Consider a dynamic web application that receives votes in real-time. Traditionally, architecting such applications meant building out your infrastructure to support both "spiky" and sustained usage over a finite amount of time. In most cases, this required your operations team to overprovision resources, leading to waste outside of high-volume voting periods.

By combining AWS Lambda with other serverless AWS services such as Amazon API Gateway, you can build a powerful, highly available web application that automatically scales up and down to handle large amounts of concurrent votes - all with zero administrative effort required. You can also store and analyze your data in fault-tolerant services like Amazon DynamoDB and Amazon Simple Storage Service.

Users text a vote to a phone number or shortcode provided by a third party like Twilio.

The third party is configured to send the content of the message to an endpoint created by Amazon API Gateway, which then forwards the response to a function built in AWS Lambda. This function extracts the vote from the message content and writes the result and any metadata into a table in Amazon DynamoDB. This table has DynamoDB Streams enabled, which allows you to track changes to your tables on a rolling basis.

Upon update, DynamoDB Streams notifies a second AWS Lambda function, which tallies the votes and writes them back to a second aggregate DynamoDB table that only stores the sum of the votes for each category.

A dashboard to display a summary of votes is created using HTML and JavaScript, and hosted as a static website in Amazon Simple Storage Service (Amazon S3). This page uses the AWS Javascript SDK to query the aggregate DynamoDB table and display the voting results in real-time.

Finally, Amazon Route 53 is used as a DNS provider to create a hosted zone pointing a custom domain name to the Amazon S3 bucket.