Design and approach Document

Introduction:

This document will give you the overall information on the design, selection of tools and packages used in different stages of API development.

Tools and packages used:

Python: 3.8.10

Flask: 2.0.2, Flask-HTTPAuth: 4.5.0

Database (MySQL): 8.0.23

Design:

**API:** The API is built in flask which provides data over HTTP using basic authentication in Flask. The input parameter to the API is **{event, from timestamp, to timestamp}**. The API is using MySQL databases to fetch data from “**events” table** and serve to the client.

Right now, only two users are defined for basic auth with hard-coded values in the code. I have created a table structure for users where password can be stored with md5 or similar digest method. You can use these credentials while consuming API.

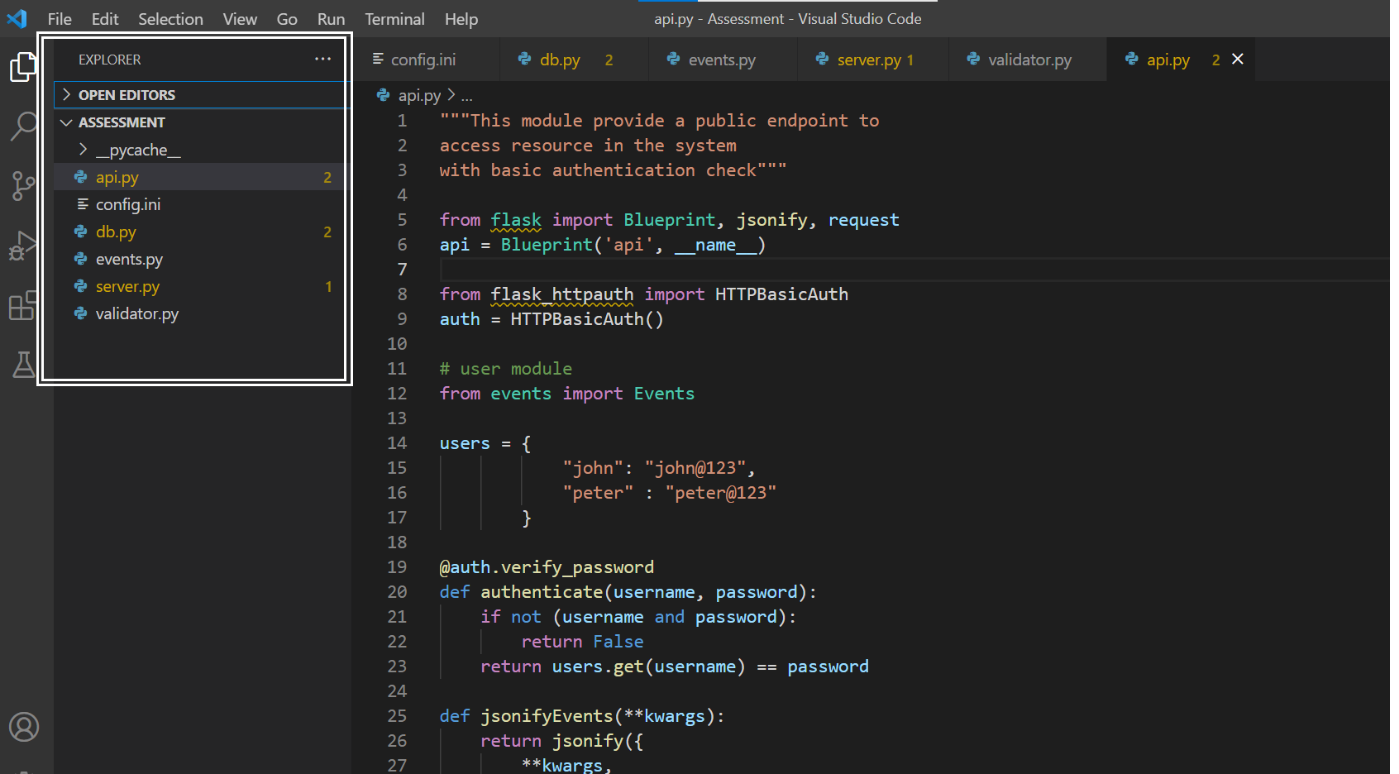
users = {

"john": "john@123",

"peter" : "peter@123"

