**Build Restful Services Using Flask, Flask – Restful, Python, Mongo Engine & Flask-JWT**

Table Contents

[Introduction 3](#_Toc48811272)

[Revision History 3](#_Toc48811273)

[Prerequisites: 3](#_Toc48811274)

[Project specific dependencies: 3](#_Toc48811275)

[1. Create a new directory and installation file in VS code 3](#_Toc48811276)

[Install required dependency 4](#_Toc48811277)

[2. Create entry point of application: 4](#_Toc48811278)

[3. Configuring properties with config.py 5](#_Toc48811279)

[Sub-directory & Sub-modules](#_Toc48811280)**[:](#_Toc48811280)** [5](#_Toc48811280)

[1. Create python application initializer: 6](#_Toc48811281)

[2. Create Rest Controller for CRUD operations: 6](#_Toc48811282)

[3. Create model for MongoDB interaction: 7](#_Toc48811293)

[4. Configure password security & JWT authentication: 7](#_Toc48811294)

[5. Create WSGI component: 8](#_Toc48811295)

[Flask Server & MongoDB](#_Toc48811296)**[:](#_Toc48811296)** [9](#_Toc48811296)

[1. Run MongoDB : 9](#_Toc48811297)

[2. Access Flask Server: 10](#_Toc48811298)

[3. Access MongoDB using Robo 3T: 10](#_Toc48811299)

[Test Restful-Service: 11](#_Toc48811300)

[1. Test using Postman: 11](#_Toc48811301)

[2. Insert data into MongoDB 11](#_Toc48811302)

[3. Authenticate user by JWT: 12](#_Toc48811303)

[4. Add authorization token to request header : 12](#_Toc48811304)

[5. Access user session on JWT authentication: 13](#_Toc48811305)

Introduction:

The tutorial provides a step-by-step guide for building and configuring a Restful services in Python using Flask, Flask-Restful, Mongo Engine & Flask-JWT components. You will learn how easy it is to create your Python web application and test with Postman. WSGI server is a deployment component added on top of Flask Server to make application production ready and deployed with Web Servers or on various Cloud Platforms. Postman is used to test the rest webservices.

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Changes Name** | **Updated By** | **Date** | **Version** |
| New Document | Sudhanshu Satyam | 2020/08/17 | 1.0.0 |
| Revision |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Prerequisites:

These are the softwares required to installed on local machine before starting with the application.

Visual Studio Code

Python 3.5 or above

Flask 1.1.2

Flask-Restful 0.3.8

Flask-JWT 0.3.2

Mongo Engine 0.20.0

pymongo 3.11.0

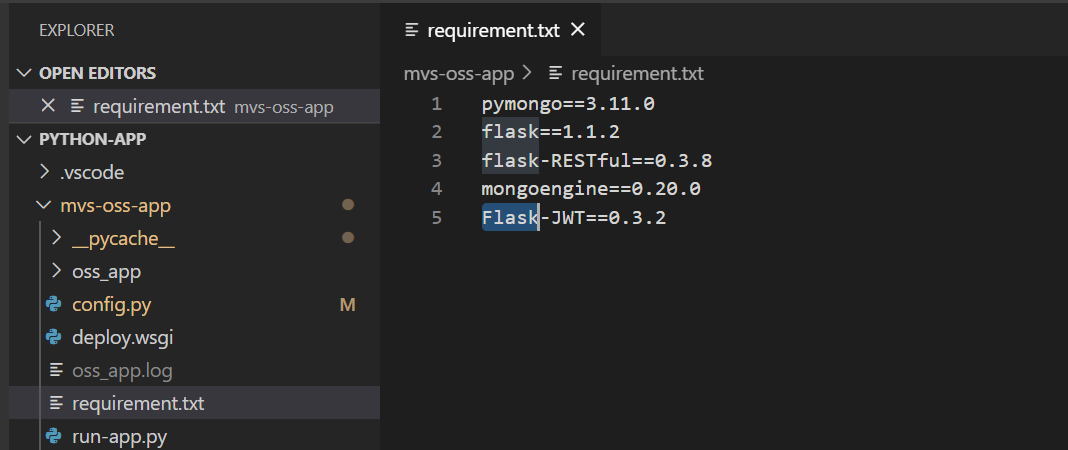
Postman

# Project specific dependencies:

## Create a new directory and installation file in VS code

Create a new directory and name your directory as ‘mvs-oss-app’ and create new text file and name it as ‘requirements.txt’ .

Add all the required dependencies needed for your application in this text file as shown in screenshot :

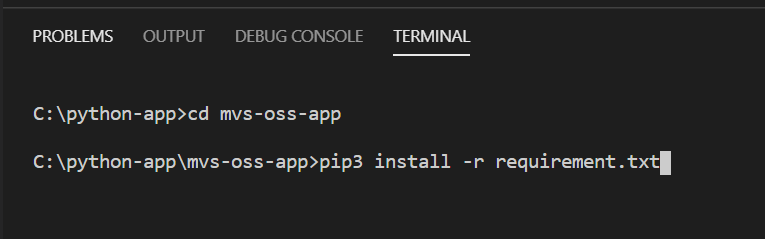


Install required dependency **:**

Change current directory to new directory from terminal and execute the command to install dependent libraries

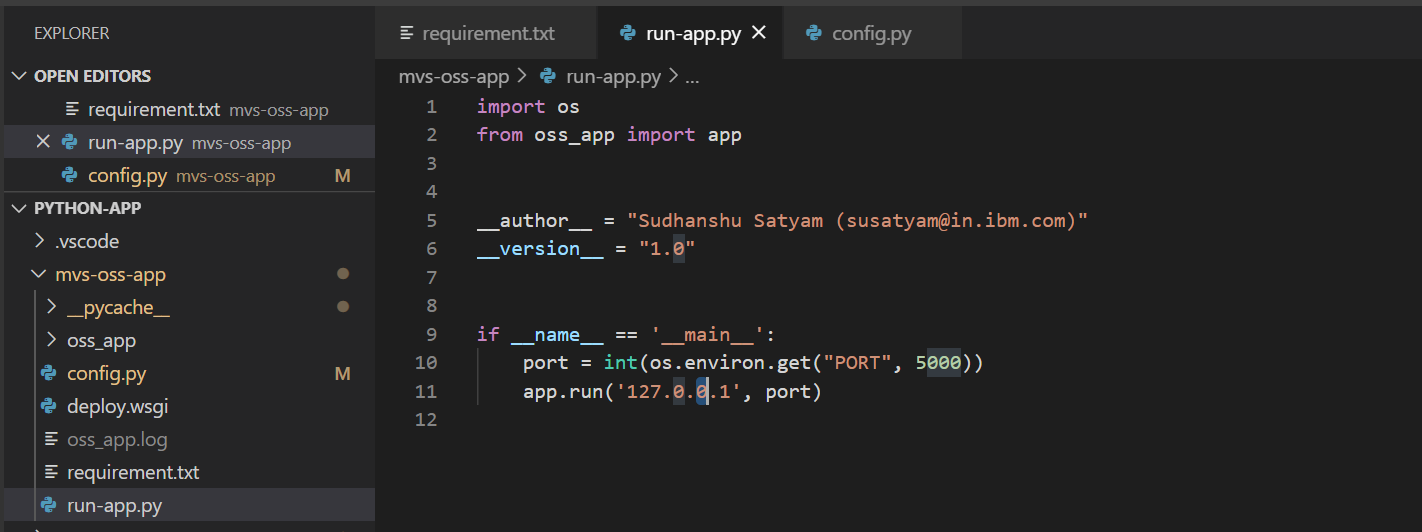
Command:

pip3 install -r requirements.txt



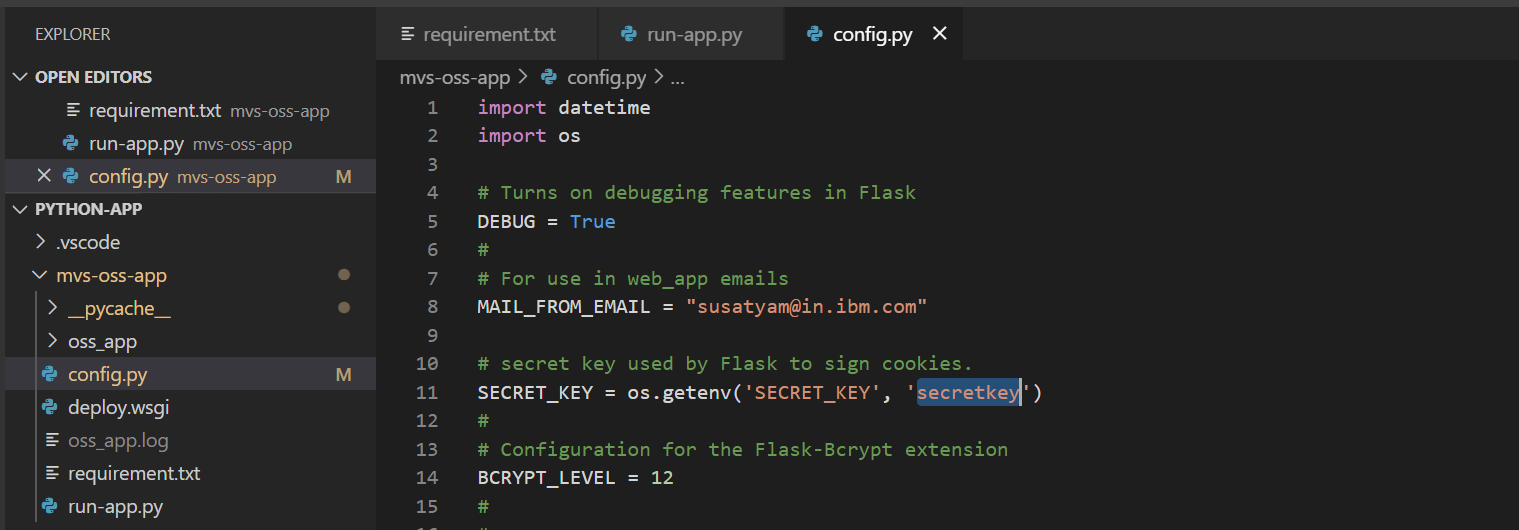
## Create entry point of application:

Under ‘mvs-oss-app’ directory, create new file and name this file as ‘run-app.py’. This file contains the main method and serve as entry point for Python web application.



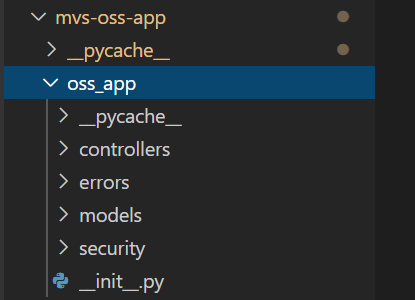
## Configuring properties with config.py

Now , We have to configure project specific properties by creating config.py file under ‘mvs-oss-app’ directory.



