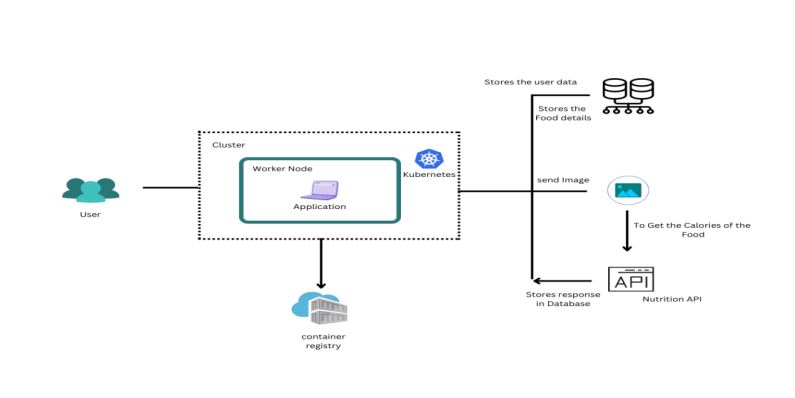
**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 05 October 2022 |
| Team ID | PNT2022TMID24304 |
| Project Name | Project - **Nutrition Assistant Application** |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Web UI. | HTML, CSS, JavaScript |
|  | Application Logic-1 | The user will upload the food picture. Then the user will see the food nutrition value the process will compute | Python, Flask (web Framework), HTML, CSS, JavaScript. |
|  | Application Logic-2 | Get the user’s name, mail and stores the food calories value. Data types: integer, string, Float Number and etc., | MySQL or PostgreSQL |
|  | Application Logic-3 | Through is the application Will compose to the internet | Kubernetes, Docker |
|  | Database | To predict the image that user will upload in the upload image page | Clarifai’s AI-driven Food detection Model API |
|  | Cloud Database | Food API’s for to the nutritional value for the identified food | Food API |
|  | File Storage | Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration: | Local, Cloud Foundry, Kubernetes, etc. Docker. |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | We are using both front and back end here to runs the web application | Flask (Microweb framework) Vue.js |
|  | Security Implementations | List all the security / access controls implemented, use of firewalls etc. | e.g. SHA-256, Encryptions, IAM Controls, OWASP etc. |
|  | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro-services) | Presentation tier- HTML/ CSS/ JavaScript Application tier- Python (API) Data tier- MySQL, PostgreSQL |
|  | Availability | Justify the availability of application (e.g. use of load balancers, distributed servers etc.) | working to reduce the severity and likelihood of problems, closely monitoring applications and infrastructure, keeping technical debt in check, automating recovering mechanisms, and regularly putting those recovery mechanisms to the test. |
|  | Performance | Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc. | Optimize image sizes, use a content delivery network, use website caching and adopt cloud based website monitoring |