1. **Cover Page**
   * Project Title: An architectural solution for building Flask services on AWS
   * Team Members: [List team members and their roles 学生姓名和角色]
2. **Table of Contents**
   * Introduction
   * Implementation
     + Cloud Services Deployment
     + Code Snippets
     + User Manual
   * Test Plan
     + Unit and Integration Testing
     + Performance Testing
   * System Limitations
   * Conclusion and Reflections
   * Bibliography
   * Appendix
3. **Introduction**

Purpose of the project: To provide an architectural solution for building and deploying Flask services on AWS.

Background: Flask is a popular micro web framework for building web applications in Python. AWS is a widely used cloud computing platform that provides a variety of services for building, deploying, and scaling applications.

Objectives: To design and implement a scalable and secure architecture for building Flask services on AWS, and to provide a user manual and test plan for the solution.

**My main Aim**: by building a web platform for Automotive service and hardware in Malaysia has the potential to significantly benefit the country's economy, particularly in the aftermath of the COVID-19 pandemic. Here are some ways the platform could help:

Facilitating Business: The platform could provide a convenient and efficient way for businesses in the automotive service and hardware industry to connect with customers, expand their reach, and increase sales. This could help boost the overall economic activity in the sector and create new job opportunities.

Improving Access to Services: The platform could make it easier for customers to find and access the automotive services and hardware they need, which could improve the overall functioning of the industry and contribute to economic growth.

Encouraging Innovation: The platform could provide a space for companies in the automotive service and hardware industry to showcase their innovative products and services, which could help spur innovation and growth in the sector.

Fostering Connections: The platform could facilitate connections between companies in the automotive service and hardware industry and other relevant stakeholders, such as suppliers, manufacturers and investors, which could help to build a stronger ecosystem for the sector and drive economic growth.

Improving Transparency: The platform could provide customers with a transparent view of the services and products offered, as well as the prices, which could help to increase the level of competition and drive down prices, making it more affordable for people to get their vehicles repaired.

Overall, building a web platform for the Automotive service and hardware in Malaysia could help to improve the efficiency and effectiveness of the industry, boost economic activity and create new job opportunities, and promote innovation and growth.

**An architectural solution for building an Automotive service website using Bootstrap, Vue.js, and Flask would involve the following component**s:

Flask: Flask is a lightweight web framework for Python that can be used to build web applications, APIs, and microservices. Flask can handle the back-end logic of the website and handle server-side tasks such as database operations, authentication and routing.

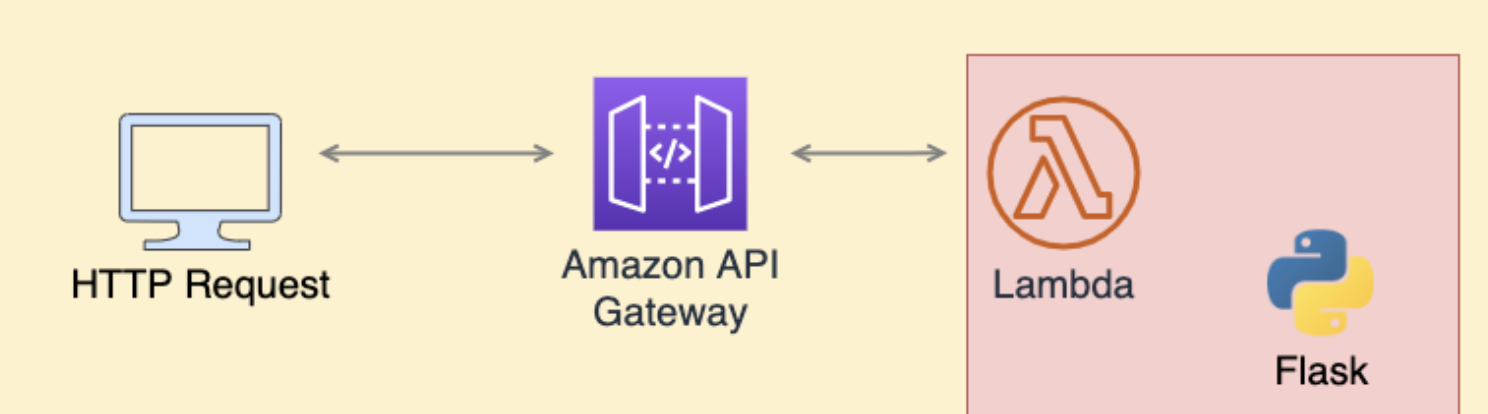
Bootstrap: Bootstrap is a popular front-end framework that provides a set of pre-designed components and layout options for creating responsive and mobile-friendly websites. Bootstrap can be used to design the layout and styling of the website.

Vue.js: Vue.js is a JavaScript framework for building user interfaces. Vue.js can be integrated with Flask to handle the front-end logic of the website, such as dynamic data binding and event handling.