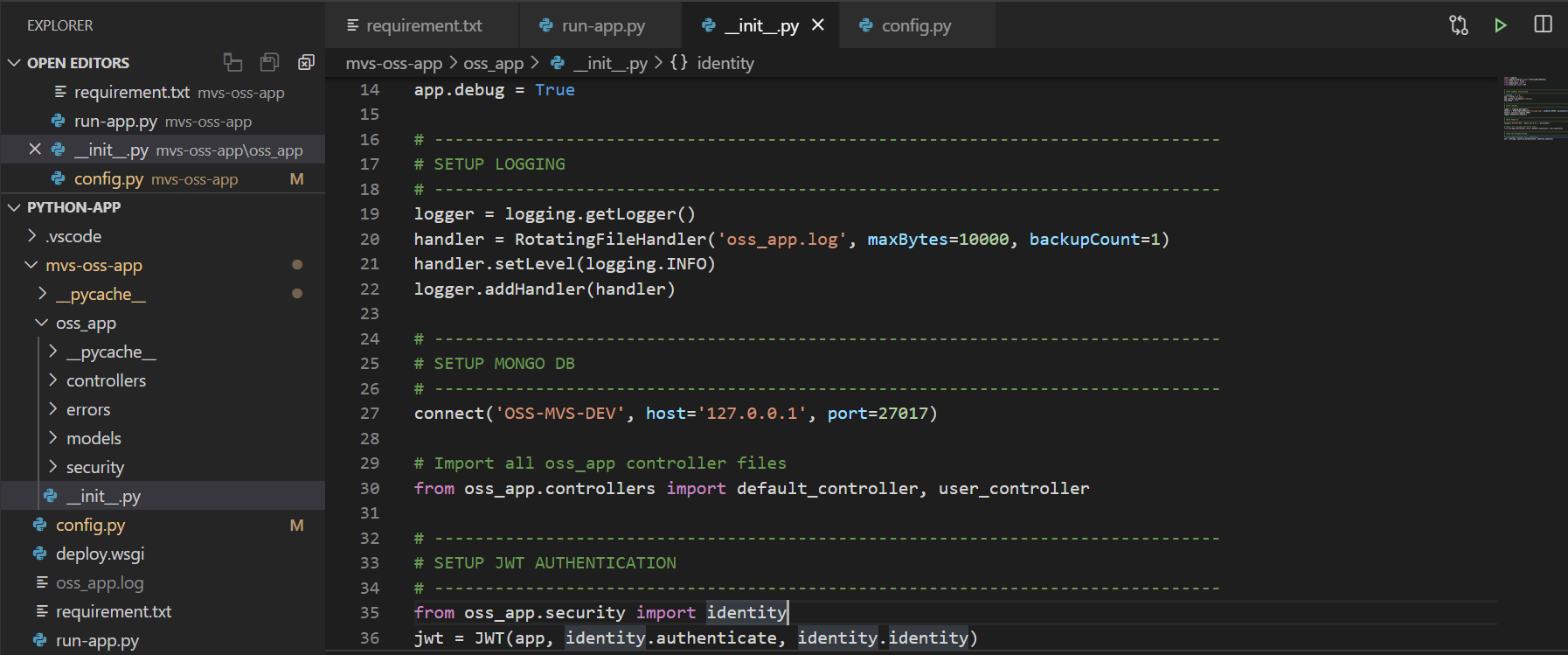
# Sub-directory & Sub-modules**:**

We have to create a sub-directory(oss-app) under main directory(mvs-oss-app). The directory structure should look like as shown in screenshot below:



## Create python application initializer:

Create python application initializer class and name it \_\_init\_\_.py under ‘oss-app’ sub-directory



## Create Rest Controller for CRUD operations:

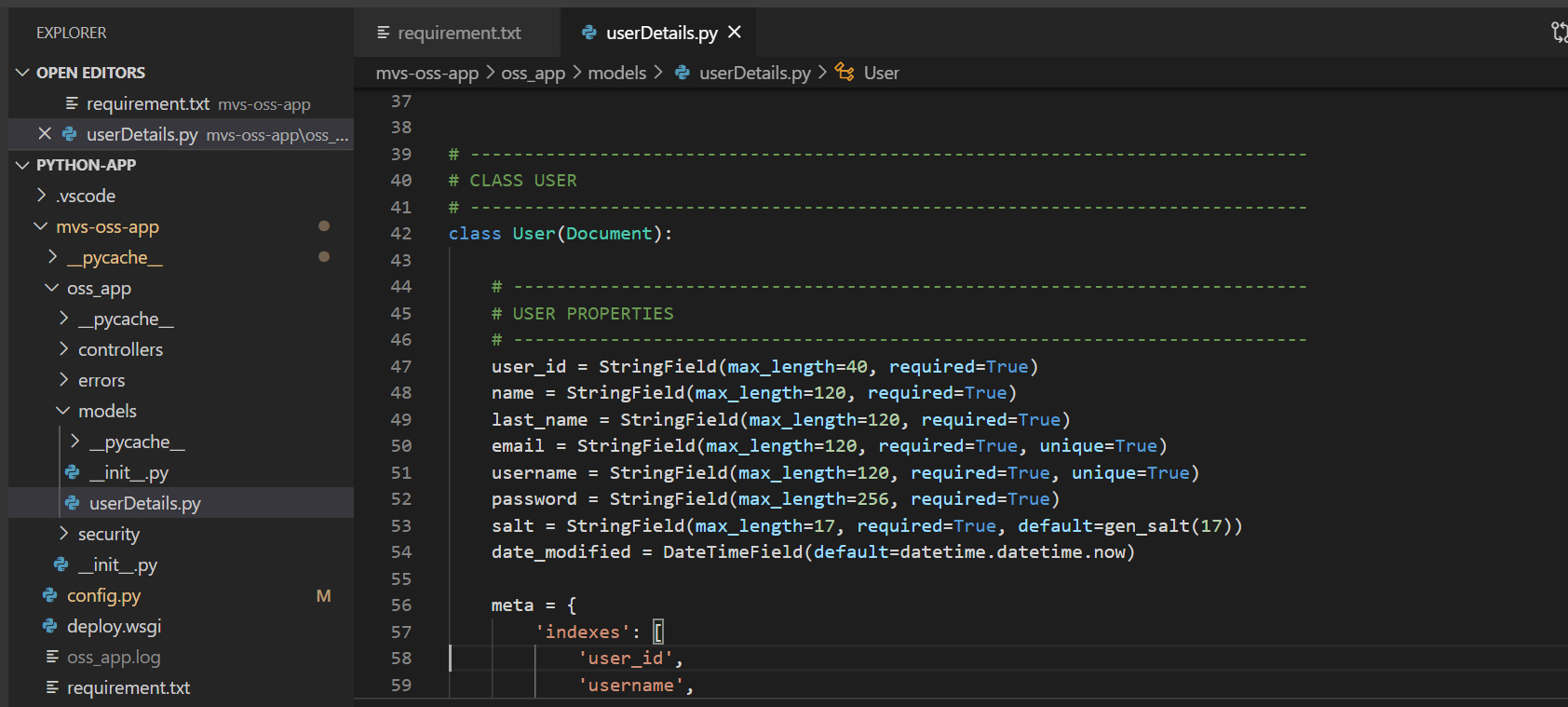
Create a Rest Controller class under ‘controller’ sub-directory and name it ‘user\_controller.py’





