Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	14 November 2022
Team ID	591955
Project Name	IMAGE CAPTION GENERATION
Maximum Marks	4 Marks

Technical Architecture:

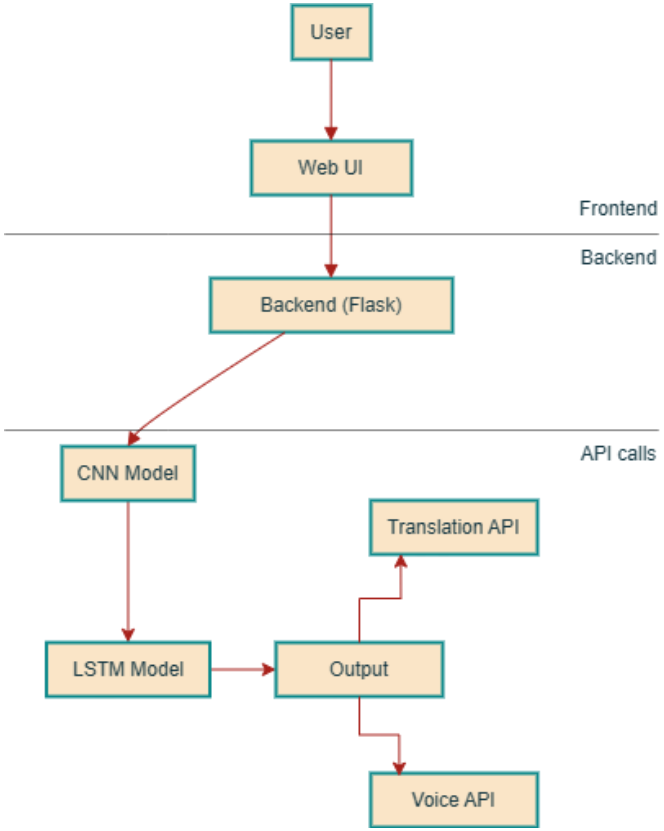

Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web UI with HTML, CSS, JavaScript, and Bootstrap or Tailwind for advanced CSS UI	HTML, CSS, JavaScript, Bootstrap or Tailwind
2.	Application Logic-1	Core logic for the backend process using Flask	Flask (Python Framework)
3.	Application Logic-2	Backend processing for uploaded images using CNN, features are fed into LSTM for caption generation	CNN/RNN
4.	Application Logic-3	Caption generation logic based on processed data from LSTM model	LSTM
5.	Auth Database	User Authentication and user account profile data storing in NoSQL format for use understanding and manipulation	Firebase authentication, Firebase storage (NOSQL)
6.	Cloud Database	Database for storing structured data in SQL format for rapid storage and better analysis of data	Supabase Cloud Storage (Postgre based SQL)
7.	External VOICE API-1	Third party api for direct conversion of converted image to text to voice.	RapidAPI
8.	External API-2	Integration with a Language Translation API for multi-language support and caption translation.	Google Cloud Translation, Microsoft Azure Translator, or open source api
9.	Deep Learning Model	Utilizing ML for image recognition and caption generation	Text Recognition Model, etc.
10.	Infrastructure (Server / Cloud)	Local Server Configuration: local python based environment for testing the web application. Cloud Server Configuration : Can be hosted in any suitable python based environment which can use Deep learning model in backend to process image and send output to client system.	Local System, Platforms like: Python Anywhere, Render, Heroku, Google Cloud, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Utilization of open-source frameworks, such as Flask (Python Framework) for backend and Bootstrap/Tailwind for advanced CSS UI in the front end.	Flask (Python Framework), Bootstrap, Tailwind, HTML, CSS, JS
2.	Security Implementations	Implementation of security measures including the use of AES-256-GCM for encryption for data protection, Single signin methods (only one user can sign in at a time) and Gmail authentication	Single login system, Gmail auth, AES-256-GCM encryption
3.	Scalable Architecture	Adoption of a scalable architecture, utilizing microservices for modular development and can be upgraded from flask backend to Django for robust performance and management.	Flask can be extended to Django application, changing infrastructure from monolithic to microservices.
4.	Availability	Ensuring high availability through the use of load balancers for efficient distribution of incoming traffic, distributed servers for redundancy, and other measures to prevent service downtime.	Load Balancers, Distributed Servers, Redundancy Measures, HTTPS/TLS.
5.	Performance	Design considerations to optimize performance, including managing the number of requests per second, implementing caching mechanisms for quicker data retrieval, and leveraging Content Delivery Networks (CDNs) for efficient content distribution.	Caching, CDNs, Gunicorn for optimizing performance