Python Real World Challenge using Flask

Before starting, keep following things in mind:
1. Write a well structured, formatted and commented code (according to best practices).
2. All of the code should be in a public github repository.
3. The git commit history should reflect the steps you took to solve the problem.
4. A report explaining your code structure. Any problems faced should also be included in the report.
5. After completing the task, please send an email mentioning the public github repo link and the report attached.

Requirements:
You have been given a real world challenge to generate reports on registration of cars. Let's suppose a small startup wants to build this app which would help the end user to search and view the reports generated. Based on this use case, you got the opportunity to join this startup as a backend Intern. You must provide clear and complete documentation about how to run your program. You should be able to handle different routes in your app. The following features
Restful APIs implementation is given to you as a first task to implement.
Dataset:
https://www.back4app.com/database/back4app/car-make-model-dataset/get-started/python/rest-api/requests
You can generate new access credentials for your application as per specified in the documentation provided with the link OR you can use the following headers to access the dataset in your application:
'X-Parse-Application-Id':'gP38fEGPgSSBvvO4Kz9McQD2UpUrcpIlrXDyHLWc'
'X-Parse-REST-API-Key': '72gJMaTFClPr90oA7bkRYdUy0PJIcKQ8tj8bQvtP'
Please use the Flask framework to achieve the functionality of a web server.

1. SignUp/ Login Functionality: A user should be able to register into your application, and a
registered user should be able to login to your application.
2. Periodic Sync of Dataset: Using the URL provided above, make automated calls once a day
to retrieve and store data into a local relational database. You only need to maintain a data set for
the last 10 years i.e. 2012-2022. This operation should be performed as a background task. Keep
in mind that discovered data should only update & not overwrite the current data stored. (HINT:
you can use Celery for Background processing)
3. Search Functionality: Write down an API call to get the reports generated. This API should
be able to filter the result and retrieve the reports based on date. For this use case, a user should
be able to query the car dataset based on make model and make year.
4. Schema Validation: The APIs should be able to properly validate input/output schemas.
(HINT: you can use Marshmallow for validating schemas)
Notes:
● Please make sure to use PAGINATION where necessary.
● Follow Database NORMALIZATION concepts as needed.
● Make sure to VERIFY and VALIDATE users on authentication.
● Follow BEST practices while implementing this application
Testing:
You can use POSTman or any other REST client to test out the functionality of your app. You do
not need frontend/GUI for your app.