1. Voting Endpoint:
   * URL: POST /api/vote
   * Description: This endpoint allows users to vote for a specific product feature.
   * Request Body: JSON object containing the following fields:
     1. "feature\_id": The ID of the product feature being voted on.
     2. "user\_id": The ID of the user casting the vote.
   * Response: HTTP status code indicating the success or failure of the vote.

Example Request:

POST /api/vote

Content-Type: application/json

{ "feature\_id": "P1", "user\_id": "12345" }

1. Result Endpoint:
   * URL: GET /api/result
   * Description: This endpoint retrieves the voting results, including the current cumulative vote count for each product feature
   * Response: JSON object containing the following fields:
     1. "cumulative\_votes": An object mapping each product feature ID to its cumulative vote count.

Example Request:

GET /api/result

Example Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"cumulative\_votes": {

"P1": 100,

"P2": 75,

"P3": 50

}

}

Programming Language: Python

Database: Amazon RDS (Relational Database Service) with PostgreSQL

Cache: Amazon ElastiCache with Redis

Backend API: AWS Lambda and API Gateway to handle the backend API endpoints.

Python with FastAPI

Setting up the environment:

1. Create an AWS account if you don't have one already.
2. Set up an S3 bucket for hosting the frontend application.
3. Set up an RDS instance with PostgreSQL as the database.
4. Create an ElastiCache Redis cluster for caching.
5. Set up AWS Lambda functions and API Gateway for the backend API.
6. Configure security measures like IAM roles, encryption, and network access control.
7. Deploy the frontend application to the S3 bucket.
8. Deploy the backend API using AWS CloudFormation or AWS CDK.