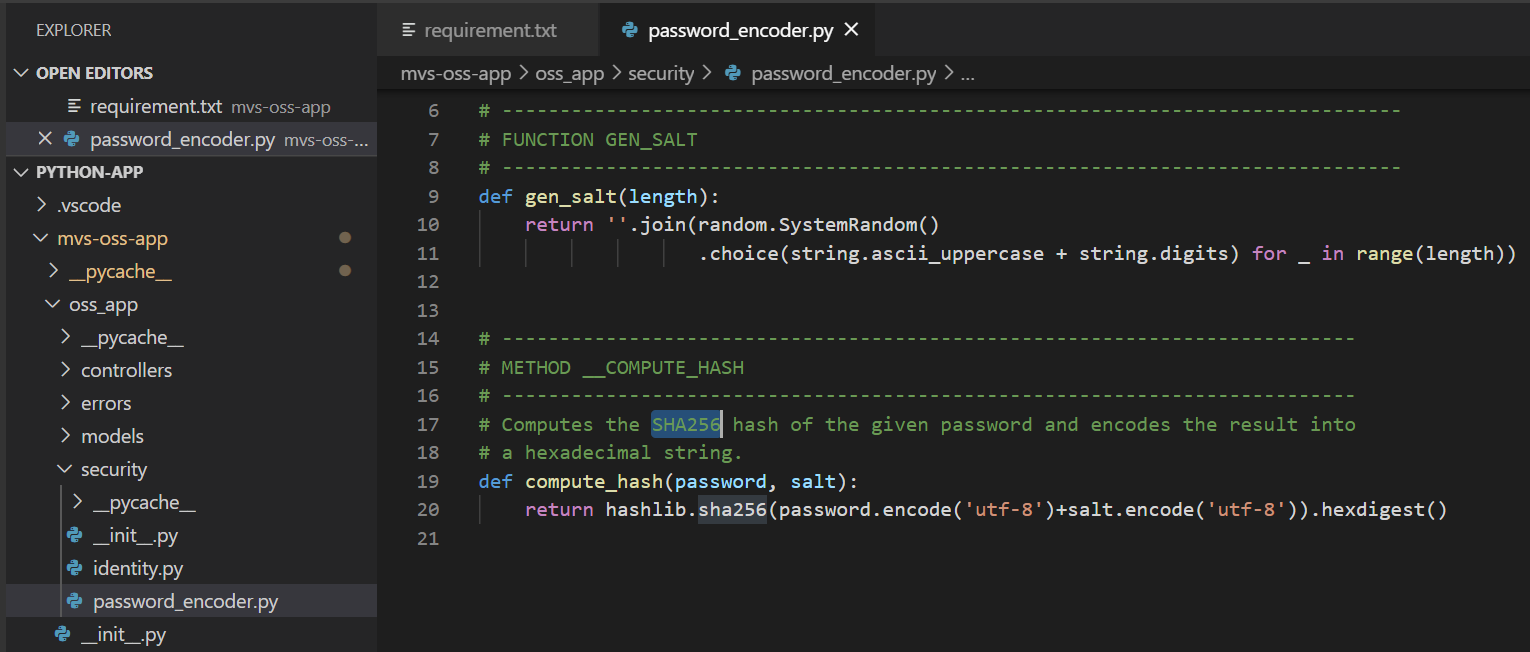
## Create model for MongoDB interaction:

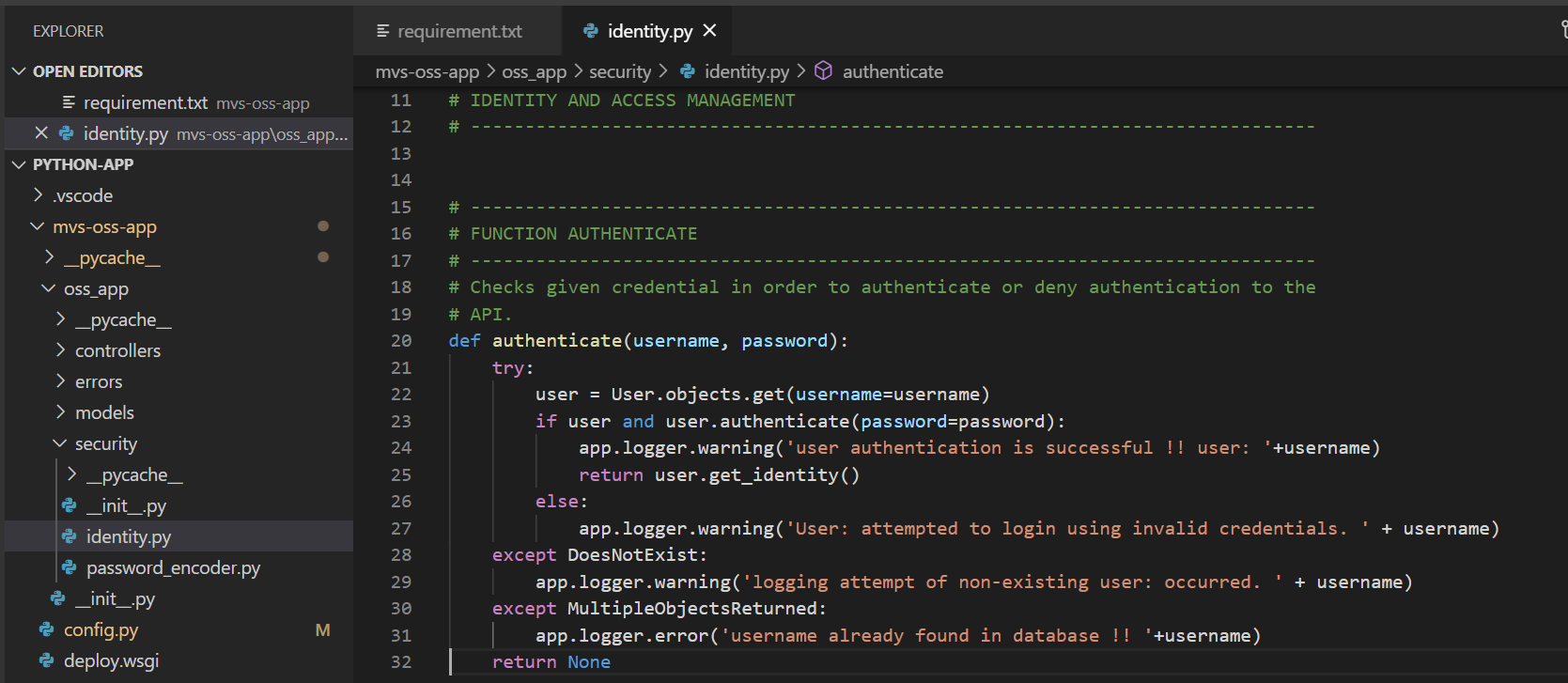
Create a model class and name it ‘userDetails.py’ under ‘models’ sub-directory



