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

| | |
|---|---|
| Date | 19-november-2023 |
| TeamID | Team-592198 |
| ProjectName | Project-ImageCaptionGeneration |
| MaximumMarks | 4 Marks |

## Technical Architecture:

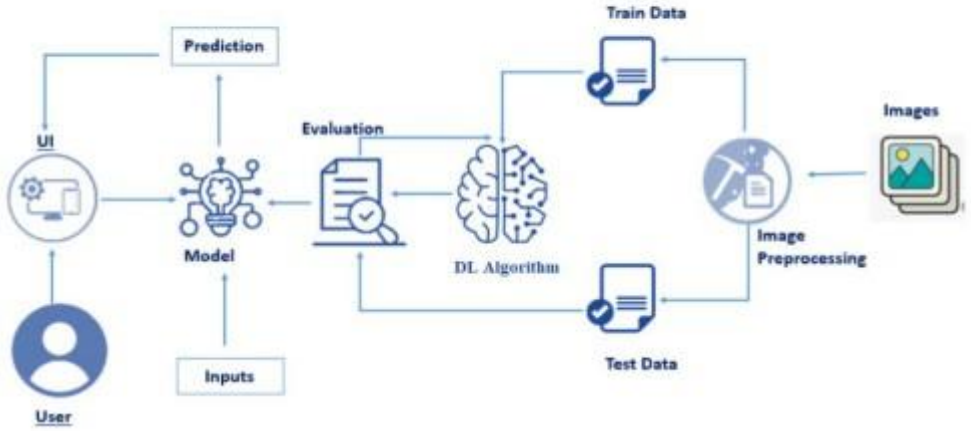

**Table-1:Components&Technologies:**

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1. | User Interface | Enables user interaction to choose and upload images. Utilizes HTML for structure, CSS for styling, and JavaScript for dynamic functionalities. Server-side functionality is implemented using Flask (Python) for handling user requests. | HTML, CSS, JavaScript/Angular Js /React Js etc. |
| 2. | Application Logic | Manages the core logic and processes of the application, handling user requests, integrating with the machine learning model, and coordinating interactions between components. Implemented using Python and Flask for web application logic. | Python, Flask for web application |
| 5. | Database | N/A (Not explicitly used in the project. If needed, could use a database for storing additional data or user-related information.) | N/A |
| 9. | ExternalAPI-2 | N/A (Not explicitly used in the project. If needed, could integrate with external APIs for additional image-related information or services.) | N/A |
| 10. | Machine Learning Model | Utilizes a combination of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to generate image captions. TensorFlow and Keras are used for model implementation. | Tensorflow, keras, python |
| 11. | Infrastructure(Server/ Cloud) | Hosts and deploys the application. The infrastructure is set up on cloud services such as AWS, Google Cloud, or hosted by providers like Heroku, ensuring the application is accessible online. | Cloud Services (e.g., AWS, Google Cloud), Hosting Providers (e.g., Heroku) |

**Table-2:ApplicationCharacteristics:**

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | Utilizes open-source frameworks for various functionalities. Anaconda Navigator, Jupyter Notebook, Spyder, and Flask are mentioned in the project. | Anaconda Navigator, Jupyter Notebook, Spyder, Flask |
| 2. | Security Implementations | Security measures to ensure data protection and secure user interactions are not explicitly mentioned in the project. However, it's advisable to implement secure coding practices, such as input validation, secure communication (HTTPS), and user authentication if handling sensitive data. | Best practices in secure coding, secure communication (HTTPS), user authentication (if applicable) |
| 3. | Scalable Architecture | The scalability of the architecture is not explicitly used in the project. To make the architecture scalable, you might consider containerization (e.g., Docker), orchestration (e.g., Kubernetes), and optimizing the model for efficient inference. Cloud services like AWS or Google Cloud can also provide scalable infrastructure. | Docker, Kubernetes (if needed), Cloud Services (e.g., AWS, Google Cloud) |

| | | | |
|---|---|---|---|
| | | | |

| | | | |
|---|---|---|---|
| 4. | Availability | Ensures the availability of the application for users. Hosting the application on reliable cloud services, implementing load balancing, and monitoring for potential downtime are crucial for maintaining availability. | Cloud Services (e.g., AWS, Google Cloud), Load Balancing, Monitoring Tools |
| 5. | Performance | Focuses on optimizing the performance of the application. This includes efficient algorithms, model optimization, and potentially utilizing caching mechanisms. Performance can be monitored and improved iteratively.. | Loss function: Categorical cross entropy<br>Optimization: Adams optimizer |