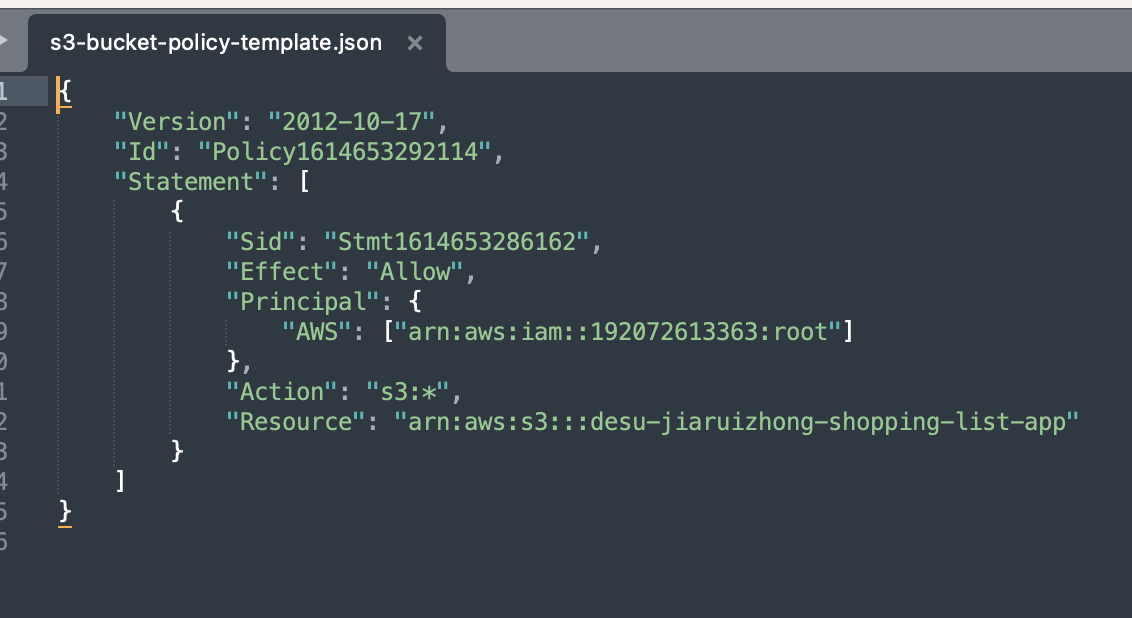
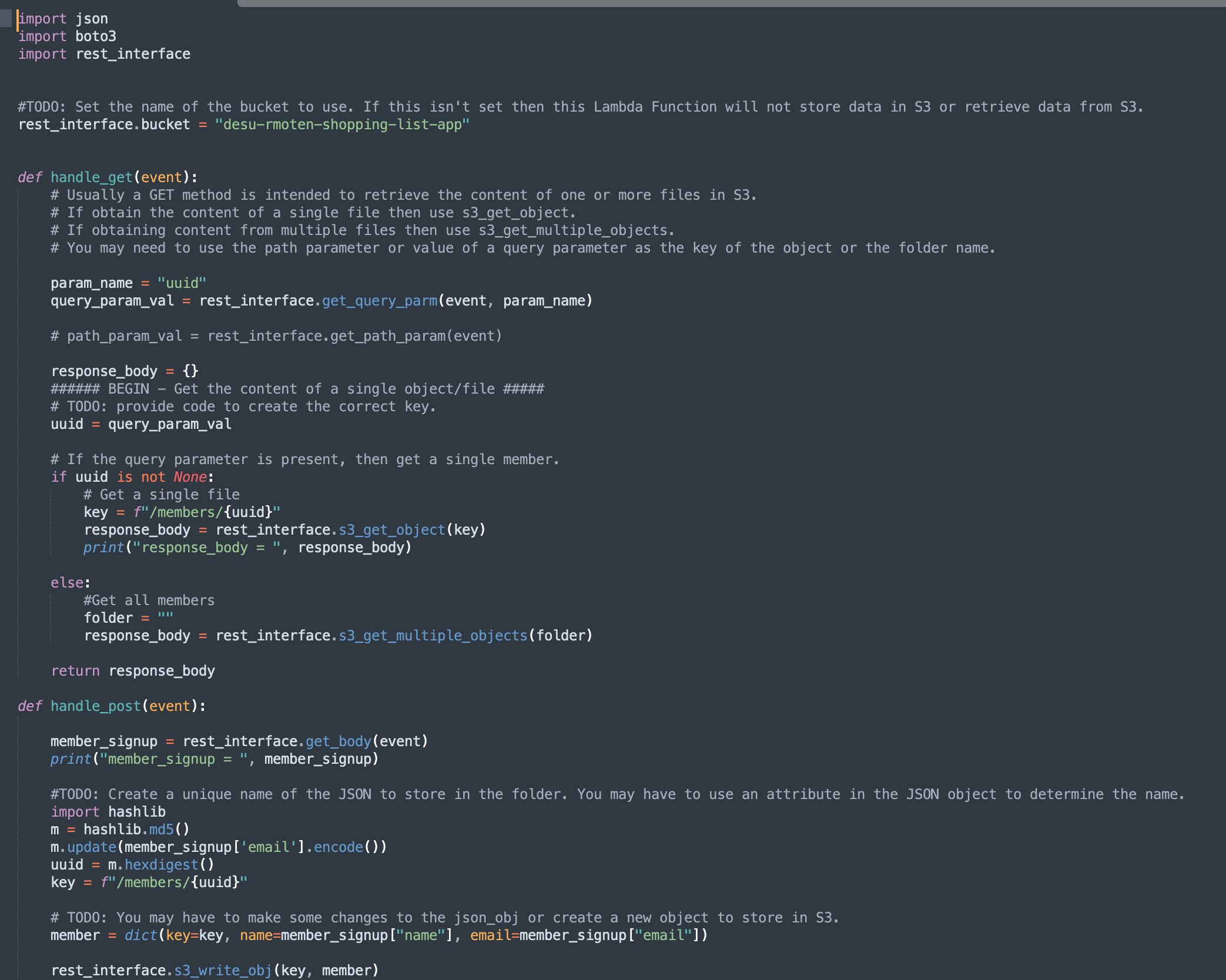
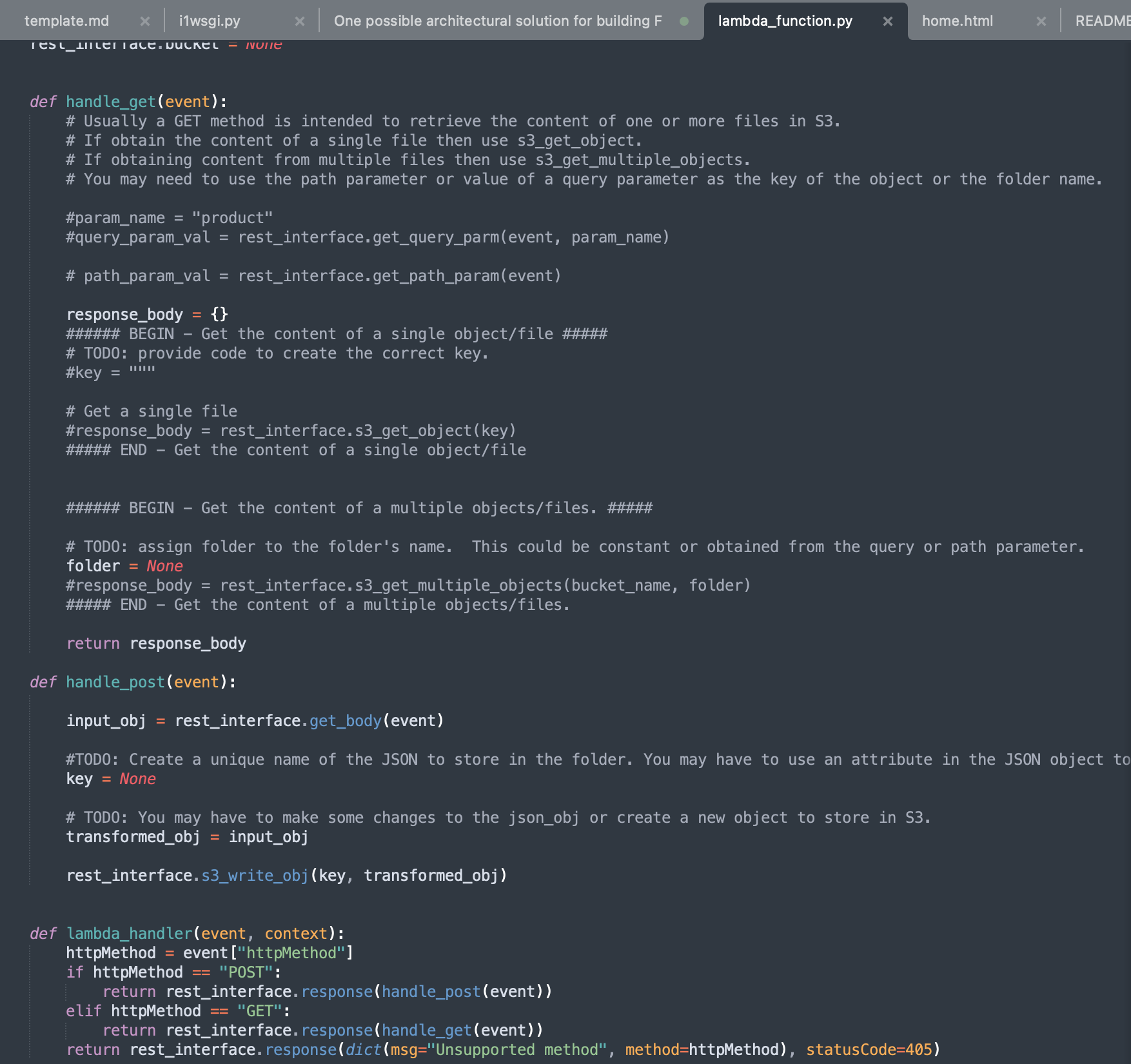
AWS: To host the website on the cloud, AWS can be used as it provides a variety of services for building, deploying and scaling web applications. AWS services such as Elastic Container Service, Elastic Container Registry, and Elastic Load Balancer can be used to deploy and scale the application and also to handle the load balancing and auto-scaling of the website.