## Configure password security & JWT authentication:

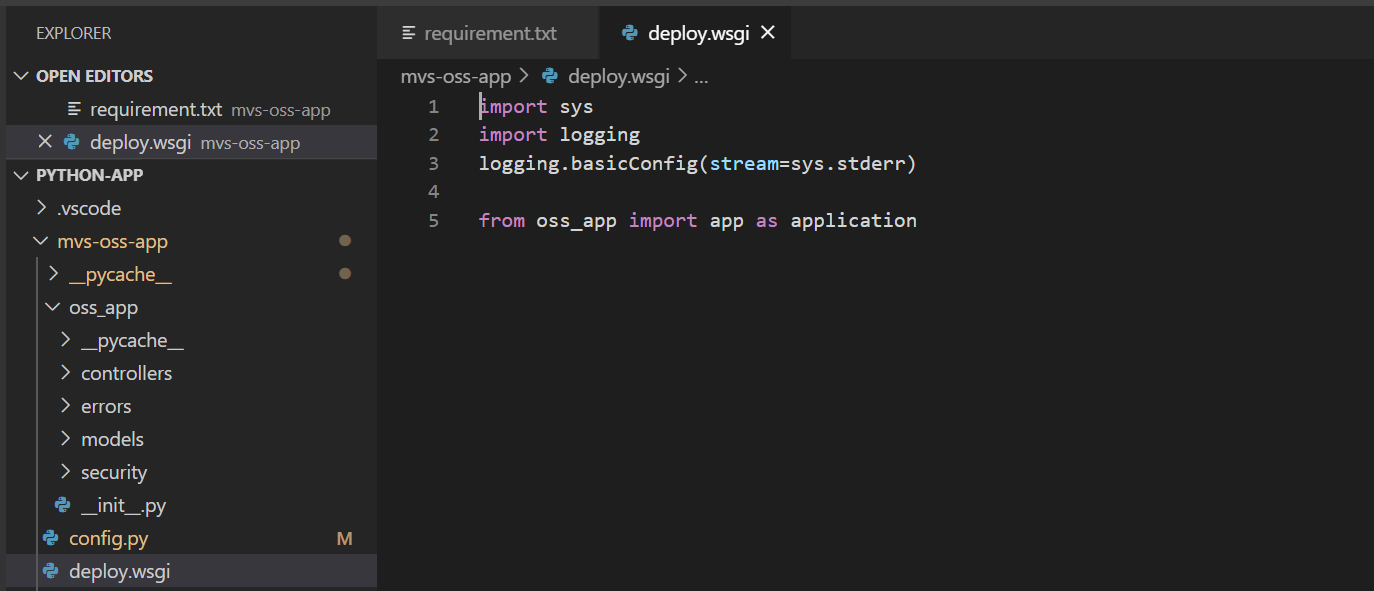
Create ‘password\_encoder.py’ class under ‘security’ sub-directory for password encryption based on Salt & SHA-256 algorithm. Create a separate class (identity.py) under same sub-directory for identity and access management based on JWT authentication.





## Create WSGI component:

Create a WSGI component(deploy.wsgi) on top of Flask Server to make it a production ready WSGI server. This extended feature can then be integrated with Web Servers (Apache, IIS, NGINX …) or even can be hosted to various Cloud platforms (AWS, GCP….)



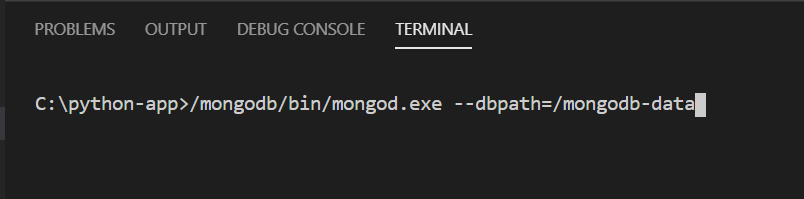
# Flask Server & MongoDB**:**

By default, Flask is registered at port 5000 and MongoDB is registered at 27017.

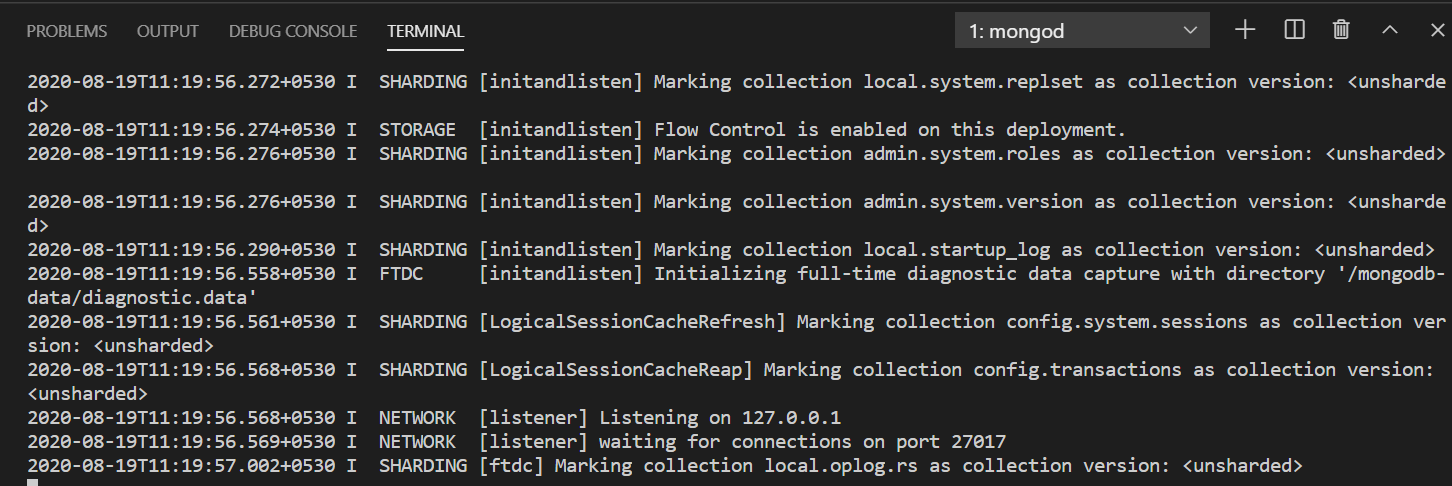
## Run MongoDB :

Open terminal window and enter below command :

/mongodb/bin/mongod.exe --dbpath=/mongodb-data



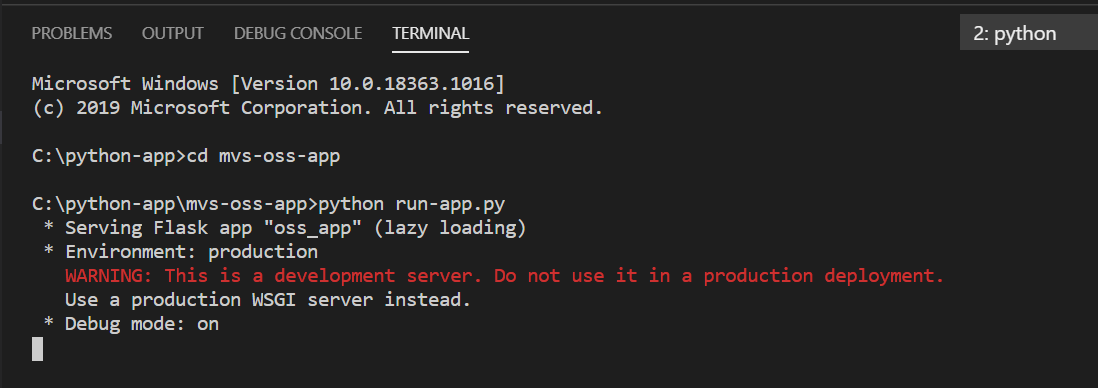
This command ‘ll make MongoDB is up and running at host - 127.0.0.1 and port – 27017. Database UI can be accessed with Robo 3T client.



