German Vocabulary Learning Web Application

Introduction

In this document, we will provide a comprehensive overview of the project, detailing its objectives, architecture, and the technical resources that will be employed in the development of the webapp.

Project Overview

My German Vocabulary Learning Web Application aims to help me and other users seeking to learn German gain vocabulary through an interactive dashboard that will display a word in German, a sentence as an example and the meaning. The webapp will count with a large array of words from which users will be able to practice.

Technical resources

The webapp's backend will be created with Flask web framework enabling rapid development of dynamic web content. In addition, Infrastructure as Code (IaC) principles are embraced through the utilization of Terraform. Amazon Web Services (AWS) forms the foundation of our infrastructure, by utilizing services such as Amazon CloudFront, Amazon S3, Amazon EC2, Amazon RDS, and AWS WAF.

Architecture review

The app aims to support user requests by the following process: users are first inspected and filtered by AWS WAF to protect against security threats. CloudFront then delivers static assets efficiently from S3, reducing latency. EC2 hosts the web application logic, interacting with both S3 and RDS to provide a complete user experience. RDS serves as the backend database for data storage and retrieval. The process is graphically explained in the diagram below.

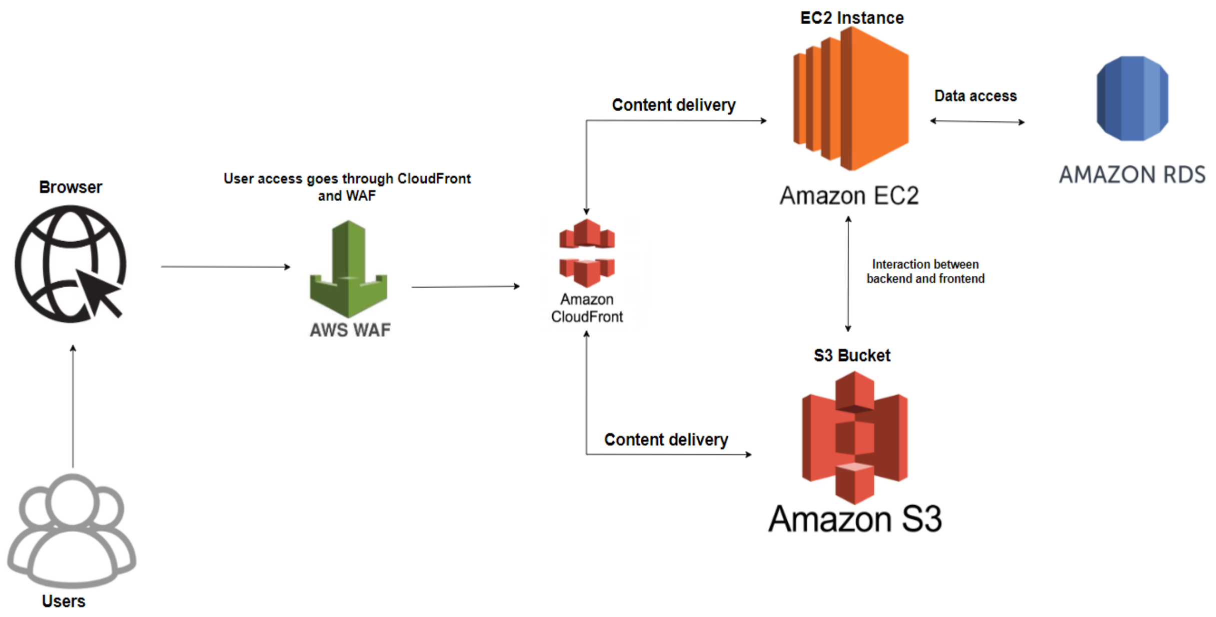

Diagram components:

AWS WAF (Web Application Firewall):

AWS WAF protects the web application by filtering out malicious traffic, preventing common web exploits, and ensuring the security of user interactions.

Amazon CloudFront (CDN):

CloudFront caches and delivers static assets like HTML, CSS, and images to users, reducing latency and improving content delivery speed. It also provides a Content Delivery Network (CDN) to distribute content globally.

Amazon S3 (Static Assets):

S3 is used to store and serve the static components of your web application, ensuring efficient content delivery and storage of non-dynamic resources.

Amazon EC2 (Flask Backend):

EC2 is responsible for dynamic content generation and logic execution. It interacts with both S3 for static content and RDS for database data.

Amazon RDS (Relational Database):

RDS stores user data, vocabulary lists, and other structured information. It allows your Flask application to read and write data securely, providing a reliable and scalable data storage solution.