest framework requires Python (2.7 and above and for this project using Python 3.7 version.) and Django framework. This project using Django Rest Framework for build API.

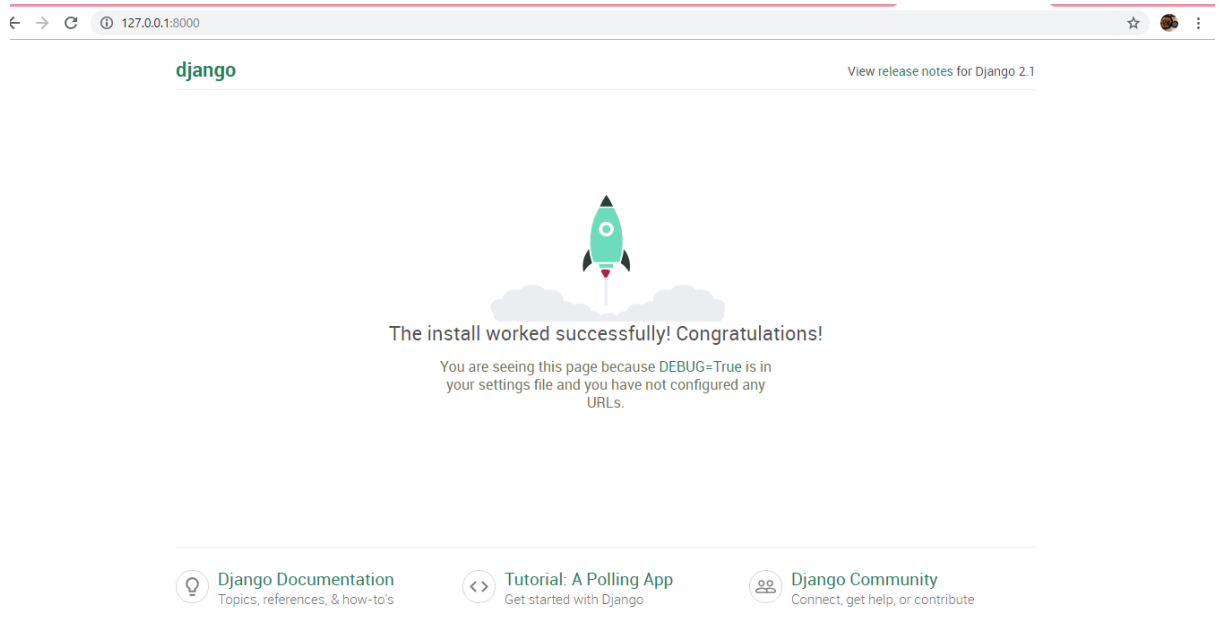


Figure 3.7: Django framework successfully install

There are packages optional to comes with this rest framework such as coreapi (1.322.0+), markdown (2.1 0+), Django-filter (1.0.1+), Django-crispy-forms, Django-guardian (1.1.1+). Installing Rest framework using pip and packages for this project

```
PS C:\WINDOWS\system32> pip install djangorestframework
Collecting djangorestframework
  Downloading https://files.pythonhosted.org/packages/99/0b/d37a5a96c5d30
/djangorestframework-3.9.0-py2.py3-none-any.whl (924kB)
    100% |#####| 931kB 33kB/s
Installing collected packages: djangorestframework
Successfully installed djangorestframework-3.9.0
PS C:\WINDOWS\system32>
```

Figure 3.8: Install DRF

markdown and Django-filter. Markdown support for the browsable API and Django for filtering support.

```
pip install djangorestframework
pip install markdown          # Markdown support for the browsable API.
pip install django-filter     # Filtering support
```

Figure 3.9: Optional packages DRF

Then, create a dedicated migration file and migrate change to the project database. `PYTHON MANAGE.PY MAKEMIGRATIONS` to create a file and `PYTHON MANAGE.PY MIGRATE` the file to launch.

For the next step proof of concept. This project needs to create the super user to login as admin.

```
PS C:\users\user\desktop\izzatiapi> python manage.py createsuperuser
Username (leave blank to use 'user'): izzati
Email address: 043772@putra.unisza.edu.my
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
PS C:\users\user\desktop\izzatiapi>
```

Figure 3.10 Create super user

Add the 'rest_framework' in `INSTALLED_APPS` settings. We need our own serializer and view to display all existing users. Create a new file in PyCharm `users/serializers.py`. After all the configuration is done, login Django as admin and gets into Django rest framework.

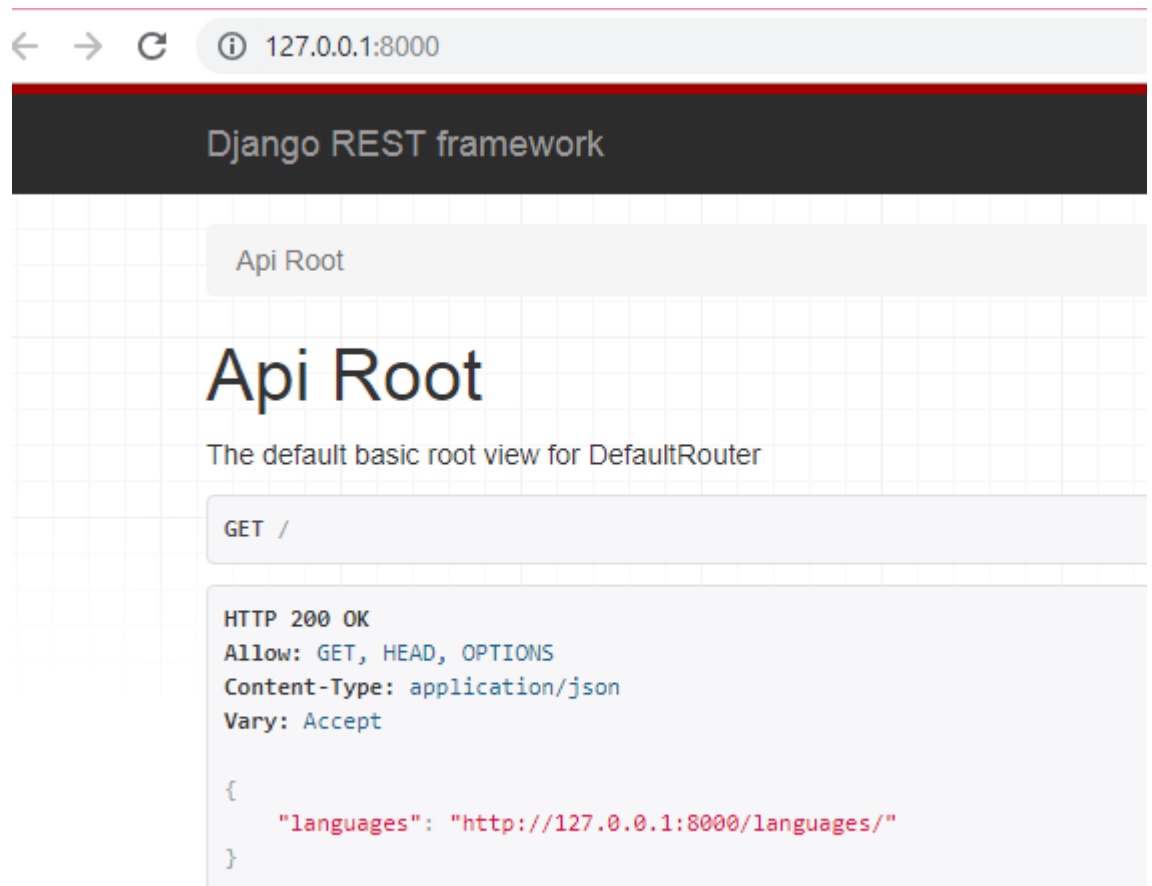


Figure 3.11 Django Rest Framework

3.4 Summary

In this chapter, the methodology of this project is explained. The flow of this project is being shown in the framework and detailed view of how the whole system works is display on the data model. Therefore, the requirements of the project are being shown in perfect order and can carry out the minimum error.