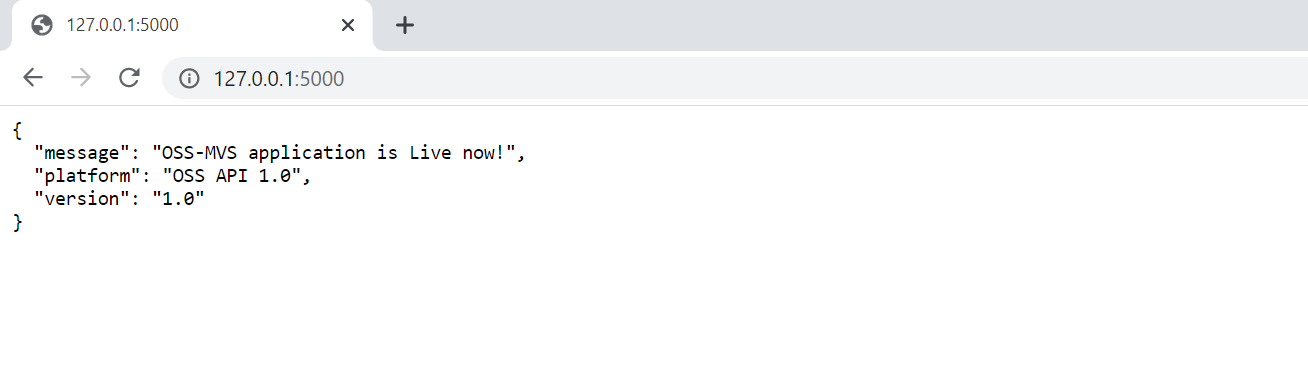
## Access Flask Server:

Open a new terminal window & execute the following command to start Flask Server:

python run-app.py

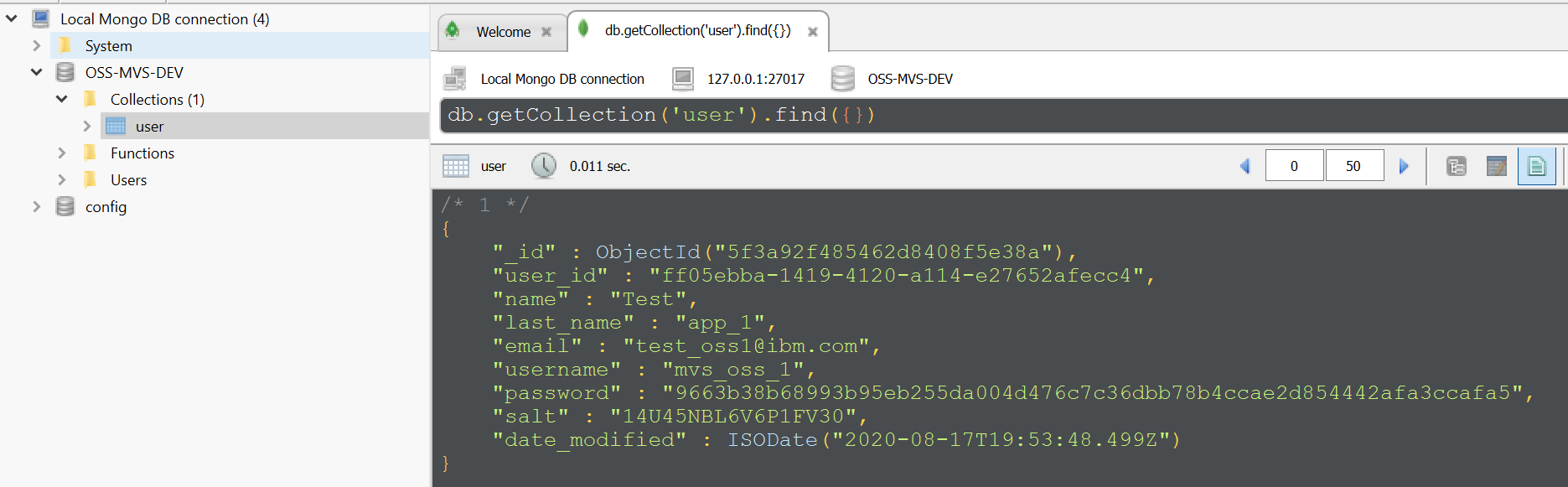


Flask is started at given port: 5000 and application get started. Now the application can be accessed through browser at <http://127.0.0.1:5000/> in up & running state.



## Access MongoDB using Robo 3T:

MongoDB can be accessed using Robo 3T client.

