Project Design Phase-II

Technology Stack (Architecture & Stack)

Date	2 November 2023
Team ID	Team-591769
Project Name	ASL - Alphabet Image Recognition
Maximum Marks	4 Marks

Technical Architecture:

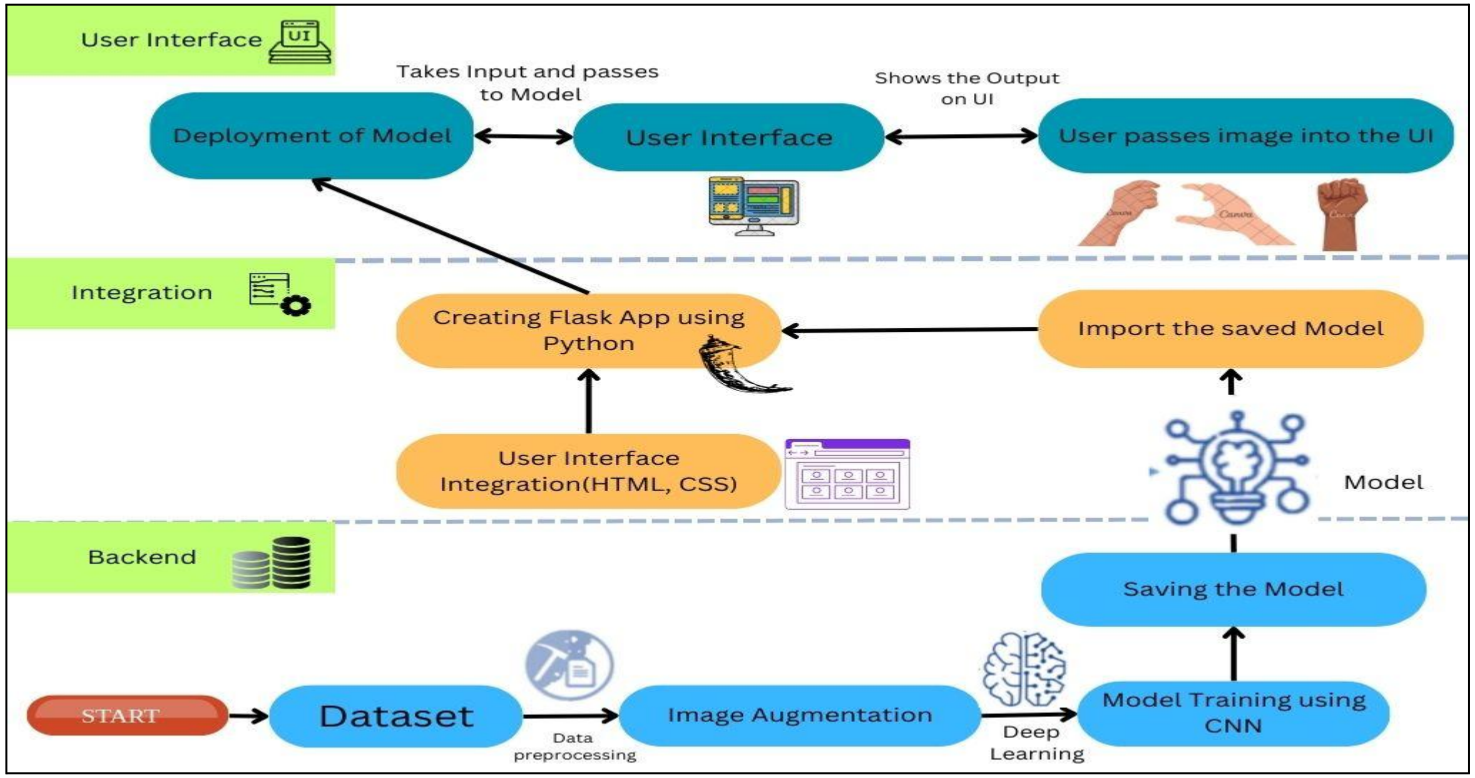

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	Image Acquisition	Capture or obtain images of ASL alphabet signs.	Digital cameras, smartphones, or image databases.
2.	Preprocessing	Prepare images for recognition by applying various transformations and enhancements.	OpenCV (Computer Vision Library), Python.
3.	Image Segmentation	Separate hand signs from the background and identify individual fingers if needed.	OpenCV, Image Processing Techniques.
4.	Feature Extraction	Extract relevant features from the segmented images, such as shape, color, and texture.	Feature extraction algorithms (e.g., Histogram of Oriented Gradients, Color Histograms), Python.
5.	Machine Learning model	Train a machine learning model to recognize ASL alphabet signs based on extracted features.	TensorFlow, PyTorch, Scikit-Learn, Keras, or a custom model using deep learning or traditional machine learning algorithms.
6.	Training data	A dataset of labeled ASL alphabet sign images for model training.	ASL image datasets, data augmentation techniques.
7.	Model Evaluation	Assess the model's accuracy, precision, recall, F1 score, and other relevant metrics.	Cross-validation, evaluation metrics in Python.
8.	Model Deployment	Deploy the trained model for real-time or batch processing of ASL signs.	Cloud platforms (e.g., AWS, Azure, GCP), web servers, APIs.
9.	User Interface	Create a user-friendly interface for users to interact with the ASL alphabet recognition system.	Web development (HTML, CSS, JavaScript), mobile app development (e.g., React Native, Flutter).
10.	Integration with Sign Language	Integrate the ASL recognition system with a sign	APIs, libraries for natural language

	Interpreter	language interpreter to provide translations or responses.	processing (NLP).
11.	Infrastructure (Server / Cloud)	Regularly update and improve the system by collecting user feedback and enhancing the model.	Agile development practices, version control (e.g., Git).

Table-2: Application Characteristics:

S.No	**Characteristics**	**Description**	**Technology**
1.	Open-Source Frameworks	Utilizing open source frameworks can significantly speed up development, reduce costs, and benefit from a collaborative community.	Python for machine learning and image processing (NumPy, OpenCV, Scikit-Learn), TensorFlow or PyTorch for deep learning, and Flask or Django for web application development.
2.	Security Implementations	Ensuring the security of user data and the system is critical. Implement various security measures to protect against data breaches and unauthorized access.	SSL/TLS for secure data transmission and implement encryption for data at rest.

3.	Scalable Architecture	Designing the system to handle increasing loads and users by scaling horizontally or vertically as needed.	Docker for packaging applications and Kubernetes for container orchestration, Auto-scaling on cloud platforms to dynamically allocate resources based on demand, tools like Nginx or HAProxy for distributing traffic across multiple instances.
4.	Availability	Ensuring that the system is always accessible and minimizes downtime.	Setting up failover mechanisms and replicate critical components for high availability, tools likeAWS CloudWatch to monitor system health and performance. Implement backup and recovery strategies to restore the system in case of failures. Use CDNs to distribute content and reduce latency.
5.	Performance	Optimize the system for quick response times and efficient resource utilization.	Implement caching mechanisms (e.g., Redis, Memcached) for frequently accessed data. Tools like Python's cProfile to identify bottlenecks. Apache Spark or Hadoop for distributed data processing.