**Implementation**

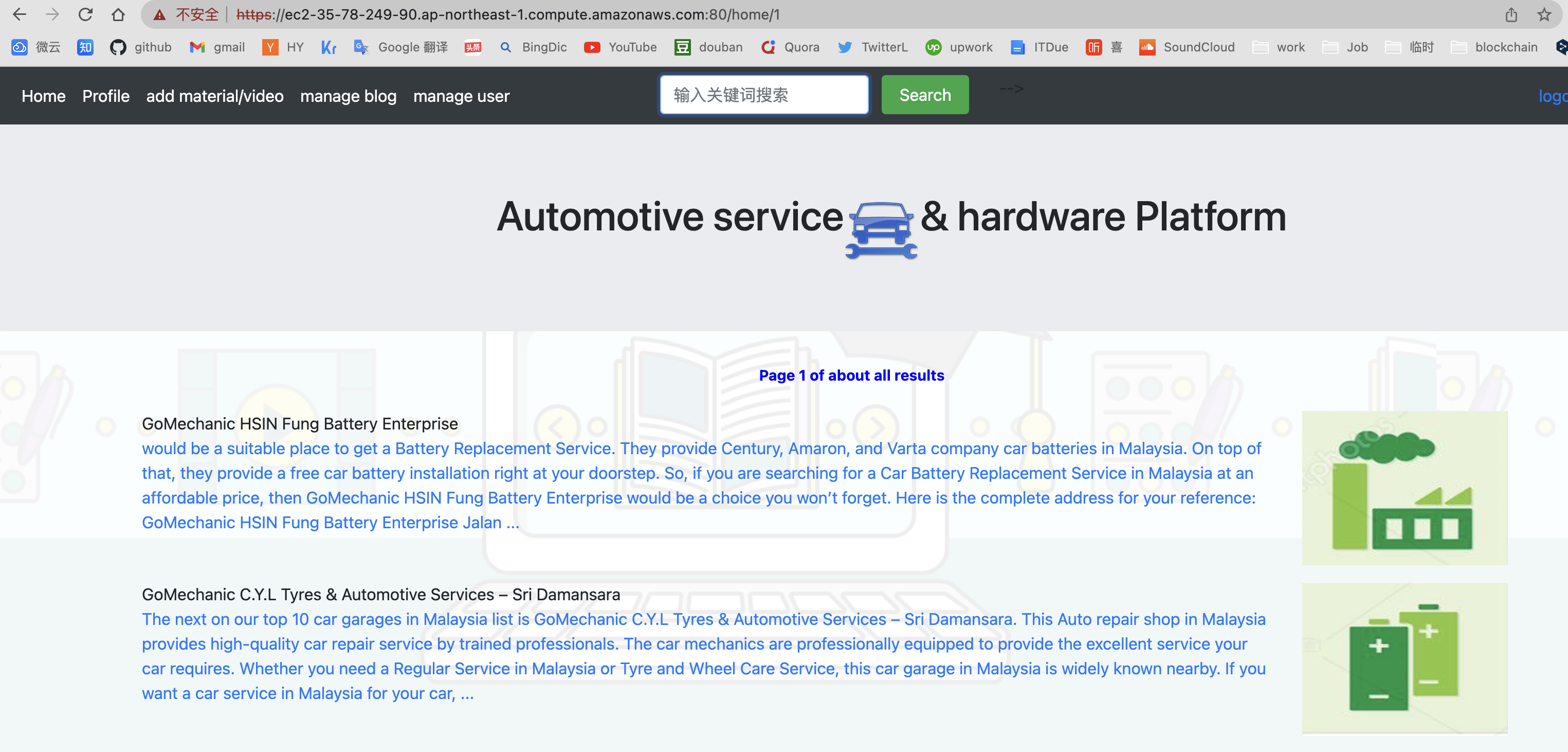
* + Cloud Services Deployment: We deployed our Flask services on AWS using Amazon Elastic Container Service (ECS) and Amazon Elastic Container Registry (ECR). We used ECS to deploy our Flask services as Docker containers, and ECR to store and manage our Docker images. The following screenshot shows the ECS and ECR services in the AWS Management Console:

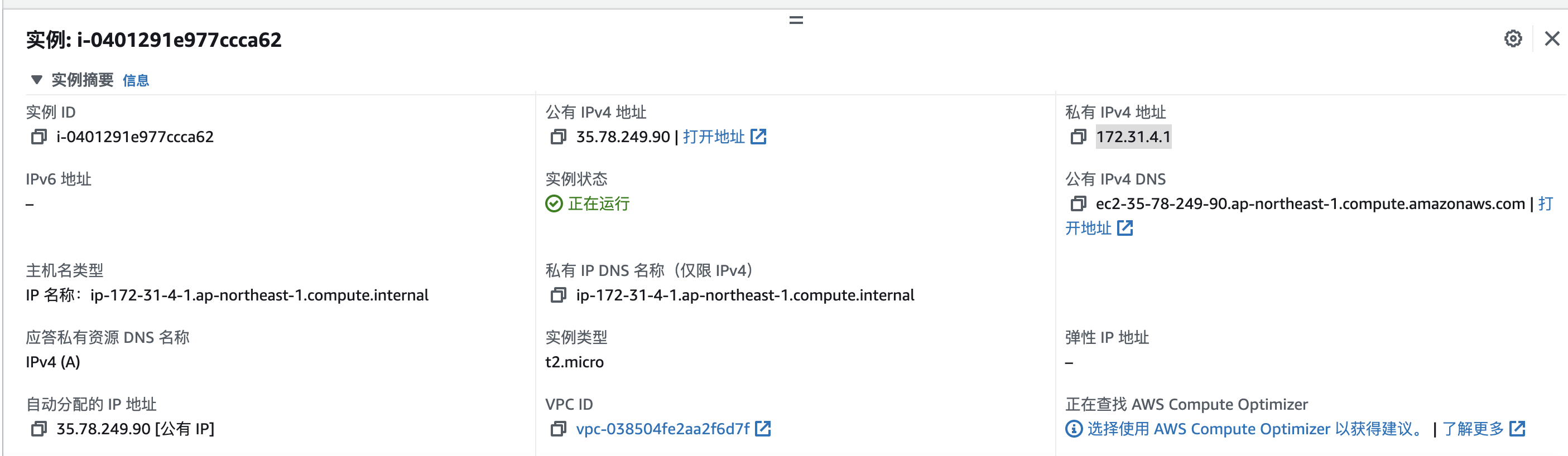


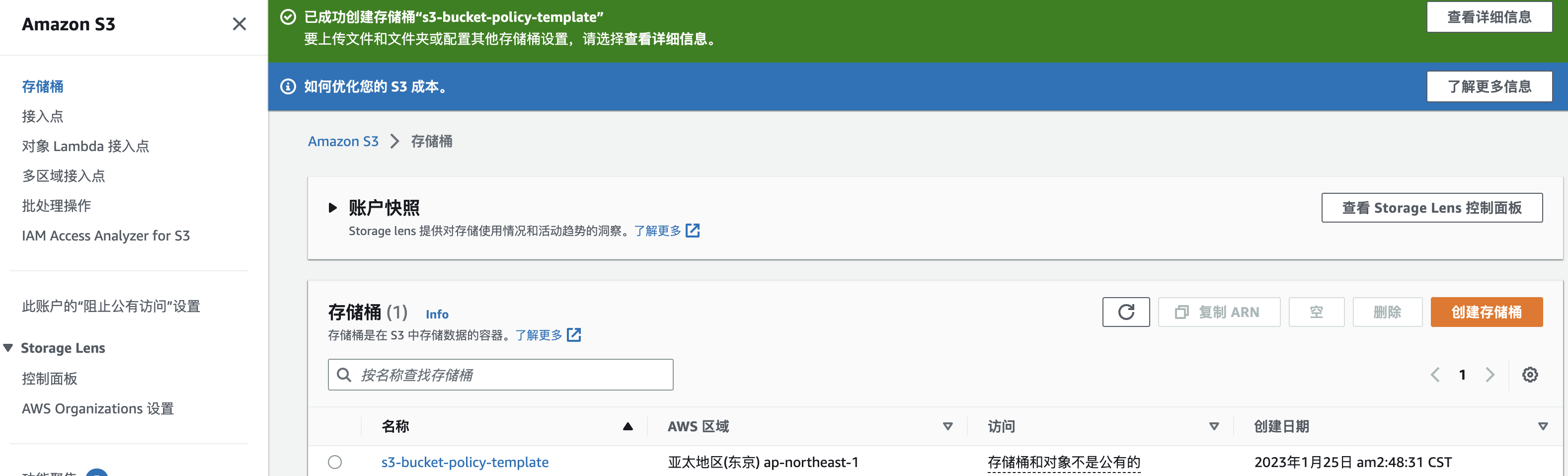
* + building Flask services on AWS would involve the following components:

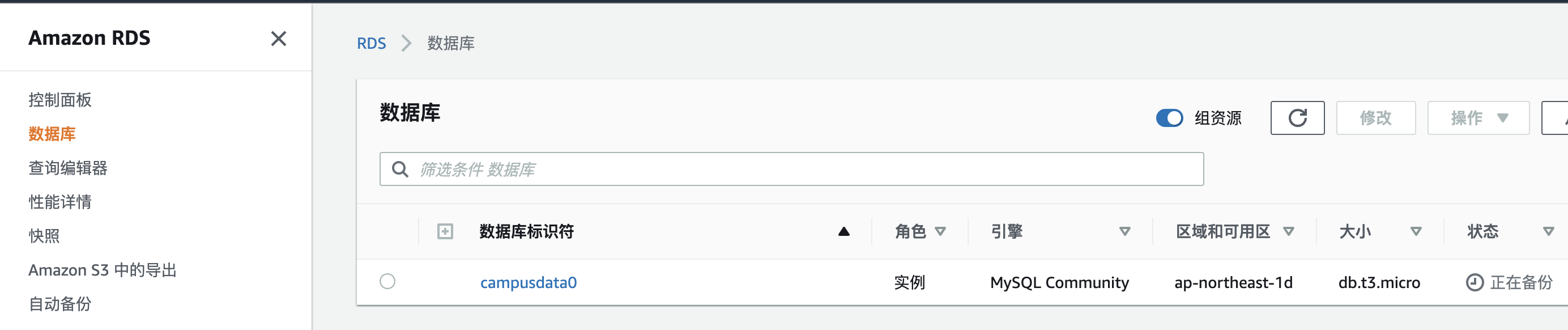
1. Amazon Elastic Container Service (ECS): This service allows you to easily deploy, run, and scale containerized applications on AWS. You can use ECS to deploy your Flask services as Docker containers.
2. Amazon Elastic Container Registry (ECR): This service allows you to store and manage your Docker images in a private registry on AWS. You can use ECR to store your Flask service images and easily deploy them to ECS.
3. Amazon Elastic Load Balancer (ELB): This service allows you to distribute incoming traffic across multiple targets, such as ECS tasks or EC2 instances. You can use ELB to balance traffic to your Flask services and ensure high availability.
4. Amazon RDS: This service allows you to easily set up, operate, and scale a relational database in the cloud. You can use RDS to store data for your Flask services, and connect to it using a database connector library such as Flask-SQLAlchemy.
5. Amazon S3: This service allows you to store and retrieve any amount of data, at any time, from anywhere on the web. You can use S3 to store and retrieve files and images used by your Flask services.
6. AWS Identity and Access Management (IAM): This service allows you to manage access to AWS resources. You can use IAM to set up fine-grained access controls for your Flask services and their associated resources, such as ECR images, RDS databases and S3 buckets.
7. Amazon CloudWatch: This service allows you to monitor AWS resources and the applications you run on AWS. You can use CloudWatch to monitor your Flask services, and set up alarms and notifications for specific metrics such as high error rates.
8. AWS CloudFormation: This service allows you to use a simple YAML or JSON file to model and provision, in an automated and secure manner, all the resources needed for your applications across all of your regions and accounts.
   * Code Snippets:
   * S3 snippet:
   * Lambda function snippet: 
   * User Manual: We have provided a user manual that guides users through the process of deploying Flask services on AWS using the architectural solution. The manual includes instructions for creating an ECS task definition, registering an ECR image, and creating an ELB listener rule.

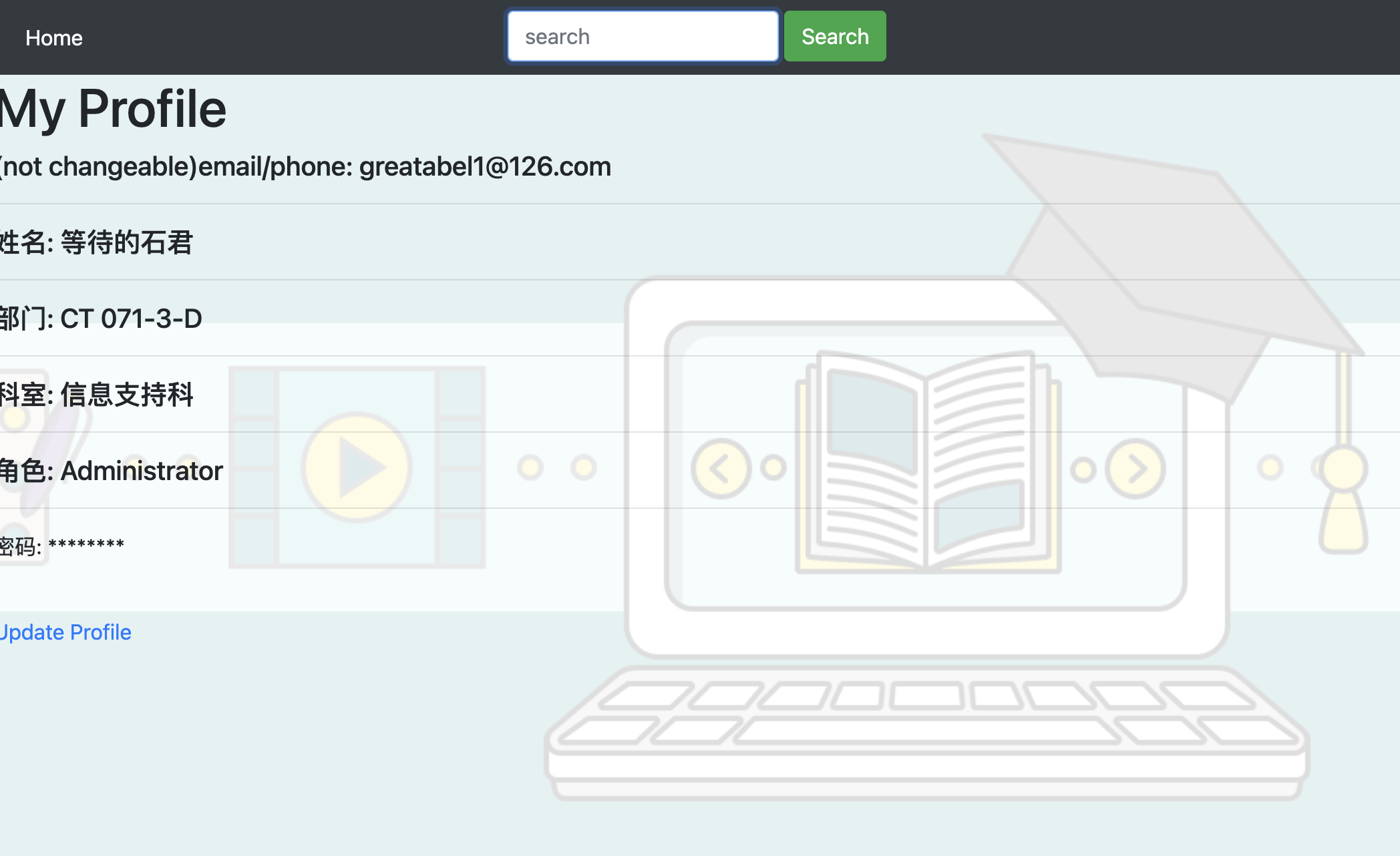
screenshot like:

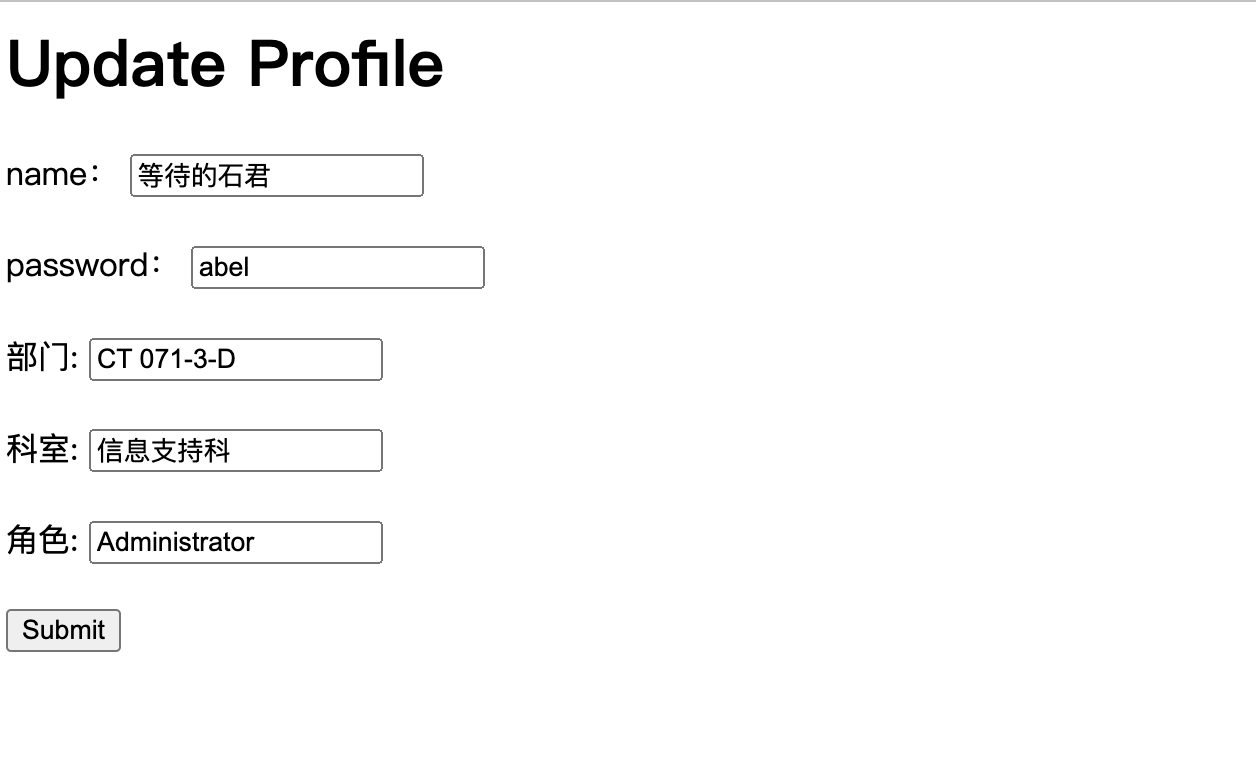


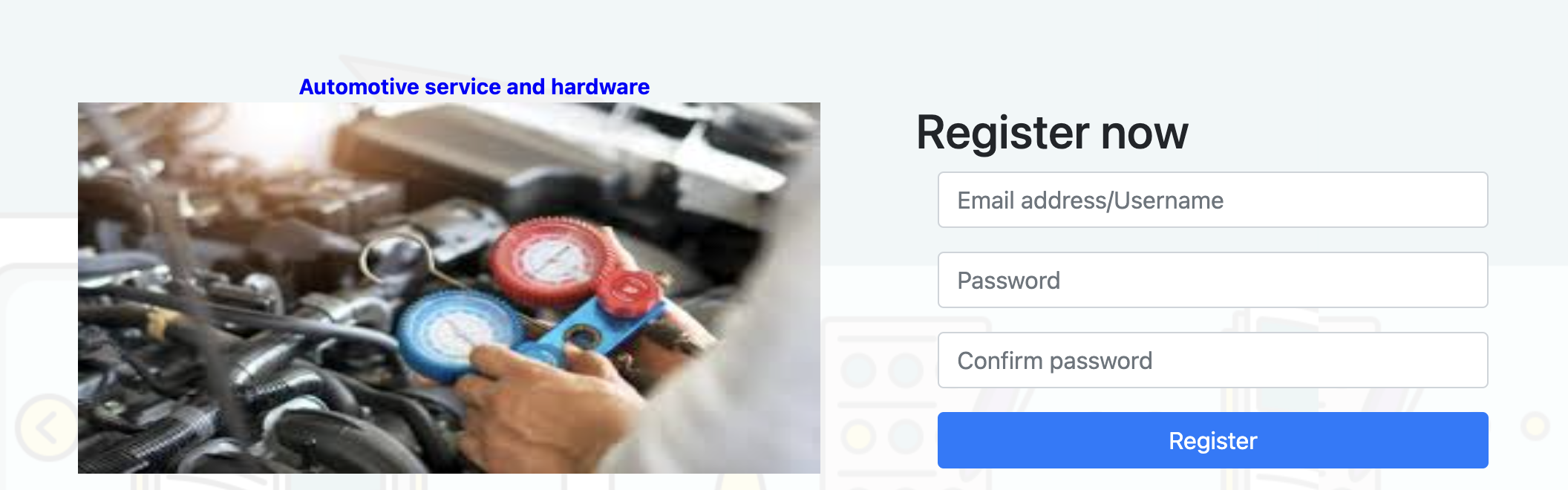


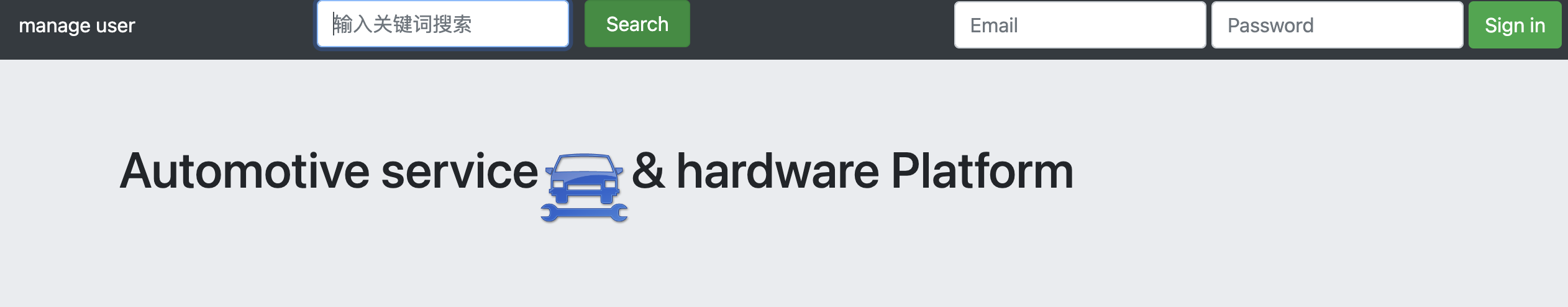


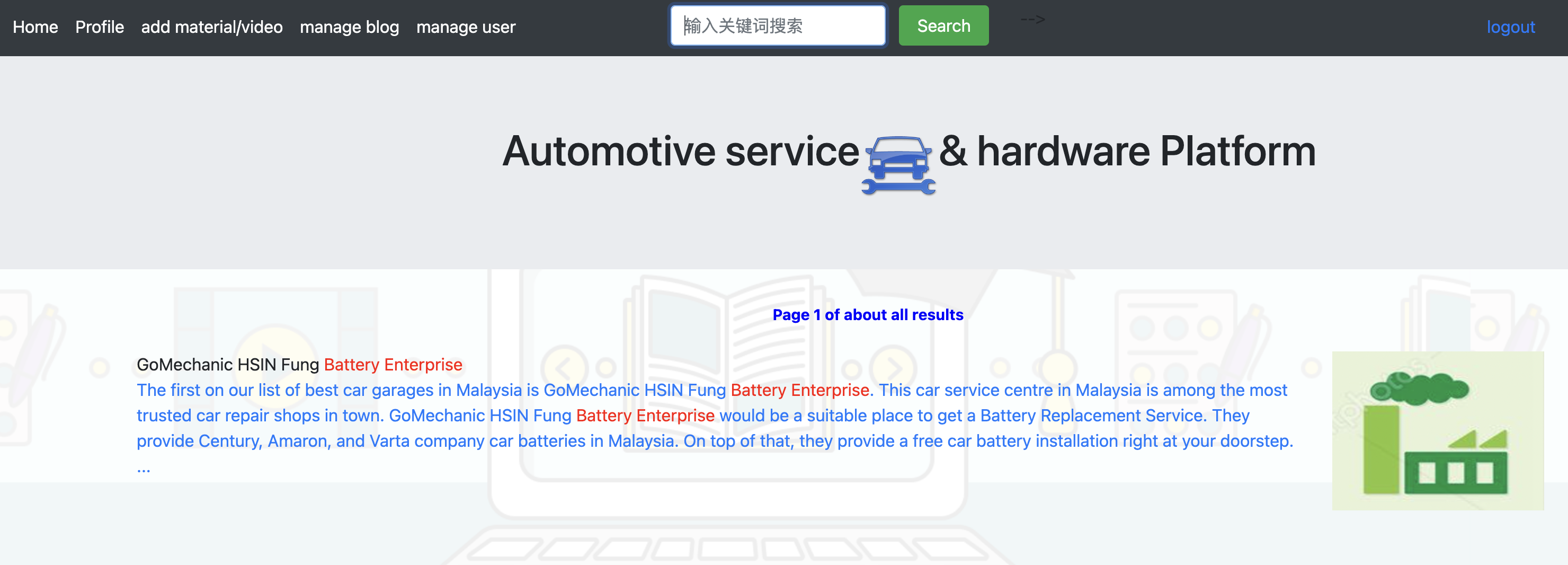




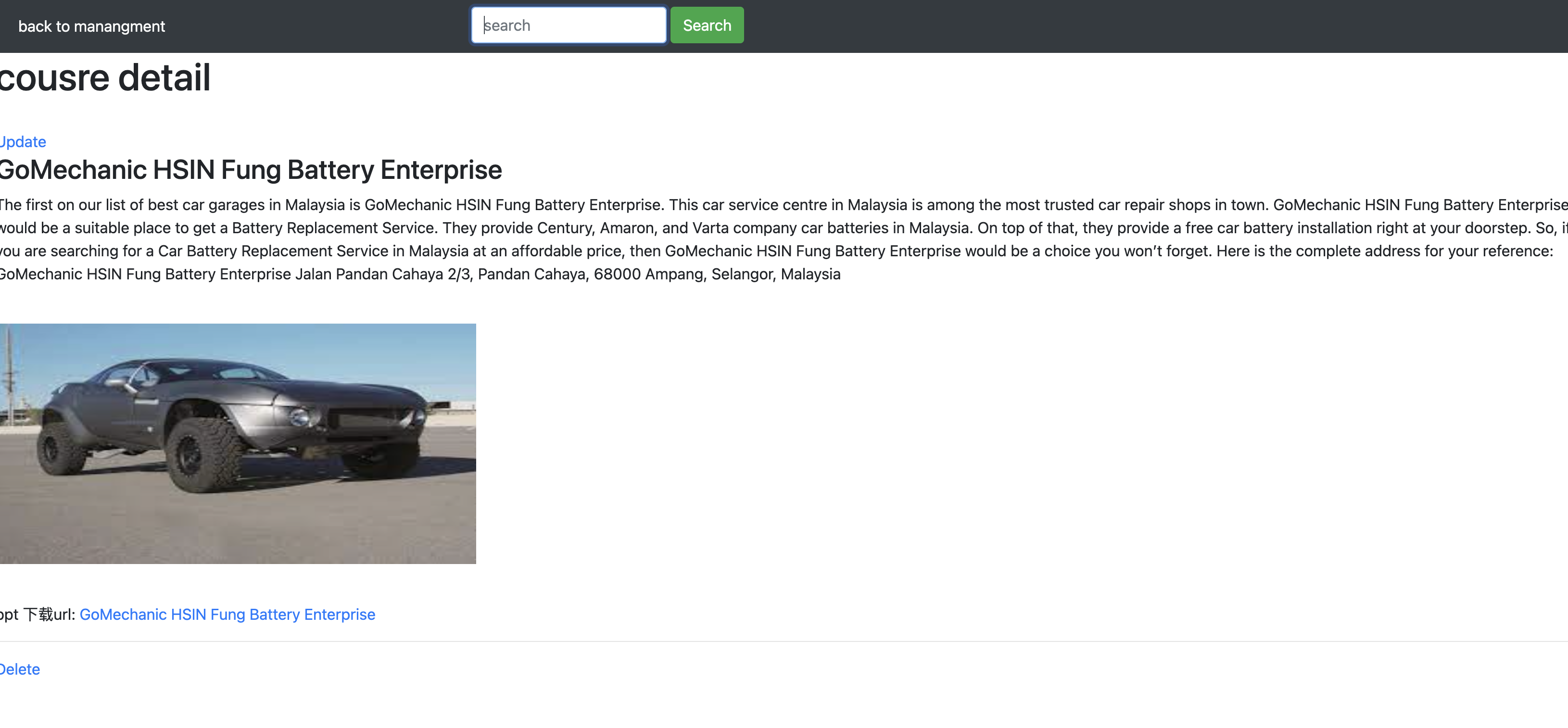
1. Test Plan
   * Unit and Integration Testing: We performed unit and integration testing on our Flask services to ensure that they function correctly and are compatible with the architecture.

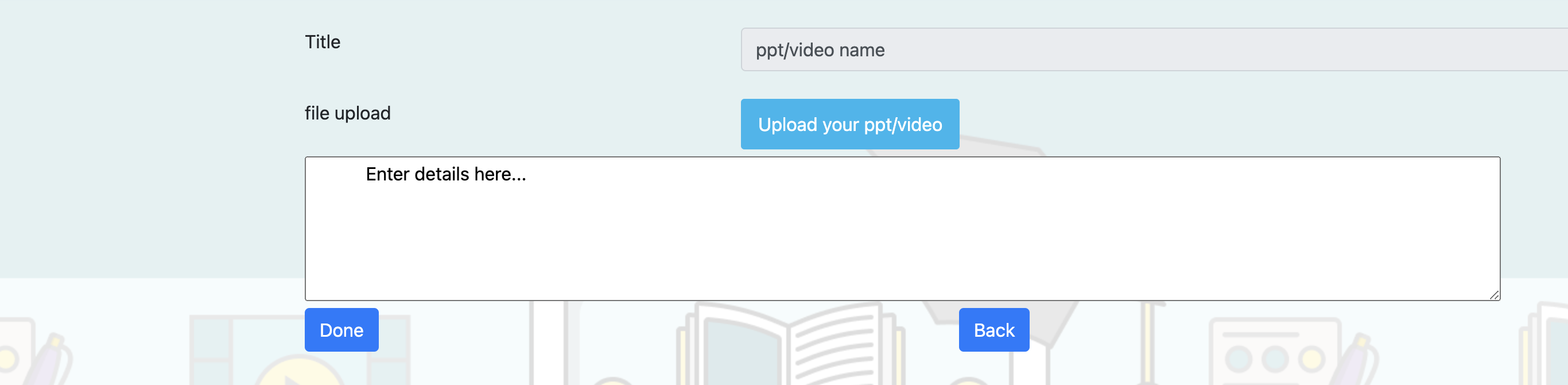












* + Performance Testing: We also performed performance testing to measure the scalability and responsiveness of the services under different loads.

1. System Limitations
   * Our solution is designed for small to medium-sized web applications, and may not be suitable for large-scale applications with high traffic.
   * The solution may not be suitable for applications with strict compliance requirements, as it may not meet certain security and regulatory standards.
2. Conclusion and Reflections
   * We have successfully developed an architectural solution for building and deploying Flask services on AWS. This solution provides a scalable and secure way to build web applications in Python.
   * However, as with any solution, there are limitations that should be considered. Our team will continue to improve and update the solution as necessary.
3. Bibliography
   * <https://aws.amazon.com/cn/cloud-data-migration/#>
   * <https://flask.palletsprojects.com/en/2.2.x/>
   * <https://getbootstrap.com/>
4. Appendix
   * https://docs.aws.amazon.com/?nc2=h\_ql\_doc\_do