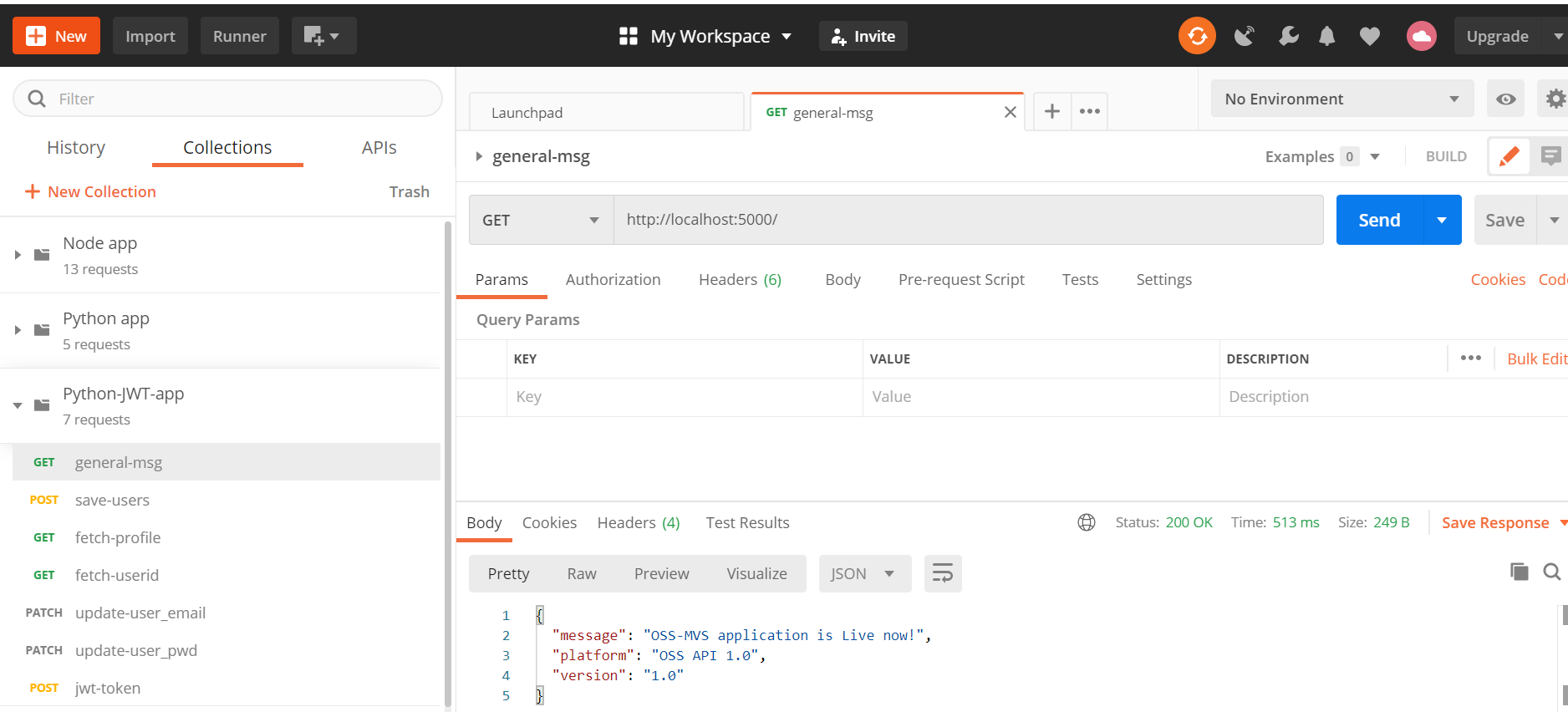


# Test Restful-Service:

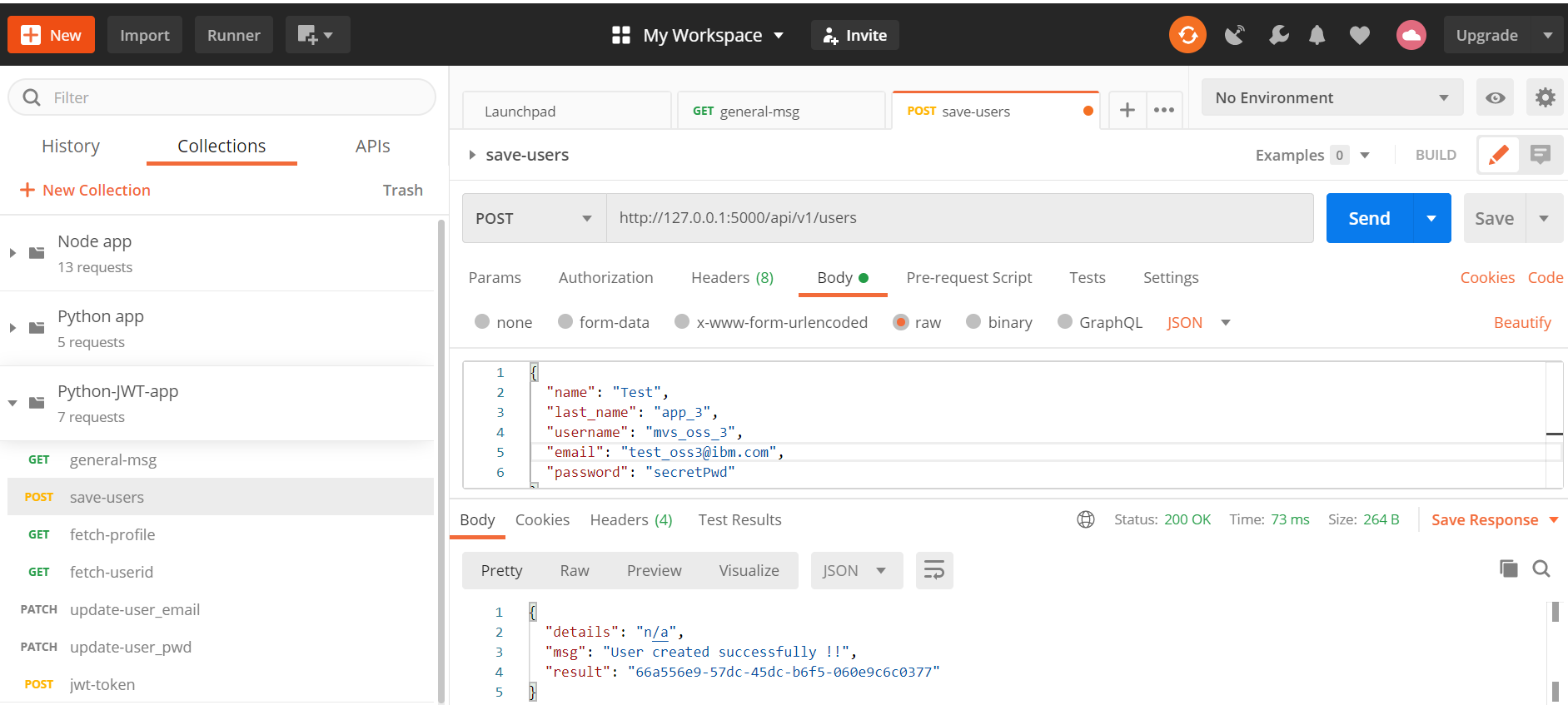
After deploying on Flask server, the application can be accessed by Postman for testing API.

## Test using Postman:

On Postman, create a new collection and under that collection create a new request as shown below and send the request to fetch response.

