Technology Stack

ASL – Alphabet Image Recognition

Team-592820

Kanishk Prasad – 21BCE2054   Ebi John Sinjin – 21BCE2082

Mudit Agrawal – 21BCE2008   Divyansh Jain – 21BCE2072

Table 1 – Components and Technology

S.No.	Component	Description	Technology
1.	User Interface	Web UI for the user interaction	HTML, CSS, JavaScript, React, etc.
2.	Application Logic	Backend Logic for image recognition	Python (with Machine Learning)
3.	Database	Data Storage and retrieval	MySQL, NoSQL (e.g., MongoDB)
4.	Cloud Database	Cloud based database service	AWS RDS, Google Cloud
5.	File Storage	Storage for image data and models	AWS S3, Azure Blob Storage, Local File system
6.	External API-1	For additional data or features	Google Cloud Vision API, IBM Watson Language Translator, etc.
7.	Machine Learning Model	Model for ASL Alphabet Image Recognition	Tensorflow, PyTorch, or similar Machine Learning frameworks
8.	Infrastructure	Deployment and server configuration	Cloud Platform (AWS, Google Cloud, Azure, etc.)

Table 2 – Application Characteristics

S. No.	Characteristics	Description	Technology
1.	Open-Source Frameworks	Utilizes open-source frameworks for application development and machine learning. For example, Flask for API, TensorFlow for the machine learning model, scikit-learn for data preprocessing, and OpenCV for image processing.	Flask, TensorFlow, scikit-learn, OpenCV, etc.
2.	Security Implementations	Implements robust security measures including HTTPS (TLS) for secure communication, API tokens for authentication and authorization, Role-Based Access Control (RBAC) for fine-grained access control, strong hashing algorithms such as SHA-256 for password storage, and follows OWASP best practices for web application security.	HTTPS (TLS), API tokens, Role-Based Access Control (RBAC), Hashing (e.g., SHA-256), OWASP best practices
3.	Scalable Architecture	Adopts a scalable architecture using Docker containers for application components, Kubernetes for container orchestration, load balancers for distributing traffic, and AWS Lambda for serverless functions. This architecture ensures the application can handle increased load and maintain responsiveness.	Load balancers, Redundant Servers, High Availability (HA) Architectures
4.	Availability	Ensures high availability with load balancers that distribute traffic across multiple redundant servers. It implements high availability (HA) architectures and disaster recovery strategies to minimize downtime and ensure continuous operation.	Load balancers, Redundant Servers, High Availability (HA) Architectures
5.	Performance	Focuses on optimizing performance by implementing caching mechanisms (e.g., Redis) to reduce database load, utilizing Content Delivery Networks (CDN) for fast content delivery, conducting load testing to identify and address performance bottlenecks, and optimizing database queries for efficient data retrieval.	Caching (Redis), Content Delivery Networks (CDN), Load Testing, Optimal Database Queries