CHAPTER IV

IMPLEMENTATION AND RESULTS

4.1 INTRODUCTION

This chapter discusses the analysis of the implementation and testing of this project. To achieve the objectives of the project, we need to make sure that make use virtual environments to create and manage separate environments for Python projects. This is because Python virtual environments are to create an isolated environment for the Python project. This means that each project did not overlap because it has own dependencies. After Python virtual environment success. Install Django and Django Rest Framework and integrates Django Rest Framework with Token Based Authentication and make migrations all the third-party applications. Lastly, set up the dummy social network to connect with backend token authentication. The implementation of the token in the Django framework with a dummy social network was successfully implemented. This authentication using Django backend Token has been tested to ensure that this application reaches the objectives and meets the requirement.

4.2 Project Implementation

4.2.1 Setting up virtual environment

- 1) Settings up the virtual environment are an important step for this research but make sure Python and PIP installed. Python is the programming languages, it has a unique way of downloading, storing, and resolving packages. It has all this advantage but python also has some problem which is the packages are stored in the same place that can make them affect with another project. I need to install pip because the operating system that I use is windows.

```
PS C:\WINDOWS\system32> cd ~
PS C:\Users\User> cd desktop
PS C:\Users\User\desktop> python --version
Python 3.7.1
PS C:\Users\User\desktop> pip --version
pip 19.0.3 from c:\users\user\appdata\local\programs\python\python37-32\lib\site-packages\pip (python 3.7)
PS C:\Users\User\desktop>
```

Figure 4.1: python and pip version

- 2) A virtual environment is a tool that keeps dependencies required by different project separate by creating isolated python virtual environment for the project. In addition, the dependencies of every project are isolated from the system and each other.

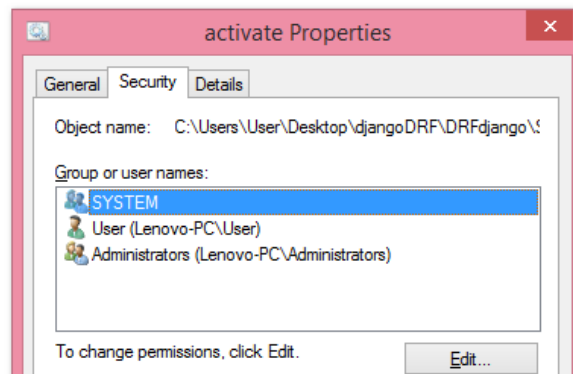
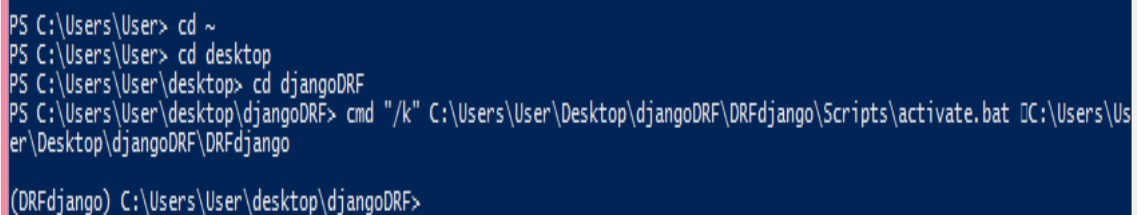


Figure 4.2: Path of the project

- 3) The task-based command line shell that uses in this project is the windows power shell. PowerShell is more complicated than the traditional command prompt, but it's also much more powerful. The command is a little bit different when using a command prompt and power shell. The command used in PowerShell is cmd "/k" and the path of the python virtual environment file. The path we just need to take it from bat file to activate the environment. Once the virtual environment is activated, the name of the virtual environment will appear on the left side of the terminal.

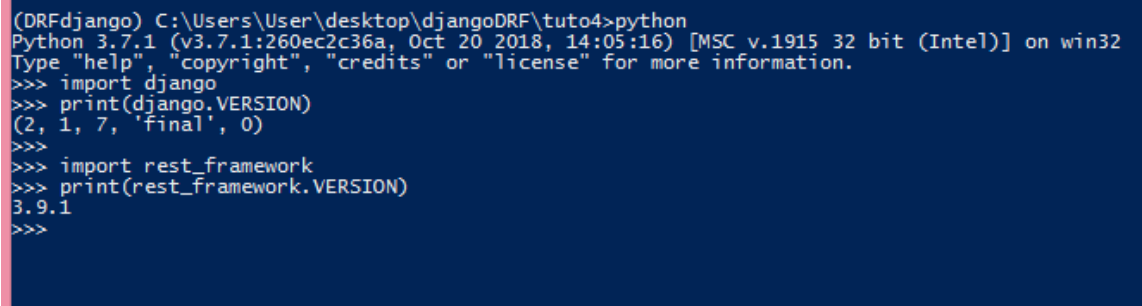


```
PS C:\Users\User> cd ~
PS C:\Users\User> cd desktop
PS C:\Users\User\desktop> cd djangoDRF
PS C:\Users\User\desktop\djangoDRF> cmd "/k" C:\Users\User\Desktop\djangoDRF\DRFdjango\Scripts\activate.bat CC:\Users\User\Desktop\djangoDRF\DRFdjango
(DRFdjango) C:\Users\User\desktop\djangoDRF>
```

Figure 4.3: Activate the python virtual environment

4.2.2 Download framework and configurations

Now install dependencies related to the project in this virtual environment. In this project the package requirement user to install Django, Django Rest Framework and pygments.



```
(DRFdjango) C:\Users\User\Desktop\djangoDRF\tuto4>python
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> print(django.VERSION)
(2, 1, 7, 'final', 0)
>>>
>>> import rest_framework
>>> print(rest_framework.VERSION)
3.9.1
>>>
```

Figure 4.4: Django and Django Rest Framework version in python virtual environment

4.2.3 Create API and configurations

- 1) The first step use is PowerShell to create API.

Command: python manage.py startapp snippets

This is how this project connects with each other. The code in Pycharm, use Python languages, configure in PowerShell and connect with localhost. If any error with the code, PowerShell as the server will show error and can't run with the localhost. After add snippets add rest_framework app to INSTALLED_APPS.

```
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     #third -party Apps
41     'rest_framework',
42     'rest_framework.authtoken',
43     #my project
44     'snippets.apps.SnippetsConfig',
```

Figure 4.5: List of INSTALLED_APPS, third-party Apps and project

- 2) Snippets is a programming term for a small region of the re-usable source code, machine code, or text. In this project, this is important to render code when using API. Start with creating a Snippet model that is used to store code snippets and use PowerShell to make migrations snippets that are responsible for creating new migrations based on the changes that have made to models. After this, migrate it which is responsible for applying migrations as well as applying and listing their status.