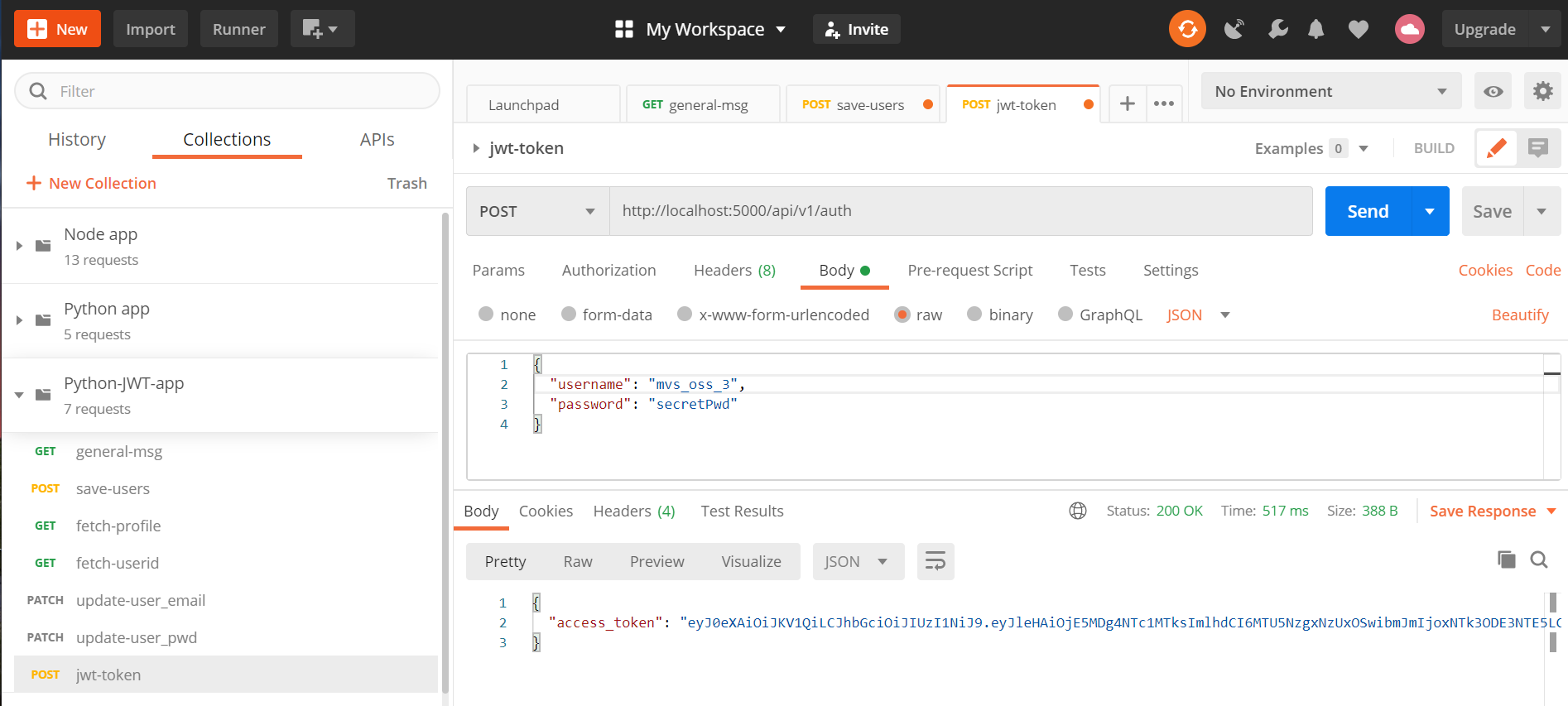


## Insert data into MongoDB



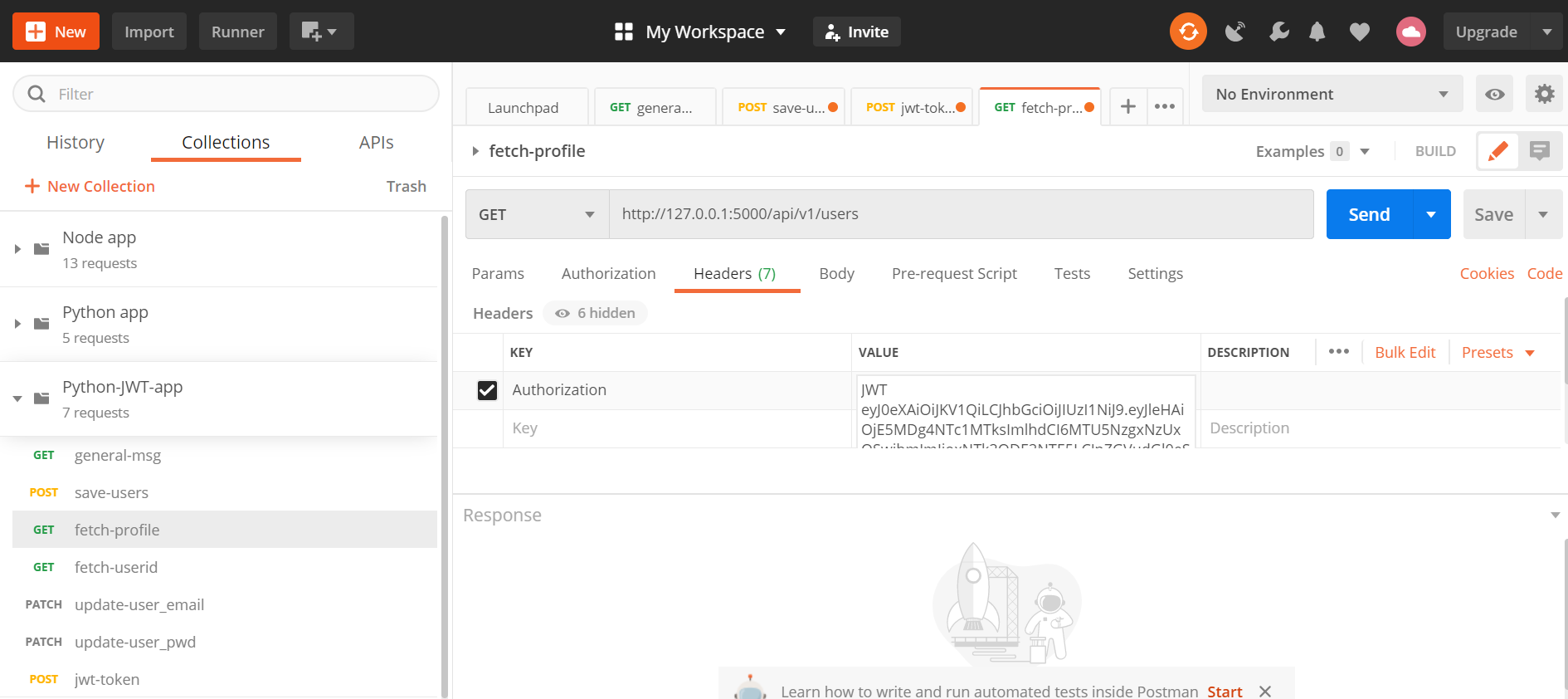
## Authenticate user by JWT:

Add request body with username and password as shown in screen shot:



## Add authorization token to request header :

Add token to request header and save the request body



## Access user session on JWT authentication:

