Systems Configuration explanation:

The overall systems infrastructure used in AWS is highly-available and fault-tolerant architecture to ensure scalability and availability of the service. The architecture uses cross-region replication and utilizes AWS' multiple availability zones. The microservices architecture uses ECS/ECR features like service scaling and security features. The infrastructure also has optional support for third-party end-users that uses AWS API gateway for future features.

Listed below are the AWS service and other middleware used building the systems infrastructure draft.

- Docker - Microservices
- AWS Native Services (IAM, ECS, ECR, etc)
- AWS IAM
- AWS ECS/ECR
- Nginx
- Ansible
- AWS Cloudformation
- AWS service discovery ECS/Route53 - application cluster discovery on AWS
- AWS Aurora DB/DynamoDB
- API Gateway
- AWS Application Load Balancers
- AWS Lambda with python
- AWS KMS and EC2 Systems Manager Parameter Store
- Infrastructure lifecycle management and Auto scaling ECS applications, EC2 Auto Scaling lifecycle hooks
- Continuous delivery pipeline using CodePipeline, Codebuild and Cloudformation services.

Docker with AWS services -

Leverage AWS to overcome the challenges of running distributed Docker application. Docker provides the best way to build, package, and run modern applications, while Amazon Web Services (AWS) is the world's most popular cloud computing platform.

AWS IAM -

Using AWS IAM to focus on establishing and providing secure access to the AWS account using both the AWS console and command line tools. We can establish a simple, yet effective identity and access management framework that defines administrative roles and enforces the requirement for multi-factor authentication, which is a security best practice that you should always be adopted when securing access to AWS.

AWS ECS and ECR -
AWS ECS provides a managed service for running Docker applications on AWS EC2 infrastructure. ECS is the best way to run Docker applications in AWS. with ECR providing a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. Use **ECR** to host images in a highly available and scalable architecture, which allows to reliably deploy containers for the applications.

NGINX -
Nginx is in each microservices ensuring they are connected, served, authenticated, secured, cached and scaled.

IAC (Infrastructure as Code using Ansible and Cloudformation) -
There will be a strong focus on adopting an Infrastructure as Code approach that leverages Ansible and the AWS CloudFormation service to build and deploy all of the AWS environments and resources.
The primary benefit of this approach is that we can create, update, and destroy complex AWS environments on demand with the click of a button, or the tap of an Enter key.

Ansible is a great open-source tool for orchestrating a complex set of tasks, while CloudFormation will allows us to define all of the AWS resources declaratively via CloudFormation templates.

AWS Aurora DB -
Use Aurora to provide performances five times than a traditional MySQL database and 3 times than a PostgreSQL. Other areas that Aurora delivers is scalability, cost-effectiveness, security, high-availability and is fully managed.

AWS Application Load Balancer -
For servicing requests from internal and external clients, and other supporting resources that collectively form the end-to-end infrastructure required to run our application.

AWS Lambda with Python -
Lambda provides a serverless platform for running custom provisioning code. An important requirement for any deployment methodology is the ability to extend our toolchain to be able to accommodate custom provisioning tasks. Using AWS Lambda service to extend AWS CloudFormation to perform such tasks.

AWS KMS and EC2 Systems Manager Parameter Store -
Used on focusing on managing secrets securely for our environments, creating a simple yet powerful secrets management solution based upon the AWS KMS, or Key Management Service, and EC2 Systems Manager Parameter Store that will inject secrets into the Docker applications at container startup, and also securely provision secrets for other resources in the environments.

Continuous Delivery Pipeline using CodePipeline, Codebuild and Cloudformation Services - This will provide an end-to-end pipeline that automatically and continuously tests, builds, and publishes Docker images on every application source code commit, which then proceeds to continuously deploy the application into a development environment running on AWS, and provides a manual approval step before automatically deploying your application changes into production.

Questions for the PO:

What is the target audience region of the service? This will help in designing the overall infrastructure to lessen latency for the users globally.

Will the service cater to third-party end-users as well? This will aid in the decision if an API gateway is to be used for features to be supported.