Command makemigrations: python manage.py makemigrations

Command migrate: python manage.py migrate

```
1  from django.db import models
2  from pygments.lexers import get_all_lexers
3  from pygments.styles import get_all_styles
4
5  from pygments.lexers import get_lexer_by_name
6  from pygments.formatters.html import HtmlFormatter
7  from pygments import highlight
8
9  #untuk store code snippet
10 LEXERS = [item for item in get_all_lexers() if item[1]]
11 LANGUAGE_CHOICES = sorted([(item[1][0], item[0]) for item in LEXERS])
12 STYLE_CHOICES = sorted((item, item) for item in get_all_styles())
13
14
15 class Snippet(models.Model):
16     created = models.DateTimeField(auto_now_add=True)
17     title = models.CharField(max_length=100, blank=True, default='')
18     code = models.TextField()
19     linenos = models.BooleanField(default=False)
20     language = models.CharField(choices=LANGUAGE_CHOICES, default='python', max_length=100)
21     style = models.CharField(choices=STYLE_CHOICES, default='friendly', max_length=100)
22     owner = models.ForeignKey('auth.User', related_name='snippets', on_delete=models.CASCADE)
23     highlighted = models.TextField(max_length=50, default=style)
24
25
26     class Meta:
27         ordering = ('created',)
28
29     # () save method
30     def save(self, *args, **kwargs):
31         """
```

Figure 4.6: Models.py code

```

# () save method
def save(self, *args, **kwargs):
    """
    Use the `pygments` library to create a highlighted HTML
    representation of the code snippet.
    """
    lexer = get_lexer_by_name(self.language)
    linenos = 'table' if self.linenos else False
    options = {'title': self.title} if self.title else {}
    formatter = HtmlFormatter(style=self.style, linenos=linenos,
                              full=True, **options)
    self.highlighted = highlight(self.code, lexer, formatter)
    super(Snippet, self).save(*args, **kwargs)

```

Figure 4.7: Models.py code

- 3) The next step is to provide a way of serializing and deserializing the snippet instances into representations as JSON. Create serializers file the def (define function) used to utilize code more than one place in the program.

```

1 from rest_framework import serializers
2 from snippets.models import Snippet, LANGUAGE_CHOICES, STYLE_CHOICES
3 from django.contrib.auth.models import User
4
5 #a web API endpoints returns JSON and available HTTP verbs
6 #convert native python datatype render to JSON or XML
7 class SnippetSerializer(serializers.HyperlinkedModelSerializer):
8     owner = serializers.ReadOnlyField(source='owner.username')
9     highlight = serializers.HyperlinkedIdentityField(view_name='snippet-highlight', format='html')
10
11     class Meta:
12         model = Snippet
13         fields = ('url', 'id', 'highlight', 'owner',
14                  'title', 'code', 'linenos', 'language', 'style')
15
16 #add representations of those users to our API
17 class UserSerializer(serializers.HyperlinkedModelSerializer):
18     snippets = serializers.HyperlinkedRelatedField(many=True, view_name='snippet-detail', read_only=True)
19
20     class Meta:
21         model = User
22         fields = ('url', 'id', 'username', 'snippets',)

```

Figure 4.8: Serializer.py code

- 4) This project uses ModelSerializer to make the code a bit concise. The serializer class defines the fields that get serialized and deserialized. The field flags can also control how the serializer should be displayed in certain circumstances, such when rendering to HTML.
- 5) We will a view which corresponds to an individual snippet and can be used to retrieve, update or delete the snippets. The @api_view is wrappers use to write API views. Its decorator for working with function-based views. Wrappers provide a few bits of functionality such as making sure to receive Request instances in view, and adding context to Response objects so that content negotiation can be performed.

```
1 from snippets.models import Snippet
2 from snippets.serializers import SnippetSerializer, UserSerializer
3 from rest_framework import generics, permissions
4 from django.contrib.auth.models import User
5 from snippets.permissions import IsOwnerOrReadOnly
6 from rest_framework.decorators import api_view
7 # wrapperb untuk make sure receive request instances
8 # in view and adding content response object
9 from rest_framework.reverse import reverse
10 from rest_framework import renderers
11 from rest_framework.response import Response
12
13
14 @api_view(['GET'])
15 def api_root(request, format=None):
16     return Response({
17         'users': reverse('user-list', request=request, format=format),
18         'snippets': reverse('snippet-list', request=request, format=format)
19     })
20
21 class SnippetList(generics.ListCreateAPIView):
22     queryset = Snippet.objects.all()
23     serializer_class = SnippetSerializer
24     permission_classes = (permissions.IsAuthenticated, )
25     # untuk handle any information that is implicit in the incoming request or requested URL.
26     def perform_create(self, serializer):
27         serializer.save(owner=self.request.user)
28
```

Figure 4.9: views.py code

```

29
30 class SnippetDetail(generics.RetrieveUpdateDestroyAPIView):
31     queryset = Snippet.objects.all()
32     serializer_class = SnippetSerializer
33     permission_classes = (permissions.IsAuthenticated, IsOwnerOrReadOnly, )
34
35 class UserList(generics.ListAPIView):
36     queryset = User.objects.all()
37     serializer_class = UserSerializer
38
39
40 class UserDetails(generics.RetrieveAPIView):
41     queryset = User.objects.all()
42     serializer_class = UserSerializer
43
44 class SnippetHighlight(generics.GenericAPIView):
45     queryset = Snippet.objects.all()
46     renderer_classes = (renderers.StaticHTMLRenderer,)
47
48     def get(self, request, *args, **kwargs):
49         snippet = self.get_object()
50         return Response(snippet.highlighted)
51

```

Figure 4.10: views.py code

```

1 from django.urls import path
2 from rest_framework.urlpatterns import format_suffix_patterns
3
4 from snippets import views
5 from rest_framework.authtoken.views import obtain_auth_token
6
7
8 # API endpoints
9 urlpatterns = format_suffix_patterns([
10     path('', views.api_root),
11     path('snippets/', views.SnippetList.as_view(), name='snippet-list'),
12     path('snippets/<int:pk>/', views.SnippetDetail.as_view(), name='snippet-detail'),
13
14     path('snippets/<int:pk>/highlight/', views.SnippetHighlight.as_view(), name='snippet-highlight'),
15
16     path('users/', views.UserList.as_view(), name='user-list'),
17     path('users/<int:pk>/', views.UserDetail.as_view(), name='user-detail'),
18     path('api-token-auth/', obtain_auth_token, name='api_token_auth'),
19     path('login/', views.login, name='login')
20 ]

```

Figure 4.11: snippet/urls.py code

4.2.4 Permissions

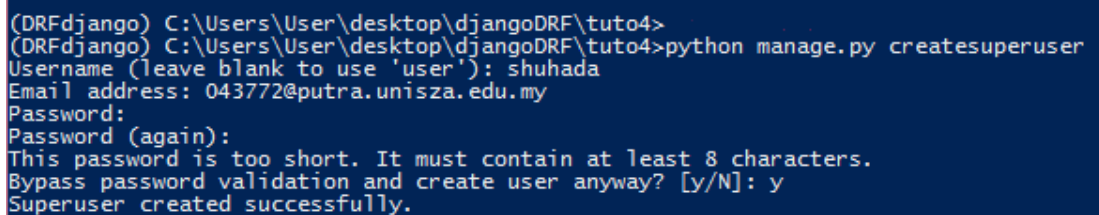
- 1) For now, there is no restriction to API, anyone can edit or delete code snippets.
Add a couple of fields one of those fields will be used to represent the user who created the code snippet. The other field will be used to store the highlighted HTML representation of the code.
- 2) After all the configurations, delete all the databases and start again to avoid them from effect with the data input from an authenticated user.
- 3) The command uses in PowerShell. Django default is using db.sqlite3 as a lightweight relational database. Thus, for the next step this project will change database.

Command: del -f tmp.db db.sqlite3

Command: del -r Snippets/migrations

Command: python manage.py makemigrations snippets

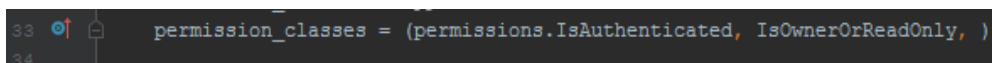
Command: python manage.py migrate



```
(DRFdjango) C:\Users\User\desktop\djangoDRF\tuto4>
(DRFdjango) C:\Users\User\desktop\djangoDRF\tuto4>python manage.py createsuperuser
Username (leave blank to use 'user'): shuhada
Email address: 043772@putra.unisza.edu.my
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Figure 4.12: create super user

- 4) Create superuser by using this command: python manage.py createsuperuser
- 5) All the credentials will add to database db.sqlite3.



```
33 permission_classes = (permissions.IsAuthenticated, IsOwnerOrReadOnly, )
34
```

Figure 4.13: permission is authenticated to someone who has credentials

- 6) The next step is adding a login to the browsable API because before this we create a restriction. If anyone wants to edit, it no longer available.
- 7) Add path ('api-auth/', include('rest_framework.urls')) to make login page for user login to Django localhost.

```
16 from django.contrib import admin
17 from django.urls import path
18 from django.conf.urls import include
19
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('api-auth/', include('rest_framework.urls')),
24     path('', include('snippets.urls')),
25 ]
```

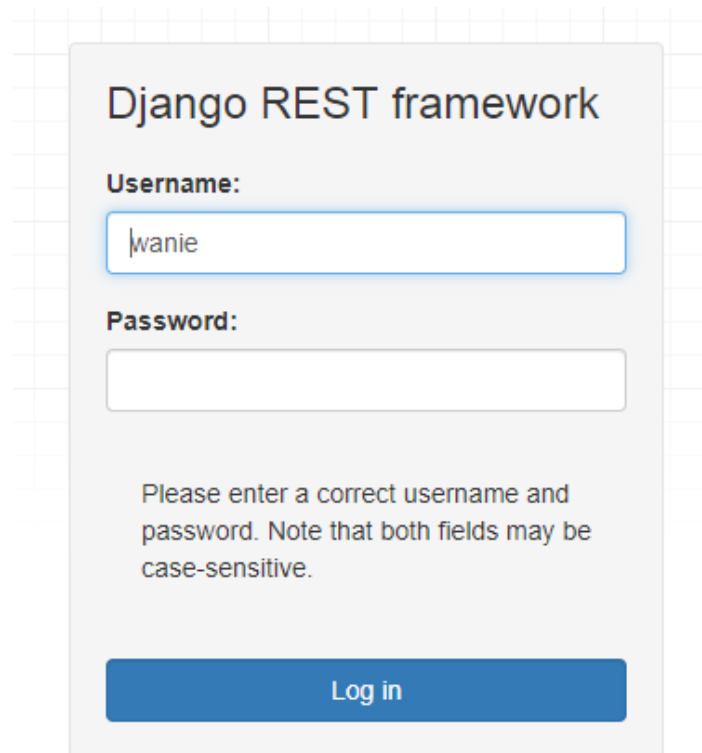
Figure 4.14: urls.py code

- 8) Create permission file to make sure code snippets is able to update or delete it.

```
1 from rest_framework import permissions
2
3
4 class IsOwnerOrReadOnly(permissions.BasePermission):
5     """
6     Custom permission to only allow owners of an object to edit it.
7     """
8
9     def has_object_permission(self, request, view, obj):
10         # Read permissions are allowed to any request,
11         # so we'll always allow GET, HEAD or OPTIONS requests.
12         if request.method in permissions.SAFE_METHODS:
13             return True
14
15         # Write permissions are only allowed to the owner of the snippet.
16         return obj.owner == request.user
```

Figure 4.15: Permission code who can edit and delete

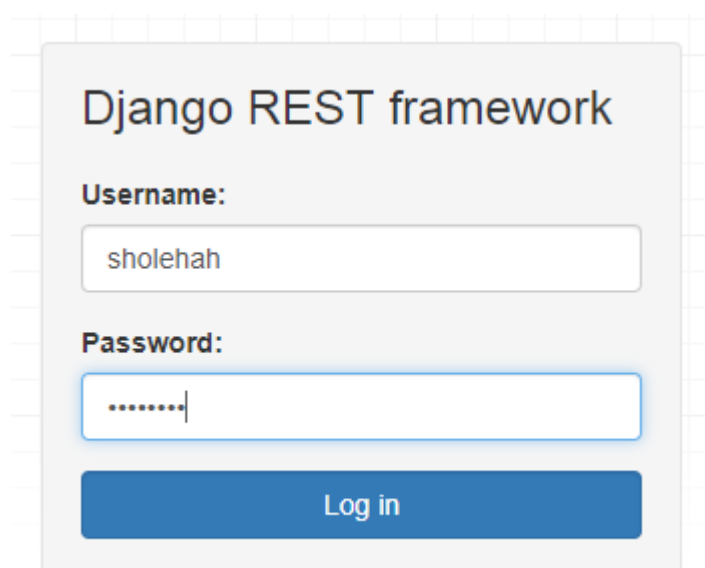
- 9) This is the login page when the user login with the wrong username and password. The output will be shown “please enter a correct username and password. Note that both fields may be case-sensitive.”



The screenshot shows a login form titled "Django REST framework". It has two input fields: "Username:" with the value "lwanie" and "Password:" which is empty. Below the fields is a message: "Please enter a correct username and password. Note that both fields may be case-sensitive." At the bottom is a blue "Log in" button.

Figure 4.16: wrong username and password

- 10) This is the correct username and password input by the user.



The screenshot shows the same login form titled "Django REST framework". The "Username:" field contains "sholehah" and the "Password:" field contains masked characters ".....". A blue "Log in" button is at the bottom.

Figure 4.17 Right credential

- 11) At the right of the page, the name of the user will be shown for example “sholehah”. If the user does not log in, the right of the page will show “log in” and anyone who wants log in can click to input username and password. After login, the user will see detail.

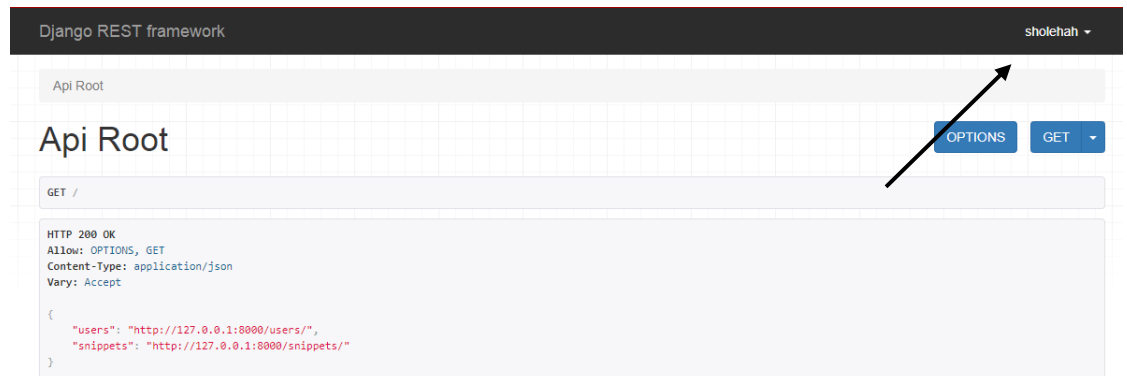


Figure 4.18: This picture show user input with correct credential and can edit or delete snippets

4.2.5 Implementing the Token Based Authentication

- 1) Add `rest_framework.authtoken` to `Installed_apps` and. This is the third - party Apps that this project use. The default of Django doesn't have token authentication. To empower the Django security this project, Token-based approach is used, Hence, it needs some modification of the Django. After add, do migrate by using PowerShell and its success.

```
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     #third -party Apps
41     'rest_framework',
42     'rest_framework.authtoken',
43     #my project
44     'snippets.apps.SnippetsConfig',
45 ]
```

Figure 4.19: Add `rest_framework.authtoken` as a third-part application.

- 2) Include `rest_framework.authentication.TokenAuthentication` to `REST_FRAMEWORK`. After some research Token authentication cannot stand alone. It needs session authentication and for the basic authentication, it is just for testing. Thus, my project needs two components of authentication Token-based Authentication and session-based authentication.

```
133 REST_FRAMEWORK = {
134
135
136     'DEFAULT_AUTHENTICATION_CLASSES': (
137         'rest_framework.authentication.TokenAuthentication',
138         'rest_framework.authentication.SessionAuthentication',
139         #'rest_framework.authentication.BasicAuthentication',
140     ),
141
142     'DEFAULT_PERMISSION_CLASSES': (
143         'rest_framework.permissions.IsAuthenticated',
144     ),
145 }
```

Figure 4.20: Add to rest framework token and session

- 3) For testing the authentication, using power shell try to input `http http://127.0.0.1:8000/snippets/`. In the figure below shown `www-Authentication: Token`. This project set is authenticated only for those who have credentials. It can be set anyone can see but cannot edit/delete and it can be set anyone can edit/delete. If the link input without a token, the output is “detail”: Authentication credentials were not provided.”

```
(DRFdjango) C:\Users\User\desktop\djangoDRF\tuto4>http http://127.0.0.1:8000/snippets/
HTTP/1.1 401 Unauthorized
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 58
Content-Type: application/json
Date: Mon, 29 Apr 2019 08:02:12 GMT
Server: WSGIServer/0.2 CPython/3.7.1
Vary: Accept, Cookie
WWW-Authenticate: Token
X-Frame-Options: SAMEORIGIN

{
  "detail": "Authentication credentials were not provided."
}
```

Figure 4.21: Testing without token

- 4) Input username and password in power shell to see token.

```
http post http://127.0.0.1:8000/api-token-auth/ username=sholehah  
password=sholehah
```

```
(DRFdjango) C:\Users\User\desktop\djangoDRF\tuto4>http post http://127.0.0.1:8000/api-token-auth/ username=sholehah password=sholehah  
HTTP/1.1 200 OK  
Allow: POST, OPTIONS  
Content-Length: 52  
Content-Type: application/json  
Date: Mon, 29 Apr 2019 07:31:38 GMT  
Server: WSGIServer/0.2 CPython/3.7.1  
Vary: Cookie  
X-Frame-Options: SAMEORIGIN  
  
{  
  "token": "98c56dff138e9ac3993776a7f311134b4d29a910"  
}
```

Figure 4.22: Token

- 5) The token that got from above can be used to get credentials with the snippets detail. As this project set anyone who authenticated only can see the detail. Input the token user and the credentials will provide as the figure below. It is one way to testing using http command. Another way is the user tool postman or curl in a terminal.

```
(DRFdjango) C:\Users\User\desktop\djangoDRF\tuto4>http http://127.0.0.1:8000/snippets/ "Authorization: Token 98c56dff138e9ac3993776a7f311134b4d29a910"  
HTTP/1.1 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Length: 481  
Content-Type: application/json  
Date: Mon, 29 Apr 2019 08:04:41 GMT  
Server: WSGIServer/0.2 CPython/3.7.1  
Vary: Accept  
X-Frame-Options: SAMEORIGIN  
  
{  
  "count": 2,  
  "next": null,  
  "previous": null,  
  "results": [  
    {  
      "code": "java",  
      "highlight": "http://127.0.0.1:8000/snippets/1/highlight/",  
      "id": 1,  
      "language": "bugs",  
      "linenos": false,  
      "owner": "sholehah",  
      "style": "abap",  
      "title": "Hello world",  
      "url": "http://127.0.0.1:8000/snippets/1/"  
    },  
    {  
      "code": "C",  
      "highlight": "http://127.0.0.1:8000/snippets/2/highlight/",  
      "id": 2,  
      "language": "html+php",  
      "linenos": false,  
      "owner": "sholehah",  
      "style": "lovelace",  
      "title": "Save world",  
      "url": "http://127.0.0.1:8000/snippets/2/"  
    }  
  ]  
}
```

Figure 4.23: Testing with Token

4.2.6 Connect framework to phpMyAdmin and change database

- 1) Django by default automatically creates an SQLite database for any project. In addition to SQLite, Django officially supports three other popular relational databases that include: PostgreSQL, MySQL and Oracle. The Django configuration to connect to a database is done inside the `setting.py` of the Django project.

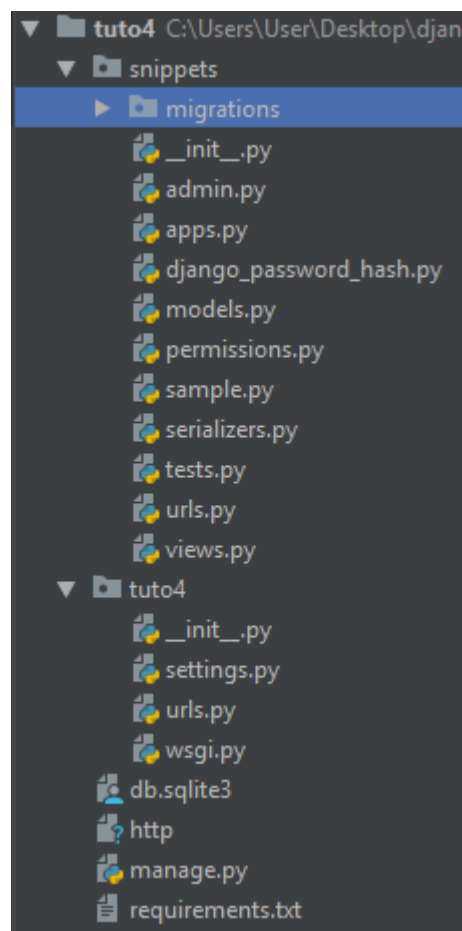


Figure 4.24: List of Django default with `db.sqlite3` database

- 2) Open setting.py file of Django, there is DATABASES variable has a default dictionary with the values illustrated.

```
80
81 DATABASES = {
82     'default': {
83         'ENGINE': 'django.db.backends.mysql',
84         # 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
85         'NAME': 'db_spkpta',
86         'USER': 'root',
87         'PASSWORD': '',
88         'HOST': '127.0.0.1',
89         'PORT': '3306',
90     }
91 }
```

Figure 4.25: Change database to mysql

- 3) Change db.sqlite to mysql. Create a new database, in this project, using the db_spkpta because this database connects with the dummy website to replace social networks. Set user and password, in this project host is the IP address that databases are hosted and we using 3306 port because using MySQL.

Module	PID(s)	Port(s)
Apache	11600 3968	80, 443
MySQL	12092	3306
FileZilla	6920	21, 14147

Figure 4.26: Start all port to connect with server.

- 4) Install mysql python connector in powershell and make sure the version compatible with the python version.

```
>>> try:
...     connection = mysql.connector.connect(host='localhost',
...                                         database='db_spkpta',
...                                         user='root',
...                                         password='')
...     if connection.is_connected():
...         db_Info = connection.get_server_info()
...         print("Connected to MySQL database... MySQL Server version on ",db_Info)
...         cursor = connection.cursor()
...         cursor.execute("select database();")
...         record = cursor.fetchone()
...         print ("Your connected to - ", record)
... except Error as e :
...     print ("Error while connecting to MySQL", e)
... finally:
...     #closing database connection.
...     if(connection.is_connected()):
...         cursor.close()
...         connection.close()
...         print("MySQL connection is closed")
...
Connected to MySQL database... MySQL Server version on 5.6.21
Your connected to - ('db_spkpta',)
True
MySQL connection is closed
>>>
```

Figure 4.27: Testing mysql python connector

- 5) Import the pymysql module and used pymysql.install_as_MySQLdb() to convert from pymysql to Mysql to import data to Mysql database. Add this code at Init.py for the main project.

```
1  import pymysql
2
3  pymysql.install_as_MySQLdb()
```

Figure 4.28: _init.py_ code

4.2.7 Matching hashing database and give Token to user

- 1) The username and password that we create in task-based command line shell will be at in mysql databases. Django framework will hash the password to secure the credential of user.

id	password	last_login	is_superuser	username
1	pbkdf2_sha256\$120000\$CXAfkLWOt0Ec\$laZsrMJT1y3UFR6+...	2019-04-20 08:07:10.469711	1	sholehah
2	pbkdf2_sha256\$120000\$XQ0Pve4ehEgx\$mlkrA24S+O91AykM...	2019-04-20 08:40:47.838772	1	luqman
3	pbkdf2_sha256\$120000\$XklKZWbhTraH\$1Nv1AdF19KIHNxl5...	NULL	1	jannah




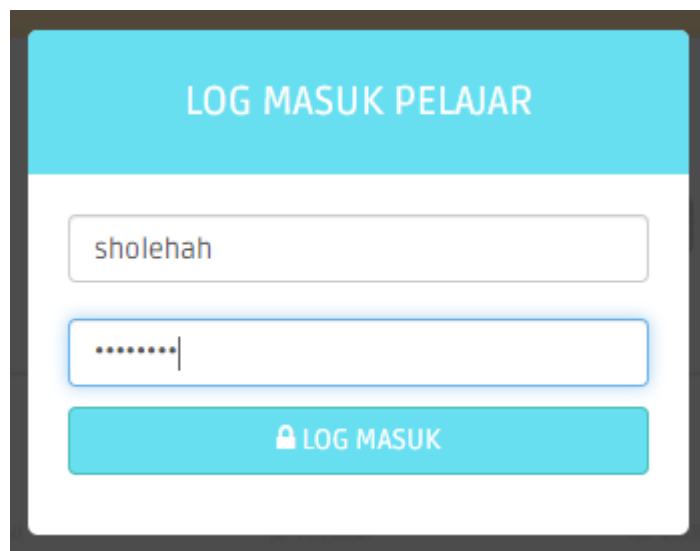
selected:  Change  Delete  Export

Figure 4.29: Hashing password from database

- 2) Django provides a flexible password storage system and uses PBKDF by default. This project needs a dummy website to testing the password and token. This is the credentials that used, for example, username: sholehah, password: sholehah.



The image shows a login interface titled "LOG MASUK PELAJAR" (Student Login). It features a light blue header with the title. Below the header, there are two input fields: the first contains the username "sholehah", and the second contains a masked password represented by seven dots. Below these fields is a large blue button with a white lock icon and the text "LOG MASUK".

Figure 4.30: Login interface to input credentials



Figure 4.31: Main interface when user can login and matching with the hashing password

- 3) The password that user input will be matching with the password in MySQL databases but in this case, the system will hashing with sha256 and salt first before matching with password on the db_sptkpta database.

```

3 session_start();
4
5 include "connection.php";
6
7 $query2 = "SELECT * FROM auth_user WHERE username='".$$_POST['no_matrik']."'";
8 $result2 = @mysql_query($query2);
9 $row2 = @mysql_fetch_array($result2);
10 $pw2 = $row2['password']; //fetch password table (hashing)
11 $id = $row2['id']; //fetch password table (hashing)
12
13
14
15 $pw=$_POST['kata_laluan'];
16 include 'test.php';
17 $check = django_password_verify("$pw","$pw2"); // compare guna function dlm test.php
18 if((django_password_verify("$pw","$pw2"))==true){
19
20 $query3 = "SELECT * FROM authtoken_token WHERE user_id='$id'";
21 $result3 = @mysql_query($query3);
22 $row3 = @mysql_fetch_array($result3);
23 $token = $row3['key'];
24 $pw2 = $row2['password'];
25
26 // $query = "SELECT * FROM tbl_pelajar WHERE no_matrik='".$$_POST['no_matrik']."' AND kata_laluan='".$$_POST['kata_laluan']."'";
27 $query = "SELECT * FROM auth_user WHERE username='".$$_POST['no_matrik']."' AND password='$pw2'";
28 $result = @mysql_query($query);
29 }else
30 {
31 //echo"<script>alert('Id dan Kata Laluan Salah!');document.location.href='logmasukpelajar.php';</script>";
32

```

Figure 4.32: PHP code login from dummy website

4) The test.php is the algorithm that uses to matching with databases.

```
$check = django_password_verify("$pw","$pw2");
```

```
34 function django_password_verify($password, $djangoHash)
35 {
36
37     $pieces = explode('$', $djangoHash);
38
39     if (count($pieces) != 4) {
40         throw new Exception("Illegal hash format");
41     }
42     list($header, $iter, $salt, $hash) = $pieces;
43     // Get the hash algorithm used:
44     if (preg_match('#^pbkdf2_([a-z0-9A-Z]+)$#', $header, $m)) {
45         $algo = $m[1];
46     } else {
47         throw new Exception(sprintf("Bad header (%s)", $header));
48     }
49     if (!in_array($algo, hash_algos())) {
50         throw new Exception(sprintf("Illegal hash algorithm (%s)", $algo));
51     }
52
53     $calc = hash_pbkdf2(
54         $algo,
55         $password,
56         $salt,
57         (int) $iter,
58         32,
59         true
60     );
61     return hash_equals($calc, base64_decode($hash));
62 }
```

Figure 4.33: PHP code to matching hashing password

```
if(!function_exists('hash_equals'))
{
function hash_equals($str1, $str2)
{
    if(strlen($str1) != strlen($str2))
    {
        return false;
    }
    else
    {
        $res = $str1 ^ $str2;
        $ret = 0;
        for($i = strlen($res) - 1; $i >= 0; $i--)
        {
            $ret |= ord($res[$i]);
        }
        return !$ret;
    }
}
}
```

Figure 4.34: PHP code to matching hashing password

```

1  <?php
2  session_start();
3  include "connection.php";
4
5  if(!empty($_SESSION['id'])) {
6      $id=$_SESSION['id'];
7      $no_matrik=$_SESSION['no_matrik'];
8      $token=$_SESSION['token'];//fetch from session
9
10 }
11 else
12 {
13     echo"<script>alert('Please Log In!');document.location.href='logmasukpelajar.php';</script>";
14 }
15
16 $query3 = "SELECT * FROM authtoken_token WHERE user_id='$id'";
17 $result3 = @mysql_query($query3);
18 $row3 = @mysql_fetch_array($result3);
19 $token2 = $row3['key'];//fetched from db
20
21
22 if($token != $token2)
23     echo"<script>alert('Invalid Token!');document.location.href='logmasukpelajar.php';</script>";
24 ?>

```

Figure 4.35: PHP code token function

4.2.8 Try CSRF attack on website

- 1) By using RawCap, try sniff some packet from localhost and see stream php in Wireshark.

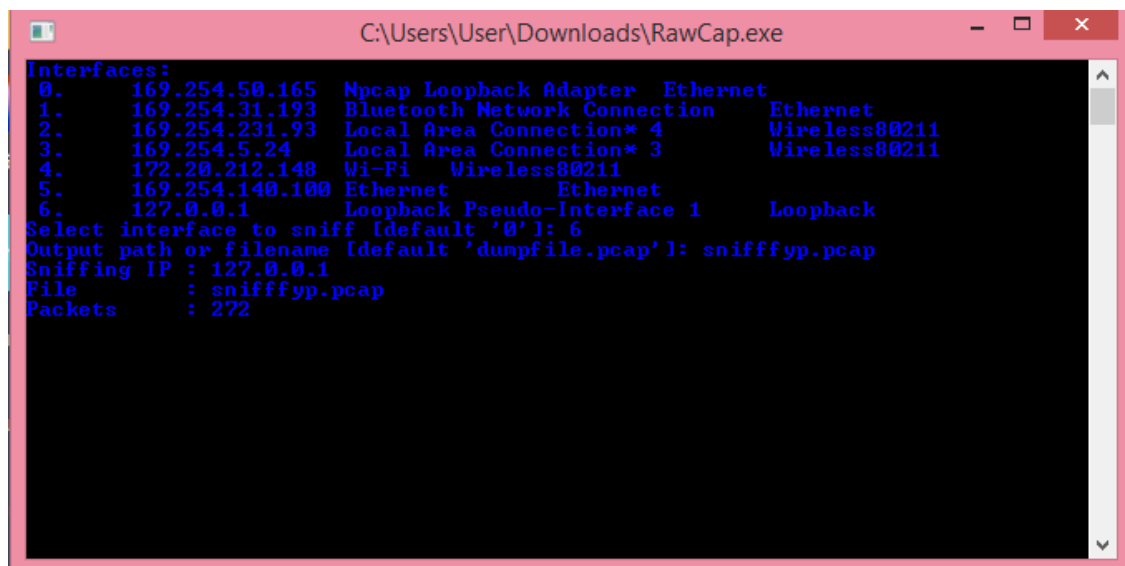


Figure 4.36: Using RawCap to sniff packets

Time	Source	Destination	Protocol	Length	Info
112.19.328040	127.0.0.1	127.0.0.1	HTTP	642	HTTP/1.1 200 OK (text/html)
113.19.328040	127.0.0.1	127.0.0.1	TCP	40	64878 → 80 [ACK] Seq=641 Ack=7903 Win=65536 Len=0
114.19.348041	127.0.0.1	127.0.0.1	HTTP	537	GET /FYPsystem/assets/js/jquery-1.8.3.min.js HTTP/1.1
115.19.348041	127.0.0.1	127.0.0.1	TCP	40	80 → 64877 [ACK] Seq=22655 Ack=2830 Win=65024 Len=0
116.19.352041	127.0.0.1	127.0.0.1	TCP	618	80 → 64877 [PSH, ACK] Seq=22655 Ack=2830 Win=65024 Len=578 [TCP segment of a reassembled PDU]
117.19.352041	127.0.0.1	127.0.0.1	TCP	40	64877 → 80 [ACK] Seq=2830 Ack=23233 Win=64768 Len=0
118.19.352041	127.0.0.1	127.0.0.1	TCP	493	80 → 64877 [PSH, ACK] Seq=23233 Ack=2830 Win=65024 Len=453 [TCP segment of a reassembled PDU]
119.19.352041	127.0.0.1	127.0.0.1	TCP	40	64877 → 80 [ACK] Seq=2830 Ack=23686 Win=64512 Len=0
120.19.352041	127.0.0.1	127.0.0.1	TCP	371	80 → 64877 [PSH, ACK] Seq=23686 Ack=2830 Win=65024 Len=331 [TCP segment of a reassembled PDU]
121.19.352041	127.0.0.1	127.0.0.1	TCP	40	64877 → 80 [ACK] Seq=2830 Ack=24017 Win=65536 Len=0
122.19.362042	127.0.0.1	127.0.0.1	TCP	236	80 → 64877 [PSH, ACK] Seq=24017 Ack=2830 Win=65024 Len=106 [TCP segment of a reassembled PDU]

Figure 4.37: Packets that got from localhost