}

Code Structure:



api.py: This module contains the endpoints defined to be accessed over HTTP. As of now authentication is also written here but it can be moved authentication.py later. The users are added in dictionary in the same file itself for basic auth. This function takes input parameters from API and send it to controller file which is **event.py**.

event.py: This module first validates the input parameters with the help of validator.py module and if all okay then calls the db.py function which query the data from database and return to it and in turn give it to api.py

validator.py: The validator function validates our input parameters, here it is checking if from\_timestamp and to\_timestamp is in “%Y-%m-%d %H:%M:%S” or not. If the format is not correct then sends the error message to the caller function.

db.py: This module interacts with the database which means it connect to the database, prepare the query and execute it and revert the query output to the caller function. “pymysql” package is used to connect to MySQL database server.

config.py: This file stores the common values which can be used across the project and it offers an advantage where in case of changes in values you do not need to go every function to search and update. I have provided database parameters, table name, column names which is getting used in the db.py and other files.

server.py: The app is initiated from here to separate the main code from server file.

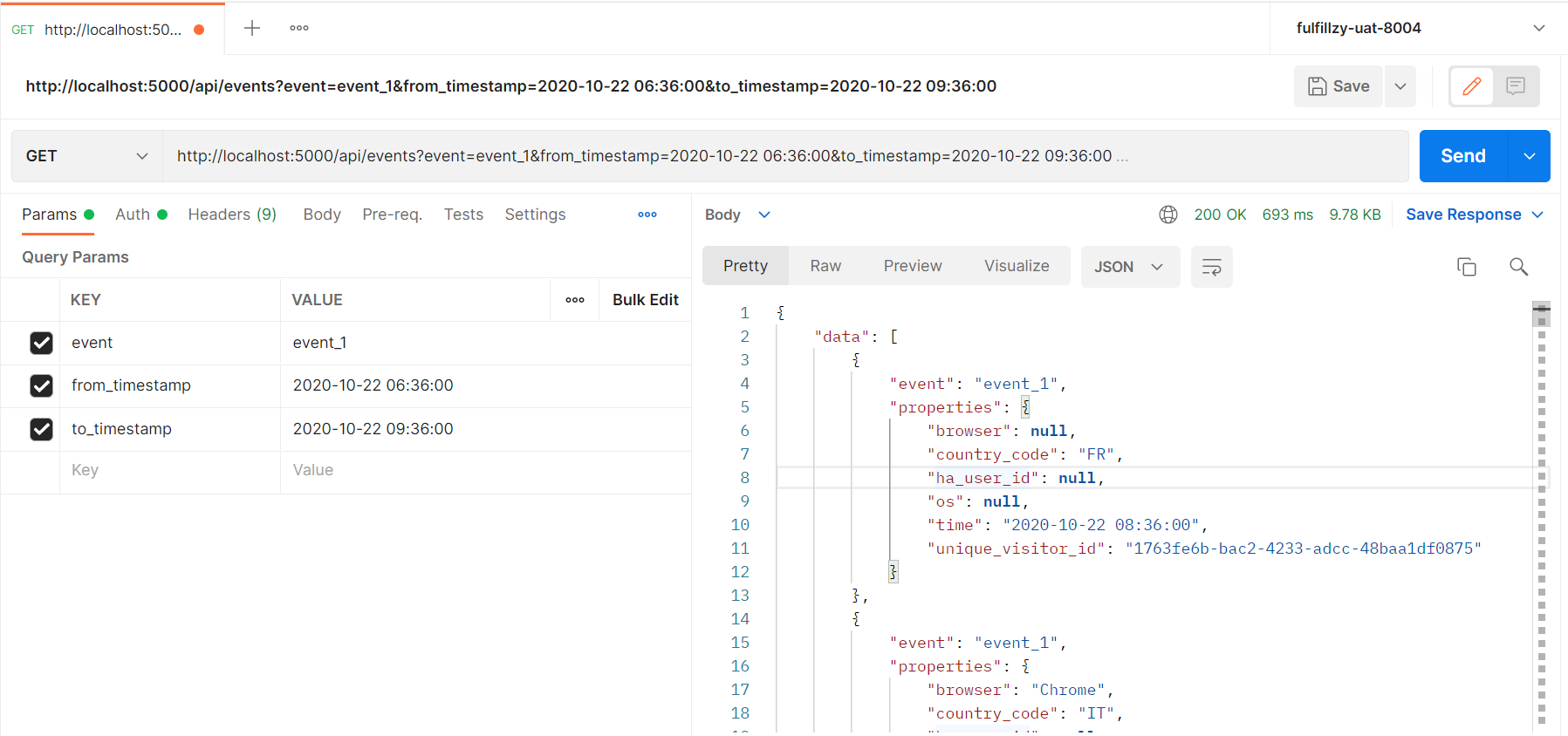
API Endpoint:

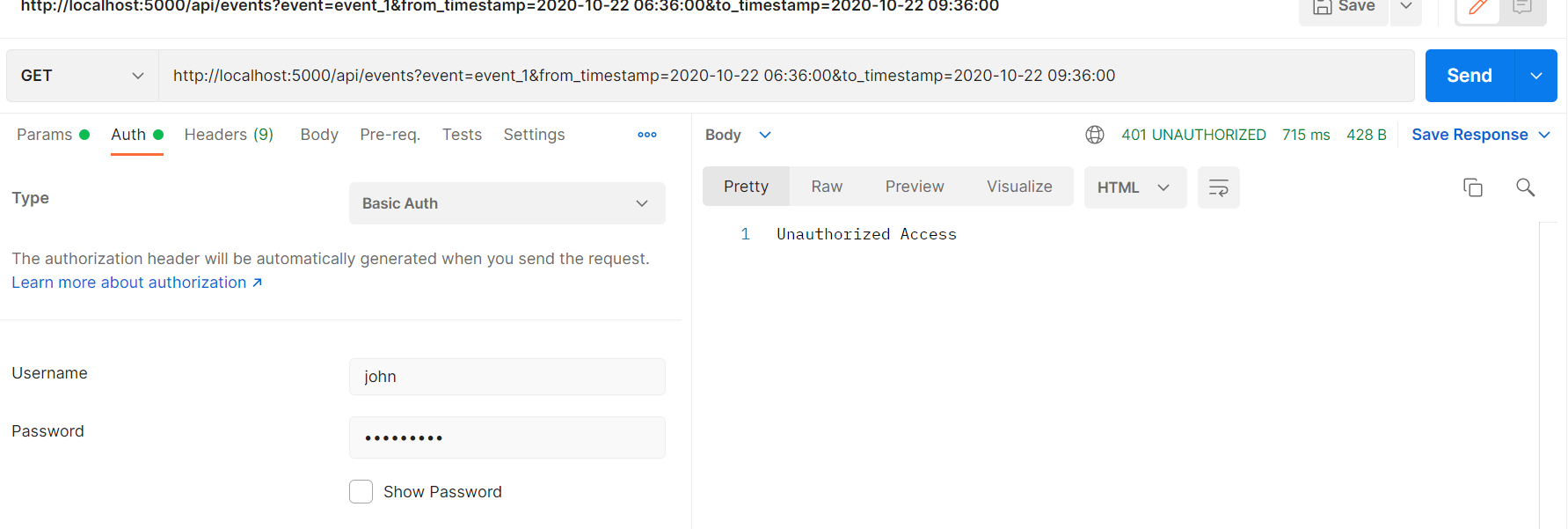
[http://localhost:5000/api/events?event={event}&from\_timestamp={from](http://localhost:5000/api/events?event=%7bevent%7d&from_timestamp=%7bfrom)\_timestamp}&to\_timestamp={to\_timestamp}

Content-Type: application/x-www-form-urlencoded

Method: GET

Auth: Basic Auth



****

**Decision on selection of tools:**

Flask:

I have used Flask because of it offers great flexibility in creating microservices and provides easy integration with different adapters, data sources and data bases. If we need concurrency then definitely, we should go for Fast API.

MySQL:

MySQL is great tool for read and write I/O throughput. MySQL is quite fast in querying the data.

Note: I can speak more about on the decision part but for the document I am making it short.