```

GET /FYPsystem/lamanutama-pelajar.php HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://127.0.0.1/FYPsystem/loginpelajar.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: csrftoken=o03HvL9i076JvWCFK8XNcU9g6UjUa9h5ieV0eOR0bXwkDgZJxzCg9EmQhhAIHKU7; PHPSESSID=5s6hplv3fim7irghci188lg0v7
  
```

Figure 4.38: Data that sniff from website

4.3 Summary

In conclusion, we can conclude authentication in a social network is important. The social network is for anyone, child, teenagers, and adult. They don't know any words that save to the system will have history and hackers can get their data anytime and anywhere. Django backend Token-based authentication is a new era that needs to use in the social network because of the security. The backend that generates a token and hashing password will not be known by the hackers if they can attack the database of the system.

CHAPTER V

CONCLUSION

5.1 INTRODUCTION

This chapter discusses the contributions of validations of user credentials in social networks by using Django backend Authentication. Moreover, project constraints, limitations, and future works also will be discussed in this chapter.

5.2 PROJECT CONTRIBUTION

A Django Based System allows integration between Tokens Based Authentication with Django Framework. Incorporating between both components enable to share permissions of any applications.

5.3 PROJECT CONSTRAINTS AND LIMITATIONS

Generally, the project constraints and limitations are discussed in terms of the obstacles and limitations of the project. There have a few constraints and limitations such as this project using the dummy social network. Social networks such as Facebook are to secure to make corporations and Social networks such as Twitter have many policies and only certain files can be accessed. The important file such as login files, of course, cannot be accessed because of security.

Besides, the process of integrated Django Authentication to Social Network is complicated and needs through a study about it before I can proceed to the next steps. It takes a lot of effort and time to explore and study the architecture a component of the authentication in a social network.

5.4 FUTURE WORKS

In the future, there are still a lot of features that can be added to this project. Implementing Django authentication in a real-world social network is one way to improve this project. Even though, it's not easy to implement, maybe by cooperation with any real-world social network is a good way to improve security in the social network. Lastly, the token generator changes within 10-20 seconds. The more often the number of tokens change the higher security be implemented because if hackers can get in the database, hackers have to attempt within 10 seconds of the token is impossible. If it is successfully developed, then this application will be highly contributing to others.

5.5 SUMMARY

Towards, Industrial Revolution 4.0, security now has become one of the big issues as we cannot protect the data from the intruders. We have to secure data from the start, its mean login page. The login such as Facebook and Twitter were noticed used by people across the world on a daily basis. We can say that authentication nowadays is important as it allows us to securely protect our data. The development of this project has reached the objective that has been stated at the beginning of the semester.

References

- [1] Bansal, S. (n.d.). Token Based Authentication for Django Rest Framework. Retrieved from medium.com: <https://medium.com/quick-code/token-based-authentication-for-django-rest-framework-44586a9a56fb>
- [2] Denis, S. (2018, January 10). The Pros and Cons of Different Two-Factor Authentication Types and Methods. Retrieved from protectimus.com: <https://www.protectimus.com/blog/two-factor-authentication-types-and-methods/>
- [3] DICKSON, B. (n.d.). 5 authentication methods putting passwords to shame. Retrieved from thenextweb.com: <https://thenextweb.com/insider/2016/03/31/5-technologies-will-flip-world-authentication-head/>
- [4] DigitalOcean. (2011-2018). django-rest-framework.org. Retrieved from Django Rest Framework: <https://www.django-rest-framework.org/#installation>
- [5] Django software, f. (2015-2018). Djangoproject. Retrieved from Django: <https://docs.djangoproject.com/en/2.1/topics/auth/customizing/>
- [6] Felix. (2014, April 23). felixthea. Retrieved from Frontend vs Backend: Learn Before Hiring a Freelance Developer: <http://felixthea.com/frontend-vs-backend/>
- [7] justmobiledev. (2017, November 12). justmobiledev. Retrieved from HOME: <http://justmobiledev.com/token-based-versus-cookie-based-authentication-methods/>

- [8] Kasulani, E. K. (n.d.). Let's build an API with Django REST Framework. Retrieved from medium.com: <https://medium.com/backticks-tildes/lets-build-an-api-with-django-rest-framework-32fcf40231e5>
- [9] Kevin Hennessy, C. A. (n.d.). Token-based authentication. Retrieved from www.oreilly.com: <https://www.oreilly.com/library/view/angular-6-by/9781788835176/22c89ed6-64f4-466b-8c2b-280f76185bc3.xhtml>
- [10] L., J. (n.d.). Token-Based Authentication | 5 Steps to Get Started. Retrieved from swoopnow.com: <https://swoopnow.com/token-based-authentication/>
- [11] Lee, J. (2017, November 14). makeuseof.com. Retrieved from The Pros and Cons of Two-Factor Authentication Types and Methods: <https://www.makeuseof.com/tag/pros-cons-2fa-types-methods/>
- [12] Markus Jakobsson, D. I. (2011). Bootstrapping mobile PINs using passwords. Information Risk management PayPal.
- [13] MICHAEL, C. (n.d.). *compukol*. Retrieved from Social Media vs Social Networking: <https://www.compukol.com/social-media-vs-social-networking>
- [14] P.Austel S Bhola, S. .. (2008). Secure Delegation for web 2.0 and Mashups. New york: IBM T.J watson reseacrh center.
- [15] Samanage. (2014, june 24). samanage.com. Retrieved from What is Multifactor Authentication?: <https://blog.samanage.com/it-asset-management/multifactor-authentication-challenges-and-benefits/>
- [16] Sarita Yardi, N. F. (2008, August 18). Photo Based Authentication using social networks. p. 5.

- [17] Sevilleja, C. (2015, January 21). The Ins and Outs of Token Based Authentication. Retrieved from scotch.io: <https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication>
- [18] Soós, G. (n.d.). Angular authentication revisited. Retrieved from medium.com: <https://medium.com/@blacksonic86/angular-2-authentication-revisited-611bf7373bf9>
- [19] sslayo. (2015, october). I Will Create PHP Scripts That Use Any Web Api. Retrieved from fiverr.com: <https://www.fiverr.com/sslayo/create-php-scripts-that-use-any-web-api>
- [20] Steven, H. (23, May 2017). Advantages and Disadvantages of Django. Retrieved from hackernoon.com: <https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5>
- [21] Unity, D. (n.d.). Retrieved from Inc. Designed by Divine Unity 1 Unlimited, LLC.: <http://divineunity1.org/social-networking/>
- [22] Vincent, W. S. (n.d.). Django Login/Logout Tutorial (Part 1). Retrieved from wsvincent.com: <https://wsvincent.com/django-user-authentication-tutorial-login-and-logout/>
- [23] Vincent, W. S. (n.d.). Django Rest Framework User Authentication Tutorial - Custom User Model + Social Auth. Retrieved from wsvincent.com: <https://wsvincent.com/django-rest-framework-user-authentication-tutorial/>
- [24] geeksforgeeks. (n.d.). Python Virtual Environment | Introduction. Retrieved from geeksforgeeks.org: <https://www.geeksforgeeks.org/python-virtual-environment/>

- [25] What are the best python full stack framework. (n.d.). Retrieved from www.slant.co:
<https://www.slant.co/topics/533/~best-python-full-stack-frameworks>
- [26] 3v4l. (2016, Nov). verify a Django password (PBKDF2-SHA256). Retrieved from
3v4l.org: <https://3v4l.org/6e0ZR>
- [27] Purusothaman, 1. V. (2011). A Secure Simple Authenticated Key Exchange Algorithm. Computer Science, 5.
- [28] A.Al-Ani, A. H.-H. (2005). A New Approach for Authentication Technique. Journal of Computer Science, 4.
- [29] Shakir M. Hussain, H. A.-B. (2008). A Non-Exchanged Password Scheme for Password-Based. Journal of Computer Science, 5.
- [30] Vasiljevic, I. (2017). How to do Twitter authentication with React and RESTful API. Retrieved from medium.com: <https://medium.com/@robince885/how-to-do-twitter-authentication-with-react-and-restful-api-e525f30c62bb>

APPENDIX

GANTT CHART

Activity \ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Discuss the title of the final year project with supervisor.																
Submission of the title and abstract of the project.																
Specification of problem statement, objectives, scope, and literature review.																
Preparation for proposal presentation.																
Proposal presentation.																
Proposal correction and proposed solution methodology.																
Research of literature review.																
Documentation of proposal.																

Activity \ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Project meeting with supervisor.																
Project development.																
Project meeting with supervisor.																
Project progress presentation.																
Project development and project testing.																
Online submission of poster link.																
Final presentation for FYP 2.																
Final thesis submission & log book to supervisor.																
Submission hardcover to Faculty.																

Gantt Chart (FYP 2)